Refactoring Composite to Visitor and Inverse Transformation in Java

Akram Ajouli, Julien Cohen

▶ **To cite this version:**

Akram Ajouli, Julien Cohen. Refactoring Composite to Visitor and Inverse Transformation in Java. 2013. hal-00652872v3

HAL Id: hal-00652872
https://hal.science/hal-00652872v3

Submitted on 1 Jul 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Refactoring Composite to Visitor and Inverse Transformation in Java*

Akram Ajouli[1] & Julien Cohen[2]

1: INRIA – ASCOLA team (EMN - INRIA - LINA)
2: Université de Nantes – LINA (UMR 6241, CNRS, Univ. Nantes, EMN)

We describe how to use refactoring tools to transform a Java program conforming to the Composite design pattern into a program conforming to the Visitor design pattern with the same external behavior. We also describe the inverse transformation. We use the refactoring tools provided by IntelliJ IDEA and Eclipse.

Contents

*This is the version 3 of the report. The main difference with previous versions is the description of the computation of the minimum precondition for the base round-trip transformation, for variations and for the use case (see appendices).

1 Introduction

Composite and Visitor patterns have dual properties with respect to modularity: while the Composite pattern (as well as the Interpreter pattern and classic class hierarchies) provides modularity along subtypes and leaves operation definitions crosscut, the Visitor pattern provides modularity along operations and leaves behavior definitions crosscutting with respect to subtypes [GHJV95].

One solution to have modularity along operations *and* subtypes would be to be able to transform automatically a program conforming to the Composite pattern into a program with the same behavior, but which structure would conform to the Visitor pattern, and vice-versa [CDA12].

Chains of elementary refactorings can be used to make design patterns appear [OCN99, Ker04], for instance to introduce the Visitor pattern [MT04, Ker04], or to replace the Visitor pattern by the Interpreter pattern [HKVDSV11]. However, such transformations are not fully automated yet, and our proposal of navigation between several architectures for a same program [CDA12] is not currently workable.

In this report we do preliminary work before automating refactoring based Composite↔Visitor transformations:

1. We give chains of refactoring operations that provide Composite→Visitor and Visitor→Composite transformations for a simple Java program. Each refactoring operation is supported by at least one refactoring tool.

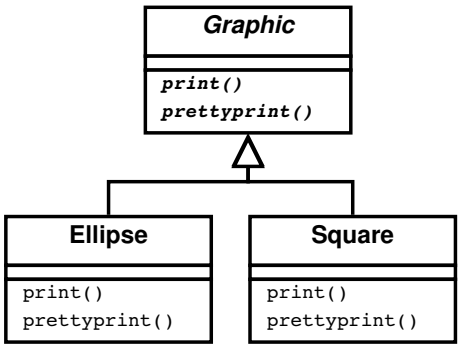
2. We explain how to use the refactoring tools IntelliJ IDEA and Eclipse to perform the needed refactoring operations (composition of several operations of the tools, specific options, applying some operations before being able to perform another one, bugs to overcome, missing operations...).

3. We study variants of the transformations for several variations in the implementation of the patterns.

Our algorithms are validated on a running toy example and on the JHotDraw program [GI].

2 General Approach

We consider the Java program of Fig. 1. It contains a classic class hierarchy: the abstract class *Graphic* has two subclasses, *Square* and *Ellipse*, and two methods, *print* and *prettyprint* implemented in the subclasses. We also consider that two classes *Printer* and *PrettyPrinter* already exist in the program: they will become visitor subclasses.




```
abstract class Graphic {

    abstract public void print();

    abstract public void prettyprint();

}
```

```
class Square extends Graphic {

    int l;

    public void print() {
        System.out.print("Square(" + l + ")");
    }

    public void prettyprint(){
        System.out.print("Square.");
    }
}
```

```
class Ellipse extends Graphic{

    int l1, l2;

    public void print() {
        System.out.print("Ellipse: (" + l1 +"," + l2 + ")");
    }

    public void prettyprint(){
        System.out.print("Ellipse.");
    }
}
```

Figure 1: Base Program (classic class hierarchy)

```
1. ForAll m in LM, c in LC do
     Let visitorname = V(m) in
       MoveMethodWithDelegate(c, m, visitorname)
       RenameMethod(visitorname, m, "visit")
   done

2. AddAbstractSuperClass("Visitor", LV)

3. ForAll c in LC do
     PullUpAbstract(LV, "visit", c, "Visitor")

4. ForAll c in LC do
     ExtractMethod(c, LM, "accept")

5. ForAll m in LM do
     PullUpConcrete(LC, m, S)
```

Figure 2: Simple Class Hierarchy → Visitor transformation [MT04].

In the following algorithms, we make abstraction of the class and method names and number: let LM be the set of traversal functions, LC the set of *concrete* classes in the composite structure, and S the superclass of the composite structure.

Here, LM = {print,prettyprint}, LC = {Ellipse,Square} and S=Graphic.

We also define a function V that maps a name of visitor class to a name of method. We consider here $V(\text{print}) = \textit{Printer}$ and $V(\text{prettyprint}) = \textit{PrettyPrinter}$. We also define $\mathsf{LV} = V(\mathsf{LM}) = \{V(m)\}_{m \in \mathsf{LM}}$.

2.1 Guidelines in the Literature

We start by considering some guidelines given in the literature for introducing an instance of the Visitor pattern into a typical object-oriented class hierarchy. We consider the guidelines of Mens and Tourwé [MT04], rephrased in Fig. 2.

To introduce a visitor pattern, the first obvious step is to move the business code[1] from the class hierarchy to visitor classes (in this section, we consider the target classes for the moved methods already exist in the project). This is done in step 1 (Fig. 2). We move the business code but, in order not to change the interface of the class hierarchy, we keep in the class hierarchy some methods with the same profiles as the original ones, which will be delegators to visitor's methods (see *Move Method* in Fowler [Fow99]).

The new methods in visitor classes are named *visit* so that the visitor classes will all be able to implement the abstract class *Visitor*, which is added afterward (step 2). In visitor classes, there is one method *visit* for each concrete class of the class hierarchy LC (with overloading). They are introduced as abstract methods in the *Visitor* class (step 3).

To introduce the double dispatch which is typical of the visitor pattern without changing the interface of the class hierarchy, another delegation is introduced inside the concrete classes of LC (step 4). The delegate method is named *accept*.

Since the initial methods are now delegators to *accept*, the overriding bodies are the same in the concrete classes of LC, and it can be defined once for all in the super class (step 5).

The refactoring results in the program given in Figs. 3 and 4.

2.2 Automation

If we refer to Fowler [Fow99], a refactoring is manual with checks under the responsibility of the programmer. In the same way, the general guidelines given in Fig. 2 must be *interpreted* by someone which will adapt them to his particular program.

We now consider that the programmer uses a refactoring tool. We consider IntelliJ IDEA but the same is also possible with Eclipse with small variations.

Prepare the move. A first problem occurs with the *Move Method* operation. The refactoring tool cannot move instance methods to a class if there is no reference of the destination class in that method (in parameters or in body). The reason is that the receiver object cannot be inferred otherwise (we consider *instance* methods).

Before moving the methods, we have to create delegates for these methods (to keep the initial interface), then add a parameter of the convenient visitor type to the delegates, then move them (see Fig. 5, step 1).

[1] We call business code the code that defines the operations, here the bodies of *print* and *prettyprint*, which are spread over several classes (by the means of overriding).

```
abstract class Graphic {

    public void print() {
        accept(new PrintVisitor());
    }

    public void prettyprint() {
        accept(new PrettyPrintVisitor());
    }

    public abstract void accept(Visitor v);

}
```

```
class Square extends Graphic {

    int l;

    public void accept(Visitor v) {
        v.visit(this);
    }

}
```

```
class Ellipse extends Graphic{

    int l1, l2;

    public void accept(Visitor v) {
        v.visit(this);
    }

}
```

Figure 3: Program with Visitor (classic class hierarchy)

```
public abstract class Visitor {
  public abstract void visit(Square square);

  public abstract void visit(Ellipse ellipse);
}
```

```
public class PrintVisitor extends Visitor {
  public void visit(Square square) {
    System.out.print("Square(" + square.l + ")");
  }

  public void visit(Ellipse ellipse) {
    System.out.print("Ellipse: (" + ellipse.l1 +"," + ellipse.l2 + ")");
  }
}
```

```
public class PrettyPrintVisitor extends Visitor {
  public void visit(Square s){
    System.out.print("Square.");
  }

  public void visit(Ellipse e){
    System.out.print("Ellipse.");
  }
}
```

Figure 4: Program with Visitor (classic class hierarchy – visitor part)

1. ForAll (m,param) in LM, c in LC do
   Let visitorname = V(m) in
   AddParameterWithDelegate(c,m,param,visitorname)
   MoveMethod(c, m, param+visitorname, visitorname)
   RenameMethod(visitorname, m,param+c, "visit")
   done
2. ExtractSuperClass(LV, "Visitor") // with *visit* abstract methods
3. ForAll c in LC do
   ExtractGeneralMethod(c, LM, "accept", "Visitor")
4. PullUpAbstract(LC, "accept", "Visitor", S)
5. ForAll m in LM do
   PullUpConcrete(LC, m, S)

Figure 5: Simple Class Hierarchy → Visitor transformation (adapted to IntelliJ IDEA)

Restore object type after move. In our example, the pretty-print method does not access to any instance variables or methods (see Fig. 1) of the receiver object. In this case, when the *prettyprint* delegate methods are moved, the tool does not make a parameter of type *Ellipse* or *Square* appear in the resulting method.

This is problematic because we want overloaded *visit* methods (it's a design choice, here we could also use different method names) but the lack of these parameters introduces a name clash.

To solve this, it is sufficient to apply the *Add Parameter* refactoring to the methods which have been moved. We do not make this appear into the algorithm of Fig. 5 because we encapsulate this behavior into the *Move Method* operation. We consider *Move Method* is an abstract operation, which can be implemented by a refactoring tool with a single operation or with a composition/chain of several basic operations. We make the correspondence between abstract operation and tool operations in App. A (see App. A.6).

ExtractSuperClass. Introducing a new superclass and pulling up methods (steps 2 and 3 of Fig. 2) is known as *Extract Superclass* in Fowler [Fow99]. That composite operations is also available in IntelliJ IDEA and Eclipse. For that reason, we use it in Fig. 5 (step 2).

However, in IntelliJ IDEA, we have had to provide an extension of that operation that applies to several classes simuleanously[2] (it was already possible in Eclipse).

Extract Method Accept. In the following code (from *Square* or *Ellipse*), the instruction *o.visit(this)* occurs twice (with a different object *o*).

```
public void print() {
    new PrintVisitor().visit(this);
}

public void prettyprint() {
    new PrettyPrintVisitor().visit(this);
}
```

That instruction has to be extracted into a method *accept* with *o* as a parameter, and the occurrences of that expression will be replaced by *accept(o)*.

The tool IntelliJ IDEA will accept to extract a same method for the two instances only after we introduce a same type for the receiver objects. In practice, we first introduce a new local variable for *new PrintVisitor()* (resp. *new PrettyPrintVisitor()*), then change the type of that variable form *PrintVisitor* (resp. *PrettyPrintVisitor*) to *Visitor*, and then the extraction of the method successes (the two instances are replaced by invocations of that method). The operations used in IntelliJ IDEA are *Introduce Variable* and *Type Migration* (as many other refactoring operations *Type Migration* checks that the change is type safe). One would may also find useful to rename the local variables or the parameter of accept to *v* or *visitor* (operation *Rename*).

The local variables can be inlined afterward (operation *Inline*).

Note that the task of making *accept* act on *Visitors* is left implied in the guidelines of Mens and Tourwé (Fig 2). This task is not explained either by Fowler (*Extract Method* [Fow99]).

[2] *Pull up method refactoring extension* plugin: `http://plugins.intellij.net/plugin/?idea_ce&id=6889`

Again, we encapsulate these elementary changes in the *ExtractGeneralMethod* refactoring operation, defined in App. A.27.

Pull Up. Note that when *accept* is pulled up (step 4 of Fig. 5), IntelliJ IDEA does not add the *@Override* annotation to all the subclasses, but only in the one the operation is called on.

Also, when *print* and *prettyprint* are pulled up (step 5 of Fig. 5), the tool cannot take several classes simultaneously into account, so that the pull up does not verifies that the code are the same in all the concrete classes (in fact they are). Note that for *Pull Up*, Eclipse can take several classes into account (it allows to remove overriding methods in these classes) but it does not checks that the behavior is preserved by this change.

Visibility. In the example program, instance variables are public (package). If they were private or protected, we would have had to make them public so that the moved methods can access them. This does not depend on the way we implement the transformation, but rather to the nature of the Visitor pattern. Note that Eclipse *Move* makes the change automatically while with IntelliJ IDEA you have to do it after or before the *Move*.

Conclusion. We have seen that as soon as we consider a refactoring tool,

1. the guidelines have to be adapted and
2. an algorithm can be defined (and automated).

We have seen also that some steps are implied in the guidelines, and that, on the opposite, some chains of operations of the guidelines can be done with a single tool's operation.

Finally, we have seen that we also have to adapt the chain of operation to characteristics of the initial program. In the following, after having studied a reverse transformation to get the program back to its initial structure, we will see how the algorithm is adapted to variations in the initial program.

3 Composite↔Visitor Transformation Scheme

We now consider an instance of the Composite pattern as the initial program (Fig. 6). The difference between the classic object structure considered before and the Composite structure is *recursion*: the data type is recursive (subclasses make references to the superclass) and the operations are recursive (to traverse trees of that datatype which depth in unknown).

In this section, all the business methods we handle take no parameter and do not return any result, and the traversal process is stateless. These constraints are relaxed in Sec. 4.

We also consider that the visitor classes are not part of the project in the Composite state (unlike in previous section).

3.1 Composite→Visitor Transformation

Let us consider this part in the code of the CompositeGraphic class:

```
    public void print() {
        System.out.print("Composite: " + this + " with: (");
        for (Graphic graphic : childGraphics) {
            graphic.print();
        }
        System.out.println(")");
    }
```

If we apply the previous transformation algorithm (Fig. 5), after the operation *AddParameterWithDelegate* (step 1), we get the following (with IntelliJ IDEA):

```
public void print() {
        print(new PrintVisitor());
    }

    public void print(PrintVisitor v) {
        System.out.print("Composite: " + this + " with: (");
        for (Graphic graphic : childGraphics) {
            graphic.print();
        }
        System.out.println(")");
    }
```

```
abstract class Graphic {

        abstract public void print();

        abstract public void show();

}
```

```
class Ellipse extends Graphic{

 public void print() {
  System.out.println("Ellipse :" + this);
 }

 public void show(){
  System.out.println("Ellipse corresponding to the object " + this + ".");
 }
}
```

```
class CompositeGraphic extends Graphic {

    private ArrayList<Graphic> mChildGraphics = new ArrayList<Graphic>();

    public void print() {
        System.out.println("Composite:");
        for (Graphic graphic : mChildGraphics) {
            graphic.print();
        }
    }

    public void prettyprint(){
        System.out.println("Composite " + this + " composed of:");
        for (Graphic graphic : mChildGraphics) {
            graphic.prettyprint();
        }
        System.out.println("(end of composite)");
    }

    public void add(Graphic graphic) {
        mChildGraphics.add(graphic);
    }

    public void remove(Graphic graphic) {
        mChildGraphics.remove(graphic);
    }
}
```

Figure 6: Base Program (class hierarchy)

1) ForAll m in $\mathbb{M}$ do **CreateEmptyClass**(V(m))

2) ForAll m in $\mathbb{M}$ do **CreateIndirectionInSuperClass**(S,m, aux(m))

3) ForAll m in $\mathbb{M}$, c in $\mathbb{C}$ do **InlineMethodInvocations**(c, m, aux(m))

4) ForAll m in $\mathbb{M}$ do **AddParameterWithReuse**(S, aux(m), V(m))

5) ForAll m in $\mathbb{M}$, c in $\mathbb{C}$ do **MoveMethodWithDelegate**(c, aux(m), V(m), "visit")

6) **ExtractSuperClass**($\mathbb{V}$, "Visitor")

7) ForAll m in $\mathbb{M}$ do **UseSuperType**(S, aux(m), V(m), "Visitor")

8) **MergeDuplicateMethods**(S, $\{aux(\mathsf{m})\}_{\mathsf{m}\in\mathbb{M}}$, "accept")

Figure 7: Base Composite→Visitor transformation

We observe that the recursive invocation to *graphic.print()* in the *for* loop has been left unchanged. The code is still functionally correct, but it is problematic for the following reason: if we look at the definition of *Graphic.print()* (in the program at that moment of the transformation, you cannot tell which instance of *print()* will be invoked because *print()* is abstract in the class *Graphic*, but we know that *print()*, as a delegator, will be pulled up to the class *Graphic*), we can see that each invocation of *print()* will result in the construction of a new *PrintVisitor* object.

Here, if possible, one would choose to use a single *PrintVisitor* object instead of creating useless new ones. In fact, there is a means to do this with the IntelliJ IDEA refactorer, but, in order to do that, the *print()* delegator method must be pulled up,[3] which impacts the rest of the algorithm (for instance, the pull-up of step 1 is already done).

This shows that, as soon as we rely on a refactoring tool, the chain of refactoring operations depends on the characteristics of the tool.

For this reason, here we cannot encapsulate the small change in the transformation into a variation of one of the steps of the algorithm, but we have to adapt the whole algorithm. Our algorithm for basic Composite→Transformation is given in Fig. 7.

We use the following notations to abstract the transfomration algorithmds :

- $\mathbb{M}$: set of business methods, here $\mathbb{M}$ =$\{$print,prettyprint$\}$.
- $\mathbb{C}$: set of Composite hierarchy classes except its root, here $\mathbb{C}$ =$\{$Ellipse, CompositeGraphic$\}$
- S: root of the Composite hierarchy, here S = Graphic.
- *vis*: function that generates a visitor class name from a business method name, here V(print) = PrintVisitor.
- $\mathbb{V}$: set of visitor classes, here $\mathbb{V} = \{V(m)\}_{m\in\mathbb{M}}$ =$\{$PrintVisitor, PrettyPrintVisitor$\}$.
- *aux*: function used to generate names of temporary methods from business methods, here aux(print)= printAux.

Note that two bugs were encountered with IntelliJ IDEA at the beginning of our work, but were solved by JetBrains, so that no manual intervention is needed now.

The result of this transformation is given in Figs. 8 and 9.

3.2 Visitor→Composite Transformation

Composite→Visitor transformation is based on moving business code from the data-type class hierarchy to the visitor classes. Now we do the opposite (move business code from visitor classes to composite classes). We proceed with three coarse steps (Fig 10):

i. Replace dynamic dispatch with static dispatch.

ii. In-line the business code from the *visitor* structure to the *composite* structure.

iii. Make some small changes to get the initial Composite pattern structure back.

[3]The trick is to first introduce an indirection (directly in the superclass), then inline the delegator invocation inside the loop, then add the parameter to the delegate, so that the tool is able to insert as new parameter in invocations existing objects instead of using a default value.

```
abstract class Graphic {

        public void print() {
            accept(new PrintVisitor());
        }

        public void prettyprint() {
                accept(new PrettyPrintVisitor());
        }

        public abstract void accept(Visitor v);

}
```

```
class Ellipse extends Graphic{

    public void accept(Visitor v) {
                v.visit(this);
        }
}
```

```
class CompositeGraphic extends Graphic {

    ArrayList<Graphic> mChildGraphics = new ArrayList<Graphic>();

        public void accept(Visitor v) {
                v.visit(this);
        }

    public void add(Graphic graphic) {
        mChildGraphics.add(graphic);
    }

    public void remove(Graphic graphic) {
        mChildGraphics.remove(graphic);
    }
}
```

Figure 8: Program with Visitor (data classes)

```
public abstract class Visitor {

        public abstract void visit(Ellipse ellipse);

        public abstract void visit(CompositeGraphic compositeGraphic);

}
```

```
public class PrintVisitor extends Visitor {

        public void visit(CompositeGraphic compositeGraphic) {
                System.out.println("Composite:");
            for (Graphic graphic : compositeGraphic.mChildGraphics) {
                graphic.accept(this);
            }
        }

        public void visit(Ellipse ellipse) {
                System.out.println("Ellipse :" + ellipse);
        }
}
```

```
public class PrettyPrintVisitor extends Visitor {

        public void visit(CompositeGraphic compositeGraphic) {
                System.out.println("Composite " + compositeGraphic + " composed of:");
                for (Graphic graphic : compositeGraphic.mChildGraphics) {
                graphic.accept(this);
            }
                System.out.println("(end of composite)");
        }

        public void visit(Ellipse ellipse) {
                System.out.println("Ellipse corresponding to the object " + ellipse + ".");
        }
}
```

Figure 9: Program with Visitor (visitor classes)

I) ForAll v in $\mathbb{V}$ do **AddSpecializedMethodInHierarchy**(S, "accept","Visitor",v)

II) **DeleteMethodInHierarchy**(S,accept,"Visitor")

III) ForAll c in $\mathbb{C}$ do **PushDownAll**("Visitor","visit",c)

IV) ForAll v in $\mathbb{V}$, c in $\mathbb{C}$ do **InlineMethod**(v,"visit",c)

V) ForAll m in $\mathbb{M}$ do **RenameMethod**(S,accept,$V(\mathsf{m})$,$aux(\mathsf{m})$)

VI) ForAll m in $\mathbb{M}$ do **RemoveParameter**(S,$aux(\mathsf{m})$,$V(\mathsf{m})$)

VII) ForAll m in $\mathbb{M}$ do **ReplaceMethodDuplication**(S,m)

VIII) ForAll m in $\mathbb{M}$ do **PushDownImplementation**(S,m)

IX) ForAll m in $\mathbb{M}$ do **PushDownAll**(S,$aux(\mathsf{m})$)

X) ForAll m in $\mathbb{M}$, c in $\mathbb{C}$ do **InlineMethod**(c,$aux(\mathsf{m})$)

XI) ForAll v in $\mathbb{V}$ do **DeleteClass**(v)

XII) **DeleteClass**("Visitor")

Figure 10: Base Visitor $\rightarrow$ Composite transformation

Remove Dynamic Dispatch (Fig. 10, steps I and III). We replace the *accept(Visitor)* method by some overloaded methods *accept*, one for each subtype of *Visitor*. This removes all dynamic dispatch in *visit* method invocations, so that their invocations can be inlined afterward. The *visit* methods can also be removed from the *Visitor* class (but not from the concrete visitor classes before they are inlined).

```
abstract class Graphic {

        public void print() {
            accept(new PrintVisitor());
        }

        public void prettyprint() {
                accept(new PrettyPrintVisitor());
        }

    public abstract void accept(PrintVisitor v);

    public abstract void accept(PrettyPrintVisitor v);
}
```

```
class Ellipse extends Graphic{

    public void accept(PrettyPrintVisitor v) {
        v.visit(this);
    }

    public void accept(PrintVisitor v) {
        v.visit(this);
    }
}
```

```
class CompositeGraphic extends Graphic {

    ArrayList<Graphic> mChildGraphics = new ArrayList<Graphic>();

    public void accept(PrettyPrintVisitor v) {
        v.visit(this);
    }

    public void accept(PrintVisitor v) {
        v.visit(this);
    }

    public void add(Graphic graphic) {
        mChildGraphics.add(graphic);
    }

    public void remove(Graphic graphic) {
        mChildGraphics.remove(graphic);
    }
}
```

Figure 11: Reverse-State 1 (data classes)

The result of this is given in Figs. 11 and 12.

Move Business Code (Fig. 10, step IV). The business code in the *visitor* classes is inlined: invocations of the *visit* methods in the *composite* classes are replaced by the corresponding body (the business code) and the *visit* methods are deleted.

The result of this step is given in Fig. 13 (visitor classes are empty).

Remove Visitors and Recover Initial Structure (Fig. 10, steps V to XII). Once the business code has been moved into the convenient classes, the rest of the refactoring operations are common refactoring operations allowing to recover the composite structure (the important part is done before).

The result of this step is given in Fig. 14.

```
public abstract class Visitor {

}
```

```
public class PrintVisitor extends Visitor {

        public void visit(CompositeGraphic compositeGraphic) {
                System.out.println("Composite:");
            for (Graphic graphic : compositeGraphic.mChildGraphics) {
                graphic.accept(this);
            }
        }

        public void visit(Ellipse ellipse) {
                System.out.println("Ellipse :" + ellipse);
        }
}
```

```
public class PrettyPrintVisitor extends Visitor {

        public void visit(CompositeGraphic compositeGraphic) {
                System.out.println("Composite " + compositeGraphic + " composed of:");
                for (Graphic graphic : compositeGraphic.mChildGraphics) {
                graphic.accept(this);
            }
                System.out.println("(end of composite)");
        }

        public void visit(Ellipse ellipse) {
                System.out.println("Ellipse corresponding to the object " + ellipse + ".");
        }
}
```

Figure 12: Reverse-State 1 (visitor classes)

3.3 Result after Round Trip Transformation

After this transformation, the program conforms to the Composite pattern (Fig. 14).

A few more refactorings are necessary to recover exactly the original program: make private the fields that were made public during the Composite→Transformation, reorder method definitions.

Note also that some comments are altered or lost during the transformation (which is not shown by our example).

3.4 Precondition

We use the calculus of Kniesel and Koch [KK04] to compute the minimum precondition that ensures the success of the transformation. Our use of the calculus is described in [CA13]. The computed preconditions are given in App. B. The formal descriptions of the refactoring operations used in the calculus are given in App. A.

```
abstract class Graphic {

        public void print() {
            accept(new PrintVisitor());
        }

        public void prettyprint() {
                accept(new PrettyPrintVisitor());
        }

    public abstract void accept(PrintVisitor v);

    public abstract void accept(PrettyPrintVisitor v);
}
```

```
class Ellipse extends Graphic{

    public void accept(PrettyPrintVisitor v) {
        System.out.println("Ellipse corresponding to the object " + this + ".");     }

    public void accept(PrintVisitor v) {
        System.out.println("Ellipse :" + this);
    }
}
```

```
class CompositeGraphic extends Graphic {

    ArrayList<Graphic> mChildGraphics = new ArrayList<Graphic>();

    public void accept(PrettyPrintVisitor v) {
        System.out.println("Composite " + this + " composed of:");
        for (Graphic graphic : mChildGraphics) {
            graphic.accept(v);
        }
        System.out.println("(end of composite)");
    }

    public void accept(PrintVisitor v) {
        System.out.println("Composite:");
        for (Graphic graphic : mChildGraphics) {
            graphic.accept(v);
        }
    }

    public void add(Graphic graphic) {
        mChildGraphics.add(graphic);
    }

    public void remove(Graphic graphic) {
        mChildGraphics.remove(graphic);
    }
}
```

Figure 13: Reverse-State 2 (data classes)

```
abstract class Graphic {

        public abstract void print();

    public abstract void prettyprint();

}
```

```
class Ellipse extends Graphic{

    public void print() {
        System.out.println("Ellipse :" + this);
    }

    public void prettyprint() {
        System.out.println("Ellipse corresponding to the object " + this + ".");
    }
}
```

```
class CompositeGraphic extends Graphic {

    ArrayList<Graphic> mChildGraphics = new ArrayList<Graphic>();

    public void add(Graphic graphic) {
        mChildGraphics.add(graphic);
    }

    public void remove(Graphic graphic) {
        mChildGraphics.remove(graphic);
    }

    public void print() {
        System.out.println("Composite:");
        for (Graphic graphic : mChildGraphics) {
            graphic.print();
        }
    }

    public void prettyprint() {
        System.out.println("Composite " + this + " composed of:");
        for (Graphic graphic : mChildGraphics) {
            graphic.prettyprint();
        }
        System.out.println("(end of composite)");
    }
}
```

Figure 14: Result after Back Transformations

4 Variants of Transformations for Various Pattern Instances

In this section we consider some variants of either Composite pattern or Visitor pattern and we adapt the algorithm of transformation.

4.1 Methods with Parameters

4.1.1 Considered variation

We consider that some business methods in the Composite structure have parameters, as exemplified by the following method setColor :

```
// in Graphic
abstract void setColor(int c)
```

```
// in Ellipse
int color;
void setColor(int c) { this.color = c; }
```

```
// in CompositeGraphic
void setColor(int c) {
   for (Graphic child : children){child.setColor(c);} }
```

Note that the parameter c of the method setColor is passed to each recursive call (in the class CompositeGraphic).

4.1.2 Target structure

In the Visitor structure (Figs. 8 and 9), the visitor object, which is created by the interface methods of the class Graphic, is passed recursively as parameter of *accept* and as receiver of *visit* invocations. So, to take the parameter c into account, we put it into the state of that visitor object, so that it is available during the traversal:

```
class SetColorVisitor extends Visitor{

 final int c;

 SetColorVisitor (int c){ this.c = c ; }

 void visit(Ellipse e){ e.color = c; }

 void visit(CompositeGraphic g){
  for(Graphic child : g.children){child.accept(this);} }}
```

The method setColor of the Graphic abstract class passes the parameter c to the constructor of the class SetColorVisitor, then passes the resulting visitor object (with c in its state) to the *accept* method:

```
// in Graphic
void setColor(int c) { accept(new SetColorVisitor(c)); }
```

The implementation of *accept* in Ellipse and CompositeGraphic is left unchanged.

4.1.3 Composite→Visitor Transformation

The refactoring operation of step 4 of the basic transformation (Fig. 7) add a visitor parameter to the methods that becomes *accept* later. Here, we do not want to add the visitor parameter to the initial method parameter (such as c), but we want to replace the initial parameter with the visitor. To do that we apply the operation **IntroduceParameterObject** (step 4.A below). Note that the refactoring operation **IntroduceParameterObject** could not be used with methods without parameters.

For that reason, we distinguish methods with parameters and methods without parameters and we introduce the following notation to introduce different treatments in the transformation algorithm:

- $\mathbb{M}_{\mathbb{P}}$: set of methods with parameters, here $\mathbb{M}_{\mathbb{P}} = \{\text{setColor(int c)}\}$.
- $\mathbb{M}_{\mathbb{W}}$: set of methods without parameters, with $\mathbb{M}_{\mathbb{P}} \cup \mathbb{M}_{\mathbb{W}} = \mathbb{M}$ and $\mathbb{M}_{\mathbb{P}} \cap \mathbb{M}_{\mathbb{W}} = \emptyset$.

Introducing a parameter object of type A to a method m(B b) for example creates a class A, moves the parameter b to A as an instance variable and finally changes m(B b) into m(A a). Any old access to b in the body of m will be replaced by a.b.

The initial step 1 is omitted for methods with parameters because the operation **IntroduceParameterObject** creates the new class (step 1.A below replaces step 1).

Here are the deviations from the basic algorithm for this variation:

1.A) ForAll m in $\mathbb{M}_{\mathbb{W}}$ do **CreateEmptyClass**(V(m)) (replaces step 1)

4.A) ForAll m in $\mathbb{M}_{\mathbb{P}}$ do
IntroduceParameterObject(S, aux(m), V(m))

ForAll m in $\mathbb{M}_{\mathbb{W}}$ do
AddParameterWithReuse(S, aux(m), V(m)) (replaces step 4)

4.1.4 Visitor→Composite Transformation

Before deleting visitor classes (step XI) we have to check that there is no references to them in the Composite hierarchy. For the methods without parameters, we just remove the parameters corresponding to the visitor (step VI.A : restriction of step VI to methods without parameters) since at this moment those methods do not use that parameter. For example, at this moment (before step VI), the intermediate method for print in Ellipse is as follows:

```
// in Ellipse
void printaux(PrintVisitor v){
       System.out.println("Ellipse");}
```

For the methods with parameters, instead of deleting the visitor parameter, we have to inline the occurrences of visitor classes to recover the initial parameter c. After applying step X (before deleting visitor classes), the method setColor is as follows:

```
// in Ellipse
void setColor(int c){
       this.color = new SetColorVisitor(c).c;}
```

At this point we apply the operation **InlineParameterObject** which will replace new SetColorVisitor(c).c by c (step XI.A), and then we can delete visitor classes (step XII).

Here is the extension of the back transformation:

VI.A ForAll m in $\mathbb{M}_{\mathbb{W}}$ do (replaces step VI)
RemoveParameter(S,aux(m),V(m))

XI.A ForAll m in $\mathbb{M}_{\mathbb{P}}$ do (before step XI)
InlineParameterObject(S, aux(m), V(m))

4.2 Methods with different return types

4.2.1 Considered variation

We consider now business methods with different return types. For example we consider a program with two methods: Integer perimeter() and String toString().

4.2.2 Target Structure

Since we have methods with different return types, we cannot use void to the *accept* method. One solution is to have an *accept* method variant for each return type by the means of overloading. But this breaks the beauty of the Visitor pattern (one *accept* method for each business method instead of one *accept* method to implement an abstract traversal). To avoid that, we use generic types. In the abstract class Graphic, the *accept* method becomes generic:

```
abstract <T> T accept(Visitor <T> v)
```

Note that the returned type is bound by the type of the visitor class which appears as parameter. Each visitor class represents a business method and its return type. The parameterized visitor structure is as follows:

```
abstract class Visitor <T> {...}
```

```
class PerimeterVisitor extends Visitor <Integer> {...}
```

```
class ToStringVisitor extends Visitor <String> {...}
```

Remark Because of the restriction in the use of generic types in Java, returned types which are raw types, such as int or bool, must be converted to object types such as Integer or Boolean. In the case of void, one can use Object and add a return null statement (we use a refactoring operation to do that).

4.2.3 Composite→Visitor Transformation

We use the following notations in the algorithm corresponding to this variation:

- $\mathbb{R}$: Set of methods and their corresponding return types, here $\mathbb{R}$ ={(prettyprint,String), (eval,Integer)}.

In step 6 of the basic algorithm, the operation **ExtractSuperClass** creates a new abstract class and pulls up abstract declarations of visit methods. In the considered variation, we have to use an extension of the pull up operation that introduces generic types in the super class to be able to insert abstract declarations for methods with different return types.

To deal with this variation we apply the operation **ExtractSuperClassWithoutPullUp** then the operation **PullUpWithGenerics**[4] instead of the operation **ExtractSuperClass** of the step 6 (step 6.B).

6.B **ExtractSuperClassWithoutPullUp**($\mathbb{V}$, "Visitor") ;
ForAll m in $\mathbb{M}$, c in $\mathbb{C}$ do
PullUpWithGenerics(V(m), "visit","Visitor") (replaces 6)

4.2.4 Visitor→Composite Transformation

At the step I of the base algorithm, we must specify the return type of each *accept* method. The convenient return types could be identified directly from return types of visit methods existing in concrete visitors. This is done by the operation **AddSpecialisedMethodWithGenerics** (step I.B).

I.B ForAll v in $\mathbb{V}$ do
AddSpecializedMethodWithGenerics(S,"accept",$\mathbb{R}$,
"Visitor",v) (replaces I)

4.3 Class Hierarchies with Several Levels

4.3.1 Considered variation

We consider that the Composite hierarchy has multiple levels, with a random repartition of business code: some business methods are inherited, and some other are overridden.

For example, we consider the class Ellipse has a subclass ColoredEllipse where the method print is overridden whereas the second method prettyprint is inherited:

```
class ColoredEllipse extends Ellipse{
 int color;
 ColoredEllipse (int c){ this.color = c; }

 void print{System.out.println(
       "Ellipse colored with " + color); } }
```

4.3.2 Target Structure

In order to have in visitor classes one *visit* method for each class of the Composite hierarchy, the code of the method prettyprint() defined in Ellipse in the Composite structure and inherited by ColoredEllipse, is placed in the methods visit(ColoredEllipse c) and visit(Ellipse e) in PrettyPrintVisitor:

```
class PrettyPrintVisitor extends Visitor{
 void visit(CompositeGraphic g){...}

 void visit(Ellipse e){
            System.out.println("Ellipse :"+ e +".");}

 void visit(ColoredEllipse c){
           System.out.println("Ellipse :"+ c +".");}}
```

[4] http://plugins.jetbrains.com/plugin/?idea_ce&id=6889

4.3.3 Composite→Visitor Transformation

In order to push down a duplicate of the inherited method to the right subclass, we apply the operation **PushDownCopy**[5] (step 1.C) before running the basic algorithm.

We use the following notations in the algorithm corresponding to this variation:

- $i(c)$: a function that gives the list of inherited methods of a class ; here $i(\mathsf{ColoredEllipse}) = \{\mathsf{prettyprint()}\}$.
- $s(c)$: a function that gives the superclass of a class.

1.C ForAll c in $\mathbb{C}$, ForAll m in $i(c)$ do (before 1)
PushDownCopy(c,m,$s(c)$)

4.3.4 Visitor→Composite Transformation

First we apply the basic algorithm. Then, in order to get back the initial structure we delete methods (step XII.C) that were initially added in these classes in the step 1.C of the forward transformation.

XII.C ForAll (c,m) in $\mathbb{C}$, ForAll m in $i(c)$ do
DeleteMethod(c,m) (after XII)

4.4 Interface instead of Abstract Class

4.4.1 Considered Variation

We now consider that the root of the Composite hierarchy is not an abstract class but an interface and that there is an intermediary abstract class between it and its subclasses. This architecture is found in real softwares: libraries are often provided by the means of an interface and compiled byte-code (*Facade* pattern).

We suppose that there are no other subclasses implementing the interface.

```
interface Graphic{
 void print();
}
```

```
abstract class AbstractGraphic implements Graphic {
 abstract void print();
}
```

```
class CompositeGraphic extends AbstractGraphic{

ArrayList<Graphic> children = ...

 void print(){
     ...
     for(Graphic child : children){ child.print();}}
}
```

4.4.2 Target Structure

Here is a possible target structure corresponding to the considered variation:

```
interface Graphic{
 void print();
 void accept(Visitor v);
}
```

```
abstract class AbstractGraphic implements Graphic{
 void print(){accept(new PrintVisitor());}
}
```

```
class CompositeGraphic extends AbstractGraphic{
 ArrayList<Graphic> children = ...
 void accept(Visitor v){v.visit(this);}
}
```

[5] **PushDownCopy** consists in applying **Extract Method**, then **Push Down Method**.

```
class PrintVisitor extends Visitor{
  void visit(CompositeGraphic g){
    for(Graphic child : g.children){ child.accept(this);}}
}
```

Note that the loop in visit(CompositeGraphic) is done on objects of type Graphic (not AbstractGraphic).

4.4.3 Composite→Visitor Transformation

To reach the target structure, we have to create a delegator print(){printaux(..)} in the class AbstractGraphic and inline the recursive call of print in CompositeGraphic (steps 2 and 3). But that recursive call refers to the method print declared in the Graphic interface whereas the delegator is defined in the abstract class AbstractGraphic. To solve that, we introduce a downcast to the class AbstractGraphic in the recursive call to print as follows: ((AbstractGraphic) child).print() (step 3.D). This makes the inlining by the refactoring tool possible. This downcast is legal because we suppose that the interface has no other implementation than the abstract class.

After creating the method accept (step 8), we pull up its declaration to the interface Graphic, then we delete the downcast (step 8.D).

```
3.D ForAll m in 𝕄, c in ℂ do
      IntroduceDownCast(c,m,S)                                   (before 3)

8.D pullupAbstractMethod(S, "accept", I)
    ForAll v in 𝕍 do
      DeleteDownCast(v,"accept")                                 (after 8)
```

Real practice of the transformation The algorithms shown above represent the ideal solution to get a Visitor structure. In fact, there is no operation in the refactoring tools we use to manage downcasts. In order to automate the full transformation, we do not use downcasts and do not inline the delegator. As a result we get a Visitor with indirect recursion as follows:

```
// In Graphic
   void print();
```

```
// In AbstractGraphic
   abstract void accept(Visitor v);
   void print(){accept(new PrintVisitor();}
```

```
// In PrintVisitor
   void visit(CompositeGraphic g){
     for(Graphic child : g.children){ child.print();}}
```

We can see that at each recursive invocation a new instance of a Visitor is created. The result is legal but shows a poor use of memory. This problem disappears when the initial Composite structure is recovered. Moreover, if needed, the downcast can be introduced manually (or the refactoring operation can be implemented).

So, in practice, the variation in the algorithm is: do not apply step 3 (nor 3.D); do not apply step 8.D (but step 8).

4.4.4 Visitor→Composite Transformation

After the *practical* Composite→Visitor transformation, the base Visitor→Composite transformation can be applied without performing the step VII.

After the full Composite→Visitor transformation described above (with downcasts), we also have to add and remove some downcasts to recover the Composite structure (before step VII and after VII).

4.5 Preconditions

The preconditions for the transformations described in this section are given in App. B.

5 Application to JHotDraw

In this section we we apply our transformation to the JHotDraw framework.

We used a pattern detection tool to detect the Composite pattern in JHotDraw. There are two Composite structures, a simple one (2 classes and 1 method) and a larger one with 18 classes and 6 business methods. We consider the largest one in our study since it shows the four variations presented above. We aliment the transformation algorithm with the following data:

- S = AbstractFigure.
- $\mathbb{C}$ = { EllipseFigure, DiamondFigure, RectangleFigure, RoundRectangleFigure, TriangleFigure, TextFigure, BezierFigure, TextAreaFigure, ... }.
- $\mathbb{M}_{\mathbb{P}}$ = { basicTransform (AffineTransform tx), contains(Point2D.Double p), setAttribute(AttributeKey key,Object value), findFigureInside(Point2D.Double p), addNotify(Const "Drawing d),removeNotify(Drawing d)}.
- $\mathbb{M}_{\mathbb{W}} = \emptyset$.
- $\mathbb{R}$ = { (basicTransform,Void), (contains,Boolean), (setAttribute, Void), (findFigureInside,Figure), (addNotify, Void), (removeNotify, Void)}.
- i(LineConnectionFigure) = {findFigureInside, setAttribute,contains}, $i(...) = ...$
- s(LineConnectionFigure)= {BezierFigure}, $s(...)= ...$

5.1 From Composite to Visitor

To switch from the Composite structure of JHotDraw to its Visitor structure we apply the following sequence of steps: 1.C ; 2 ; 4.A ; 5 ; 6.B ; 7 ; 8.

5.2 From Visitor to Composite

To recover the initial structure, we apply the following: steps I.B ; II; III; IV ; V ; VI.A; VIII ; IX ; X ; XI.A ; XI ; XII ; XII.C.

5.3 Usability of JHotDraw transformation

Since JHotDraw Composite contains 18 classes and 6 business methods, if one add another functionality (a business method) to the program he must make all these classes aware about this modification. Thanks to our transformation, we could perform this evolution as a modular maintenance by applying the Composite to Visitor transformation and adding the new functionality as only one module in the Visitor structure. The reverse way of the transformation could put then the added code in the right place of the initial structure.

5.4 Generated Precondition

The computed minimum precondition that ensures success of the round-trip transformation is given in App. C.

6 Related work

6.1 Refactoring to Patterns

Using chains of elementary refactoring operations to introduce design patterns into programs is not new. The idea is first proposed by Batory and Takuda [BT95].

Ó Cinnéide [OC00] give transformation to introduce several patterns but not the Visitor (he considers in [OC00] that automating the introduction of a visitor pattern is impractical).

Kerievsky [Ker04] proposes two sets of guidelines to introduce Visitor patterns. The first one is similar to the one from Mens and Tourwé [MT04] described in Sec. 2. The second one applies to an "external accumulation": instead of transforming an operation defined by overriding methods in the class hierarchy, it applies to an operation defined outside of the class hierarchy by a switch on the type of an object with *instanceof* and type casts. Neither Mens and Tourwé [MT04] nor Kerievsky [Ker04] give the inverse transformation.

Hills et al. [HKVDSV11] have transformed a program based on a Visitor pattern to introduce a Visitor pattern instead (the Visitor pattern is similar to the Composite pattern). Their transformation is automated, with a few interactions with the user. Since their transformation is dedicated to a specific program and is not abstractly described, it requires some work to be applied to other programs.

Jeaon et al. [JLB02] provide automatic inference of sequence of refactoring operations allowing to reach design pattern based versions of programs. Sudan et al. [PRSK10] provide an inference of a sequence of refactoring operations allowing to pass from a given version of a program to a second given version. Such tools could be used to infer variations of our transformation algorithms for variations in initial programs, or to infer transformations between other patterns.

6.2 Building Complex Refactoring Operations

The transformations we aim at can be seen as complex/composed refactoring operations. As each refactoring operation has specific preconditions, and as we use a large number of elementary transformations, assistance for building such transformations would be valuable. Several works provide languages to build or compose refactoring operations. Ó Cinnéide and Nixon [OCN00] show how to compose elementary refactoring operations with pre/post-conditions, as well as Kniesel and Koch [KK04].

Verbaere et al. propose a language dedicated to building refactoring operations [VEdM06], and Klint et al. propose a language dedicated to program manipulation [KSV09], which they have used to build the Visitor→Interpreter transformation [HKVDSV11].

6.3 Design Patterns Discovery

To provide a fully automated transformation, detection of the occurrences of the initial design pattern must be automated. Several work exist in that domain. Smith and Scott provide a tool that discovers variants of a design pattern in a given program [SS03]. Such tools can be used to automatically provide inputs to our transformations.

On the opposite, some tools detect pattern precursors, anti-patterns or code smells [RJ04, MGL06], but, in our approach, we consider that the initial program has already a good design.

7 Conclusion

In this report:

- We have shown how to use refactoring operations to transform a Java program conforming to the Composite pattern (or Interpreter pattern) into a program (still in Java) conforming to the Visitor pattern and vice versa.
- We have explained how to use a refactoring tool (IntelliJ IDEA) to perform these transformations.
- We have discussed some variations in transformations to adapt to variations in the initial programs.
- We have computed preconditions for the proposed transformations.

This work is a first step towards automation of these transformations so that the user does not have to perform each basic refactoring with a refactoring tool. On the example of the JHotDraw program, automation can reduce transformation time from 8 hours to a few minutes. This kind of automated transformation can be used to provide different versions of a same programs with different properties with respect to modularity [CDA12].

References

[BT95] Don Batory and Lance Tokuda. Automated software evolution via design pattern transformations. Technical report, University of Texas at Austin, Austin, TX, USA, 1995.

[CA13] Julien Cohen and Akram Ajouli. Practical use of static composition of refactoring operations. In *ACM Symposium On Applied Computing*, Portugal, March 2013.

[CDA12] Julien Cohen, Rémi Douence, and Akram Ajouli. Invertible program restructurings for continuing modular maintenance. In *Soft. Maintenance and Reengineering (CSMR), 16th European Conf. on*, pages 347–352, 2012.

[Fow99] Martin Fowler. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley, 1999.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[GI] Erich Gamma and IFA Informatik. JHotDraw as Open-Source Project. `http://www.jhotdraw.org/`.

[HKVDSV11] Mark Hills, Paul Klint, Tijs Van Der Storm, and Jurgen Vinju. A case of visitor versus interpreter pattern. In *Proceedings of the 49th international conference on Objects, models, components, patterns*, TOOLS'11, pages 228–243, Berlin, Heidelberg, 2011. Springer-Verlag.

[JLB02] Sang-Uk Jeon, Joon-Sang Lee, and Doo-Hwan Bae. An automated refactoring approach to design pattern-based program transformations in Java programs. In *Proceedings of the Ninth Asia-Pacific Software Engineering Conference*, APSEC '02, pages 337–, Washington, DC, USA, 2002. IEEE Computer Society.

[Ker04] Joshua Kerievsky. *Refactoring to Patterns.* Pearson Higher Education, 2004.

[KK04] Günter Kniesel and Helge Koch. Static composition of refactorings. *Science of Computer Programming*, 52(Issues 1-3):9–51, Aug. 2004. Special Issue on Program Transformation.

[Koc02] Helge Koch. Ein refactoring-framework fr Java (in german). Diploma thesis, CS Dept. III, University of Bonn, Germany, April 2002.

[KSV09] Paul Klint, Tijs van der Storm, and Jurgen Vinju. Rascal: A domain specific language for source code analysis and manipulation. In *Proceedings of the 2009 Ninth IEEE International Working Conference on Source Code Analysis and Manipulation*, SCAM '09, pages 168–177, Washington, DC, USA, 2009. IEEE Computer Society.

[MGL06] Naouel Moha, Yann-Gael Gueheneuc, and Pierre Leduc. Automatic generation of detection algorithms for design defects. In *Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering*, pages 297–300, Washington, DC, USA, 2006. IEEE Computer Society.

[MT04] Tom Mens and Tom Tourwé. A survey of software refactoring. *IEEE Trans. Softw. Eng.*, 30:126–139, February 2004.

[OC00] Mel Ó Cinnéide. *Automated Application of Design Patterns: A Refactoring Approach.* PhD thesis, Trinity College, Dublin, Oct. 2000.

[OCN99] M. Ó Cinnéide and P. Nixon. A methodology for the automated introduction of design patterns. In *Proceedings of the IEEE International Conference on Software Maintenance*, ICSM '99, pages 463–, Washington, DC, USA, 1999. IEEE Computer Society.

[OCN00] Mel Ó Cinnéide and Paddy Nixon. Composite refactorings for Java programs. In *Proc. of the Workshop on Formal Techniques for Java Programs, ECOOP*, 2000.

[PRSK10] Kyle Prete, Napol Rachatasumrit, Nikita Sudan, and Miryung Kim. Template-based reconstruction of complex refactorings. In *IEEE International Conference on Software Maintenance (ICSM)*, Sept. 2010.

[RJ04] J. Rajesh and D. Janakiram. Jiad: a tool to infer design patterns in refactoring. In *Proceedings of the 6th ACM SIGPLAN international conference on Principles and practice of declarative programming*, PPDP '04, pages 227–237, New York, NY, USA, 2004. ACM.

[SS03] Jason M. Smith and David Stotts. SPQR: Flexible automated design pattern extraction from source code. In *18th IEEE Intl. Conf. on Automated Soft. Eng.*, pages 215–224. IEEE Computer Society Press, 2003.

[VEdM06] Mathieu Verbaere, Ran Ettinger, and Oege de Moor. Jungl: a scripting language for refactoring. In *Proceedings of the 28th international conference on Software engineering*, ICSE '06, pages 172–181, New York, NY, USA, 2006. ACM.

A Refactoring Operations

In this appendix, we define refactoring operations we use in our transformations. For each operation, we give an example to explain its behavior, and we tell how it is performed with IntelliJ IDEA or Eclipse. We give some preconditions when an operation applies only in a specific case. These preconditions are neither minimal (they can be refined into weaker conditions) nor complete (they are sufficient in our basic examples, but not in some situations we have not considered). Also, some preconditions dealing with name clashes are left implied.

The given *backward descriptions* are used for static composition of refactorings [KK04] (see App. B).

A.1 CreateEmptyClass

Overview: CreateEmptyClass (classname c): this operation is used t add a new class c.

Refactoring tools. *new Class* in Eclipse and IntelliJ IDEA.

Precondition.
$(\neg existsType(c))$

Backward Description.

$\texttt{ExistsClass}(c) \mapsto \top$

$\texttt{ExistsType}(c) \mapsto \top$

$\texttt{IsAbstractClass}(c) \mapsto \bot$

$\texttt{ExistsMethodDefinition}(c, Y) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(c, Y, [\,]) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(c, Y, [T1]) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(c, Y, [T1; T2]) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(c, Y, [T1; T2; T3]) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(c, Y, [T1; T2; T3; T4]) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(c, Y, [T1; T2; T3; T4; T5]) \mapsto \bot$

$\texttt{IsInheritedMethod}(c, Y) \mapsto \texttt{IsVisible}(\mathit{java.lang.Object}, Y, c)$

$\texttt{IsInheritedMethodWithParams}(c, Y, [\,]) \mapsto \texttt{IsVisibleMethod}(\mathit{java.lang.Object}, Y, [\,], c)$

$\texttt{IsInheritedMethodWithParams}(c, Y, [T1]) \mapsto \texttt{IsVisibleMethod}(\mathit{java.lang.Object}, Y, [T1], c)$

$\texttt{IsInheritedMethodWithParams}(c, Y, [T1; T2]) \mapsto \texttt{IsVisibleMethod}(\mathit{java.lang.Object}, Y, [T1; T2], c)$

$\texttt{IsInheritedMethodWithParams}(c, Y, [T1; T2; T3]) \mapsto \texttt{IsVisibleMethod}(\mathit{java.lang.Object}, Y, [T1; T2; T3], c)$

$\texttt{IsInheritedMethodWithParams}(c, Y, [T1; T2; T3; T4]) \mapsto \texttt{IsVisibleMethod}(\mathit{java.lang.Object}, Y, [T1; T2; T3; T4], c)$

$\texttt{IsInheritedMethodWithParams}(c, Y, [T1; T2; T3; T4; T5]) \mapsto \texttt{IsVisibleMethod}(\mathit{java.lang.Object}, Y, [T1; T2; T3; T4; T5], c)$

$\texttt{IsSubType}(c, X) \mapsto \bot(\mathit{condition})$

$\texttt{ExtendsDirectly}(c, X) \mapsto \bot(\mathit{condition})$

$\texttt{ExistsMethodDefinitionWithParams}(B, Y, [c]) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(B, Y, [c; T1]) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(B, Y, [T1; c]) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(B, Y, [T1; T2; c]) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(B, Y, [T1; c; T2]) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(B, Y, [c; T1; T2]) \mapsto \bot$

$\texttt{IsInheritedMethodWithParams}(B, Y, [c]) \mapsto \bot$

$\texttt{IsInheritedMethodWithParams}(B, Y, [c; T1]) \mapsto \bot$

$\texttt{IsInheritedMethodWithParams}(B, Y, [T1; c]) \mapsto \bot$

$\texttt{IsInheritedMethodWithParams}(B, Y, [T1; T2; c]) \mapsto \bot$

$\texttt{IsInheritedMethodWithParams}(B, Y, [T1; c; T2]) \mapsto \bot$

$\texttt{IsInheritedMethodWithParams}(B, Y, [c; T1; T2]) \mapsto \bot$

$\texttt{ExistsParameterWithName}(B, Y, [c], P) \mapsto \bot$

$\texttt{ExistsParameterWithName}(B, Y, [c; T1], P) \mapsto \bot$

$\texttt{ExistsParameterWithName}(B, Y, [T1; c], P) \mapsto \bot$

$\texttt{ExistsParameterWithName}(B, Y, [T1; c; T2], P) \mapsto \bot$

$\texttt{ExistsParameterWithName}(B, Y, [T1; T2; c], P) \mapsto \bot$

$\texttt{ExistsParameterWithName}(B, Y, [c; T1; T2], P) \mapsto \bot$

$\texttt{ExistsParameterWithType}(B, Y, [c], P) \mapsto \bot$

$\texttt{ExistsParameterWithType}(B, Y, [c; T1], P) \mapsto \bot$

$\texttt{ExistsParameterWithType}(B, Y, [T1; c], P) \mapsto \bot$

$\texttt{ExistsParameterWithType}(B, Y, [T1; c; T2], P) \mapsto \bot$

$\texttt{ExistsParameterWithType}(B, Y, [T1; T2; c], P) \mapsto \bot$

$\texttt{ExistsParameterWithType}(B, Y, [c; T1; T2], P) \mapsto \bot$

$\texttt{IsUsedMethodIn}(c, B, Y) \mapsto \bot$

$\texttt{IsUsedMethod}(c, B, [T1]) \mapsto \bot$

$\texttt{IsUsedMethod}(c, B, [T1; T2]) \mapsto \bot$

$\texttt{IsUsedMethod}(c, B, [T1; T2; T3]) \mapsto \bot$

$\texttt{IsUsedMethod}(c, B, [T1; T2; T3; T4]) \mapsto \bot$

$\texttt{IsUsedConstructorAsMethodParameter}(c, B, Y) \mapsto \bot$

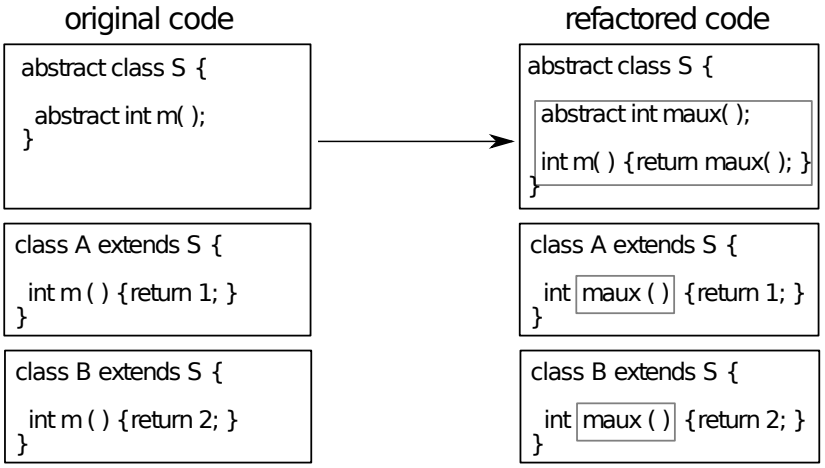
$\texttt{IsUsedConstructorAsInitializer}(c, B, Y) \mapsto \bot$

$\texttt{IsUsedConstructorAsObjectReceiver}(c, B, Y) \mapsto \bot$

$\texttt{IsUsedConstructorAsMethodParameter}(B, c, Y) \mapsto \bot$
$\texttt{IsUsedConstructorAsInitializer}(B, c, Y) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(B, c, Y) \mapsto \bot$
$\texttt{IsSubType}(B, c) \mapsto \bot$
$\texttt{ExtendsDirectly}(B, c) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(B, Y, [P], [c]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(B, Y, [P], [c; T1]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(B, Y, [P], [T1; c]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(B, Y, [P], [T1; c; T2]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(B, Y, [P], [T1; T2; c]) \mapsto \bot$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(c, T1, T2, T3) \mapsto \top$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(c, T1, T2) \mapsto \top$
$\texttt{BoundVariableInMethodBody}(c, T1, T2) \mapsto \bot$
$\texttt{ExistsField}(c, F) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(c, Y, T1, X) \mapsto \bot$
$\texttt{ExistsAbstractMethod}(c, Y) \mapsto \bot$
$\texttt{IsIndirectlyRecursive}(c, Y) \mapsto \bot$
$\texttt{IsVisibleMethod}(c, Y, [T1], B) \mapsto \bot$
$\texttt{IsVisibleMethod}(c, Y, [T1; T2], B) \mapsto \bot$
$\texttt{IsVisibleMethod}(c, Y, [T1; T2; T3], B) \mapsto \bot$
$\texttt{IsVisibleMethod}(B, Y, [B], c) \mapsto \bot$
$\texttt{IsVisibleMethod}(B, Y, [B; T1], c) \mapsto \bot$
$\texttt{IsVisibleMethod}(B, Y, [T1; B], c) \mapsto \bot$
$\texttt{IsVisibleMethod}(B, Y, [T1; T2; B], c) \mapsto \bot$
$\texttt{IsVisibleMethod}(B, Y, [T1; B; T2], c) \mapsto \bot$
$\texttt{IsVisibleMethod}(B, Y, [B; T1; T2], c) \mapsto \bot$
$\texttt{IsInverter}(c, Y, T1, T2) \mapsto \bot$
$\texttt{IsDelegator}(c, Y, X) \mapsto \bot$
$\texttt{IsAbstractClass}(c) \mapsto \bot$
$\texttt{IsUsedMethodIn}(c, Y, B) \mapsto \bot$
$\texttt{IsUsedMethodIn}(B, Y, c) \mapsto \bot$
$\texttt{IsPrimitiveType}(c) \mapsto \bot$
$\texttt{IsPublic}(c, Y) \mapsto \bot$
$\texttt{IsProtected}(c, Y) \mapsto \bot$
$\texttt{IsPrivate}(c, Y) \mapsto \bot$
$\texttt{IsUsedAttributeInMethodBody}(c, X, Y) \mapsto \bot$
$\texttt{IsGenericsSubtype}(c, [T1], B, [T2]) \mapsto \bot$
$\texttt{IsGenericsSubtype}(c, [T1; T2], B, [T4; T3]) \mapsto \bot$
$\texttt{IsGenericsSubtype}(c, [T1; T2; T3], B, [T4; T5; T6]) \mapsto \bot$
$\texttt{IsInheritedField}(c, F) \mapsto \bot$
$\texttt{IsOverridden}(c, Y) \mapsto \bot$
$\texttt{IsOverloaded}(c, Y) \mapsto \bot$
$\texttt{IsOverriding}(c, Y) \mapsto \bot$
$\texttt{IsRecursiveMethod}(c, Y) \mapsto \bot$
$\texttt{IsRecursiveMethod}(c, Y) \mapsto \bot$
$\texttt{HasReturnType}(c, Y, T1) \mapsto \bot$
$\texttt{HasParameterType}(c, T1) \mapsto \bot$
$\texttt{HasParameterType}(B, c) \mapsto \bot$
$\texttt{MethodHasParameterType}(c, Y, T1) \mapsto \bot$
$\texttt{AllSubclasses}(c, [C1; C2; C3]) \mapsto \bot$
$\texttt{ExtendsDirectly}(c, java.lang.Object) \mapsto \top$

A.2 CreateIndirectionInSuperClass

Overview: CreateIndirectionInSuperclass (classname s, subclasses [a,b], methodname m,types [t,t'], returntype q, newname n) : this operation is used to create an indirection of the method s::m to the method n in all the hierarchy.



Refactoring tools. With IntelliJ IDEA:

- Use *Change Signature* on the method m in class s (select "delegate via overloading method", specify the new name n, specify the desired visibility).

With Eclipse:

- Use *Change Method Signature* on the method m in class s (specify to "keep original method as delegate to changed method", and specify the new name n).

Precondition.
$(\texttt{ExistsClass}(s)$
$\wedge\, \texttt{IsAbstractClass}(s)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t; t'])$
$\wedge\, \texttt{ExistsAbstractMethod}(s, m)$
$\wedge\, \neg\texttt{IsInheritedMethod}(s, n)$
$\wedge\, \neg\texttt{IsInheritedMethodWithParams}(s, n, [t; t'])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(s, n, [t; t'])$
$\wedge\, \texttt{AllSubclasses}(s, [a; b])$
$\wedge\, \texttt{HasReturnType}(s, m, r)$
$\wedge\, \neg\texttt{IsPrivate}(s, m)$
$\wedge\, \neg\texttt{IsPrivate}(a, m)$
$\wedge\, \neg\texttt{IsPrivate}(b, m)$
$\wedge\, \texttt{ExistsMethodDefinition}(s, m)$
$\wedge\, \texttt{ExistsMethodDefinition}(a, m)$
$\wedge\, \texttt{ExistsMethodDefinition}(b, m)$
$\wedge\, \neg\texttt{ExistsMethodDefinition}(s, n)$
$\wedge\, \neg\texttt{ExistsMethodDefinition}(a, n)$
$\wedge\, \neg\texttt{ExistsMethodDefinition}(b, n))$

Backward Description.
$\texttt{ExistsAbstractMethod}(s, n) \mapsto \top$
$\texttt{ExistsAbstractMethod}(s, m) \mapsto \bot$
$\texttt{IsDelegator}(s, m, n) \mapsto \top$
$\texttt{IsInheritedMethodWithParams}(s, n, [t; t']) \mapsto \bot$
$\texttt{IsOverriding}(s, n) \mapsto \bot$
$\texttt{ExistsType}(r) \mapsto \top$
$\texttt{HasReturnType}(s, n, r) \mapsto \texttt{HasReturnType}(s, m, r)$
$\texttt{HasReturnType}(a, n, r) \mapsto \texttt{HasReturnType}(s, m, r)$

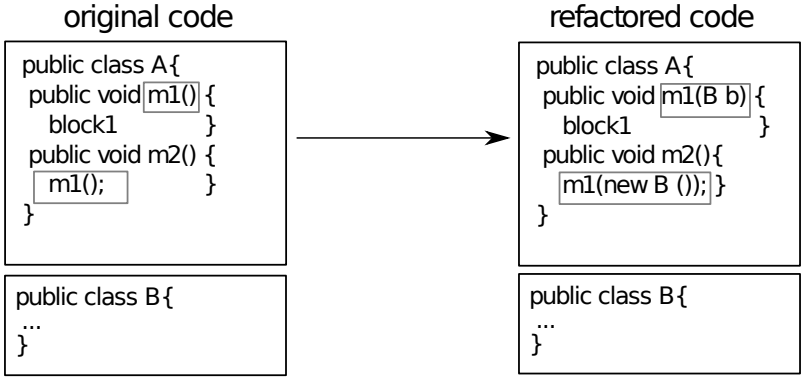
$\texttt{HasReturnType}(b, n, r) \mapsto \texttt{HasReturnType}(s, m, r)$
$\texttt{ExistsMethodDefinition}(s, n) \mapsto \top$
$\texttt{ExistsMethodDefinition}(a, n) \mapsto \top$
$\texttt{ExistsMethodDefinition}(b, n) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(s, n, [t; t']) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(a, n, [t; t']) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(b, n, [t; t']) \mapsto \top$
$\texttt{ExistsParameterWithName}(s, n, [t; t'], N) \mapsto \bot$
$\texttt{ExistsParameterWithName}(a, n, [t; t'], N) \mapsto \bot$
$\texttt{ExistsParameterWithName}(b, n, [t; t'], N) \mapsto \bot$
$\texttt{ExistsParameterWithName}(s, n, [V], N) \mapsto \texttt{ExistsParameterWithName}(s, m, [V], N)$
$\texttt{ExistsParameterWithName}(a, n, [V], N) \mapsto \texttt{ExistsParameterWithName}(a, m, [V], N)$
$\texttt{ExistsParameterWithName}(b, n, [V], N) \mapsto \texttt{ExistsParameterWithName}(b, m, [V], N)$
$\texttt{ExistsMethodDefinition}(a, m) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(b, m) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(a, m, [t; t']) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [t; t']) \mapsto \bot$
$\texttt{IsIndirectlyRecursive}(a, n) \mapsto \texttt{IsRecursiveMethod}(a, m)$
$\texttt{IsIndirectlyRecursive}(b, n) \mapsto \texttt{IsRecursiveMethod}(b, m)$
$\texttt{ExistsMethodInvocation}(a, n, s, m) \mapsto \texttt{IsRecursiveMethod}(a, m)$
$\texttt{ExistsMethodInvocation}(b, n, s, m) \mapsto \texttt{IsRecursiveMethod}(b, m)$
$\texttt{ExistsMethodInvocation}(s, m, a, n) \mapsto \top$
$\texttt{ExistsMethodInvocation}(s, m, b, n) \mapsto \top$
$\texttt{BoundVariableInMethodBody}(s, n, V) \mapsto \texttt{BoundVariableInMethodBody}(s, m, V)$
$\texttt{BoundVariableInMethodBody}(a, n, V) \mapsto \texttt{BoundVariableInMethodBody}(a, m, V)$
$\texttt{BoundVariableInMethodBody}(b, n, V) \mapsto \texttt{BoundVariableInMethodBody}(b, m, V)$
$\texttt{IsOverloaded}(s, n) \mapsto \bot$
$\texttt{IsOverloaded}(a, n) \mapsto \bot$
$\texttt{IsOverloaded}(b, n) \mapsto \bot$
$\texttt{IsOverridden}(s, n) \mapsto \bot$
$\texttt{IsOverridden}(a, n) \mapsto \texttt{IsOverridden}(a, m)$
$\texttt{IsOverridden}(b, n) \mapsto \texttt{IsOverridden}(b, m)$
$\texttt{IsOverriding}(a, n) \mapsto \texttt{IsOverriding}(a, m)$
$\texttt{IsOverriding}(b, n) \mapsto \texttt{IsOverriding}(b, m)$
$\texttt{IsRecursiveMethod}(s, n) \mapsto \bot$
$\texttt{IsRecursiveMethod}(a, n) \mapsto \bot$
$\texttt{IsRecursiveMethod}(b, n) \mapsto \bot$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(s, n, N, V) \mapsto \texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(s, m, N, V)$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(a, n, N, V) \mapsto \texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(a, m, N, V)$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(b, n, N, V) \mapsto \texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(b, m, N, V)$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(s, n, N) \mapsto \texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(s, m, N)$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(a, n, N) \mapsto \texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(a, m, N)$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(b, n, N) \mapsto \texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(b, m, N)$
$\texttt{IsPrivate}(s, V) \mapsto \bot$
$\texttt{IsPrivate}(a, V) \mapsto \bot$
$\texttt{IsPrivate}(b, V) \mapsto \bot$
$\texttt{IsPrivate}(s, n) \mapsto \bot$

$\texttt{IsPrivate}(a, n) \mapsto \bot$
$\texttt{IsPrivate}(b, n) \mapsto \bot$
$\texttt{IsOverriding}(a, n) \mapsto \texttt{IsOverriding}(a, m)$
$\texttt{IsOverriding}(b, n) \mapsto \texttt{IsOverriding}(b, m)$
$\texttt{IsDelegator}(a, n, V) \mapsto \texttt{IsDelegator}(a, m, V)$
$\texttt{IsDelegator}(b, n, V) \mapsto \texttt{IsDelegator}(b, m, V)$
$\texttt{IsDelegator}(a, V, n) \mapsto \texttt{IsDelegator}(a, V, m)$
$\texttt{IsDelegator}(b, V, n) \mapsto \texttt{IsDelegator}(b, V, m)$
$\texttt{IsInheritedMethodWithParams}(a, n, [t; t']) \mapsto \texttt{IsVisibleMethod}(s, m, [t; t'], a)$
$\texttt{IsInheritedMethodWithParams}(b, n, [t; t']) \mapsto \texttt{IsVisibleMethod}(s, m, [t; t'], b)$
$\texttt{IsVisibleMethod}(s, m, [t; t'], a) \mapsto \top$
$\texttt{IsVisibleMethod}(s, m, [t; t'], b) \mapsto \top$
$\texttt{MethodIsUsedWithType}(a, n, [t; t'], [t; t']) \mapsto \texttt{MethodIsUsedWithType}(a, m, [t; t'], [t; t'])$
$\texttt{MethodIsUsedWithType}(b, n, [t; t'], [t; t']) \mapsto \texttt{MethodIsUsedWithType}(b, m, [t; t'], [t; t'])$
$\texttt{IsUsedMethod}(a, n, [t; t']) \mapsto \texttt{IsUsedMethod}(a, m, [t; t'])$
$\texttt{IsUsedMethod}(b, n, [t; t']) \mapsto \texttt{IsUsedMethod}(b, m, [t; t'])$
$\texttt{IsUsedMethodIn}(a, n, V) \mapsto \texttt{IsUsedMethodIn}(a, m, V)$
$\texttt{IsUsedMethodIn}(b, n, V) \mapsto \texttt{IsUsedMethodIn}(b, m, V)$
$\texttt{IsInverter}(a, n, V, V1) \mapsto \texttt{IsInverter}(a, m, V, V1)$
$\texttt{IsInverter}(b, n, V, V1) \mapsto \texttt{IsInverter}(b, m, V, V1)$
$\texttt{ExistsMethodInvocation}(a, V, V1, n) \mapsto \texttt{ExistsMethodInvocation}(a, V, V1, m)$
$\texttt{ExistsMethodInvocation}(b, V, V1, n) \mapsto \texttt{ExistsMethodInvocation}(b, V, V1, m)$
$\texttt{IsIndirectlyRecursive}(a, n) \mapsto \texttt{IsIndirectlyRecursive}(a, m)$
$\texttt{IsIndirectlyRecursive}(b, n) \mapsto \texttt{IsIndirectlyRecursive}(b, m)$
$\texttt{BoundVariableInMethodBody}(a, n, V) \mapsto \texttt{BoundVariableInMethodBody}(a, n, V)$
$\texttt{BoundVariableInMethodBody}(b, n, V) \mapsto \texttt{BoundVariableInMethodBody}(b, n, V)$
$\texttt{IsOverridden}(a, n) \mapsto \texttt{IsOverridden}(a, m)$
$\texttt{IsOverridden}(b, n) \mapsto \texttt{IsOverridden}(b, m)$

A.3 AddParameter

(*Add Parameter* in Fowler [Fow99] et [Koc02])

AddParameter(class c, method m, parameterType t, parameterName n, **defaultvalue** e):
Add a parameter of type t to a method m in class c. In method invocations, use the expression e as new parameter.



Refactoring tools. *Change Method signature* in Eclipse tool and *Change Signature* in IntelliJ IDEA.

A.4 AddParameterWithReuse

Overview: AddParameterWithReuse (classname s,subclasses [a,b], methodname m,methodparameters [] ,paramType t, paramName p, usedvalueofparamType defaultvalue): this operation is used to add the parameter p of type t to the parameters of the method s::m, a::m and b::m. Same as AddParameter, but instead of adding a default value for the additional parameter in invocations, use any value with the specified type that is visible from the invocation site.

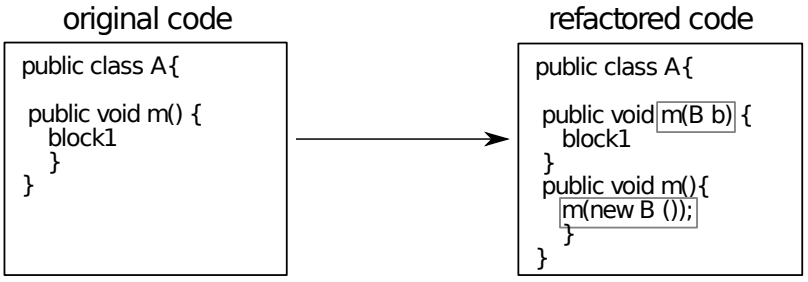
In IntelliJ IDEA, this is specified with the *Any Var* option in *Change Signature*. This is not supported by Eclipse.

Note that when several variables of the specified type are visible, the result in unspecified. In the example of use in this report, the type of the added parameter is a fresh type, and in recursive methods, the only variable of this type is the parameter being introduced so that there is not ambiguity.

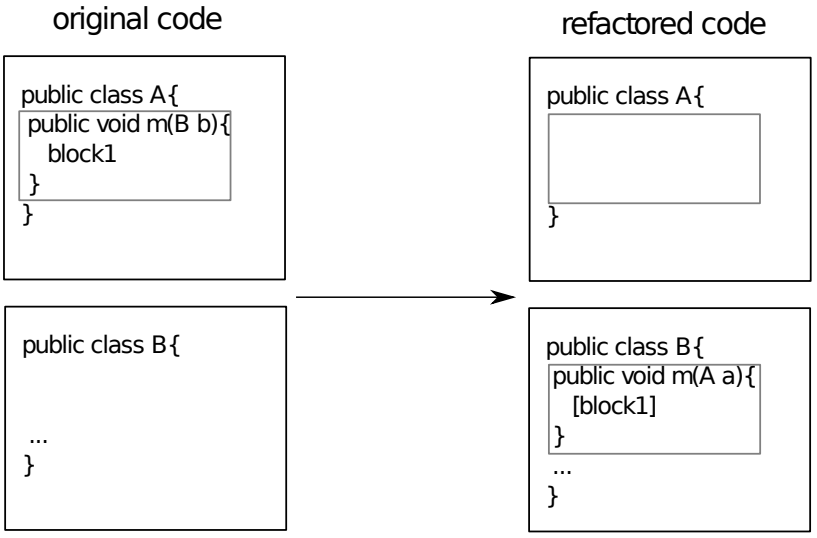
Precondition.
$(\neg\texttt{BoundVariableInMethodBody}(s, m, p)$
$\wedge\, \texttt{ExistsClass}(s)$
$\wedge\, \texttt{ExistsMethodDefinition}(s, m)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [\,])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(s, m, [t])$
$\wedge\, \neg\texttt{IsInheritedMethodWithParams}(s, m, [t])$
$\wedge\, \neg\texttt{ExistsParameterWithName}(s, m, [\,], p)$
$\wedge\, \texttt{ExistsType}(t)$
$\wedge\, \texttt{AllSubclasses}(s, [a; b]))$

Backward Description. $\texttt{ExistsMethodDefinitionWithParams}(s, m, [\,]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(s, m, [t]) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(a, m, [\,]) \mapsto \neg\texttt{ExistsMethodDefinitionWithParams}(a, m, [\,])$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [\,]) \mapsto \neg\texttt{ExistsMethodDefinitionWithParams}(b, m, [\,])$
$\texttt{ExistsMethodDefinitionWithParams}(a, m, [t]) \mapsto \texttt{ExistsMethodDefinitionWithParams}(a, m, [\,])$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [t]) \mapsto \texttt{ExistsMethodDefinitionWithParams}(b, m, [\,])$
$\texttt{ExistsParameterWithName}(s, m, [t], p) \mapsto \top$
$\texttt{ExistsParameterWithName}(a, m, [t], p) \mapsto \top$
$\texttt{ExistsParameterWithName}(b, m, [t], p) \mapsto \top$
$\texttt{ExistsParameterWithType}(s, m, [t], t) \mapsto \top$
$\texttt{ExistsParameterWithType}(a, m, [t], t) \mapsto \top$
$\texttt{ExistsParameterWithType}(b, m, [t], t) \mapsto \top$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(s, m, p, T) \mapsto \top$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(a, m, p, T) \mapsto \top$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(b, m, p, T) \mapsto \top$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(s, m, p) \mapsto$
$(\neg\texttt{IsOverloaded}(s, m)$
$\wedge\, \neg\texttt{IsOverloaded}(a, m)$
$\wedge\, \neg\texttt{IsOverloaded}(b, m))$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(a, m, p) \mapsto$
$(\neg\texttt{IsOverloaded}(s, m)$
$\wedge\, \neg\texttt{IsOverloaded}(a, m)$
$\wedge\, \neg\texttt{IsOverloaded}(b, m))$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(b, m, p) \mapsto$
$(\neg\texttt{IsOverloaded}(s, m)$
$\wedge\, \neg\texttt{IsOverloaded}(a, m)$
$\wedge\, \neg\texttt{IsOverloaded}(b, m))$
$\texttt{IsUsedConstructorAsMethodParameter}(t, s, m) \mapsto \top$
$\texttt{IsUsedConstructorAsMethodParameter}(t, a, m) \mapsto \top$
$\texttt{IsUsedConstructorAsMethodParameter}(t, b, m) \mapsto \top$

A.5 AddParameterWithDelegate



A.6 MoveMethod



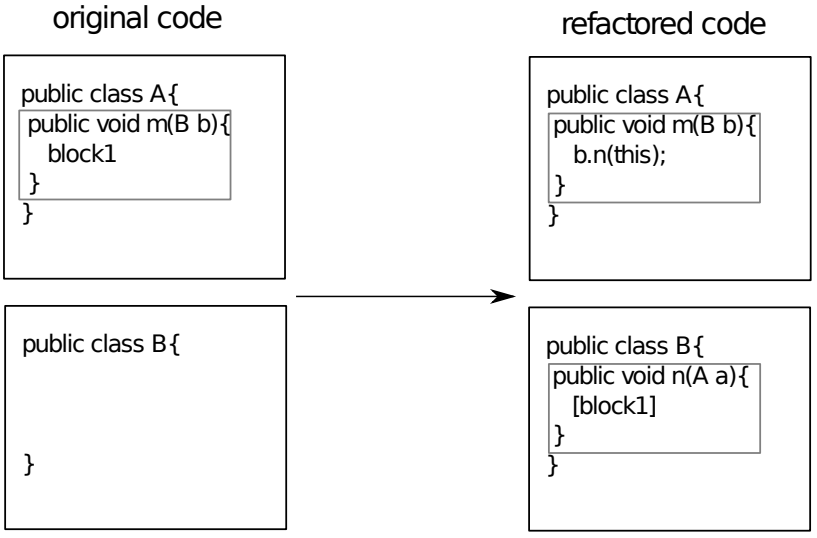
Refactoring tools. If the receiver object is not used in the body of the initial class, it will not be included as parameter in the destination class, so that you have to add it (see *AddParameter*).

A.7 MoveMethodWithDelegate

(*Move Method* in Fowler [Fow99])

Overview: MoveMethodWithDelegate (classname s,attributes [att1,att2], targetclass a,methodtobemoved m, parameterstypes [t,a], returntype r, movedmethod n, receivingobjectname o, newreceivingobjectname o'): this operation is used to move the method s::m to the class a and rename it as n.

Transform a method m of a class s into a delegator to a method n in an other class a. The code of m has been moved to n (and adapted).



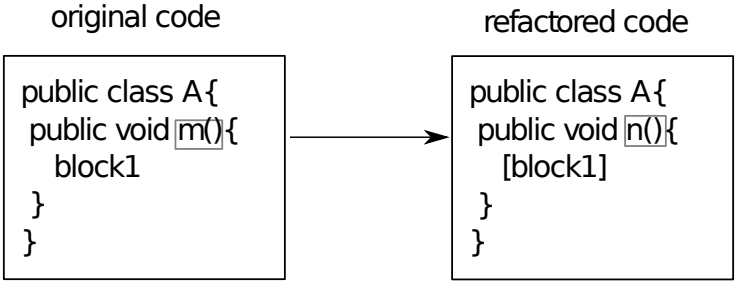
Refactoring tools. *Move* in Eclipse tool. In IntelliJ IDEA, first introduce a local delegate (with *Change Signature*), then *Move*.

Precondition.
$(\texttt{ExistsClass}(s)$
$\wedge\, \texttt{ExistsClass}(a)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t; a])$
$\wedge\, \texttt{ExistsParameterWithType}(s, m, [t; a], a)$
$\wedge\, \texttt{ExistsParameterWithName}(s, m, [t; a], o)$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(a, n, [t; s])$
$\wedge\, \texttt{HasReturnType}(s, m, r)$
$\wedge\, \neg\texttt{IsPrivate}(s, m)$
$\wedge\, \neg\texttt{IsPrivate}(s, att1)$
$\wedge\, \neg\texttt{IsPrivate}(s, att2))$

Backward Description.
$\texttt{ExistsMethodDefinitionWithParams}(s, m, [t; a]) \mapsto \top$
$\texttt{ExistsMethodDefinition}(a, n) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(a, n, [t; s]) \mapsto \top$
$\texttt{HasReturnType}(a, n, r) \mapsto \texttt{HasReturnType}(s, m, r)$
$\texttt{BoundVariableInMethodBody}(a, n, M) \mapsto \texttt{BoundVariableInMethodBody}(s, m, M)$
$\texttt{ExistsParameterWithName}(a, n, [t; s], N) \mapsto \texttt{ExistsParameterWithName}(s, m, [t; a], N)(condition)$
$\texttt{ExistsParameterWithName}(a, n, [t; s], o') \mapsto \top$
$\texttt{ExistsParameterWithName}(a, n, [t; s], o) \mapsto \bot$
$\texttt{ExistsParameterWithType}(a, n, [t; s], s) \mapsto \top$
$\texttt{ExistsParameterWithType}(a, n, [t; s], a) \mapsto \bot$
$\texttt{ExistsParameterWithType}(a, n, [t; s], T) \mapsto \texttt{ExistsParameterWithType}(s, m, [t; a], T)(condition)$
$\texttt{ExistsMethodInvocation}(s, m, a, n) \mapsto \top$
$\texttt{IsInverter}(s, m, a, r) \mapsto \top$
$\texttt{IsPrivate}(s, att1) \mapsto \bot$
$\texttt{IsPrivate}(s, att2) \mapsto \bot$
$\texttt{IsUsedAttributeInMethodBody}(s, att1, m) \mapsto \bot$
$\texttt{IsUsedAttributeInMethodBody}(s, att2, m) \mapsto \bot$

A.8 RenameMethod

(*Rename* in Fowler [Fow99] et [Koc02])
RenameMethod(class c, method m, newname n): Rename the method m of class c into n.



Refactoring tools. *Rename* in Eclipse and IntelliJ IDEA.
We have identified two types of this operation. The first one does not accept the overloading, the second one accepts overloading.

A.8.1 RenameInHierarchyNoOverloading

Overview RenameInHierarchyNoOverloading (class c,subclasses [a,b],method m ,types [t,t'], newname n) : this operation is used to rename the method (c,a,b)::m into n if n does not already exist with another signature in the hierarchy.

Precondition.
($\texttt{ExistsClass}(c)$
$\wedge\, \texttt{ExistsClass}(a)$
$\wedge\, \texttt{ExistsClass}(b)$
$\wedge\, \texttt{ExistsMethodDefinition}(c, m)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(c, m, [t; t'])$
$\wedge\, \texttt{AllSubclasses}(c, [a; b])$
$\wedge\, \neg\texttt{ExistsMethodDefinition}(c, n)$
$\wedge\, \neg\texttt{ExistsMethodDefinition}(a, n)$
$\wedge\, \neg\texttt{ExistsMethodDefinition}(b, n)$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(c, n, [t; t'])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(a, n, [t; t'])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(b, n, [t; t'])$
$\wedge\, \neg\texttt{IsOverloaded}(c, m)$
$\wedge\, \neg\texttt{IsOverloaded}(a, m)$
$\wedge\, \neg\texttt{IsOverloaded}(b, m)$
$\wedge\, \neg\texttt{IsInheritedMethod}(c, n))$

Backward Description.
$\texttt{ExistsMethodDefinition}(c, n) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(c, n, [t; t']) \mapsto \top$
$\texttt{ExistsMethodDefinition}(a, n) \mapsto \texttt{ExistsMethodDefinition}(a, m)$
$\texttt{ExistsMethodDefinition}(b, n) \mapsto \texttt{ExistsMethodDefinition}(b, m)$
$\texttt{ExistsMethodDefinition}(c, m) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(a, m) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(b, m) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(a, n, [t; t']) \mapsto \texttt{ExistsMethodDefinitionWithParams}(a, m, [t; t'])$
$\texttt{ExistsMethodDefinitionWithParams}(b, n, [t; t']) \mapsto \texttt{ExistsMethodDefinitionWithParams}(b, m, [t; t'])$
$\texttt{ExistsMethodDefinitionWithParams}(c, m, [t; t']) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(a, m, [t; t']) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [t; t']) \mapsto \bot$
$\texttt{IsInheritedMethod}(a, n) \mapsto \texttt{IsInheritedMethod}(a, m)$
$\texttt{IsInheritedMethod}(b, n) \mapsto \texttt{IsInheritedMethod}(b, m)$
$\texttt{IsDelegator}(c, n, V) \mapsto \texttt{IsDelegator}(c, m, V)$
$\texttt{IsDelegator}(a, n, V) \mapsto \texttt{IsDelegator}(a, m, V)$
$\texttt{IsDelegator}(b, n, V) \mapsto \texttt{IsDelegator}(b, m, V)$
$\texttt{IsDelegator}(c, V, n) \mapsto \texttt{IsDelegator}(c, V, m)$
$\texttt{IsDelegator}(a, V, n) \mapsto \texttt{IsDelegator}(a, V, m)$
$\texttt{IsDelegator}(b, V, n) \mapsto \texttt{IsDelegator}(b, V, m)$
$\texttt{IsOverloaded}(c, V) \mapsto \texttt{IsOverloaded}(c, V)(\mathit{condition})$
$\texttt{IsOverloaded}(a, V) \mapsto \texttt{IsOverloaded}(a, V)(\mathit{condition})$
$\texttt{IsOverloaded}(b, V) \mapsto \texttt{IsOverloaded}(b, V)(\mathit{condition})$
$\texttt{IsOverriding}(a, n) \mapsto \texttt{IsOverriding}(a, m)$
$\texttt{IsOverriding}(b, n) \mapsto \texttt{IsOverriding}(b, m)$
$\texttt{IsOverridden}(a, n) \mapsto \texttt{IsOverridden}(a, m)$
$\texttt{IsOverridden}(b, n) \mapsto \texttt{IsOverridden}(b, m)$
$\texttt{ExistsParameterWithName}(c, n, [t; t'], V1) \mapsto \texttt{ExistsParameterWithName}(c, m, [t; t'], V1)$
$\texttt{ExistsParameterWithName}(a, n, [t; t'], V1) \mapsto \texttt{ExistsParameterWithName}(a, m, [t; t'], V1)$
$\texttt{ExistsParameterWithName}(b, n, [t; t'], V1) \mapsto \texttt{ExistsParameterWithName}(b, m, [t; t'], V1)$
$\texttt{ExistsParameterWithType}(c, n, [t; t'], V1) \mapsto \texttt{ExistsParameterWithType}(c, m, [t; t'], V1)$

$\texttt{ExistsParameterWithType}(a, n, [t; t'], V1) \mapsto \texttt{ExistsParameterWithType}(a, m, [t; t'], V1)$
$\texttt{ExistsParameterWithType}(b, n, [t; t'], V1) \mapsto \texttt{ExistsParameterWithType}(b, m, [t; t'], V1)$
$\texttt{IsRecursiveMethod}(c, n) \mapsto \texttt{IsRecursiveMethod}(c, m)$
$\texttt{IsRecursiveMethod}(a, n) \mapsto \texttt{IsRecursiveMethod}(a, m)$
$\texttt{IsRecursiveMethod}(b, n) \mapsto \texttt{IsRecursiveMethod}(b, m)$
$\texttt{ExistsAbstractMethod}(c, n) \mapsto \texttt{ExistsAbstractMethod}(c, m)$
$\texttt{ExistsAbstractMethod}(a, n) \mapsto \texttt{ExistsAbstractMethod}(a, m)$
$\texttt{ExistsAbstractMethod}(b, n) \mapsto \texttt{ExistsAbstractMethod}(b, m)$
$\texttt{IsInheritedMethodWithParams}(a, n, [t; t']) \mapsto \texttt{IsVisibleMethod}(c, m, [t; t'], a)$
$\texttt{IsInheritedMethodWithParams}(b, n, [t; t']) \mapsto \texttt{IsVisibleMethod}(c, m, [t; t'], b)$
$\texttt{MethodIsUsedWithType}(c, n, [t; t'], [t; t']) \mapsto \texttt{MethodIsUsedWithType}(c, m, [t; t'], [t; t'])$
$\texttt{MethodIsUsedWithType}(a, n, [t; t'], [t; t']) \mapsto \texttt{MethodIsUsedWithType}(a, m, [t; t'], [t; t'])$
$\texttt{MethodIsUsedWithType}(b, n, [t; t'], [t; t']) \mapsto \texttt{MethodIsUsedWithType}(b, m, [t; t'], [t; t'])$
$\texttt{IsUsedMethod}(c, n, [t; t']) \mapsto \texttt{IsUsedMethod}(c, m, [t; t'])$
$\texttt{IsUsedMethod}(a, n, [t; t']) \mapsto \texttt{IsUsedMethod}(a, m, [t; t'])$
$\texttt{IsUsedMethod}(b, n, [t; t']) \mapsto \texttt{IsUsedMethod}(b, m, [t; t'])$
$\texttt{IsUsedMethodIn}(c, n, V) \mapsto \texttt{IsUsedMethodIn}(c, m, V)$
$\texttt{IsUsedMethodIn}(a, n, V) \mapsto \texttt{IsUsedMethodIn}(a, m, V)$
$\texttt{IsUsedMethodIn}(b, n, V) \mapsto \texttt{IsUsedMethodIn}(b, m, V)$
$\texttt{HasReturnType}(c, n, V1) \mapsto \texttt{HasReturnType}(c, m, V1)$
$\texttt{HasReturnType}(a, n, V1) \mapsto \texttt{HasReturnType}(a, m, V1)$
$\texttt{HasReturnType}(b, n, V1) \mapsto \texttt{HasReturnType}(b, m, V1)$
$\texttt{IsInverter}(c, n, V, V1) \mapsto \texttt{IsInverter}(c, m, V, V1)$
$\texttt{IsInverter}(a, n, V, V1) \mapsto \texttt{IsInverter}(a, m, V, V1)$
$\texttt{IsInverter}(b, n, V, V1) \mapsto \texttt{IsInverter}(b, m, V, V1)$
$\texttt{ExistsMethodInvocation}(c, V, V1, n) \mapsto \texttt{ExistsMethodInvocation}(c, V, V1, m)$
$\texttt{ExistsMethodInvocation}(a, V, V1, n) \mapsto \texttt{ExistsMethodInvocation}(a, V, V1, m)$
$\texttt{ExistsMethodInvocation}(b, V, V1, n) \mapsto \texttt{ExistsMethodInvocation}(b, V, V1, m)$
$\texttt{ExistsMethodInvocation}(c, m, V, V1) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(a, m, V, V1) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(b, m, V, V1) \mapsto \bot$
$\texttt{IsIndirectlyRecursive}(c, n) \mapsto \texttt{IsIndirectlyRecursive}(c, m)$
$\texttt{IsIndirectlyRecursive}(a, n) \mapsto \texttt{IsIndirectlyRecursive}(a, m)$
$\texttt{IsIndirectlyRecursive}(b, n) \mapsto \texttt{IsIndirectlyRecursive}(b, m)$
$\texttt{BoundVariableInMethodBody}(c, n, V) \mapsto \texttt{BoundVariableInMethodBody}(c, n, V)$
$\texttt{BoundVariableInMethodBody}(a, n, V) \mapsto \texttt{BoundVariableInMethodBody}(a, n, V)$
$\texttt{BoundVariableInMethodBody}(b, n, V) \mapsto \texttt{BoundVariableInMethodBody}(b, n, V)$
$\texttt{IsOverridden}(a, n) \mapsto \texttt{IsOverridden}(a, m)$
$\texttt{IsOverridden}(b, n) \mapsto \texttt{IsOverridden}(b, m)$
$\texttt{IsUsedConstructorAsMethodParameter}(V, c, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V, a, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V, b, m) \mapsto \bot$

A.8.2 RenameOverloadedMethodInHierarchy

Overview RenameOverloadedMethodInHierarchy (class c,subclasses [a,b],method m ,usedconstrcutorsInM [c1,c2], newname n, types [t]) : this operation is used to rename the method (c,a,b)::m into n nevertheless n will be overloaded or not.

Precondition.
$(\texttt{ExistsClass}(c)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(c, m, [t])$
$\wedge\, \neg\texttt{IsInheritedMethodWithParams}(c, n, [t])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(c, n, [t])$

$\wedge \neg\texttt{ExistsMethodDefinitionWithParams}(a, n, [t])$
$\wedge \neg\texttt{ExistsMethodDefinitionWithParams}(b, n, [t])$
$\wedge \neg\texttt{IsInheritedMethodWithParams}(c, m, [t])$
$\wedge \texttt{AllSubclasses}(c, [a; b]))$

Backward Description.
$\texttt{ExistsMethodDefinition}(c, m) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(c, n) \mapsto \top$
$\texttt{IsOverriding}(c, n) \mapsto \bot$
$\texttt{IsOverridden}(c, n) \mapsto \bot$
$\texttt{IsPublic}(c, n) \mapsto \texttt{IsPublic}(c, m)$
$\texttt{ExistsMethodDefinitionWithParams}(c, n, [t]) \mapsto \texttt{ExistsMethodDefinitionWithParams}(c, m, [t])$
$\texttt{ExistsMethodDefinitionWithParams}(a, n, [t]) \mapsto \texttt{ExistsMethodDefinitionWithParams}(a, m, [t])$
$\texttt{ExistsMethodDefinitionWithParams}(b, n, [t]) \mapsto \texttt{ExistsMethodDefinitionWithParams}(b, m, [t])$
$\texttt{ExistsMethodDefinition}(a, n) \mapsto \top$
$\texttt{ExistsMethodDefinition}(b, n) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(c, m, [t]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(a, m, [t]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [t]) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(a, m) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(b, m) \mapsto \bot$
$\texttt{isOverridingMethod}(a, n, [t]) \mapsto \texttt{isOverridingMethod}(a, m, [t])$
$\texttt{isOverridingMethod}(b, n, [t]) \mapsto \texttt{isOverridingMethod}(b, m, [t])$
$\texttt{ExistsParameterWithName}(c, n, [t], V) \mapsto \texttt{ExistsParameterWithName}(c, m, [t], V)$
$\texttt{ExistsParameterWithName}(a, n, [t], V) \mapsto \texttt{ExistsParameterWithName}(a, m, [t], V)$
$\texttt{ExistsParameterWithName}(b, n, [t], V) \mapsto \texttt{ExistsParameterWithName}(b, m, [t], V)$
$\texttt{ExistsParameterWithType}(c, n, [t], V) \mapsto \texttt{ExistsParameterWithType}(c, m, [t], V)$
$\texttt{ExistsParameterWithType}(a, n, [t], V) \mapsto \texttt{ExistsParameterWithType}(a, m, [t], V)$
$\texttt{ExistsParameterWithType}(b, n, [t], V) \mapsto \texttt{ExistsParameterWithType}(b, m, [t], V)$
$\texttt{IsDelegator}(c, n, V) \mapsto \texttt{IsDelegator}(c, m, V)$
$\texttt{IsDelegator}(a, n, V) \mapsto \texttt{IsDelegator}(a, m, V)$
$\texttt{IsDelegator}(b, n, V) \mapsto \texttt{IsDelegator}(b, m, V)$
$\texttt{IsDelegator}(c, V, n) \mapsto \texttt{IsDelegator}(c, V, m)$
$\texttt{IsDelegator}(a, V, n) \mapsto \texttt{IsDelegator}(a, V, m)$
$\texttt{IsDelegator}(b, V, n) \mapsto \texttt{IsDelegator}(b, V, m)$
$\texttt{IsRecursiveMethod}(c, n) \mapsto \texttt{IsRecursiveMethod}(c, m)$
$\texttt{IsRecursiveMethod}(a, n) \mapsto \texttt{IsRecursiveMethod}(a, m)$
$\texttt{IsRecursiveMethod}(b, n) \mapsto \texttt{IsRecursiveMethod}(b, m)$
$\texttt{ExistsAbstractMethod}(c, n) \mapsto \texttt{ExistsAbstractMethod}(c, m)$
$\texttt{ExistsAbstractMethod}(a, n) \mapsto \texttt{ExistsAbstractMethod}(a, m)$
$\texttt{ExistsAbstractMethod}(b, n) \mapsto \texttt{ExistsAbstractMethod}(b, m)$
$\texttt{IsInheritedMethodWithParams}(a, n, [t]) \mapsto \texttt{IsVisibleMethod}(c, m, [t], a)$
$\texttt{IsInheritedMethodWithParams}(b, n, [t]) \mapsto \texttt{IsVisibleMethod}(c, m, [t], b)$
$\texttt{MethodIsUsedWithType}(c, n, [t], [t]) \mapsto \texttt{MethodIsUsedWithType}(c, m, [t], [t])$
$\texttt{MethodIsUsedWithType}(a, n, [t], [t]) \mapsto \texttt{MethodIsUsedWithType}(a, m, [t], [t])$
$\texttt{MethodIsUsedWithType}(b, n, [t], [t]) \mapsto \texttt{MethodIsUsedWithType}(b, m, [t], [t])$
$\texttt{IsUsedMethod}(c, n, [t]) \mapsto \texttt{IsUsedMethod}(c, m, [t])$
$\texttt{IsUsedMethod}(a, n, [t]) \mapsto \texttt{IsUsedMethod}(a, m, [t])$
$\texttt{IsUsedMethod}(b, n, [t]) \mapsto \texttt{IsUsedMethod}(b, m, [t])$
$\texttt{IsUsedMethodIn}(c, n, V) \mapsto \texttt{IsUsedMethodIn}(c, m, V)$
$\texttt{IsUsedMethodIn}(a, n, V) \mapsto \texttt{IsUsedMethodIn}(a, m, V)$
$\texttt{IsUsedMethodIn}(b, n, V) \mapsto \texttt{IsUsedMethodIn}(b, m, V)$

$\texttt{HasReturnType}(c, n, V) \mapsto \texttt{HasReturnType}(c, m, V)$
$\texttt{HasReturnType}(a, n, V) \mapsto \texttt{HasReturnType}(a, m, V)$
$\texttt{HasReturnType}(b, n, V) \mapsto \texttt{HasReturnType}(b, m, V)$
$\texttt{IsInverter}(c, n, V, V1) \mapsto \texttt{IsInverter}(c, m, V, V1)$
$\texttt{IsInverter}(a, n, V, V1) \mapsto \texttt{IsInverter}(a, m, V, V1)$
$\texttt{IsInverter}(b, n, V, V1) \mapsto \texttt{IsInverter}(b, m, V, V1)$
$\texttt{ExistsMethodInvocation}(c, V, V1, n) \mapsto \texttt{ExistsMethodInvocation}(c, V, V1, m)$
$\texttt{ExistsMethodInvocation}(a, V, V1, n) \mapsto \texttt{ExistsMethodInvocation}(a, V, V1, m)$
$\texttt{ExistsMethodInvocation}(b, V, V1, n) \mapsto \texttt{ExistsMethodInvocation}(b, V, V1, m)$
$\texttt{IsIndirectlyRecursive}(c, n) \mapsto \texttt{IsIndirectlyRecursive}(c, m)$
$\texttt{IsIndirectlyRecursive}(a, n) \mapsto \texttt{IsIndirectlyRecursive}(a, m)$
$\texttt{IsIndirectlyRecursive}(b, n) \mapsto \texttt{IsIndirectlyRecursive}(b, m)$
$\texttt{BoundVariableInMethodBody}(c, n, V) \mapsto \texttt{BoundVariableInMethodBody}(c, m, V)$
$\texttt{BoundVariableInMethodBody}(a, n, V) \mapsto \texttt{BoundVariableInMethodBody}(a, m, V)$
$\texttt{BoundVariableInMethodBody}(b, n, V) \mapsto \texttt{BoundVariableInMethodBody}(b, m, V)$
$\texttt{IsOverridden}(a, n) \mapsto \texttt{IsOverridden}(a, m)$
$\texttt{IsOverridden}(b, n) \mapsto \texttt{IsOverridden}(b, m)$
$\texttt{IsUsedConstructorAsMethodParameter}(V, c, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V, a, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V, b, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(V, c, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(V, a, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(V, b, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V, c, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V, a, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V, b, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V, c, n) \mapsto \texttt{IsUsedConstructorAsMethodParameter}(V, c, m)$
$\texttt{IsUsedConstructorAsMethodParameter}(V, a, n) \mapsto \texttt{IsUsedConstructorAsMethodParameter}(V, a, m)$
$\texttt{IsUsedConstructorAsMethodParameter}(V, b, n) \mapsto \texttt{IsUsedConstructorAsMethodParameter}(V, b, m)$
$\texttt{IsUsedConstructorAsObjectReceiver}(c1, c, n) \mapsto \top$
$\texttt{IsUsedConstructorAsObjectReceiver}(c2, c, n) \mapsto \top$
$\texttt{IsUsedConstructorAsObjectReceiver}(c1, a, n) \mapsto \top$
$\texttt{IsUsedConstructorAsObjectReceiver}(c2, a, n) \mapsto \top$
$\texttt{IsUsedConstructorAsObjectReceiver}(c1, b, n) \mapsto \top$
$\texttt{IsUsedConstructorAsObjectReceiver}(c2, b, n) \mapsto \top$
$\texttt{IsUsedConstructorAsMethodParameter}(V, c, n) \mapsto \texttt{IsUsedConstructorAsMethodParameter}(V, c, m)$
$\texttt{IsUsedConstructorAsMethodParameter}(V, a, n) \mapsto \texttt{IsUsedConstructorAsMethodParameter}(V, a, m)$
$\texttt{IsUsedConstructorAsMethodParameter}(V, b, n) \mapsto \texttt{IsUsedConstructorAsMethodParameter}(V, b, m)$

A.8.3 RenameDelegatorWithOverloading

Overview RenameDelegatorWithOverloading (classname s, subclasses [a,b], method m,paramtype t,paramName pn,super typeOfparamtype t', newname n) : this operation is used to rename the method (c,a,b)::m into n and accepts overloaded methods. This operation is an ad-hoc use of the operation RenameOverloadedMethodInHierarchy (we need in this use more details about the signature of the method to be renamed).

Precondition.
$(\texttt{ExistsClass}(s)$
$\wedge\, \texttt{ExistsClass}(a)$
$\wedge\, \texttt{ExistsClass}(b)$
$\wedge\, \texttt{ExistsMethodDefinition}(s, m)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t])$
$\wedge\, \texttt{AllSubclasses}(s, [a; b])$

$\wedge \neg \texttt{ExistsMethodDefinitionWithParams}(s, n, [t])$
$\wedge \neg \texttt{ExistsMethodDefinitionWithParams}(a, n, [t])$
$\wedge \neg \texttt{ExistsMethodDefinitionWithParams}(b, n, [t])$
$\wedge \neg \texttt{IsInheritedMethod}(s, n)$
$\wedge \neg \texttt{ExistsMethodDefinitionWithParams}(s, n, [t]))$

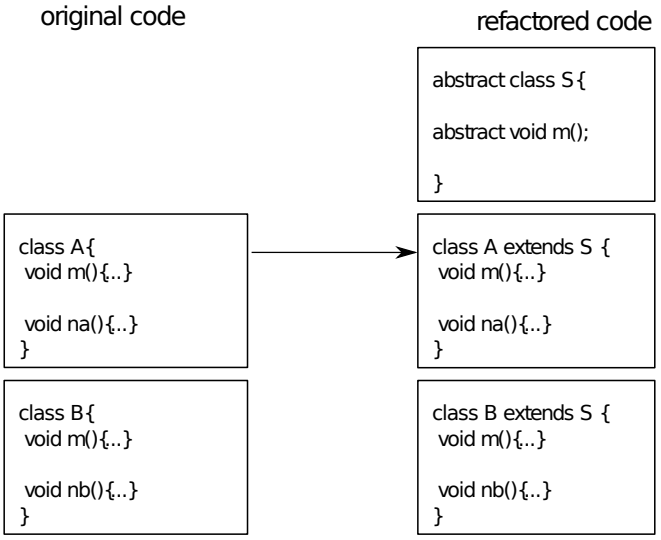
Backward Description.
$\texttt{ExistsMethodDefinition}(s, n) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(s, n, [t]) \mapsto \top$
$\texttt{IsPublic}(s, n) \mapsto \texttt{IsPublic}(s, m)$
$\texttt{ExistsMethodDefinition}(s, m) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(a, m) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(b, m) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(s, m, [t]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(a, m, [t]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [t]) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(a, n) \mapsto \texttt{ExistsMethodDefinition}(a, m)$
$\texttt{ExistsMethodDefinition}(b, n) \mapsto \texttt{ExistsMethodDefinition}(b, m)$
$\texttt{ExistsMethodDefinitionWithParams}(a, n, [t]) \mapsto \texttt{ExistsMethodDefinitionWithParams}(a, m, [t])$
$\texttt{ExistsMethodDefinitionWithParams}(b, n, [t]) \mapsto \texttt{ExistsMethodDefinitionWithParams}(b, m, [t])$
$\texttt{IsInheritedMethod}(a, n) \mapsto \texttt{IsInheritedMethod}(a, m)$
$\texttt{IsInheritedMethod}(b, n) \mapsto \texttt{IsInheritedMethod}(b, m)$
$\texttt{MethodIsUsedWithType}(s, n, [t], [t]) \mapsto \texttt{MethodIsUsedWithType}(s, m, [t], [t])$
$\texttt{MethodIsUsedWithType}(a, n, [t], [t]) \mapsto \texttt{MethodIsUsedWithType}(a, m, [t], [t])$
$\texttt{MethodIsUsedWithType}(b, n, [t], [t]) \mapsto \texttt{MethodIsUsedWithType}(b, m, [t], [t])$
$\texttt{MethodIsUsedWithType}(s, m, [t], [t]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(a, m, [t], [t]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(b, m, [t], [t]) \mapsto \bot$
$\texttt{ExistsParameterWithName}(s, n, [t], V) \mapsto \texttt{ExistsParameterWithName}(s, m, [t], V)$
$\texttt{ExistsParameterWithName}(a, n, [t], V) \mapsto \texttt{ExistsParameterWithName}(a, m, [t], V)$
$\texttt{ExistsParameterWithName}(b, n, [t], V) \mapsto \texttt{ExistsParameterWithName}(b, m, [t], V)$
$\texttt{ExistsParameterWithType}(s, n, [t], V) \mapsto \texttt{ExistsParameterWithType}(s, m, [t], V)$
$\texttt{ExistsParameterWithType}(a, n, [t], V) \mapsto \texttt{ExistsParameterWithType}(a, m, [t], V)$
$\texttt{ExistsParameterWithType}(b, n, [t], V) \mapsto \texttt{ExistsParameterWithType}(b, m, [t], V)$
$\texttt{ExistsMethodInvocation}(s, V1, V, n) \mapsto \texttt{ExistsMethodInvocation}(s, V1, V, m)$
$\texttt{ExistsMethodInvocation}(a, V1, V, n) \mapsto \texttt{ExistsMethodInvocation}(a, V1, V, m)$
$\texttt{ExistsMethodInvocation}(b, V1, V, n) \mapsto \texttt{ExistsMethodInvocation}(b, V1, V, m)$
$\texttt{IsDelegator}(s, n, V) \mapsto \texttt{IsDelegator}(s, m, V)$
$\texttt{IsDelegator}(a, n, V) \mapsto \texttt{IsDelegator}(a, m, V)$
$\texttt{IsDelegator}(b, n, V) \mapsto \texttt{IsDelegator}(b, m, V)$
$\texttt{IsDelegator}(s, V, n) \mapsto \texttt{IsDelegator}(s, V, m)$
$\texttt{IsDelegator}(a, V, n) \mapsto \texttt{IsDelegator}(a, V, m)$
$\texttt{IsDelegator}(b, V, n) \mapsto \texttt{IsDelegator}(b, V, m)$
$\texttt{IsUsedMethod}(s, n, [t]) \mapsto \texttt{IsUsedMethod}(s, m, [t])$
$\texttt{IsUsedMethod}(a, n, [t]) \mapsto \texttt{IsUsedMethod}(a, m, [t])$
$\texttt{IsUsedMethod}(b, n, [t]) \mapsto \texttt{IsUsedMethod}(b, m, [t])$
$\texttt{IsUsedMethodIn}(s, n, V) \mapsto \texttt{IsUsedMethodIn}(s, m, V)$
$\texttt{IsUsedMethodIn}(a, n, V) \mapsto \texttt{IsUsedMethodIn}(a, m, V)$
$\texttt{IsUsedMethodIn}(b, n, V) \mapsto \texttt{IsUsedMethodIn}(b, m, V)$
$\texttt{HasReturnType}(s, n, V) \mapsto \texttt{HasReturnType}(s, m, V)$
$\texttt{HasReturnType}(a, n, V) \mapsto \texttt{HasReturnType}(a, m, V)$
$\texttt{HasReturnType}(b, n, V) \mapsto \texttt{HasReturnType}(b, m, V)$

$\texttt{IsInverter}(s, n, V, V1) \mapsto \texttt{IsInverter}(s, m, V, V1)$
$\texttt{IsInverter}(a, n, V, V1) \mapsto \texttt{IsInverter}(a, m, V, V1)$
$\texttt{IsInverter}(b, n, V, V1) \mapsto \texttt{IsInverter}(b, m, V, V1)$
$\texttt{IsIndirectlyRecursive}(s, n) \mapsto \texttt{IsIndirectlyRecursive}(s, m)$
$\texttt{IsIndirectlyRecursive}(a, n) \mapsto \texttt{IsIndirectlyRecursive}(a, m)$
$\texttt{IsIndirectlyRecursive}(b, n) \mapsto \texttt{IsIndirectlyRecursive}(b, m)$
$\texttt{BoundVariableInMethodBody}(s, n, V) \mapsto \texttt{BoundVariableInMethodBody}(s, n, V)$
$\texttt{BoundVariableInMethodBody}(a, n, V) \mapsto \texttt{BoundVariableInMethodBody}(a, n, V)$
$\texttt{BoundVariableInMethodBody}(b, n, V) \mapsto \texttt{BoundVariableInMethodBody}(b, n, V)$
$\texttt{IsOverridden}(a, n) \mapsto \texttt{IsOverridden}(a, m)$
$\texttt{IsOverridden}(b, n) \mapsto \texttt{IsOverridden}(b, m)$
$\texttt{IsUsedConstructorAsMethodParameter}(V, s, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V, a, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V, b, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(t, s, n) \mapsto \texttt{IsUsedConstructorAsObjectReceiver}(t, s, m)$
$\texttt{IsUsedConstructorAsObjectReceiver}(t, a, n) \mapsto \texttt{IsUsedConstructorAsObjectReceiver}(t, a, m)$
$\texttt{IsUsedConstructorAsObjectReceiver}(t, b, n) \mapsto \texttt{IsUsedConstructorAsObjectReceiver}(t, b, m)$

A.9 ExtractSuperClass

(*Extract Super Class* in Fowler [Fow99] and [Koc02])

Overview: ExtractSuperClass (subclasses[a,b], superclass s,methodsOfsubclasses [m,n],returntype t): this operation is used to extract a super-class s from the classes a and b and make an abstract declaration of methods a::m, a::n, b::m and b::n in this new super-class.



Refactoring tools. *Extract Superclass* in Eclipse tool and IntelliJ IDEA. In IntelliJ IDEA, the *Extract Superclass* operation cannot be applied to several classes simultaneously, so we have maintain the code of this operation in order to run it on several classes.

Precondition.
$(\neg\texttt{ExistsType}(s)$
$\wedge\, \texttt{ExistsClass}(a)$
$\wedge\, \texttt{ExistsClass}(b)$
$\wedge\, \texttt{ExtendsDirectly}(a, java.lang.Object)$
$\wedge\, \texttt{ExtendsDirectly}(b, java.lang.Object)$
$\wedge\, \texttt{HasReturnType}(a, m, t)$
$\wedge\, \texttt{HasReturnType}(a, n, t)$
$\wedge\, \texttt{HasReturnType}(b, m, t)$

$\wedge$ HasReturnType(b, n, t))

Backward Description.
IsAbstractClass$(s) \mapsto \top$
ExistsClass$(s) \mapsto \top$
ExistsType$(s) \mapsto \top$
ExistsMethodDefinition$(s, X) \mapsto$ (ExistsMethodDefinition(a, X)
$\wedge$ ExistsMethodDefinition(b, X))
ExistsMethodDefinitionWithParams$(s, X, [\,]) \mapsto$ (ExistsMethodDefinitionWithParams$(a, X, [\,])$
$\wedge$ ExistsMethodDefinitionWithParams$(b, X, [\,])$)
ExistsMethodDefinitionWithParams$(s, X, [Y]) \mapsto$ (ExistsMethodDefinitionWithParams$(a, X, [Y])$
$\wedge$ ExistsMethodDefinitionWithParams$(b, X, [Y])$)
IsUsedMethodIn$(s, X, Y) \mapsto \bot$
ExistsMethodDefinitionWithParams$(X, Y, [s]) \mapsto \bot$
IsUsedMethod$(s, X, [Y]) \mapsto \bot$
AllSubclasses$(s, [a; b]) \mapsto \top$
MethodIsUsedWithType$(X, Y, [Z], [s]) \mapsto \bot$
IsInheritedMethodWithParams$(X, Y, [s]) \mapsto \bot$
IsUsedConstructorAsMethodParameter$(s, X, Y) \mapsto \bot$
IsUsedConstructorAsInitializer$(s, X, Y) \mapsto \bot$
IsUsedConstructorAsObjectReceiver$(s, X, Y) \mapsto \bot$
IsUsedConstructorAsMethodParameter$(X, s, Y) \mapsto \bot$
IsUsedConstructorAsInitializer$(X, s, Y) \mapsto \bot$
IsUsedConstructorAsObjectReceiver$(X, s, Y) \mapsto \bot$
IsPrimitiveType$(s) \mapsto \bot$
IsSubType$(a, s) \mapsto \top$
IsSubType$(b, s) \mapsto \top$
IsSubType$(X, s) \mapsto$ IsSubType(X, a)
IsSubType$(X, s) \mapsto$ IsSubType(X, b)
IsPublic$(s, m) \mapsto \top$
IsPublic$(s, n) \mapsto \top$
ExistsAbstractMethod$(s, m) \mapsto \top$
ExistsAbstractMethod$(s, n) \mapsto \top$
IsOverriding$(s, m) \mapsto \bot$
IsOverriding$(s, n) \mapsto \bot$
IsOverridden$(s, m) \mapsto \top$
IsOverridden$(s, n) \mapsto \top$
IsPrivate$(s, m) \mapsto \bot$
IsPrivate$(s, n) \mapsto \bot$

A.9.1 ExtractSuperClassWithoutPullUp

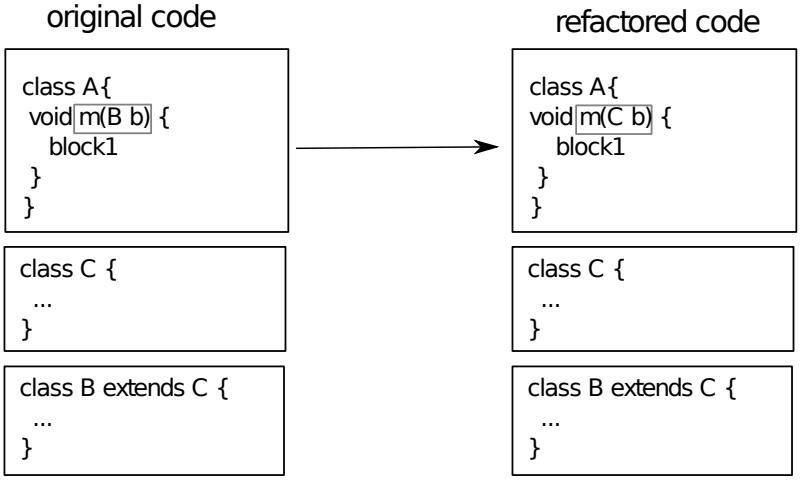
Overview: ExtractSuperClassWithouPullUp (subclasses[a,b], superclass s): this operation is a specific variant of the operation extract super-class. It is simply used to extract a super-class without pull up the methods of sub-classes.

Precondition.
($\neg$ExistsType(s)
$\wedge$ ExistsClass(a)
$\wedge$ ExistsClass(b)
$\wedge$ ExtendsDirectly$(a, java.lang.Object)$
$\wedge$ ExtendsDirectly$(b, java.lang.Object)$)

Backward Description.
$\texttt{IsAbstractClass}(s) \mapsto \top$
$\texttt{ExistsClass}(s) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(X, Y, [s]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(s, X, [Y]) \mapsto \bot$
$\texttt{ExistsType}(s) \mapsto \top$
$\texttt{AllSubclasses}(s, [a; b]) \mapsto \top$
$\texttt{IsUsedConstructorAsMethodParameter}(s, X, Y) \mapsto \bot$
$\texttt{IsUsedConstructorAsInitializer}(s, X, Y) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(s, X, Y) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(X, s, Y) \mapsto \bot$
$\texttt{IsUsedConstructorAsInitializer}(X, s, Y) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(X, s, Y) \mapsto \bot$
$\texttt{IsPrimitiveType}(s) \mapsto \bot$
$\texttt{IsUsedMethod}(s, X, [Y]) \mapsto \bot$
$\texttt{IsInheritedMethodWithParams}(X, Y, [s]) \mapsto \bot$
$\texttt{IsPrivate}(s, X) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(X, Y, [Z], [s]) \mapsto \bot$
$\texttt{IsSubType}(a, s) \mapsto \top$
$\texttt{IsSubType}(b, s) \mapsto \top$
$\texttt{IsSubType}(X, s) \mapsto \texttt{IsSubType}(X, a)$
$\texttt{IsSubType}(X, s) \mapsto \texttt{IsSubType}(X, b)$

A.10 GeneraliseParameter

Overview: GeneraliseParameter (classname s, subclasses [a,b], methodname m,paramName p,type t ,supertype st): this operation is used to change the type t of the parameter p of the methods s::m, a::m and b::m into a super-type st. All the uses of the parameter which type is changed must be legal with the new type st (method invocations, object passed as parameter of other methods). The uses of the parameter which type is changed as parameter of methods must not result in a charge of the invoked code (static resolving of overloading).



Refactoring tools. *Change Method Signature* in Eclipse tool and *Type Migration* in IntelliJ IDEA (or *Change Signature*).

Precondition.
$(\texttt{ExistsClass}(s)$
$\wedge\, \texttt{ExistsClass}(a)$
$\wedge\, \texttt{ExistsClass}(b)$
$\wedge\, \texttt{ExistsMethodDefinition}(s, m)$
$\wedge\, \texttt{ExistsMethodDefinition}(a, m)$
$\wedge\, \texttt{ExistsMethodDefinition}(b, m)$

$\wedge$ IsSubType(st, t)
$\wedge$ AllSubclasses$(s, [a; b])$
$\wedge$ AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC(s, m, p, t)
$\wedge$ AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC(a, m, p, t)
$\wedge$ AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC(b, m, p, t)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded(s, m, p)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded(a, m, p)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded$(b, m, p))$

Backward Description.
IsInverter$(s, m, t, V) \mapsto$ IsInverter(s, m, st, V)
IsInverter$(a, m, t, V) \mapsto$ IsInverter(a, m, st, V)
IsInverter$(b, m, t, V) \mapsto$ IsInverter(b, m, st, V)
ExistsMethodDefinitionWithParams$(s, m, [t]) \mapsto$ ExistsMethodDefinitionWithParams$(s, m, [st])$
ExistsMethodDefinitionWithParams$(a, m, [t]) \mapsto$ ExistsMethodDefinitionWithParams$(a, m, [st])$
ExistsMethodDefinitionWithParams$(b, m, [t]) \mapsto$ ExistsMethodDefinitionWithParams$(b, m, [st])$
ExistsMethodDefinitionWithParams$(s, m, [st]) \mapsto \bot$
ExistsMethodDefinitionWithParams$(a, m, [st]) \mapsto \bot$
ExistsMethodDefinitionWithParams$(b, m, [st]) \mapsto \bot$
IsInheritedMethodWithParams$(a, m, [t]) \mapsto \top$
IsInheritedMethodWithParams$(b, m, [t]) \mapsto \top$
IsUsedConstructorAsMethodParameter$(t, s, m) \mapsto \top$
IsUsedConstructorAsMethodParameter$(t, a, m) \mapsto \top$
IsUsedConstructorAsMethodParameter$(t, b, m) \mapsto \top$
IsUsedConstructorAsMethodParameter$(st, s, m) \mapsto \bot$
IsUsedConstructorAsMethodParameter$(st, a, m) \mapsto \bot$
IsUsedConstructorAsMethodParameter$(st, b, m) \mapsto \bot$
IsOverridden$(a, m) \mapsto$ ExistsMethodDefinition(a, m)
IsOverridden$(b, m) \mapsto$ ExistsMethodDefinition(b, m)
IsOverriding$(a, m) \mapsto$ ExistsMethodDefinition(a, m)
IsOverriding$(b, m) \mapsto$ ExistsMethodDefinition(b, m)
ExistsParameterWithName$(s, m, [t], p) \mapsto \top$
ExistsParameterWithName$(a, m, [t], p) \mapsto \top$
ExistsParameterWithName$(b, m, [t], p) \mapsto \top$
ExistsParameterWithType$(s, m, [t], t) \mapsto \top$
ExistsParameterWithType$(a, m, [t], t) \mapsto \top$
ExistsParameterWithType$(b, m, [t], t) \mapsto \top$

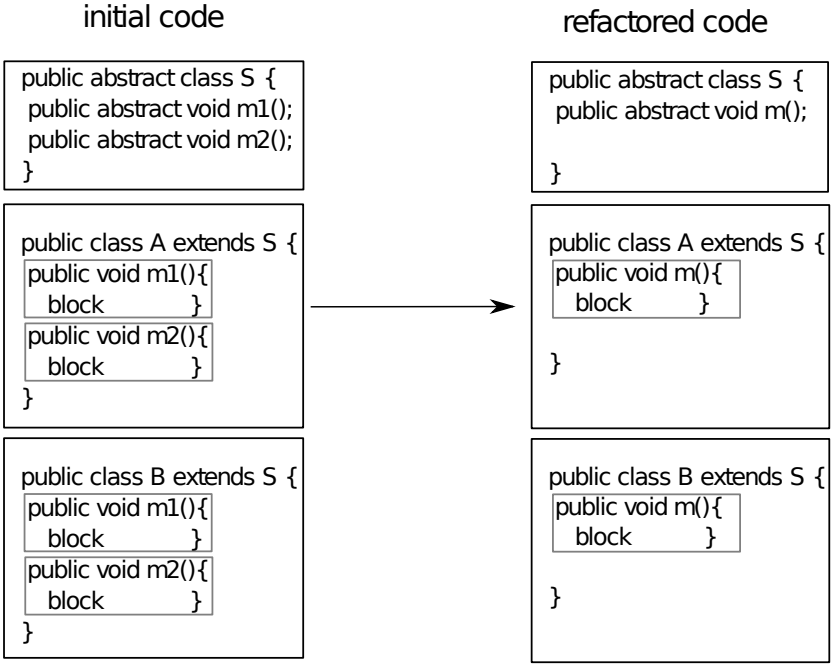
A.11 MergeDuplicateMethods

Overview MergeDuplicateMethods (classname c, subclasses [a,b],mergedmethods [m,n],newmethod m2, invertedtype t, returntype q): this operation is used to merge methods m and n existing in the hierarchy to a single method m2. The formal description of this operation is built on the formal description of five refactoring operations since it is composed of these operations.

Algorithm of the operation The operation MergeDuplicateMethods is based on four steps :

MergeDuplicateMethods (c,[a,b],[m,n],m2,t,q) =

1. ReplaceMethodcodeDuplicatesInverter (c, m, [n], t,q))
2. PullupConcreteDelegator(c, [a,b], n ,m))
3. InlineAndDelete(c,n))
4. RenameInHierarchyNoOverloading (c, [a,b], m,[t], m2)



Refactoring tools. *Rename, Replace Method duplication, Extract Method, In-line* in Eclipse, *Rename, Replace Method Code Duplicates, Pull Up, Inline* in IntelliJ IDEA:

Notes.

- ReplaceCodeDuplicates introduces a delegation. After that, the code for m2 is the same in all the subclasses of c (a delegation). Then the pull-up can be done without changing the semantics, which allows to inline (and remove) m2 afterwards.
- After the first pull-up, the IntelliJ IDEA pull-up warns that some code already exists. The first time, when some code replaces the abstract declaration, the refactorer manages to remove the abstract declaration. The next times, when a second code comes in addition of the first one, the refactorer prefers to leave the two versions (which are identical in this case), so that we have to use *safe delete* to remove one of them.

  We could provide an extension of the *pull-up* operation with the customized behavior.

Preconditions:

- The two concerned methods bodies must be syntactically equals.
- The new name must not introduce an overloading.
- The two methods must not be overloaded.

A.12 ReplaceMethodcodeDuplicatesInverter

Overview ReplaceMethodcodeDuplicatesInverter (classname c, method m, copies [n,m1],invertedtype t,returntype r) : this operation is used to replace c::[n,m1] by c::m.

Precondition.
($\texttt{ExistsClass}(c)$
$\wedge$ ($\texttt{ExistsMethodDefinition}(c, m)$
$\wedge$ $\texttt{ExistsMethodDefinition}(c, n)$
$\wedge$ $\texttt{ExistsMethodDefinition}(c, m1)$)
$\wedge$ ($\texttt{IsInverter}(c, m, t, r)$
$\wedge$ $\texttt{IsInverter}(c, n, t, r)$
$\wedge$ $\texttt{IsInverter}(c, m1, t, r)$)))

Backward Description.
$\texttt{IsDelegator}(c, n, m) \mapsto \top$
$\texttt{IsDelegator}(c, m1, m) \mapsto \top$
$\texttt{ExistsMethodInvocation}(c, n, c, m) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(c, m1, c, m) \mapsto \bot$
$\texttt{IsRecursiveMethod}(c, n) \mapsto \bot$
$\texttt{IsRecursiveMethod}(c, m1) \mapsto \bot$

A.13 SafeDeleteDelegatorOverriding

Overview SafeDeleteDelegatorOverriding (classname c, method m, superclass s, deleguee n) : this operations is used to remove useless overridings.

Precondition.
($\texttt{ExistsClass}(c)$
$\wedge$ $\texttt{ExistsClass}(s)$
$\wedge$ $\texttt{ExistsMethodDefinition}(c, m)$
$\wedge$ $\texttt{ExistsMethodDefinition}(s, m)$
$\wedge$ $\texttt{IsDelegator}(c, m, n)$
$\wedge$ $\texttt{IsDelegator}(s, m, n)$
$\wedge$ $\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(c, m, this)$)

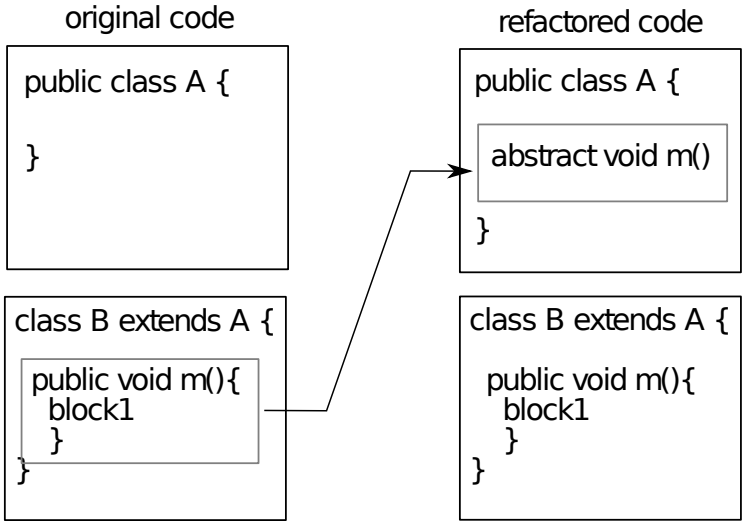
Backward Description.
$\texttt{ExistsMethodDefinition}(c, m) \mapsto \bot$
$\texttt{IsInheritedMethod}(c, m) \mapsto \top$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(c, m, X, Y) \mapsto \bot$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(c, m, X) \mapsto \bot$
$\texttt{BoundVariableInMethodBody}(c, m, X) \mapsto \bot$
$\texttt{ExistsParameterWithName}(c, m, [X], Y) \mapsto \bot$
$\texttt{ExistsParameterWithType}(c, m, [X], Y) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(c, m, X, Y) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(c, m, [X]) \mapsto \bot$
$\texttt{IsInheritedMethodWithParams}(X, m, [Y]) \mapsto \bot$
$\texttt{IsIndirectlyRecursive}(c, m) \mapsto \bot$
$\texttt{IsVisibleMethod}(c, m, [X], Y) \mapsto \bot$
$\texttt{IsInverter}(c, m, X, Y) \mapsto \bot$
$\texttt{IsDelegator}(c, m, X) \mapsto \bot$
$\texttt{IsUsedMethod}(c, m, [X]) \mapsto \bot$
$\texttt{IsUsedMethodIn}(c, m, X) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(X, c, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsInitializer}(X, c, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(X, c, m) \mapsto \bot$
$\texttt{IsPublic}(c, m) \mapsto \bot$
$\texttt{IsProtected}(c, m) \mapsto \bot$

$\texttt{IsPrivate}(c, m) \mapsto \bot$
$\texttt{IsUsedAttributeInMethodBody}(c, X, m) \mapsto \bot$
$\texttt{IsOverridden}(c, m) \mapsto \bot$
$\texttt{IsOverloaded}(c, m) \mapsto \bot$
$\texttt{IsOverriding}(c, m) \mapsto \bot$
$\texttt{IsRecursiveMethod}(c, m) \mapsto \bot$
$\texttt{HasReturnType}(c, m, X) \mapsto \bot$
$\texttt{MethodHasParameterType}(c, m, X) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(c, m, [X], [X]) \mapsto \bot$

A.14 PullUpAbstract

PullUpAbstract(set of classes C, method m, interface s)

Pull up a method implemented in a set of classes C to their superclass s: do not move the definitions, just declare the method abstract in s.



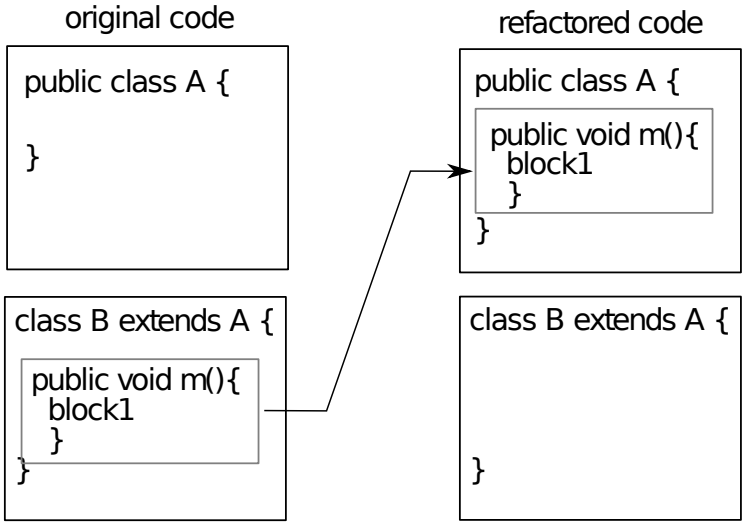
Refactoring tools. *Pull Up* in Eclipse tool and IntelliJ IDEA.

Preconditions:

- s is a superclass of each class in C.
- m is defined in all the classes of C

A.15 PullUpImplementation

Overview: PullupImplementation(a,[att1,att2],m,s): this operation is used to pull up the definition of the method a::m to s and delete it from a.



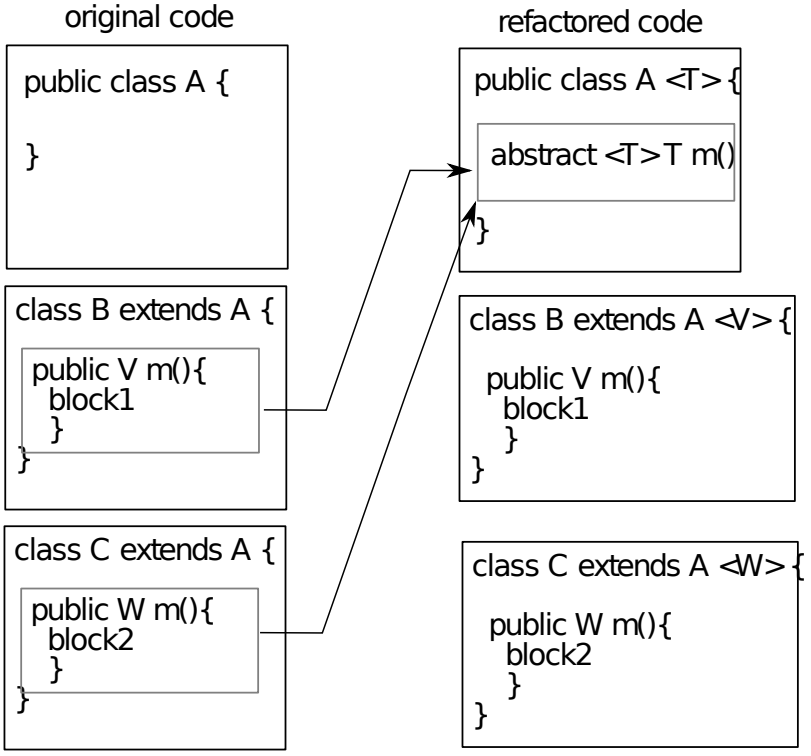
Refactoring tools. *Pull Up* in Eclipse tool and IntelliJ IDEA.

Precondition. ($\texttt{ExistsClass}(c)$
$\wedge\, \texttt{ExistsClass}(s)$
$\wedge\, \texttt{IsAbstractClass}(s)$
$\wedge\, \texttt{ExistsMethodDefinition}(c, m)$
$\wedge\, \texttt{ExistsAbstractMethod}(s, m)$
$\wedge\, \texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(c, m, this, s)$
$\wedge\, \texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(c, m, this)$
$\wedge\, \neg\texttt{IsPrivate}(c, m)$
$\wedge\, \neg\texttt{IsUsedAttributeInMethodBody}(c, att1, m)$
$\wedge\, \neg\texttt{IsUsedAttributeInMethodBody}(c, att2, m))$

Backward Description.
$\texttt{ExistsMethodDefinition}(c, m) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(s, m) \mapsto \top$
$\texttt{ExistsAbstractMethod}(s, m) \mapsto \bot$
$\texttt{IsDelegator}(s, m, X) \mapsto \texttt{IsDelegator}(c, m, X)(condition)$
$\texttt{ExistsMethodDefinitionWithParams}(c, m, [X]) \mapsto \bot$
$\texttt{IsInheritedMethodWithParams}(c, m, [X]) \mapsto \top$
$\texttt{IsInheritedMethod}(c, m) \mapsto \top$
$\texttt{IsVisibleMethod}(s, m, [X], c) \mapsto \top$
$\texttt{IsPrivate}(c, m) \mapsto \bot$
$\texttt{IsOverridden}(c, m) \mapsto \bot$
$\texttt{IsOverriding}(c, m) \mapsto \bot$
$\texttt{IsVisible}(s, m, c) \mapsto \top$
$\texttt{IsOverloaded}(s, m) \mapsto \texttt{ExistsMethodDefinition}(s, m)$
$\texttt{IsUsedAttributeInMethodBody}(c, X, m) \mapsto \bot$
$\texttt{IsOverridden}(c, m) \mapsto \bot$
$\texttt{IsOverloaded}(c, m) \mapsto \bot$
$\texttt{IsRecursiveMethod}(c, m) \mapsto \bot$
$\texttt{HasReturnType}(c, m, X) \mapsto \bot$
$\texttt{MethodHasParameterType}(c, m, X) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(c, m, [X], [X]) \mapsto \bot$
$\texttt{IsPrivate}(c, att1) \mapsto \bot$
$\texttt{IsPrivate}(c, att2) \mapsto \bot$

A.16 PullUpWithGenerics

Overview: PullupWithGenerics (classname s, subclassname a, [att1,att2],methodname m,returntype r ,parameterType T): this operation is used to pull up the method a::m to s and then creates the parameter type T to the class s (as shown in the following figure). After performing this operation a polymorphism is created in the hierarchy (Java *Generic* types).



Refactoring tools. We provide this operation as a plugin for IntelliJ IDEA (*Pull up method refactoring extension*: `http://plugins.intellij.net/plugin/?idea_ce&id=6889`).

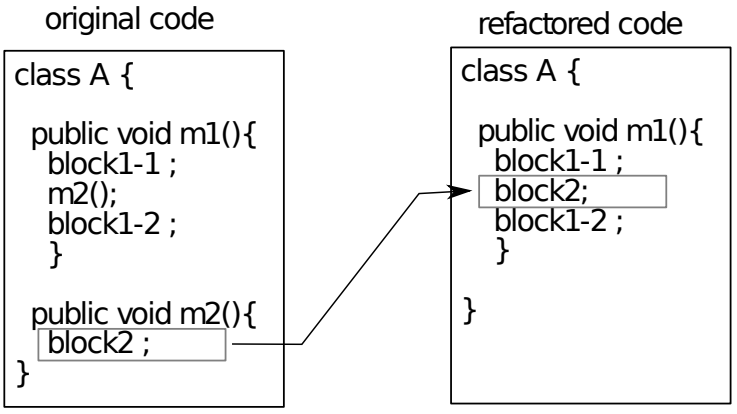
Precondition.
($\texttt{ExistsClass}(s)$
$\wedge\, \texttt{IsAbstractClass}(s)$
$\wedge\, \texttt{ExistsClass}(a)$
$\wedge\, \texttt{IsSubType}(a, s)$
$\wedge\, \texttt{HasReturnType}(a, m, r)$
$\wedge\, \neg\texttt{ExistsAbstractMethod}(s, m)$
$\wedge\, \neg\texttt{IsPrimitiveType}(r)$
$\wedge\, \neg\texttt{IsPrivate}(a, m)$
$\wedge\, \neg\texttt{HasParameterType}(a, r)$
$\wedge\, \neg\texttt{IsPrivate}(a, att1)$
$\wedge\, \neg\texttt{IsPrivate}(a, att2))$

Backward Description. $\texttt{HasReturnType}(s, m, T) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(s, m, [X]) \mapsto \top$
$\texttt{MethodHasParameterType}(s, m, T) \mapsto \top$
$\texttt{HasParameterType}(s, T) \mapsto \top$
$\texttt{extendsFromPrametricClass}(a, s, r) \mapsto \top$
$\texttt{IsGenericsSubtype}(a, [r], s, [T]) \mapsto \top$
$\texttt{IsPrivate}(a, m) \mapsto \bot$

A.17 InlineAndDelete

(*Inline Method* in [Fow99])

Overview: InlineAndDelete (classname s,methodname m,types [t,t'],invocatormethod n, othermethods [m1,m2],otherclasses [a,b,c]): this operation is used to replace one or all invocations of a given method by its body and delete it.



Refactoring tools. *In-line* in Eclipse tool and IntelliJ IDEA.

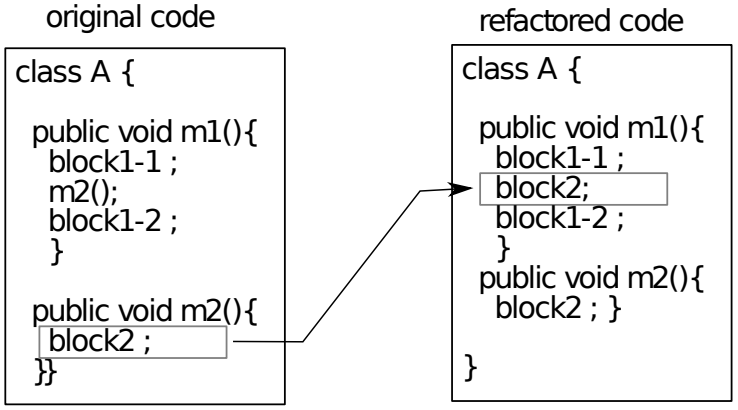
Precondition.
$(\texttt{ExistsClass}(s)$
$\wedge\, \texttt{ExistsMethodDefinition}(s, m)$
$\wedge\, \neg\texttt{IsOverriding}(s, m)$
$\wedge\, \neg\texttt{IsOverridden}(s, m)$
$\wedge\, \neg\texttt{IsRecursiveMethod}(s, m)$
$\wedge\, \neg\texttt{ExistsMethodInvocation}(s, m, s, m1)$
$\wedge\, \neg\texttt{ExistsMethodInvocation}(s, m, s, m2)$
$\wedge\, \neg\texttt{IsUsedMethodIn}(s, m, a)$
$\wedge\, \neg\texttt{IsUsedMethodIn}(s, m, b)$
$\wedge\, \neg\texttt{IsUsedMethodIn}(s, m, c))$

Backward Description. $\texttt{ExistsMethodDefinition}(s, m) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(s, m, [t; t']) \mapsto \bot$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(s, m, X, Y) \mapsto \bot$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(s, m, X) \mapsto \bot$
$\texttt{BoundVariableInMethodBody}(s, m, X) \mapsto \bot$
$\texttt{ExistsParameterWithName}(s, m, [X], Y) \mapsto \bot$
$\texttt{ExistsParameterWithType}(s, m, [X], Y) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(s, m, X, Y) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(s, n, Y, Z) \mapsto$
$(\texttt{ExistsMethodInvocation}(s, m, Y, Z) \quad \vee \texttt{ExistsMethodInvocation}(s, n, Y, Z))$
$\texttt{IsUsedConstructorAsObjectReceiver}(X, s, n) \mapsto$
$(\texttt{IsUsedConstructorAsObjectReceiver}(X, s, m) \quad \vee \texttt{IsUsedConstructorAsObjectReceiver}(X, s, n))$
$\texttt{IsIndirectlyRecursive}(s, m) \mapsto \bot$
$\texttt{IsVisibleMethod}(s, m, [X], Y) \mapsto \bot$
$\texttt{IsInverter}(s, m, X, Y) \mapsto \bot$
$\texttt{IsDelegator}(s, m, X) \mapsto \bot$
$\texttt{IsUsedMethod}(s, m, [X]) \mapsto \bot$
$\texttt{IsUsedMethodIn}(s, m, X) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(X, s, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsInitializer}(X, s, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(X, s, m) \mapsto \bot$
$\texttt{IsPublic}(s, m) \mapsto \bot$
$\texttt{IsProtected}(s, m) \mapsto \bot$
$\texttt{IsPrivate}(s, m) \mapsto \bot$
$\texttt{IsUsedAttributeInMethodBody}(s, X, m) \mapsto \bot$

$\texttt{IsOverridden}(s, m) \mapsto \bot$
$\texttt{IsOverloaded}(s, m) \mapsto \bot$
$\texttt{IsOverriding}(s, m) \mapsto \bot$
$\texttt{IsRecursiveMethod}(s, m) \mapsto \bot$
$\texttt{HasReturnType}(s, m, X) \mapsto \bot$
$\texttt{HasParameterType}(s, m) \mapsto \bot$
$\texttt{MethodHasParameterType}(s, m, X) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(s, m, [X], [X]) \mapsto \bot$

A.18 InlineMethodInvocations

Overview: InlineMethodInvocations(classname c,inlinedmethod m, classofinlinedmethod a, modifiedmethod n): this operation is used to in-line a method invocation of the method c::m inside the method a::n.



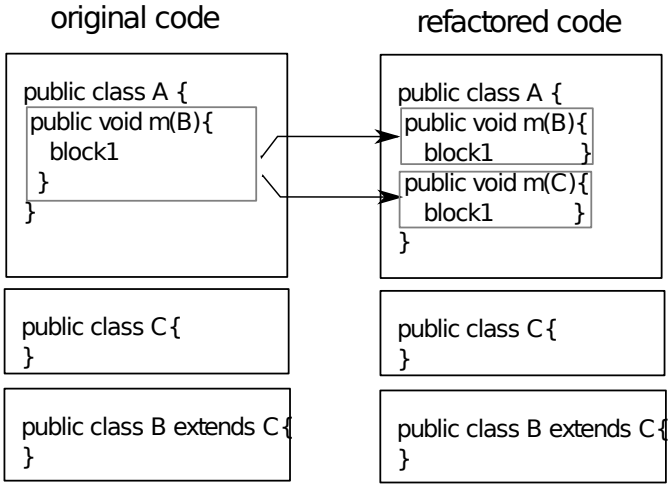
Refactoring tools. Inline in Eclipse and IntelliJ IDEA: select an invocation to inline and specify you want to inline only that one.

Precondition.
$(\texttt{ExistsClass}(c)$
$\wedge\, \texttt{IsIndirectlyRecursive}(c, m)$
$\wedge\, \texttt{IsRecursiveMethod}(c, n)$
$\wedge\, \neg\texttt{IsRecursiveMethod}(c, m)$
$\wedge\, \texttt{ExistsMethodInvocation}(c, m, a, n)$
$\wedge\, \texttt{ExistsMethodDefinition}(c, m)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(c, m, [t; t'])$
$\wedge\, \texttt{ExistsMethodDefinition}(a, n)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(a, n, [t1; t1']))$

Backward Description. $\texttt{ExistsMethodDefinitionWithParams}(c, m, [t; t']) \mapsto \top$
$\texttt{ExistsMethodDefinition}(c, m) \mapsto \top$
$\texttt{ExistsMethodInvocation}(c, m, a, n) \mapsto \bot$
$\texttt{IsRecursiveMethod}(c, m) \mapsto \texttt{ExistsMethodInvocation}(a, n, c, m)$
$\texttt{IsIndirectlyRecursive}(c, m) \mapsto$
$(\texttt{ExistsMethodInvocation}(a, n, C, X)$
$\wedge\, \texttt{ExistsMethodInvocation}(C, X, c, m))(\mathit{condition})$
$\texttt{IsUsedMethodIn}(a, n, m) \mapsto \bot$

A.19 AddSpecializedMethodInHierarchy (Composed)

Overview: AddSpecializedMethodInHierarchy(class s, subclasses [a,b], methodname m, callermethods [n,o], inkovekmethods [p,q], paramtype t, paramname pn , subtypesOfparamtype [t1,t2],newtype t'): this operation is used to get the method s::m(t' pn) instead of s::m(t pn). This new duplication takes place in s and in all its subclasses that override m.



Algorithm of the operation The operation AddSpecializedMethodInHierarchy is based on three steps :

AddSpecializedMethodInHierarchy(class s, subclasses [a,b], methodname m, callermethods [n,o], inkovekmethods [p,q], paramtype t, paramname pn , subtypesOfparamtype [t1,t2],newtype t') =

1. DuplicateMethodInHierarchy s [a,b] m [p,q] [n,o] temporaryName [t]
2. SpecialiseParameter s [a,b] temporaryName t pn [t1,t2] t';
3. RenameDelegatorWithOverloading (s, [a,b], temporaryName,t', pn,t,m)

Refactoring tools. With IntelliJ IDEA:

1. Apply DuplicateMethodInHierarchy(c, m, temp-name) (see A.20 below).
2. Apply *Change Signature* on the method temp-name in the class s, to change the parameter type t into t' (this change is propagated into subclasses).

   Note that the behavior preservation is not guaranteed by this operation in general, but here we introduce a new method so the behavior is not changed. Note also, that here we cannot use the operation *Type Migration* of IntelliJ IDEA: replacing a parameter type by one of its subtypes is not safe in general.
3. Rename temp-name into m in s with *Rename*. That renaming introduces an overloading. In general, this could change the semantics of the program, but in the case of this particular chain, and provided the preconditions given below are satisfied, the behavior is preserved (the two methods have the same body; some invocations may be dispatched on the new method, but the external behavior is the same).

A.20 DuplicateMethodInHierarchy

Overview: DuplicateMethodInHierarchy(class s, subclasses [a,b], methodname m, callermethods [m1,m2], inkovekmethods [m3,m4],newname n ,paramType [t]) : this operation is used to create a duplicate of the method s::m with the name n. All overriding methods in subclasses are also duplicated in these classes.

Refactoring tools. With IntelliJ IDEA:

1. For each implementation of the method m in the subclasses of the class s, duplicate m by applying *Extract Method* on its body (give the new name, specify the desired visibility), then inline the invocation of method n that has replaced the method's body.
2. Use *Pull Members Up* to make the new method appear in classes where the initial method is declared abstract (specify that it must appear as abstract) (see *PullUpAbstract*).

Precondition.
$(\texttt{ExistsClass}(s)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t; t'])$
$\wedge\, \texttt{ExistsMethodDefinition}(s, m)$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(s, n, [t; t'])$

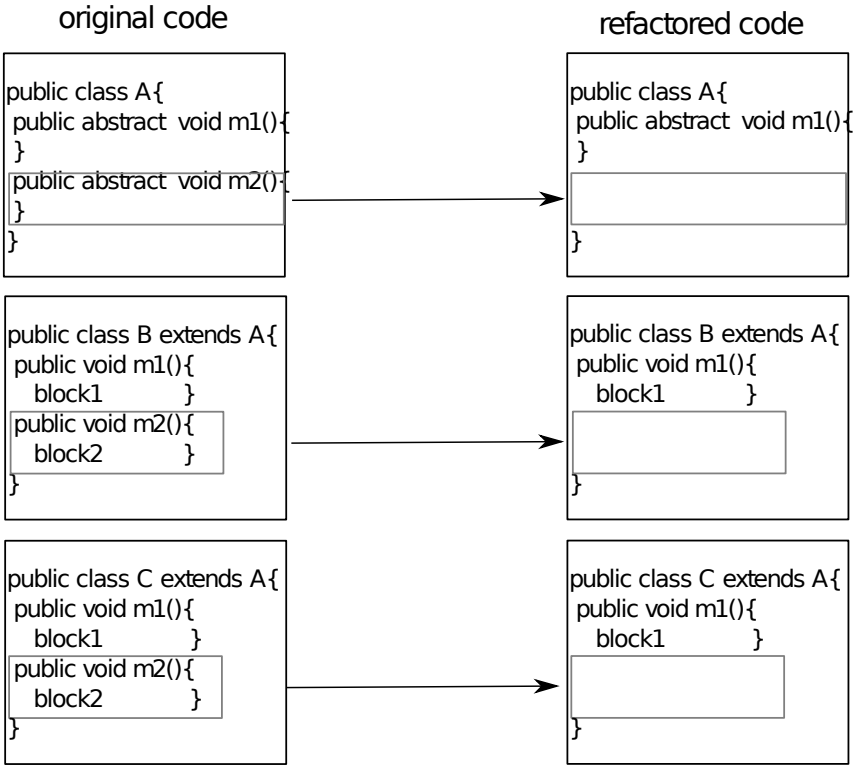
$\wedge \neg\texttt{ExistsMethodDefinitionWithParams}(a, n, [t; t'])$
$\wedge \neg\texttt{ExistsMethodDefinitionWithParams}(b, n, [t; t'])$
$\wedge \neg\texttt{IsInheritedMethodWithParams}(s, n, [t; t'])$
$\wedge \texttt{AllSubclasses}(s, [a; b]))$

Backward Description.
$\texttt{ExistsMethodDefinition}(s, n) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(s, n, [t; t']) \mapsto \top$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(s, n, V) \mapsto \top(condition)$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(s, n, V, V1) \mapsto \texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclare}$
$\texttt{BoundVariableInMethodBody}(s, n, V) \mapsto \texttt{BoundVariableInMethodBody}(s, m, V)$
$\texttt{IsPublic}(s, n) \mapsto \texttt{IsPublic}(s, m)$
$\texttt{ExistsParameterWithName}(s, n, [t; t'], V) \mapsto \texttt{ExistsParameterWithName}(s, m, [t; t'], V)$
$\texttt{ExistsParameterWithType}(s, n, [t; t'], T) \mapsto \texttt{ExistsParameterWithType}(s, m, [t; t'], T)$
$\texttt{IsIndirectlyRecursive}(s, n) \mapsto \texttt{IsIndirectlyRecursive}(s, m)$
$\texttt{IsRecursiveMethod}(s, n) \mapsto \texttt{IsRecursiveMethod}(s, m)$
$\texttt{IsInverter}(s, n, T, V) \mapsto \texttt{IsInverter}(s, m, T, V)$
$\texttt{IsUsedAttributeInMethodBody}(s, V, n) \mapsto \texttt{IsUsedAttributeInMethodBody}(s, V, m)$
$\texttt{MethodHasParameterType}(s, n, V) \mapsto \texttt{MethodHasParameterType}(s, m, V)$
$\texttt{ExistsMethodDefinitionWithParams}(a, n, [t; t']) \mapsto \texttt{ExistsMethodDefinitionWithParams}(a, m, [t; t'])$
$\texttt{ExistsMethodDefinitionWithParams}(b, n, [t; t']) \mapsto \texttt{ExistsMethodDefinitionWithParams}(b, m, [t; t'])$
$\texttt{IsDelegator}(s, n, m3) \mapsto \top$
$\texttt{IsDelegator}(s, n, m4) \mapsto \top$
$\texttt{IsDelegator}(a, n, m3) \mapsto \top$
$\texttt{IsDelegator}(a, n, m4) \mapsto \top$
$\texttt{IsDelegator}(b, n, m3) \mapsto \top$
$\texttt{IsDelegator}(b, n, m4) \mapsto \top$
$\texttt{ExistsMethodDefinition}(s, n) \mapsto \top$
$\texttt{ExistsMethodDefinition}(a, n) \mapsto \top$
$\texttt{ExistsMethodDefinition}(b, n) \mapsto \top$
$\texttt{MethodIsUsedWithType}(s, n, [t; t'], [t; t']) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(a, n, [t; t'], [t; t']) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(b, n, [t; t'], [t; t']) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(s, n, [t; t'], [T]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(a, n, [t; t'], [T]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(b, n, [t; t'], [T]) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(s, m1, V, n) \mapsto \top$
$\texttt{ExistsMethodInvocation}(s, m2, V, n) \mapsto \top$
$\texttt{ExistsMethodInvocation}(a, m1, V, n) \mapsto \top$
$\texttt{ExistsMethodInvocation}(a, m2, V, n) \mapsto \top$
$\texttt{ExistsMethodInvocation}(b, m1, V, n) \mapsto \top$
$\texttt{ExistsMethodInvocation}(b, m2, V, n) \mapsto \top$
$\texttt{IsInheritedMethodWithParams}(a, n, [t; t']) \mapsto \neg\texttt{ExistsMethodDefinitionWithParams}(a, m, [t; t'])$
$\texttt{IsInheritedMethodWithParams}(b, n, [t; t']) \mapsto \neg\texttt{ExistsMethodDefinitionWithParams}(b, m, [t; t'])$
$\texttt{IsInheritedMethod}(a, n) \mapsto \neg\texttt{ExistsMethodDefinition}(a, m)$
$\texttt{IsInheritedMethod}(b, n) \mapsto \neg\texttt{ExistsMethodDefinition}(b, m)$
$\texttt{IsOverriding}(a, n) \mapsto \neg\texttt{ExistsMethodDefinition}(a, m)$
$\texttt{IsOverriding}(b, n) \mapsto \neg\texttt{ExistsMethodDefinition}(b, m)$
$\texttt{IsOverridden}(a, n) \mapsto \neg\texttt{ExistsMethodDefinition}(a, m)$
$\texttt{IsOverridden}(b, n) \mapsto \neg\texttt{ExistsMethodDefinition}(b, m)$

A.21 DeleteMethodInHierarchy

(*Delete Method* in Fowler [Fow99] and [Koc02])

Overview: DeleteMethodInHierarchy (classname s, subclasses [a,b], method m, invokedmethodsInm [m1,m2], paramType t) : this operation is used to delete the method m from the hierarchy of classes s, a and b.



Refactoring tools. *Safe Delete* in IntelliJ IDEA and *Delete* in Eclipse.

Precondition.
$(\texttt{ExistsClass}(s)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t])$
$\wedge\, \neg\texttt{MethodIsUsedWithType}(s, m, [t], [t])$
$\wedge\, \neg\texttt{MethodIsUsedWithType}(a, m, [t], [t])$
$\wedge\, \neg\texttt{MethodIsUsedWithType}(b, m, [t], [t])$
$\wedge\, \texttt{AllSubclasses}(s, [a; b]))$

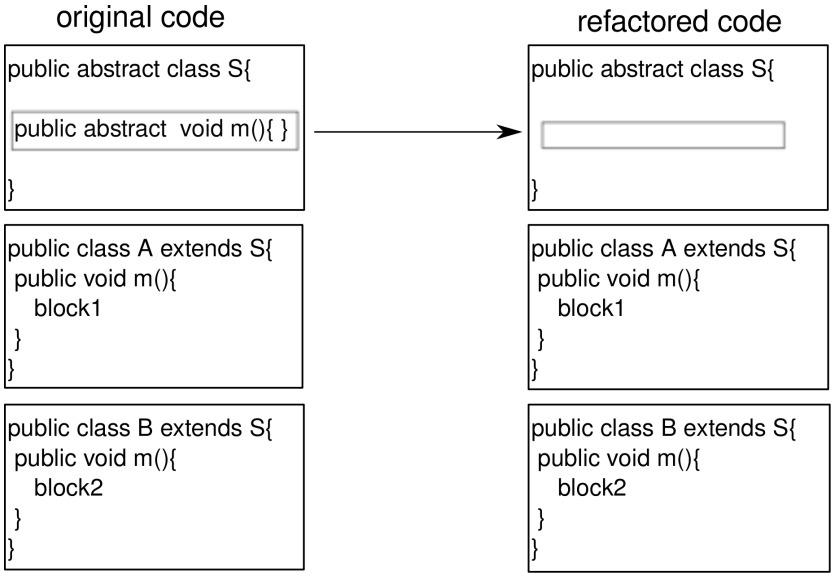
Backward Description.
$\texttt{ExistsParameterWithType}(s, m, [t], t) \mapsto \bot$
$\texttt{ExistsParameterWithType}(a, m, [t], t) \mapsto \bot$
$\texttt{ExistsParameterWithType}(b, m, [t], t) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(s, m, [t]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(a, m, [t]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [t]) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(s, m) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(a, m) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(b, m) \mapsto \bot$
$\texttt{IsUsedMethod}(s, m1, [V1]) \mapsto \bot$
$\texttt{IsUsedMethod}(s, m2, [V1]) \mapsto \bot$
$\texttt{IsUsedMethod}(a, m1, [V1]) \mapsto \bot$
$\texttt{IsUsedMethod}(a, m2, [V1]) \mapsto \bot$
$\texttt{IsUsedMethod}(b, m1, [V1]) \mapsto \bot$

$\texttt{IsUsedMethod}(b, m2, [V1]) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V1, s, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V1, a, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V1, b, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(t, s, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(t, a, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(t, b, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(t, s, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(t, a, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(t, b, m) \mapsto \bot$
$\texttt{IsInheritedMethod}(a, m) \mapsto \bot$
$\texttt{IsInheritedMethod}(b, m) \mapsto \bot$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(s, m, V1, V2) \mapsto \bot$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(a, m, V1, V2) \mapsto \bot$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(b, m, V1, V2) \mapsto \bot$
$\texttt{ExistsAbstractMethod}(s, m) \mapsto \bot$
$\texttt{ExistsAbstractMethod}(a, m) \mapsto \bot$
$\texttt{ExistsAbstractMethod}(b, m) \mapsto \bot$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(s, m, V1) \mapsto \bot$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(a, m, V1) \mapsto \bot$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(b, m, V1) \mapsto \bot$
$\texttt{BoundVariableInMethodBody}(s, m, V1) \mapsto \bot$
$\texttt{BoundVariableInMethodBody}(a, m, V1) \mapsto \bot$
$\texttt{BoundVariableInMethodBody}(b, m, V1) \mapsto \bot$
$\texttt{ExistsParameterWithName}(s, m, [t], V1) \mapsto \bot$
$\texttt{ExistsParameterWithName}(a, m, [t], V1) \mapsto \bot$
$\texttt{ExistsParameterWithName}(b, m, [t], V1) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(s, m, V1, V2) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(a, m, V1, V2) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(b, m, V1, V2) \mapsto \bot$
$\texttt{IsInheritedMethodWithParams}(a, m, [t]) \mapsto \bot$
$\texttt{IsInheritedMethodWithParams}(b, m, [t]) \mapsto \bot$
$\texttt{IsIndirectlyRecursive}(s, m) \mapsto \bot$
$\texttt{IsIndirectlyRecursive}(a, m) \mapsto \bot$
$\texttt{IsIndirectlyRecursive}(b, m) \mapsto \bot$
$\texttt{IsVisibleMethod}(s, m, [t], V1) \mapsto \bot$
$\texttt{IsVisibleMethod}(a, m, [t], V1) \mapsto \bot$
$\texttt{IsVisibleMethod}(b, m, [t], V1) \mapsto \bot$
$\texttt{IsInverter}(s, m, V1, V2) \mapsto \bot$
$\texttt{IsInverter}(a, m, V1, V2) \mapsto \bot$
$\texttt{IsInverter}(b, m, V1, V2) \mapsto \bot$
$\texttt{IsDelegator}(s, V1, m) \mapsto \bot$
$\texttt{IsDelegator}(a, V1, m) \mapsto \bot$
$\texttt{IsDelegator}(b, V1, m) \mapsto \bot$
$\texttt{IsUsedMethodIn}(s, m, V1) \mapsto \bot$
$\texttt{IsUsedMethodIn}(a, m, V1) \mapsto \bot$
$\texttt{IsUsedMethodIn}(b, m, V1) \mapsto \bot$
$\texttt{IsUsedConstructorAsInitializer}(V1, s, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsInitializer}(V1, a, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsInitializer}(V1, b, m) \mapsto \bot$
$\texttt{IsUsedAttributeInMethodBody}(s, V1, m) \mapsto \bot$
$\texttt{IsUsedAttributeInMethodBody}(a, V1, m) \mapsto \bot$

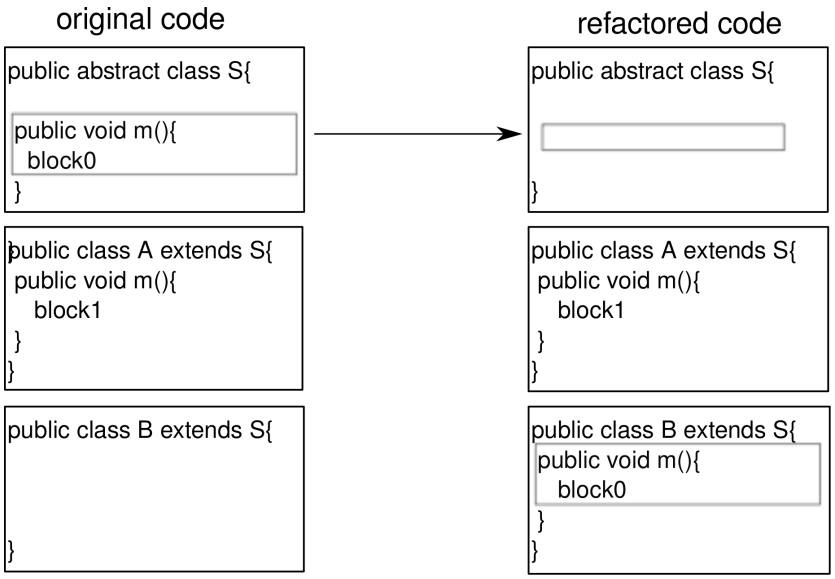
$\texttt{IsUsedAttributeInMethodBody}(b, V1, m) \mapsto \bot$
$\texttt{IsOverridden}(a, m) \mapsto \bot$
$\texttt{IsOverridden}(b, m) \mapsto \bot$
$\texttt{IsOverloaded}(s, m) \mapsto \bot$
$\texttt{IsOverloaded}(a, m) \mapsto \bot$
$\texttt{IsOverloaded}(b, m) \mapsto \bot$
$\texttt{IsOverriding}(a, m) \mapsto \bot$
$\texttt{IsOverriding}(b, m) \mapsto \bot$
$\texttt{IsRecursiveMethod}(s, m) \mapsto \bot$
$\texttt{IsRecursiveMethod}(a, m) \mapsto \bot$
$\texttt{IsRecursiveMethod}(b, m) \mapsto \bot$
$\texttt{HasReturnType}(s, m, V1) \mapsto \bot$
$\texttt{HasReturnType}(a, m, V1) \mapsto \bot$
$\texttt{HasReturnType}(b, m, V1) \mapsto \bot$
$\texttt{MethodHasParameterType}(s, m, V1) \mapsto \bot$
$\texttt{MethodHasParameterType}(a, m, V1) \mapsto \bot$
$\texttt{MethodHasParameterType}(b, m, V1) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(s, m, [t], [t]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(a, m, [t], [t]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(b, m, [t], [t]) \mapsto \bot$

A.22 PushDownAll

Overview: PushDownAll (classname s, attributes [att1,att2], subclasses [a,b], method m,paramType [t]) : this operation is used to push down the method s::m to its subclasses and delete that method from s (in *Push Down Method* by Fowler [Fow99], methods are not necessarily pushed down to all the subclasses).



Variation for non-abstract methods:



Refactoring tools. *Push Down* or *Push member Down* in Eclipse tool and IntelliJ IDEA.

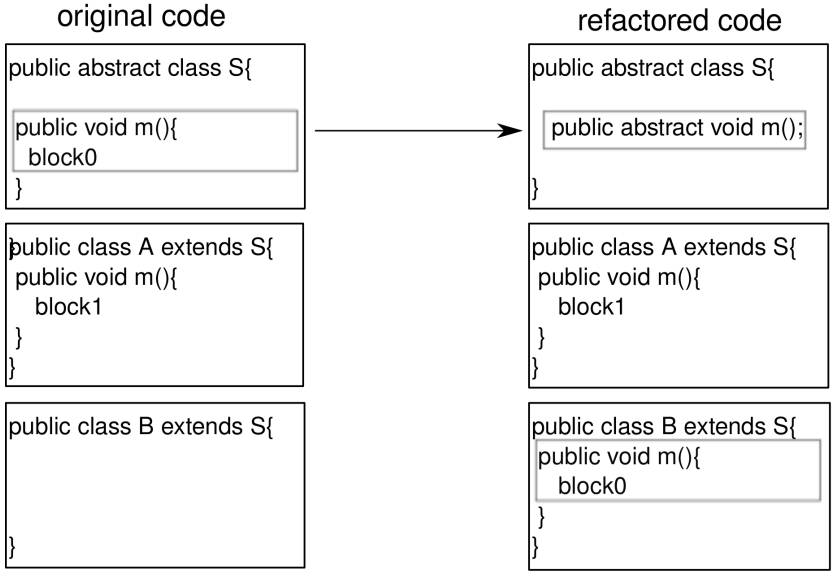
Precondition.
$(\texttt{ExistsClass}(s)$
$\wedge\, \texttt{IsAbstractClass}(s)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t])$
$\wedge\, \neg\texttt{IsUsedMethod}(s, m, [t])$
$\wedge\, \texttt{AllSubclasses}(s, [a; b])$
$\wedge\, \neg\texttt{IsPrivate}(s, m)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t])$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t])$
$\wedge\, \neg\texttt{IsPrivate}(s, att1)$
$\wedge\, \neg\texttt{IsPrivate}(s, att2))$

Backward Description. $\texttt{ExistsMethodDefinitionWithParams}(s, m, [t]) \mapsto \bot$
$\texttt{IsUsedMethodIn}(s, m, C) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(s, m) \mapsto \bot$
$\texttt{ExistsAbstractMethod}(s, m) \mapsto \bot$
$\texttt{IsDelegator}(s, m, V1) \mapsto \bot$
$\texttt{HasReturnType}(s, m, V1) \mapsto \bot$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(s, m, V1, V2) \mapsto \bot$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(s, m, V1) \mapsto \bot$
$\texttt{BoundVariableInMethodBody}(s, m, V1) \mapsto \bot$
$\texttt{ExistsParameterWithName}(s, m, [t], V1) \mapsto \bot$
$\texttt{ExistsParameterWithType}(s, m, [t], V1) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(s, m, V1, V2) \mapsto \bot$
$\texttt{IsPublic}(s, m) \mapsto \bot$
$\texttt{IsProtected}(s, m) \mapsto \bot$
$\texttt{IsPrivate}(s, m) \mapsto \bot$
$\texttt{IsOverloaded}(s, m) \mapsto \bot$
$\texttt{IsUsedAttributeInMethodBody}(s, V1, m) \mapsto \bot$
$\texttt{IsRecursiveMethod}(s, m) \mapsto \bot$
$\texttt{IsIndirectlyRecursive}(s, m) \mapsto \bot$
$\texttt{HasReturnType}(s, m, V1) \mapsto \bot$
$\texttt{MethodHasParameterType}(s, m, V1) \mapsto \bot$

$\texttt{MethodIsUsedWithType}(s, m, [t], [t]) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V1, s, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(V1, s, m) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(a, m, [t]) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [t]) \mapsto \top$
$\texttt{ExistsMethodDefinition}(a, m) \mapsto \top$
$\texttt{ExistsMethodDefinition}(b, m) \mapsto \top$
$\texttt{IsOverriding}(a, m) \mapsto \bot$
$\texttt{IsOverriding}(b, m) \mapsto \bot$
$\texttt{IsOverridden}(a, m) \mapsto \bot$
$\texttt{IsOverridden}(b, m) \mapsto \bot$
$\texttt{IsInheritedMethodWithParams}(a, m, [t]) \mapsto \bot$
$\texttt{IsInheritedMethodWithParams}(b, m, [t]) \mapsto \bot$
$\texttt{IsVisibleMethod}(s, m, [t], a) \mapsto \bot$
$\texttt{IsVisibleMethod}(s, m, [t], b) \mapsto \bot$
$\texttt{IsVisible}(s, m, a) \mapsto \bot$
$\texttt{IsVisible}(s, m, b) \mapsto \bot$
$\texttt{IsInheritedMethod}(a, m) \mapsto \bot$
$\texttt{IsInheritedMethod}(b, m) \mapsto \bot$
$\texttt{IsPrivate}(s, att1) \mapsto \bot$
$\texttt{IsPrivate}(s, att2) \mapsto \bot$

A.23 PushDownImplementation

Overview: PushDownImplementation (classname s, attributes [att1,att2], subclasses [a,b], method m,paramType [t,t']) : same as *PushDownAll* but keep the method abstract in the superclass.



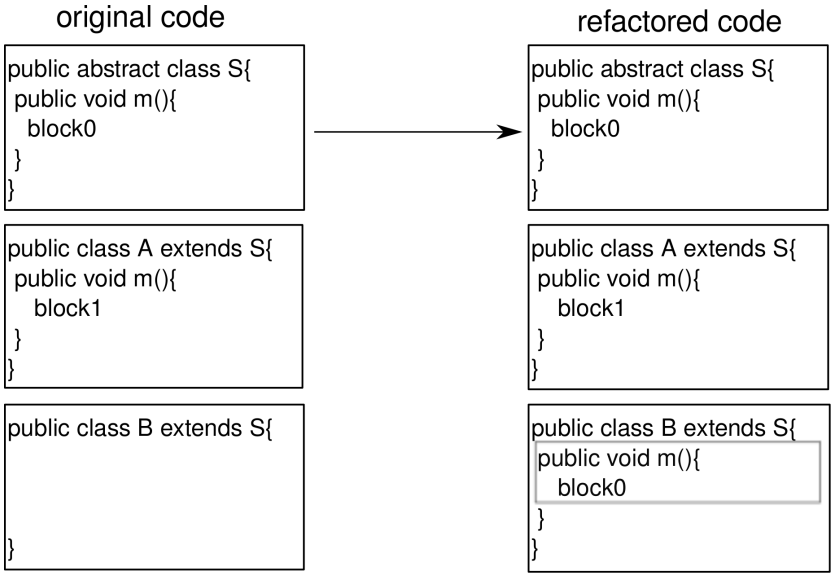
Precondition.
$(\texttt{ExistsClass}(s)$
$\wedge\, \texttt{ExistsMethodDefinition}(s, m)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t; t'])$
$\wedge\, \neg\texttt{ExistsAbstractMethod}(s, m)$
$\wedge\, \texttt{AllSubclasses}(s, [a; b])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(a, m, [t; t'])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(b, m, [t; t'])$
$\wedge\, \neg\texttt{IsPrivate}(s, att1)$

$\wedge \neg$IsPrivate$(s, att2))$

Backward Description. ExistsAbstractMethod$(s, m) \mapsto \top$
AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC$(s, m, V1, V2) \mapsto \top$
AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded$(s, m, V1) \mapsto \top$
BoundVariableInMethodBody$(s, m, V1) \mapsto \bot$
ExistsMethodInvocation$(s, m, V1, V2) \mapsto \bot$
IsInheritedMethodWithParams$(s, m, [t; t']) \mapsto \bot$
IsIndirectlyRecursive$(s, m) \mapsto \bot$
IsUsedConstructorAsInitializer$(V1, s, m) \mapsto \bot$
IsUsedConstructorAsObjectReceiver$(V1, s, m) \mapsto \bot$
IsPrivate$(s, m) \mapsto \bot$
IsUsedAttributeInMethodBody$(s, V1, m) \mapsto \bot$
IsOverridden$(s, m) \mapsto \bot$
IsOverriding$(s, m) \mapsto \bot$
IsRecursiveMethod$(s, m) \mapsto \bot$
MethodHasParameterType$(s, m, V1) \mapsto \bot$
MethodIsUsedWithType$(s, m, [t; t'], [t; t']) \mapsto \bot$
ExistsMethodDefinitionWithParams$(a, m, [t; t']) \mapsto \top$
ExistsMethodDefinitionWithParams$(b, m, [t; t']) \mapsto \top$
ExistsMethodDefinition$(a, m) \mapsto \top$
ExistsMethodDefinition$(b, m) \mapsto \top$
IsPrivate$(s, att1) \mapsto \bot$
IsPrivate$(s, att2) \mapsto \bot$

A.24 PushDownNotRedefinedMethod

pushDownNotRedefinedMethod(classname c, superclass s ,notredefinedmethods [m1,m2])
Duplicate the method m of class c into its subclasses.



Refactoring tools. *Extract Method*, *Inline*, *Push Down*, *Rename* in Eclipse and IntelliJ IDEA.

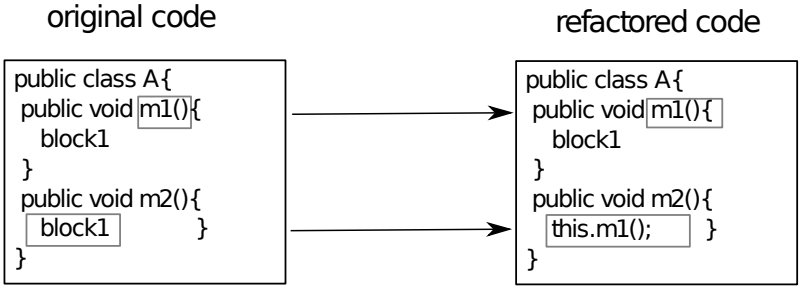
Precondition.
(ExistsType(c)

$\wedge$ ExistsClass(c)
$\wedge$ IsSubType(c, s)
$\wedge \neg$ExistsMethodDefinition$(c, m1)$
$\wedge \neg$ExistsMethodDefinition$(c, m2)$
$\wedge$ ExistsMethodDefinition$(s, m1)$
$\wedge$ ExistsMethodDefinition$(s, m2)$)

Backward Description.
ExistsMethodDefinition$(c, m1) \mapsto \top$
ExistsMethodDefinition$(c, m2) \mapsto \top$
IsOverriding$(c, m1) \mapsto \top$
IsOverriding$(c, m2) \mapsto \top$
IsOverridden$(c, m1) \mapsto \top$
IsOverridden$(c, m2) \mapsto \top$
BoundVariableInMethodBody$(c, m1, V) \mapsto$ BoundVariableInMethodBody$(s, m1, V)$
BoundVariableInMethodBody$(c, m2, V) \mapsto$ BoundVariableInMethodBody$(s, m2, V)$
HasSameBody$(c, m1, s, m1) \mapsto \top$
HasSameBody$(c, m2, s, m2) \mapsto \top$
AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC$(c, m1, this, s) \mapsto \top$
AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC$(c, m2, this, s) \mapsto \top$

A.25 ReplaceMethodDuplication

Overview: ReplaceMethodDuplication (classname s, subclasses [a,b], method m,copy n, paramType [t]) : this operation is used to replace any occurrence of method s::m.



Refactoring tools. *Replace Method Duplication* in IntelliJ IDEA.

Precondition.
(ExistsClass(s)
$\wedge$ ExistsMethodDefinition(s, m)
$\wedge$ ExistsMethodDefinition(s, n)
$\wedge$ IsDelegator(s, n, m)
$\wedge$ AllSubclasses$(s, [a; b])$)

Backward Description.
IsUsedMethod$(s, n, [t]) \mapsto \bot$
IsDelegator$(s, n, m) \mapsto \top$
ExistsMethodInvocation$(s, n, s, m) \mapsto \bot$
IsRecursiveMethod$(s, n) \mapsto \bot$
IsRecursiveMethod$(s, n) \mapsto \bot$
IsRecursiveMethod$(a, n) \mapsto \bot$
IsRecursiveMethod$(b, n) \mapsto \bot$
ExistsMethodInvocation$(a, m, a, n) \mapsto$ ExistsMethodInvocation(a, n, a, m)

$\texttt{ExistsMethodInvocation}(b, m, b, n) \mapsto \texttt{ExistsMethodInvocation}(b, n, b, m)$

A.26 DeleteClass

DeleteClass(class c): Delete a class c which is not used.

Overview: DeleteClass (classname a, classnamesuperclass s ,allclasses [s,a,b], classnamemethods [m,m1],othermethods [m2,n]) : this operation is used to delete the class a which is supposed to be not used.

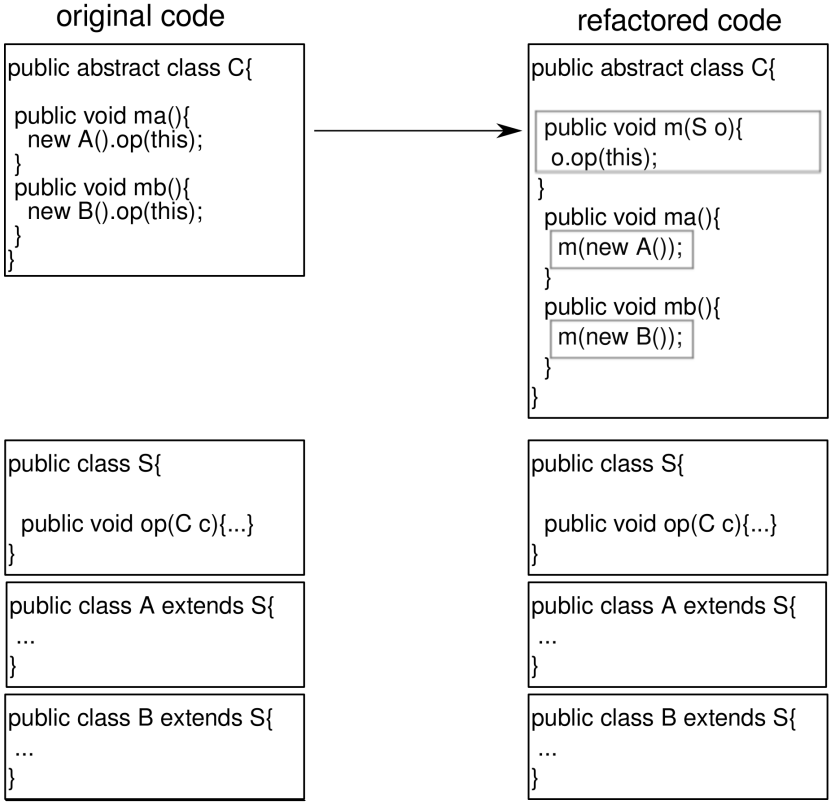
Refactoring tools. *Safe Delete* in IntelliJ IDEA, *Delete* in Eclipse.

Precondition.
$(\texttt{ExistsClass}(a)$
$\wedge\, \texttt{ExistsType}(a)$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(s, m2, [a])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(s, n, [a])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(b, m2, [a])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(b, n, [a])$
$\wedge\, \neg\texttt{IsUsedMethodIn}(a, m, s)$
$\wedge\, \neg\texttt{IsUsedMethodIn}(a, m1, s)$
$\wedge\, \neg\texttt{IsUsedMethodIn}(a, m, b)$
$\wedge\, \neg\texttt{IsUsedMethodIn}(a, m1, b)$
$\wedge\, \neg\texttt{IsUsedConstructorAsMethodParameter}(a, s, m2)$
$\wedge\, \neg\texttt{IsUsedConstructorAsMethodParameter}(a, s, n)$
$\wedge\, \neg\texttt{IsUsedConstructorAsMethodParameter}(a, b, m2)$
$\wedge\, \neg\texttt{IsUsedConstructorAsMethodParameter}(a, b, n)$
$\wedge\, \neg\texttt{IsUsedConstructorAsInitializer}(a, s, m2)$
$\wedge\, \neg\texttt{IsUsedConstructorAsInitializer}(a, s, n)$
$\wedge\, \neg\texttt{IsUsedConstructorAsInitializer}(a, b, m2)$
$\wedge\, \neg\texttt{IsUsedConstructorAsInitializer}(a, b, n)$
$\wedge\, \neg\texttt{IsUsedConstructorAsObjectReceiver}(a, s, m2)$
$\wedge\, \neg\texttt{IsUsedConstructorAsObjectReceiver}(a, s, n)$
$\wedge\, \neg\texttt{IsUsedConstructorAsObjectReceiver}(a, b, m2)$
$\wedge\, \neg\texttt{IsUsedConstructorAsObjectReceiver}(a, b, n)$
$\wedge\, \neg\texttt{IsSubType}(s, a)$
$\wedge\, \neg\texttt{IsSubType}(b, a))$

Backward Description.
$\texttt{ExistsType}(a) \mapsto \bot$
$\texttt{ExistsClass}(a) \mapsto \bot$
$\texttt{IsSubType}(a, s) \mapsto \bot$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(a, V1, V2, V3) \mapsto \bot$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(a, V1, V2) \mapsto \bot$
$\texttt{BoundVariableInMethodBody}(a, V1, V2) \mapsto \bot$
$\texttt{ExistsParameterWithName}(a, V1, [V2], V3) \mapsto \bot$
$\texttt{ExistsParameterWithType}(a, V1, [V2], V3) \mapsto \bot$
$\texttt{ExistsField}(a, V1) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(a, V1, V2, V3) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(a, V1, [V2]) \mapsto \bot$
$\texttt{ExtendsDirectly}(a, s) \mapsto \bot$
$\texttt{ExtendsDirectly}(V1, a) \mapsto \bot$
$\texttt{ExistsAbstractMethod}(a, V1) \mapsto \bot$
$\texttt{IsInheritedMethodWithParams}(a, V1, [V2]) \mapsto \bot$
$\texttt{IsIndirectlyRecursive}(a, V1) \mapsto \bot$
$\texttt{IsVisibleMethod}(a, V1, [V2], V3) \mapsto \bot$

$\texttt{IsInverter}(a, V1, V2, V3) \mapsto \bot$
$\texttt{IsDelegator}(a, V1, V2) \mapsto \bot$
$\texttt{IsAbstractClass}(a) \mapsto \bot$
$\texttt{IsUsedMethod}(a, V1, [V2]) \mapsto \bot$
$\texttt{IsUsedMethodIn}(a, V1, V2) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(V1, a, V2) \mapsto \bot$
$\texttt{IsUsedConstructorAsInitializer}(a, V1, V2) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(a, V1, V2) \mapsto \bot$
$\texttt{IsUsedConstructorAsInitializer}(V1, a, V2) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(V1, a, V2) \mapsto \bot$
$\texttt{IsPrimitiveType}(a) \mapsto \bot$
$\texttt{IsPublic}(a, V1) \mapsto \bot$
$\texttt{IsProtected}(a, V1) \mapsto \bot$
$\texttt{IsPrivate}(a, V1) \mapsto \bot$
$\texttt{IsUsedAttributeInMethodBody}(a, V1, V2) \mapsto \bot$
$\texttt{IsGenericsSubtype}(a, [V1], s, [V2]) \mapsto \bot$
$\texttt{IsGenericsSubtype}(V1, [V2], a, [V3]) \mapsto \bot$
$\texttt{IsGenericsSubtype}(V1, [a], V2, [V3]) \mapsto \bot$
$\texttt{IsInheritedField}(a, V1) \mapsto \bot$
$\texttt{IsOverridden}(a, V1) \mapsto \bot$
$\texttt{IsOverloaded}(a, V1) \mapsto \bot$
$\texttt{IsOverriding}(a, V1) \mapsto \bot$
$\texttt{IsRecursiveMethod}(a, V1) \mapsto \bot$
$\texttt{IsRecursiveMethod}(a, V1) \mapsto \bot$
$\texttt{HasReturnType}(a, V1, V2) \mapsto \bot$
$\texttt{HasParameterType}(a, V1) \mapsto \bot$
$\texttt{HasParameterType}(V1, a) \mapsto \bot$
$\texttt{MethodHasParameterType}(a, V1, V2) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(a, V1, [V2], [V3]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(V1, V2, [a], [a]) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(a, m) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(a, m1) \mapsto \bot$
$\texttt{IsInheritedMethodWithParams}(V1, m, [V2]) \mapsto \bot$
$\texttt{IsInheritedMethodWithParams}(V1, m1, [V2]) \mapsto \bot$

A.27 ExtractGeneralMethod



A.28 InlineClass

InlineClass(class c): Inline one or more references to a given class c.

Refactoring tools. *Inline* in Eclipse and IntelliJ IDEA.

A.29 SpecialiseParameter

Overview: SpecialiseParameter(classname s, subclasses [a,b], methodname m,paramType t ,paramName p, subtypes [st,q],new paramType st): this operation is used to change the type t of the parameter p of the methods s::m, a::m and b::m into one of its subtypes (st).

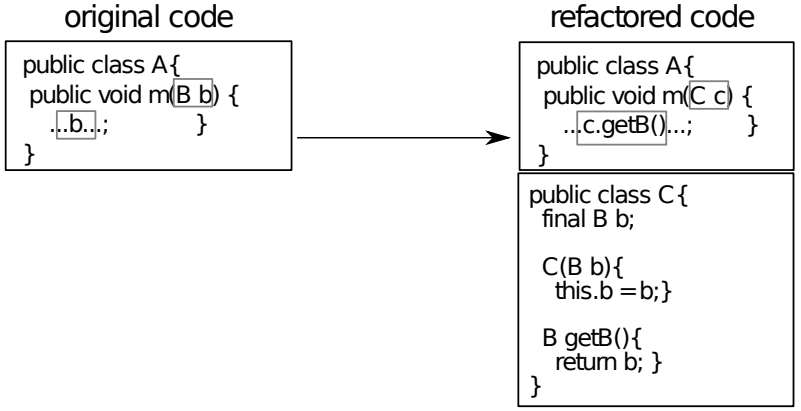
Precondition.
$(\texttt{IsSubType}(t1, t)$
$\wedge\, \neg\texttt{MethodIsUsedWithType}(s, m, [t], [t])$
$\wedge\, \neg\texttt{MethodIsUsedWithType}(a, m, [t], [t])$
$\wedge\, \neg\texttt{MethodIsUsedWithType}(b, m, [t], [t])$
$\wedge\, \texttt{ExistsClass}(s)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t])$
$\wedge\, \texttt{ExistsType}(t)$
$\wedge\, \texttt{ExistsType}(t1)$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(s, m, [t1])$
$\wedge\, \neg\texttt{IsInheritedMethodWithParams}(s, m, [t1])$
$\wedge\, \texttt{AllSubclasses}(s, [a; b])$
$\wedge\, \texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(s, m, p)$
$\wedge\, (\neg\texttt{MethodIsUsedWithType}(s, m, [t], [t2])$
$\quad \vee\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t2])))$

Backward Description.
$\texttt{ExistsMethodDefinitionWithParams}(s, m, [t1]) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(s, m, [t]) \mapsto \bot$

$\texttt{ExistsMethodDefinitionWithParams}(a, m, [t]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [t]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(a, m, [t1]) \mapsto \texttt{ExistsMethodDefinitionWithParams}(a, m, [t])$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [t1]) \mapsto \texttt{ExistsMethodDefinitionWithParams}(b, m, [t])$
$\texttt{MethodIsUsedWithType}(s, m, [t], [t]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(a, m, [t], [t]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(b, m, [t], [t]) \mapsto \bot$
$\texttt{ExistsParameterWithName}(s, m, [t1], p) \mapsto \top$
$\texttt{ExistsParameterWithName}(a, m, [t1], p) \mapsto \top$
$\texttt{ExistsParameterWithName}(b, m, [t1], p) \mapsto \top$
$\texttt{ExistsParameterWithType}(s, m, [t1], t1) \mapsto \top$
$\texttt{ExistsParameterWithType}(a, m, [t1], t1) \mapsto \top$
$\texttt{ExistsParameterWithType}(b, m, [t1], t1) \mapsto \top$
$\texttt{IsInverter}(s, m, t1, V) \mapsto \texttt{IsInverter}(s, m, t, V)$
$\texttt{IsInverter}(a, m, t1, V) \mapsto \texttt{IsInverter}(a, m, t, V)$
$\texttt{IsInverter}(b, m, t1, V) \mapsto \texttt{IsInverter}(b, m, t, V)$
$\texttt{IsInheritedMethodWithParams}(s, m, [t1]) \mapsto \top$
$\texttt{IsInheritedMethodWithParams}(a, m, [t1]) \mapsto \top$
$\texttt{IsInheritedMethodWithParams}(b, m, [t1]) \mapsto \top$
$\texttt{IsUsedConstructorAsMethodParameter}(t1, s, m) \mapsto \texttt{IsUsedConstructorAsMethodParameter}(t, s, m)$
$\texttt{IsUsedConstructorAsMethodParameter}(t1, a, m) \mapsto \texttt{IsUsedConstructorAsMethodParameter}(t, a, m)$
$\texttt{IsUsedConstructorAsMethodParameter}(t1, b, m) \mapsto \texttt{IsUsedConstructorAsMethodParameter}(t, b, m)$
$\texttt{IsOverridden}(a, m) \mapsto \texttt{ExistsMethodDefinition}(a, m)$
$\texttt{IsOverridden}(b, m) \mapsto \texttt{ExistsMethodDefinition}(b, m)$
$\texttt{IsOverriding}(a, m) \mapsto \texttt{ExistsMethodDefinition}(a, m)$
$\texttt{IsOverriding}(b, m) \mapsto \texttt{ExistsMethodDefinition}(b, m)$

A.30 IntroduceParameterObject

Overview: introduceParmeterObject `s [a,b] m [t1,t2] [p1,p2] t n`: introducing the parameter object `n` of type `t1` to the method `s::m(t p)` this creates a class of type t1 for instance `A`, moves the parameter p to A as an instance variable and finally changes `m(t p)` into `m(A a)`. Any old access to p in the body of m will be replaced by `a.p`.



Refactoring tools. *Extract Parameter Object* in IntelliJ IDEA.

Precondition.
$(\texttt{ExistsType}(s)$
$\wedge \neg\texttt{ExistsType}(t)$
$\wedge \texttt{ExistsMethodDefinitionWithParams}(s, m, [t1; t2])$
$\wedge \texttt{AllSubclasses}(s, [a; b])$
$\wedge \texttt{BoundVariableInMethodBody}(a, m, v1)$

$\wedge$ BoundVariableInMethodBody$(a, m, v2)$
$\wedge$ BoundVariableInMethodBody$(b, m, v1)$
$\wedge$ BoundVariableInMethodBody$(b, m, v2)$)

Backward Description.
ExistsClass$(t) \mapsto \top$
ExistsType$(t) \mapsto \top$
IsPrimitiveType$(t) \mapsto \bot$
IsUsedMethod$(s, m, [t]) \mapsto \top$
IsUsedMethod$(s, m, [t1; t2]) \mapsto \bot$
IsUsedMethodIn$(t, C, M) \mapsto \bot$
IsUsedMethod$(t, C, [T1]) \mapsto \bot$
IsUsedMethod$(t, C, [T1; T2]) \mapsto \bot$
IsUsedMethod$(t, C, [T1; T2; T3]) \mapsto \bot$
IsUsedMethod$(t, C, [T1; T2; T3; T4]) \mapsto \bot$
IsUsedConstructorAsMethodParameter$(t, C, m) \mapsto \bot$
IsUsedConstructorAsInitializer$(t, C, M) \mapsto \bot$
IsUsedConstructorAsMethodParameter$(C, t, M) \mapsto \bot$
IsUsedConstructorAsInitializer$(C, t, M) \mapsto \bot$
IsUsedConstructorAsObjectReceiver$(C, t, M) \mapsto \bot$
IsUsedConstructorAsInitializer$(t, C, M) \mapsto \bot$
IsSubType$(C, t) \mapsto \bot$
MethodIsUsedWithType$(s, m, [t], [t]) \mapsto \top$
MethodIsUsedWithType$(s, m, [t1; t2], [t1; t2]) \mapsto \bot$
IsInheritedMethodWithParams$(s, M, [t]) \mapsto \bot$
IsInheritedMethodWithParams$(a, m, [t]) \mapsto$ IsInheritedMethodWithParams$(a, m, [t1; t2])$
IsInheritedMethodWithParams$(b, m, [t]) \mapsto$ IsInheritedMethodWithParams$(b, m, [t1; t2])$
IsInheritedMethod$(t, M) \mapsto$ IsVisible$(java.lang.Object, M, t)$
IsInheritedMethodWithParams$(t, M, [\,]) \mapsto$ IsVisibleMethod$(java.lang.Object, M, [\,], t)$
IsInheritedMethodWithParams$(t, M, [T1]) \mapsto$ IsVisibleMethod$(java.lang.Object, M, [T1], t)$
IsInheritedMethodWithParams$(t, M, [T1; T2]) \mapsto$ IsVisibleMethod$(java.lang.Object, M, [T1; T2], t)$
IsInheritedMethodWithParams$(t, M, [T1; T2; T3]) \mapsto$ IsVisibleMethod$(java.lang.Object, M, [T1; T2; T3], t)$
IsInheritedMethodWithParams$(t, M, [T1; T2; T3; T4; T5]) \mapsto$ IsVisibleMethod$(java.lang.Object, M, [T1; T2; T3; T4; T5], t)$
ExistsMethodDefinitionWithParams$(t, M, [\,]) \mapsto \bot$
ExistsMethodDefinitionWithParams$(t, M, [T1]) \mapsto \bot$
ExistsMethodDefinitionWithParams$(t, M, [T1; T2]) \mapsto \bot$
ExistsMethodDefinitionWithParams$(t, M, [T1; T2; T3]) \mapsto \bot$
ExistsMethodDefinitionWithParams$(t, M, [T1; T2; T3; T4]) \mapsto \bot$
ExistsMethodDefinitionWithParams$(t, M, [T1; T2; T3; T4; T5]) \mapsto \bot$
IsSubType$(t, X) \mapsto \bot(condition)$
ExtendsDirectly$(t, X) \mapsto \bot(condition)$
IsInheritedMethodWithParams$(C, M, [t; T1]) \mapsto \bot$
IsInheritedMethodWithParams$(C, M, [t]) \mapsto \bot$
IsInheritedMethodWithParams$(C, M, [t; T1]) \mapsto \bot$
IsInheritedMethodWithParams$(C, M, [T1; t]) \mapsto \bot$
IsInheritedMethodWithParams$(C, M, [T1; T2; t]) \mapsto \bot$
IsInheritedMethodWithParams$(C, M, [T1; t; T2]) \mapsto \bot$
IsInheritedMethodWithParams$(C, M, [t; T1; T2]) \mapsto \bot$
ExtendsDirectly$(t, java.lang.Object) \mapsto \top$
IsUsedConstructorAsObjectReceiver$(t, a, m) \mapsto$ BoundVariableInMethodBody$(a, m, v1)$
IsUsedConstructorAsObjectReceiver$(t, a, m) \mapsto$ BoundVariableInMethodBody$(a, m, v2)$
IsUsedConstructorAsObjectReceiver$(t, b, m) \mapsto$ BoundVariableInMethodBody$(b, m, v1)$

$\texttt{IsUsedConstructorAsObjectReceiver}(t, b, m) \mapsto \texttt{BoundVariableInMethodBody}(b, m, v2)$
$\texttt{ExistsParameterWithName}(s, m, [t], n) \mapsto \top$
$\texttt{ExistsParameterWithName}(a, m, [t], n) \mapsto \top$
$\texttt{ExistsParameterWithName}(b, m, [t], n) \mapsto \top$
$\texttt{ExistsParameterWithType}(s, m, [t], t) \mapsto \top$
$\texttt{ExistsParameterWithType}(a, m, [t], t) \mapsto \top$
$\texttt{ExistsParameterWithType}(b, m, [t], t) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(s, m, [t]) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(C, M, [t]) \mapsto \bot(condition)$
$\texttt{ExistsMethodDefinitionWithParams}(C, M, [t; T1]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(C, M, [T1; t]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(C, M, [T1; T2; t]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(C, M, [T1; t; T2]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(C, M, [t; T1; T2]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(s, m, [t1; t2]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(a, m, [t1; t2]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [t1; t2]) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(a, m, [t]) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(b, m, [t]) \mapsto \top$
$\texttt{ExistsField}(t, p1) \mapsto \top$
$\texttt{ExistsField}(t, p2) \mapsto \top$
$\texttt{ExistsMethodDefinition}(t, getp1) \mapsto \top$
$\texttt{ExistsMethodDefinition}(t, getp2) \mapsto \top$
$\texttt{IsUsedConstructorAsObjectReceiver}(T1, T2, getp1) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(T1, T2, getp2) \mapsto \bot$
$\texttt{IsPrivate}(t, p1) \mapsto \top$
$\texttt{IsPrivate}(t, p2) \mapsto \top$
$\texttt{IsInheritedField}(t, p1) \mapsto \bot$
$\texttt{IsInheritedField}(t, p2) \mapsto \bot$
$\texttt{ExistsParameterWithName}(s, m, [t1; t2], p1) \mapsto \bot$
$\texttt{ExistsParameterWithName}(s, m, [t1; t2], p2) \mapsto \bot$
$\texttt{ExistsParameterWithName}(a, m, [t1; t2], p1) \mapsto \bot$
$\texttt{ExistsParameterWithName}(a, m, [t1; t2], p2) \mapsto \bot$
$\texttt{ExistsParameterWithName}(b, m, [t1; t2], p1) \mapsto \bot$
$\texttt{ExistsParameterWithName}(b, m, [t1; t2], p2) \mapsto \bot$
$\texttt{ExistsParameterWithType}(s, m, [t1; t2], t1) \mapsto \bot$
$\texttt{ExistsParameterWithType}(s, m, [t1; t2], t2) \mapsto \bot$
$\texttt{ExistsParameterWithType}(a, m, [t1; t2], t1) \mapsto \bot$
$\texttt{ExistsParameterWithType}(a, m, [t1; t2], t2) \mapsto \bot$
$\texttt{ExistsParameterWithType}(b, m, [t1; t2], t1) \mapsto \bot$
$\texttt{ExistsParameterWithType}(b, m, [t1; t2], t2) \mapsto \bot$
$\texttt{BoundVariableInMethodBody}(s, m, n) \mapsto \texttt{BoundVariableInMethodBody}(s, m, p1)$
$\texttt{BoundVariableInMethodBody}(s, m, n) \mapsto \texttt{BoundVariableInMethodBody}(s, m, p2)$
$\texttt{BoundVariableInMethodBody}(s, m, n) \mapsto \texttt{BoundVariableInMethodBody}(s, m, p1)$
$\texttt{BoundVariableInMethodBody}(s, m, n) \mapsto \texttt{BoundVariableInMethodBody}(s, m, p2)$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(s, m, n, T) \mapsto \top$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(a, m, n, T) \mapsto \top$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(b, m, n, T) \mapsto \top$
$\texttt{ExistsParameterWithName}(s, m, [t1; t2; t], n) \mapsto \top$
$\texttt{ExistsParameterWithName}(a, m, [t1; t2; t], n) \mapsto \top$
$\texttt{ExistsParameterWithName}(b, m, [t1; t2; t], n) \mapsto \top$
$\texttt{ExistsParameterWithType}(s, m, [t1; t2; t], t) \mapsto \top$

$\texttt{ExistsParameterWithType}(a, m, [t1; t2; t], t) \mapsto \top$
$\texttt{ExistsParameterWithType}(b, m, [t1; t2; t], t) \mapsto \top$

A.31 DeleteMethod

Overview: DeleteMethod c m [t1,t2] s: this operation is used to delete the method c::m which is semantically equivalent to the method m which is inherited from the class s which is the super class of c.

Refactoring tools. *Delete* in Eclipse and IntelliJ IDEA.

Precondition.
$\texttt{ExistsClass}(c)$
$\wedge\, \texttt{ExistsType}(c)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(c, m, [t1; t2])$
$\wedge\, \texttt{IsInheritedMethod}(c, m)$
$\wedge\, \texttt{IsDuplicate}(c, m, s, m)$
$\wedge\, \texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(c, m, this)$

Backward Description. $\texttt{ExistsParameterWithType}(c, m, [t1; t2], V1) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(c, m, [t1; t2]) \mapsto \bot$
$\texttt{ExistsMethodDefinition}(c, m) \mapsto \bot$
$\texttt{BoundVariableInMethodBody}(c, m, t1) \mapsto \bot$
$\texttt{BoundVariableInMethodBody}(c, m, t2) \mapsto \bot$
$\texttt{ExistsParameterWithName}(c, m, [t1; t2], V1) \mapsto \bot$
$\texttt{IsIndirectlyRecursive}(c, m) \mapsto \bot$
$\texttt{IsVisibleMethod}(c, m, [t1; t2], V1) \mapsto \bot$
$\texttt{IsInverter}(c, m, V1, V2) \mapsto \bot$
$\texttt{IsDelegator}(c, V1, m) \mapsto \bot$
$\texttt{IsOverridden}(c, m) \mapsto \bot$
$\texttt{IsOverloaded}(c, m) \mapsto \bot$
$\texttt{IsOverriding}(c, m) \mapsto \bot$
$\texttt{IsRecursiveMethod}(c, m) \mapsto \bot$
$\texttt{HasReturnType}(c, m, V1) \mapsto \bot$
$\texttt{MethodHasParameterType}(c, m, V1) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(c, m, [t1; t2], [t1; t2]) \mapsto \bot$

A.32 DuplicateMethodInHierarchyGen

Overview: DuplicateMethodInHierarchyGen (class name s, sublclasslist [a;b], methodname m, return types [r1;r2], invoked-methodsInmethodname [m1;m2], callermethods [m3;m4], newname n, methodnamparameters [t1;t2] : this operation is used to create a duplicate of the method s::m which has a generic type to two methods of name n and having respectively r1 and r2 as return types.

Refactoring tools. With IntelliJ IDEA:

1. For each implementation of the method m in the subclasses of the class s, duplicate m by applying *Extract Method* on its body (give the new name, specify the desired visibility), apply *change signature* to assign the right return type,then inline the invocation of method n that has replaced the method's body.
2. Use *Pull Members Up* to make the new method appear in classes where the initial method is declared abstract (specify that it must appear as abstract).

Precondition.
$(\texttt{ExistsClass}(s)$
$\wedge\, \texttt{ExistsMethodDefinitionWithParams}(s, m, [t1; t2])$
$\wedge\, \texttt{ExistsMethodDefinition}(s, m)$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(s, n, [t1; t2])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(a, n, [t1; t2])$
$\wedge\, \neg\texttt{ExistsMethodDefinitionWithParams}(b, n, [t1; t2])$
$\wedge\, \texttt{ExistsType}(r1)$
$\wedge\, \texttt{ExistsType}(r2)$
$\wedge\, \neg\texttt{IsInheritedMethodWithParams}(s, n, [t1; t2])$
$\wedge\, \texttt{AllSubclasses}(s, [a; b]))$

Backward Description.
$\texttt{ExistsMethodDefinition}(s, n) \mapsto \top$
$\texttt{ExistsMethodDefinitionWithParams}(s, n, [t1; t2]) \mapsto \top$
$\texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(s, n, V) \mapsto \top(condition)$
$\texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(s, n, V, V1) \mapsto \texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclare}$
$\texttt{BoundVariableInMethodBody}(s, n, V) \mapsto \texttt{BoundVariableInMethodBody}(s, m, V)$
$\texttt{IsPublic}(s, n) \mapsto \texttt{IsPublic}(s, m)$
$\texttt{ExistsParameterWithName}(s, n, [t1; t2], V) \mapsto \texttt{ExistsParameterWithName}(s, m, [t1; t2], V)$
$\texttt{ExistsParameterWithType}(s, n, [t1; t2], T) \mapsto \texttt{ExistsParameterWithType}(s, m, [t1; t2], T)$
$\texttt{IsIndirectlyRecursive}(s, n) \mapsto \texttt{IsIndirectlyRecursive}(s, m)$
$\texttt{IsRecursiveMethod}(s, n) \mapsto \texttt{IsRecursiveMethod}(s, m)$
$\texttt{IsInverter}(s, n, T, V) \mapsto \texttt{IsInverter}(s, m, T, V)$
$\texttt{IsUsedAttributeInMethodBody}(s, V, n) \mapsto \texttt{IsUsedAttributeInMethodBody}(s, V, m)$
$\texttt{MethodHasParameterType}(s, n, V) \mapsto \texttt{MethodHasParameterType}(s, m, V)$
$\texttt{ExistsMethodDefinitionWithParams}(a, n, [t1; t2]) \mapsto \texttt{ExistsMethodDefinitionWithParams}(a, m, [t1; t2])$
$\texttt{ExistsMethodDefinitionWithParams}(b, n, [t1; t2]) \mapsto \texttt{ExistsMethodDefinitionWithParams}(b, m, [t1; t2])$
$\texttt{IsDelegator}(s, n, m3) \mapsto \top$
$\texttt{IsDelegator}(s, n, m4) \mapsto \top$
$\texttt{IsDelegator}(a, n, m3) \mapsto \top$
$\texttt{IsDelegator}(a, n, m4) \mapsto \top$
$\texttt{IsDelegator}(b, n, m3) \mapsto \top$
$\texttt{IsDelegator}(b, n, m4) \mapsto \top$
$\texttt{ExistsMethodDefinition}(s, n) \mapsto \top$
$\texttt{ExistsMethodDefinition}(a, n) \mapsto \top$
$\texttt{ExistsMethodDefinition}(b, n) \mapsto \top$
$\texttt{MethodIsUsedWithType}(s, n, [t1; t2], [t1; t2]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(a, n, [t1; t2], [t1; t2]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(b, n, [t1; t2], [t1; t2]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(s, n, [t1; t2], [T]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(a, n, [t1; t2], [T]) \mapsto \bot$
$\texttt{MethodIsUsedWithType}(b, n, [t1; t2], [T]) \mapsto \bot$
$\texttt{ExistsMethodInvocation}(s, m1, V, n) \mapsto \top$
$\texttt{ExistsMethodInvocation}(s, m2, V, n) \mapsto \top$
$\texttt{ExistsMethodInvocation}(a, m1, V, n) \mapsto \top$
$\texttt{ExistsMethodInvocation}(a, m2, V, n) \mapsto \top$
$\texttt{ExistsMethodInvocation}(b, m1, V, n) \mapsto \top$
$\texttt{ExistsMethodInvocation}(b, m2, V, n) \mapsto \top$
$\texttt{IsInheritedMethodWithParams}(a, n, [t1; t2]) \mapsto \neg\texttt{ExistsMethodDefinitionWithParams}(a, m, [t1; t2])$
$\texttt{IsInheritedMethodWithParams}(b, n, [t1; t2]) \mapsto \neg\texttt{ExistsMethodDefinitionWithParams}(b, m, [t1; t2])$
$\texttt{IsInheritedMethod}(a, n) \mapsto \neg\texttt{ExistsMethodDefinition}(a, m)$
$\texttt{IsInheritedMethod}(b, n) \mapsto \neg\texttt{ExistsMethodDefinition}(b, m)$

$\texttt{IsOverriding}(a, n) \mapsto \neg\texttt{ExistsMethodDefinition}(a, m)$
$\texttt{IsOverriding}(b, n) \mapsto \neg\texttt{ExistsMethodDefinition}(b, m)$
$\texttt{IsOverridden}(a, n) \mapsto \neg\texttt{ExistsMethodDefinition}(a, m)$
$\texttt{IsOverridden}(b, n) \mapsto \neg\texttt{ExistsMethodDefinition}(b, m)$

A.33 AddSpecializedMethodInHierarchyGen (composed)

Overview: AddSpecializedMethodInHierarchyGen(class s, subclasses [a,b], methodname m, returntypes [r1,r2], callermethods [n,o], inkvokedmethods [p,q], paramtype t, paramname pn , subtypesOfparamtype [t1,t2],newtype t'): this operation is similar to the operation AddSpecializedMethodInHierarchy A.19 except that it is applied to methods having generic types. This operation is also composed (see the following algorithm).

Algorithm of the operation The operation AddSpecializedMethodInHierarchyGen is based on three steps :

AddSpecializedMethodInHierarchyGen(class s, subclasses [a,b], methodname m, returntypes [r1,r2], callermethods [n,o], inkvokedmethods [p,q], paramtype t, paramname pn , subtypesOfparamtype [t1,t2],newtype t') =

1. DuplicateMethodInHierarchyGen s [a,b] m [r1,r2] [p,q] [n,o] temporaryName [t]
2. SpecialiseParameter s [a,b] temporaryName t pn [t1,t2] t';
3. RenameDelegatorWithOverloading (s, [a,b], temporaryName,t', pn,t,m)

A.34 InlineConstructor

Overview: InlineConstructor (classname s, methodname m, inlinedConstructor c, fields [f1,f2], getters [g1,g2]: this operation is used to inline the constructor c which is used in the method c::m and whose the corresponding class has the fields [f1,f2] and the getter methods [g1,g2].

Refactoring tools. *inline* in IntelliJ IDEA.

Precondition.
$(\texttt{ExistsType}(s)$
$\wedge\, \texttt{ExistsType}(c)$
$\wedge\, \texttt{IsUsedConstructorAsObjectReceiver}(c, s, m)$
$\wedge\, (\neg\texttt{IsInheritedMethodWithParams}(s, m, [c])$
$\quad \vee\, \neg\texttt{ExistsMethodDefinitionWithParams}(s, m, [c])))$

Backward Description.
$\texttt{IsUsedConstructorAsObjectReceiver}(c, s, m) \mapsto \bot$
$\texttt{ExistsMethodDefinitionWithParams}(C, M, [c]) \mapsto \bot$
$\texttt{IsUsedMethodIn}(c, c, s) \mapsto \bot$
$\texttt{IsUsedConstructorAsMethodParameter}(c, s, m) \mapsto \bot$
$\texttt{IsUsedConstructorAsInitializer}(c, s, m) \mapsto \bot$
$\texttt{existsFieldInMethodScope}(s, m, f1) \mapsto \top$
$\texttt{existsFieldInMethodScope}(s, m, f2) \mapsto \top$
$\texttt{BoundVariableInMethodBody}(s, m, f1) \mapsto \top$
$\texttt{BoundVariableInMethodBody}(s, m, f2) \mapsto \top$
$\texttt{existslocalVariable}(s, m, f1var) \mapsto \top$
$\texttt{existslocalVariable}(s, m, f2var) \mapsto \top$
$\texttt{ExistsMethodDefinition}(s, g1) \mapsto \top$
$\texttt{ExistsMethodDefinition}(s, g2) \mapsto \top$
$\texttt{IsUsedConstructorAsObjectReceiver}(c, s, g1) \mapsto \bot$
$\texttt{IsUsedConstructorAsObjectReceiver}(c, s, g2) \mapsto \bot$
$\texttt{IsOverriding}(s, g1) \mapsto \bot$
$\texttt{IsOverriding}(s, g2) \mapsto \bot$

$\texttt{IsOverridden}(s, g1) \mapsto \bot$
$\texttt{IsOverridden}(s, g2) \mapsto \bot$
$\texttt{IsRecursiveMethod}(s, g1) \mapsto \bot$
$\texttt{IsRecursiveMethod}(s, g2) \mapsto \bot$

A.35 InlineLocalField

Overview: inlineLocalField (classname s, methodname m, fieldname f): this operation is used to inline the field f which is used in the scope of the method s::m.

Refactoring tools. *inline* in Eclipse and IntelliJ IDEA.

Precondition.
$(\texttt{ExistsType}(s)$
$\wedge\, \texttt{ExistsMethodDefinition}(s, m)$
$\wedge\, \texttt{existsFieldInMethodScope}(s, m, f))$

Backward Description.
$\texttt{existsFieldInMethodScope}(s, m, f) \mapsto \bot$

A.36 InlinelocalVariable

Overview: InlinelocalVariable (classname s, methodname m, variablename v): this operation is used to inline the local variable declared in the scope of the method s::m.

Refactoring tools. *inline* in Eclipse and IntelliJ IDEA.

Precondition.
$(\texttt{ExistsType}(s)$
$\wedge\, \texttt{ExistsMethodDefinition}(s, m)$
$\wedge\, \texttt{existslocalVariable}(s, m, v))$

Backward Description. $\texttt{existslocalVariable}(s, m, v) \mapsto \bot$

A.37 InlineParmeterObject (composed)

Overview: InlineParmeterObject (classname s, methodname m, inlinedConstructor c, inlinedgetters [g1,g2], fields [f1,f2]: this operation is used to inline a parameter object. It has the inverse role of the operation IntroduceParameterObject. This operation is composed (see algorithm below).

Algorithm of the operation The follwing algorithm describes the mechanics of this composed operation:

InlineParmeterObject (s, m, c, [g1,g2], [f1,f2]) =

1. InlineConstructor s m c [g1,g2] [f1,f2]
2. Forall f in [f1,f2] do InlineLocalField s m f
3. Forall g in [g1,g2] do InlineAnddelete (s,g,[], m, [], [])
4. Forall f in [f1,f2] do InlinelocalVariable s m (generate variable name from f)

B Precondition for all transformations

In this section, we apply a calculus of minimal precondition to the sequence of basic refactoring operations that compose our transformations. We apply the calculus of Kniesel and Koch [KK04], based on the *backward descriptions* given in the previous appendix. We use that calculus to compute a minimum precondition that ensures that the round-trip transformation succeeds, which means we determine a set of programs on which we can ensure that the preconditions of all the component refactoring operations will be satisfied when applying the *Composite→Visitor→Composite* transformation any number of time (as explained in [CA13]).

We give in the following the preconditions for the basic transformation and also those of transformation of the four variations evoked in the technical report.

B.1 Precondition for basic transformation

The computed precondition corresponding to the basic transformation is given in Fig. 17. The chains taken into account for the computation are given in Figs. 15 and 16. The difference between these chains and the algorithms of previous sections are:

- We add some parameters to the operations that were not made explicit before (in previous sections, the *project* was an implicit parameter).
- Composite operations are replaced by their component operations (MergeDuplicateMethods in step 8 of the *Composite→Visitor* chain).
- We split some operations of the tool into several abstract operations when the behavior of an operation depends on the state of the program. In that case, each possible behavior is represented by a different abstract operation. For instance, we have two operations for method renaming, one for overloaded methods and one for not-overloaded methods.

```
CreateEmptyClass(PrintVisitor) ;
CreateEmptyClass(PrettyprintVisitor) ;
CreateIndirectionInSuperClass(Graphic,[Ellipse;Composite;], print, [], void, printAux) ;
CreateIndirectionInSuperClass(Graphic, [Ellipse;Composite;], prettyprint, [], void, prettyprintAux) ;
InlineMethodInvocations(Composite, printAux, [], Graphic, print, []) ;
InlineMethodInvocations(Composite, prettyprintAux, [], Graphic, prettyprint, []) ;
AddParameterWithReuse(Graphic, [Ellipse;Composite;], printAux, [], PrintVisitor, v) ;
AddParameterWithReuse(Graphic, [Ellipse;Composite;], prettyprintAux, [], PrettyprintVisitor, v) ;
MoveMethodWithDelegate (Ellipse, [graphics;], PrintVisitor, printAux, [PrintVisitor;], void, visit) ;
MoveMethodWithDelegate (Composite, [graphics;], PrintVisitor, printAux, [PrintVisitor;], void, visit) ;
MoveMethodWithDelegate (Ellipse, [graphics;], PrettyprintVisitor, prettyprintAux, [PrettyprintVisitor;], void, visit) ;
MoveMethodWithDelegate (Composite, [graphics;], PrettyprintVisitor, prettyprintAux, [PrettyprintVisitor;], void, visit) ;
ExtractSuperClass ([PrintVisitor;PrettyprintVisitor;], Visitor, [visit;], void) ;
GeneraliseParameter (Graphic, [Ellipse;Composite;], printAux, v, PrintVisitor, Visitor) ;
GeneraliseParameter (Graphic, [Ellipse;Composite;], prettyprintAux, v, PrettyprintVisitor, Visitor) ;
ReplaceMethodcodeDuplicatesInverter (Ellipse, printAux, [prettyprintAux;], Visitor, void) ;
ReplaceMethodcodeDuplicatesInverter (Composite, printAux, [prettyprintAux;], Visitor, void) ;
PullupConcreteDelegator (Ellipse, [graphics;], prettyprintAux, Graphic) ;
SafeDeleteDelegatorWithOverridden (Composite, prettyprintAux, Graphic) ;
InlineAndDelete (Graphic, prettyprintAux) RenameInHierarchyNoOverloading (Graphic,[Ellipse;Composite;], printAux,[Visitor;], accept)
```

Figure 15: Chain of refactoring operations of the transformation Composite→Visiteur of basic program

DuplicateMethodInHierarchy (Graphic, [Ellipse;Composite;], accept,[visit;], [print;prettyprint;], acceptPrintVisitoraddspecializedMethodtmp, [Visitor;]) ;
SpecialiseParameter (Graphic, [Ellipse;Composite;], acceptPrintVisitoraddspecializedMethodtmp, Visitor, [PrintVisitor;PrettyprintVisitor;], PrintVisitor) ;
RenameDelegatorWithOverloading (Graphic, [Ellipse;Composite;], acceptPrintVisitoraddspecializedMethodtmp, PrintVisitor, v, Visitor, accept) ;
DuplicateMethodInHierarchy (Graphic, [Ellipse;Composite;], accept, [visit;], [print;prettyprint;], acceptPrettyprintVisitoraddspecializedMethodtmp, [Visitor;]) ;
SpecialiseParameter (Graphic, [Ellipse;Composite;], acceptPrettyprintVisitoraddspecializedMethodtmp, Visitor, [PrintVisitor;PrettyprintVisitor;], PrettyprintVisitor) ;
RenameDelegatorWithOverloading (Graphic, [Ellipse;Composite;], acceptPrettyprintVisitoraddspecializedMethodtmp, PrettyprintVisitor, v, Visitor, accept) ;
DeleteMethodInHierarchy (Graphic, [Ellipse;Composite;], accept, [visit;], Visitor) PushDownAll (Visitor, [PrintVisitor;PrettyprintVisitor;], visit, [Ellipse;]) ;
PushDownAll (Visitor, [PrintVisitor;PrettyprintVisitor;], visit, [Composite;]) ;
InlineMethod (Ellipse, visit, PrintVisitor, accept) ;
InlineMethod (Composite, visit, PrintVisitor, accept) ;
InlineMethod (Ellipse, visit, PrettyprintVisitor, accept) ;
InlineMethod (Composite, visit, PrettyprintVisitor, accept) ;
RenameOverloadedMethodInHierarchy (Graphic, [Ellipse;Composite;], accept, printAux, [PrintVisitor;]) ;
RenameOverloadedMethodInHierarchy (Graphic, [Ellipse;Composite;], accept, prettyprintAux, [PrettyprintVisitor;]) ;
RemoveParameter (Graphic, [Ellipse;Composite;], printAux, [PrintVisitor;], PrintVisitor, v) ;
RemoveParameter (Graphic, [Ellipse;Composite;], prettyprintAux, [PrettyprintVisitor;], PrettyprintVisitor, v) ;
ReplaceMethodDuplication (Graphic, [Ellipse;Composite;], print, printAux, []) ;
ReplaceMethodDuplication (Graphic, [Ellipse;Composite;], prettyprint, prettyprintAux, []) ;
PushDownImplementation (Graphic, [], [Ellipse;Composite;], print, []) ;
PushDownImplementation (Graphic, [], [Ellipse;Composite;], prettyprint, []) ;
PushDownAll (Graphic, [Ellipse;Composite;], printAux, []) ;
PushDownAll (Graphic, [Ellipse;Composite;], prettyprintAux, []) ;
InlineAndDelete (Ellipse, printAux) ;
InlineAndDelete (Composite, printAux) ;
InlineAndDelete (Ellipse, prettyprintAux) ;
InlineAndDelete (Composite, prettyprintAux) ;
DeleteClass (PrintVisitor, Visitor, [Ellipse;Composite;PrintVisitor;PrettyprintVisitor;Visitor;
Graphic;], [visit;], [accept;eval;prettyprint;]) ;
DeleteClass (PrettyprintVisitor, Visitor, [Ellipse;Composite;PrintVisitor;PrettyprintVisitor;Visitor;
Graphic;], [visit;], [accept;eval;prettyprint;]) ;
DeleteClass (Visitor, java.lang.Object,[Ellipse;Composite;PrintVisitor;PrettyprintVisitor;Visitor;
Graphic;], [visit;], [accept;eval;prettyprint;])

Figure 16: Chain of refactoring operations of the transformation Visiteur→Composite of basic program

$\neg$`ExistsMethod`$(Graphic, accept)$
$\wedge$ $\neg$`ExistsMethod`$(Ellipse, accept)$
$\wedge$ $\neg$`ExistsMethod`$(Composite, accept)$
$\wedge$ $\neg$`IsInheritedMethod`$(Graphic, accept)$
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`$(Composite, prettyprint, this)$
$\wedge$ `AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC`$(Ellipse, prettyprint, this, Graphic)$
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`$(Ellipse, prettyprint, this)$
$\wedge$ $\neg$`ExistsType`$(Visitor)$
$\wedge$ `ExistsClass`$(Ellipse)$
$\wedge$ $\neg$`BoundVariableInMethodBody`$(Graphic, prettyprint, v)$
$\wedge$ $\neg$`BoundVariableInMethodBody`$(Graphic, print, v)$
$\wedge$ `IsRecursiveMethod`$(Composite, prettyprint)$
$\wedge$ `ExistsClass`$(Composite)$
$\wedge$ `IsRecursiveMethod`$(Composite, print)$
$\wedge$ `ExistsMethodWithParams`$(Graphic, prettyprint, [])$
$\wedge$ `ExistsAbstractMethod`$(Graphic, prettyprint)$
$\wedge$ $\neg$`IsInheritedMethod`$(Graphic, prettyprintAux)$
$\wedge$ $\neg$`IsInheritedMethodWithParams`$(Graphic, prettyprintAux, [])$
$\wedge$ $\neg$`ExistsMethodWithParams`$(Graphic, prettyprintAux, [])$
$\wedge$ `HasReturnType`$(Graphic, prettyprint, void)$
$\wedge$ `ExistsMethod`$(Graphic, prettyprint)$
$\wedge$ `ExistsMethod`$(Ellipse, prettyprint)$
$\wedge$ `ExistsMethod`$(Composite, prettyprint)$
$\wedge$ $\neg$`ExistsMethod`$(Graphic, prettyprintAux)$
$\wedge$ $\neg$`ExistsMethod`$(Ellipse, prettyprintAux)$
$\wedge$ $\neg$`ExistsMethod`$(Composite, prettyprintAux)$
$\wedge$ `ExistsClass`$(Graphic)$
$\wedge$ `IsAbstractClass`$(Graphic)$
$\wedge$ `ExistsMethodWithParams`$(Graphic, print, [])$
$\wedge$ `ExistsAbstractMethod`$(Graphic, print)$
$\wedge$ $\neg$`IsInheritedMethod`$(Graphic, printAux)$
$\wedge$ $\neg$`IsInheritedMethodWithParams`$(Graphic, printAux, [])$
$\wedge$ $\neg$`ExistsMethodWithParams`$(Graphic, printAux, [])$
$\wedge$ `AllSubclasses`$(Graphic, [Ellipse; Composite])$
$\wedge$ `HasReturnType`$(Graphic, print, void)$
$\wedge$ $\neg$`IsPrivate`$(Graphic, print)$
$\wedge$ $\neg$`IsPrivate`$(Ellipse, print)$
$\wedge$ $\neg$`IsPrivate`$(Composite, print)$
$\wedge$ `ExistsMethod`$(Graphic, print)$
$\wedge$ `ExistsMethod`$(Ellipse, print)$
$\wedge$ `ExistsMethod`$(Composite, print)$
$\wedge$ $\neg$`ExistsMethod`$(Graphic, printAux)$
$\wedge$ $\neg$`ExistsMethod`$(Ellipse, printAux)$
$\wedge$ $\neg$`ExistsMethod`$(Composite, printAux)$
$\wedge$ $\neg$`ExistsType`$(PrettyprintVisitor)$
$\wedge$ $\neg$`ExistsType`$(PrintVisitor)$

Figure 17: Computed precondition for the round-trip transformation

B.2 Precondition for method with parameter variation

$\neg$IsUsedConstructorAsMethodParameter($SeColorVisitor, Ellipse, print$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($SeColorVisitor, CompositeGraphic, print$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($SeColorVisitor, Graphic, print$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($SeColorVisitor, Graphic, setColor$)
$\wedge$ $\neg$IsUsedConstructorAsObjectReceiver($SeColorVisitor, Ellipse, printAux$)
$\wedge$ $\neg$IsUsedConstructorAsObjectReceiver($SeColorVisitor, Ellipse, print$)
$\wedge$ $\neg$IsUsedConstructorAsObjectReceiver($SeColorVisitor, CompositeGraphic, printAux$)
$\wedge$ $\neg$IsUsedConstructorAsObjectReceiver($SeColorVisitor, CompositeGraphic, print$)
$\wedge$ ExistsType($CompositeGraphic$)
$\wedge$ ExistsType($Ellipse$)
$\wedge$ $\neg$ExistsMethodInvocation($Ellipse, setColorAux, Ellipse, print$)
$\wedge$ $\neg$IsUsedMethod($Graphic, accept_SeColorVisitor_addspecializedMethod_tmp, [SeColorVisitor]$)
$\wedge$ $\neg$ExistsMethodDefinition($Graphic, accept$)
$\wedge$ $\neg$ExistsMethodDefinition($Ellipse, accept$)
$\wedge$ $\neg$ExistsMethodDefinition($CompositeGraphic, accept$)
$\wedge$ $\neg$IsInheritedMethod($Graphic, accept$)
$\wedge$ $\neg$IsUsedMethodIn($Graphic, setColorAux, Ellipse$)
$\wedge$ $\neg$IsUsedMethodIn($Graphic, setColorAux, CompositeGraphic$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($CompositeGraphic, setColor, this$)
$\wedge$ AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC($Ellipse, setColor, this, Graphic$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($Ellipse, setColor, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($Graphic, setColor, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($Ellipse, setColor, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($CompositeGraphic, setColor, v$)
$\wedge$ $\neg$ExistsType($Visitor$)
$\wedge$ ExistsClass($Ellipse$)
$\wedge$ ExistsType($Graphic$)
$\wedge$ $\neg$ExistsType($SeColorVisitor$)
$\wedge$ BoundVariableInMethodBody($Ellipse, setColor, intc$)
$\wedge$ BoundVariableInMethodBody($CompositeGraphic, setColor, intc$)
$\wedge$ $\neg$BoundVariableInMethodBody($Graphic, print, v$)
$\wedge$ IsRecursiveMethod($CompositeGraphic, setColor$)
$\wedge$ ExistsClass($CompositeGraphic$)
$\wedge$ IsRecursiveMethod($CompositeGraphic, print$)
$\wedge$ ExistsMethodDefinitionWithParams($Graphic, setColor, [intc]$)
$\wedge$ ExistsAbstractMethod($Graphic, setColor$)
$\wedge$ $\neg$IsInheritedMethod($Graphic, setColorAux$)
$\wedge$ $\neg$IsInheritedMethodWithParams($Graphic, setColorAux, [intc]$)
$\wedge$ $\neg$ExistsMethodDefinitionWithParams($Graphic, setColorAux, [intc]$)
$\wedge$ HasReturnType($Graphic, setColor, void$)
$\wedge$ ExistsMethodDefinition($Graphic, setColor$)
$\wedge$ ExistsMethodDefinition($Ellipse, setColor$)
$\wedge$ ExistsMethodDefinition($CompositeGraphic, setColor$)
$\wedge$ $\neg$ExistsMethodDefinition($Graphic, setColorAux$)
$\wedge$ $\neg$ExistsMethodDefinition($Ellipse, setColorAux$)
$\wedge$ $\neg$ExistsMethodDefinition($CompositeGraphic, setColorAux$)
$\wedge$ ExistsClass($Graphic$)
$\wedge$ IsAbstractClass($Graphic$)
$\wedge$ ExistsMethodDefinitionWithParams($Graphic, print, [\,]$)
$\wedge$ ExistsAbstractMethod($Graphic, print$)
$\wedge$ $\neg$IsInheritedMethod($Graphic, printAux$)
$\wedge$ $\neg$IsInheritedMethodWithParams($Graphic, printAux, [\,]$)
$\wedge$ $\neg$ExistsMethodDefinitionWithParams($Graphic, printAux, [\,]$)
$\wedge$ AllSubclasses($Graphic, [Ellipse; CompositeGraphic]$)
$\wedge$ HasReturnType($Graphic, print, void$)
$\wedge$ $\neg$IsPrivate($Graphic, print$)
$\wedge$ $\neg$IsPrivate($Ellipse, print$)
$\wedge$ $\neg$IsPrivate($CompositeGraphic, print$)
$\wedge$ ExistsMethodDefinition($Graphic, print$)
$\wedge$ ExistsMethodDefinition($Ellipse, print$)
$\wedge$ ExistsMethodDefinition($CompositeGraphic, print$)
$\wedge$ $\neg$ExistsMethodDefinition($Graphic, printAux$)
$\wedge$ $\neg$ExistsMethodDefinition($Ellipse, printAux$)
$\wedge$ $\neg$ExistsMethodDefinition($CompositeGraphic, printAux$)
$\wedge$ $\neg$ExistsType($PrintVisitor$)

Figure 18: Computed precondition for the round-trip transformation of method with parameter variation

B.3 Precondition for method with different return types variation

$\neg$`ExistsMethodInvocation`($Ellipse, toStringAux, Ellipse, perimeter$)
$\wedge$ $\neg$`ExistsMethodDefinition`($Graphic, accept$)
$\wedge$ $\neg$`ExistsMethodDefinition`($Ellipse, accept$)
$\wedge$ $\neg$`ExistsMethodDefinition`($CompositeGraphic, accept$)
$\wedge$ $\neg$`IsInheritedMethod`($Graphic, accept$)
$\wedge$ $\neg$`IsUsedMethodIn`($Graphic, toStringAux, Ellipse$)
$\wedge$ $\neg$`IsUsedMethodIn`($Graphic, toStringAux, CompositeGraphic$)
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`($CompositeGraphic, toString, this$)
$\wedge$ `AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC`($Ellipse, toString, this, Graphic$)
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`($Ellipse, toString, this$)
$\wedge$ $\neg$`IsPrimitiveType`($String$)
$\wedge$ $\neg$`ExistsAbstractMethod`($Visitor, visit$)
$\wedge$ $\neg$`IsPrimitiveType`($Integer$)
$\wedge$ $\neg$`ExistsType`($Visitor$)
$\wedge$ `ExistsClass`($Ellipse$)
$\wedge$ $\neg$`BoundVariableInMethodBody`($Graphic, toString, v$)
$\wedge$ $\neg$`BoundVariableInMethodBody`($Graphic, perimeter, v$)
$\wedge$ `IsRecursiveMethod`($CompositeGraphic, toString$)
$\wedge$ `ExistsClass`($CompositeGraphic$)
$\wedge$ `IsRecursiveMethod`($CompositeGraphic, perimeter$)
$\wedge$ `ExistsMethodDefinitionWithParams`($Graphic, toString, []$)
$\wedge$ `ExistsAbstractMethod`($Graphic, toString$)
$\wedge$ $\neg$`IsInheritedMethod`($Graphic, toStringAux$)
$\wedge$ $\neg$`IsInheritedMethodWithParams`($Graphic, toStringAux, []$)
$\wedge$ $\neg$`ExistsMethodDefinitionWithParams`($Graphic, toStringAux, []$)
$\wedge$ `HasReturnType`($Graphic, toString, String$)
$\wedge$ `ExistsMethodDefinition`($Graphic, toString$)
$\wedge$ `ExistsMethodDefinition`($Ellipse, toString$)
$\wedge$ `ExistsMethodDefinition`($CompositeGraphic, toString$)
$\wedge$ $\neg$`ExistsMethodDefinition`($Graphic, toStringAux$)
$\wedge$ $\neg$`ExistsMethodDefinition`($Ellipse, toStringAux$)
$\wedge$ $\neg$`ExistsMethodDefinition`($CompositeGraphic, toStringAux$)
$\wedge$ `ExistsClass`($Graphic$)
$\wedge$ `IsAbstractClass`($Graphic$)
$\wedge$ `ExistsMethodDefinitionWithParams`($Graphic, perimeter, []$)
$\wedge$ `ExistsAbstractMethod`($Graphic, perimeter$)
$\wedge$ $\neg$`IsInheritedMethod`($Graphic, perimeterAux$)
$\wedge$ $\neg$`IsInheritedMethodWithParams`($Graphic, perimeterAux, []$)
$\wedge$ $\neg$`ExistsMethodDefinitionWithParams`($Graphic, perimeterAux, []$)
$\wedge$ `AllSubclasses`($Graphic, [Ellipse; CompositeGraphic]$)
$\wedge$ `HasReturnType`($Graphic, perimeter, Integer$)
$\wedge$ $\neg$`IsPrivate`($Graphic, perimeter$)
$\wedge$ $\neg$`IsPrivate`($Ellipse, perimeter$)
$\wedge$ $\neg$`IsPrivate`($CompositeGraphic, perimeter$)
$\wedge$ `ExistsMethodDefinition`($Graphic, perimeter$)
$\wedge$ `ExistsMethodDefinition`($Ellipse, perimeter$)
$\wedge$ `ExistsMethodDefinition`($CompositeGraphic, perimeter$)
$\wedge$ $\neg$`ExistsMethodDefinition`($Graphic, perimeterAux$)
$\wedge$ $\neg$`ExistsMethodDefinition`($Ellipse, perimeterAux$)
$\wedge$ $\neg$`ExistsMethodDefinition`($CompositeGraphic, perimeterAux$)
$\wedge$ $\neg$`ExistsType`($ToStringVisitor$)
$\wedge$ $\neg$`ExistsType`($PerimeterVisitor$)

Figure 19: Computed precondition for the round-trip transformation of method with different return type variation

B.4 Precondition for several level hierarchy variation

$\texttt{IsInheritedMethod}(ColoredRect, prettyprint)$
$\wedge \neg\texttt{ExistsMethodInvocation}(Rect, prettyprintTmpVC, Rect, print)$
$\wedge \neg\texttt{ExistsMethodDefinition}(Graphic, accept)$
$\wedge \neg\texttt{ExistsMethodDefinition}(Rect, accept)$
$\wedge \neg\texttt{ExistsMethodDefinition}(ColoredRect, accept)$
$\wedge \neg\texttt{ExistsMethodDefinition}(CompositeGraphic, accept)$
$\wedge \neg\texttt{IsInheritedMethod}(Graphic, accept)$
$\wedge \neg\texttt{IsUsedMethodIn}(Graphic, prettyprintTmpVC, Rect)$
$\wedge \neg\texttt{IsUsedMethodIn}(Graphic, prettyprintTmpVC, ColoredRect)$
$\wedge \neg\texttt{IsUsedMethodIn}(Graphic, prettyprintTmpVC, CompositeGraphic)$
$\wedge \texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(CompositeGraphic, prettyprint, this)$
$\wedge \texttt{AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC}(Rect, prettyprint, this, Graphic)$
$\wedge \neg\texttt{ExistsType}(Visitor)$
$\wedge \texttt{ExistsClass}(Rect)$
$\wedge \neg\texttt{BoundVariableInMethodBody}(Graphic, prettyprint, v)$
$\wedge \neg\texttt{BoundVariableInMethodBody}(Graphic, print, v)$
$\wedge \texttt{IsRecursiveMethod}(CompositeGraphic, prettyprint)$
$\wedge \texttt{ExistsClass}(CompositeGraphic)$
$\wedge \texttt{IsRecursiveMethod}(CompositeGraphic, print)$
$\wedge \texttt{ExistsMethodDefinitionWithParams}(Graphic, prettyprint, [])$
$\wedge \texttt{ExistsAbstractMethod}(Graphic, prettyprint)$
$\wedge \neg\texttt{IsInheritedMethod}(Graphic, prettyprintTmpVC)$
$\wedge \neg\texttt{IsInheritedMethodWithParams}(Graphic, prettyprintTmpVC, [])$
$\wedge \neg\texttt{ExistsMethodDefinitionWithParams}(Graphic, prettyprintTmpVC, [])$
$\wedge \texttt{HasReturnType}(Graphic, prettyprint, void)$
$\wedge \texttt{ExistsMethodDefinition}(Graphic, prettyprint)$
$\wedge \texttt{ExistsMethodDefinition}(CompositeGraphic, prettyprint)$
$\wedge \neg\texttt{ExistsMethodDefinition}(Graphic, prettyprintTmpVC)$
$\wedge \neg\texttt{ExistsMethodDefinition}(Rect, prettyprintTmpVC)$
$\wedge \neg\texttt{ExistsMethodDefinition}(ColoredRect, prettyprintTmpVC)$
$\wedge \neg\texttt{ExistsMethodDefinition}(CompositeGraphic, prettyprintTmpVC)$
$\wedge \texttt{ExistsClass}(Graphic)$
$\wedge \texttt{IsAbstractClass}(Graphic)$
$\wedge \texttt{ExistsMethodDefinitionWithParams}(Graphic, print, [])$
$\wedge \texttt{ExistsAbstractMethod}(Graphic, print)$
$\wedge \neg\texttt{IsInheritedMethod}(Graphic, printTmpVC)$
$\wedge \neg\texttt{IsInheritedMethodWithParams}(Graphic, printTmpVC, [])$
$\wedge \neg\texttt{ExistsMethodDefinitionWithParams}(Graphic, printTmpVC, [])$
$\wedge \texttt{AllSubclasses}(Graphic, [Rect; ColoredRect; CompositeGraphic])$
$\wedge \texttt{HasReturnType}(Graphic, print, void)$
$\wedge \neg\texttt{IsPrivate}(Graphic, print)$
$\wedge \neg\texttt{IsPrivate}(Rect, print)$
$\wedge \neg\texttt{IsPrivate}(ColoredRect, print)$
$\wedge \neg\texttt{IsPrivate}(CompositeGraphic, print)$
$\wedge \texttt{ExistsMethodDefinition}(Graphic, print)$
$\wedge \texttt{ExistsMethodDefinition}(Rect, print)$
$\wedge \texttt{ExistsMethodDefinition}(ColoredRect, print)$
$\wedge \texttt{ExistsMethodDefinition}(CompositeGraphic, print)$
$\wedge \neg\texttt{ExistsMethodDefinition}(Graphic, printTmpVC)$
$\wedge \neg\texttt{ExistsMethodDefinition}(Rect, printTmpVC)$
$\wedge \neg\texttt{ExistsMethodDefinition}(ColoredRect, printTmpVC)$
$\wedge \neg\texttt{ExistsMethodDefinition}(CompositeGraphic, printTmpVC)$
$\wedge \neg\texttt{ExistsType}(PrettyPrintVisitor)$
$\wedge \neg\texttt{ExistsType}(PrintVisitor)$
$\wedge \texttt{ExistsType}(ColoredRect)$
$\wedge \texttt{ExistsClass}(ColoredRect)$
$\wedge \texttt{IsSubType}(ColoredRect, Rect)$
$\wedge \neg\texttt{ExistsMethodDefinition}(ColoredRect, prettyprint)$
$\wedge \texttt{ExistsMethodDefinition}(Rect, prettyprint)$
$\wedge \texttt{AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded}(Rect, prettyprint, this)$

Figure 20: Computed precondition for the round-trip transformation of several level hierarchy variation

B.5 Precondition for interface variation

$\neg$ExistsMethodDefinition($Ellipse, accept$)
$\wedge$ $\neg$ExistsMethodDefinition($CompositeGraphic, accept$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($CompositeGraphic, prettyprint, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($Ellipse, prettyprint, this$)
$\wedge$ $\neg$ExistsType($Visitor$)
$\wedge$ IsRecursiveMethod($CompositeGraphic, prettyprint$)
$\wedge$ IsRecursiveMethod($CompositeGraphic, print$)
$\wedge$ HasReturnType($Ellipse, prettyprint, void$)
$\wedge$ HasReturnType($CompositeGraphic, prettyprint, void$)
$\wedge$ ExistsMethodDefinition($Ellipse, prettyprint$)
$\wedge$ ExistsMethodDefinition($CompositeGraphic, prettyprint$)
$\wedge$ $\neg$ExistsMethodDefinition($Ellipse, prettyprintTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($CompositeGraphic, prettyprintTmpVC$)
$\wedge$ HasReturnType($Ellipse, print, void$)
$\wedge$ HasReturnType($CompositeGraphic, print, void$)
$\wedge$ $\neg$IsPrivate($Ellipse, print$)
$\wedge$ $\neg$IsPrivate($CompositeGraphic, print$)
$\wedge$ ExistsMethodDefinition($Ellipse, print$)
$\wedge$ ExistsMethodDefinition($CompositeGraphic, print$)
$\wedge$ $\neg$ExistsMethodDefinition($Ellipse, printTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($CompositeGraphic, printTmpVC$)
$\wedge$ $\neg$ExistsType($PrettyPrintvisitor$)
$\wedge$ $\neg$ExistsType($PrintVisitor$)
$\wedge$ $\neg$ExistsType($AbstractGraphic$)
$\wedge$ IsInterface($Graphic$)
$\wedge$ ExistsType($Graphic$)
$\wedge$ ExistsClass($Ellipse$)
$\wedge$ ExistsClass($CompositeGraphic$)
$\wedge$ ExtendsDirectly($Ellipse, java.lang.Object$)
$\wedge$ ExtendsDirectly($CompositeGraphic, java.lang.Object$)
$\wedge$ implementsDirectly($Ellipse, Graphic$)
$\wedge$ implementsDirectly($CompositeGraphic, Graphic$))

Figure 21: Computed precondition for the round-trip transformation of interface variation

C JHotDraw transformation precondition

C.1 Chain of operations applied in the round-trip transformation of JHotDraw Composite

(Sequence of 944 operations)
PushDownNotReDefinedMethod(LineConnectionFigure, BezierFigure, [findFigureInside; setAttribute; contains]) ;
PushDownNotReDefinedMethod(SVGPath, AbstractCompositeFigure, [addNotify; removeNotify; findFigureInside; contains]) ;
PushDownNotReDefinedMethod(LabeledLineConnectionFigure, BezierFigure, [basicTransform; setAttribute; findFigureInside; contains]) ;
PushDownNotReDefinedMethod(DependencyFigure, LineConnectionFigure, [addNotify; basicTransform; setAttribute; findFigureInside; contains;]) ;
PushDownNotReDefinedMethod(NodeFigure, TextFigure, [addNotify; basicTransform; setAttribute; findFigureInside; contains]) ;
PushDownNotReDefinedMethod(GraphicalCompositeFigure, AbstractCompositeFigure, [findFigureInside]) ;
Createindirectioninsuperclass(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], basicTransform, [AffineTransform tx], Void, basicTransformTmpVC) ;
Createindirectioninsuperclass(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], contains, [Point2D.Double p], Boolean, containsTmpVC) ;
Createindirectioninsuperclass(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], setAttribute, [AttributeKey key; Object value], Void, setAttributeTmpVC) ;
Createindirectioninsuperclass(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], findFigureInside, [Point2D.Double p], Figure, findFigureInsideTmpVC) ;
Createindirectioninsuperclass(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], addNotify, [Drawing d], Void, addNotifyTmpVC) ;
Createindirectioninsuperclass(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], removeNotify, [Drawing d], Void, removeNotifyTmpVC) ;
introduceParmeterObject(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], basicTransformTmpVC, [AffineTransform tx], [AffineTransform tx], [tx], BasicTransformVisitor, v) ;
introduceParmeterObject(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], containsTmpVC, [Point2D.Double p], [Point2D.Double p], [p], ContainsVisitor, v) ;
introduceParmeterObject(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath;

DependencyFigure; LineConnectionFigure], setAttributeTmpVC, [AttributeKey key; Object value], [AttributeKey key; Object value], [key; value], SetAttributeVisitor, v) ;
introduceParmeterObject(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], findFigureInsideTmpVC, [Point2D.Double p], [Point2D.Double p], [p], FindFigureInsideVisitor, v) ;
introduceParmeterObject(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], addNotifyTmpVC, [Drawing d], [Drawing d], [d], AddNotifyVisitor, v) ;
introduceParmeterObject(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], removeNotifyTmpVC, [Drawing d], [Drawing d], [d], RemoveNotifyVisitor, v) ;
Movemethodwithdelegate(LabeledLineConnectionFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(AbstractCompositeFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(GraphicalCompositeFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(EllipseFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(DiamondFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(RectangleFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(RoundRectangleFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(TriangleFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(TextFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(BezierFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(TextAreaFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(NodeFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(SVGImage, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(SVGPath, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(DependencyFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(LineConnectionFigure, [], BasicTransformVisitor, basicTransformTmpVC, [BasicTransformVisitor], Void, visit) ;
Movemethodwithdelegate(LabeledLineConnectionFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor;], Boolean, visit) ;
Movemethodwithdelegate(AbstractCompositeFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(GraphicalCompositeFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(EllipseFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(DiamondFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(RectangleFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(RoundRectangleFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(TriangleFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(TextFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(BezierFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(TextAreaFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(NodeFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(SVGImage, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(SVGPath, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(DependencyFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(LineConnectionFigure, [], ContainsVisitor, containsTmpVC, [ContainsVisitor], Boolean, visit) ;
Movemethodwithdelegate(LabeledLineConnectionFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor;], Void, visit) ;
Movemethodwithdelegate(AbstractCompositeFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(GraphicalCompositeFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(EllipseFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(DiamondFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(RectangleFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(RoundRectangleFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(TriangleFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(TextFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(BezierFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(TextAreaFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(NodeFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(SVGImage, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(SVGPath, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(DependencyFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(LineConnectionFigure, [], SetAttributeVisitor, setAttributeTmpVC, [SetAttributeVisitor], Void, visit) ;
Movemethodwithdelegate(LabeledLineConnectionFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit)
;
Movemethodwithdelegate(AbstractCompositeFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(GraphicalCompositeFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(EllipseFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(DiamondFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(RectangleFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(RoundRectangleFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(TriangleFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(TextFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(BezierFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(TextAreaFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(NodeFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(SVGImage, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(SVGPath, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(DependencyFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;
Movemethodwithdelegate(LineConnectionFigure, [], FindFigureInsideVisitor, findFigureInsideTmpVC, [FindFigureInsideVisitor], Figure, visit) ;

Movemethodwithdelegate(LabeledLineConnectionFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(AbstractCompositeFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(GraphicalCompositeFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(EllipseFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(DiamondFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(RectangleFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(RoundRectangleFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(TriangleFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(TextFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(BezierFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(TextAreaFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(NodeFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(SVGImage, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(SVGPath, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(DependencyFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(LineConnectionFigure, [], AddNotifyVisitor, addNotifyTmpVC, [AddNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(LabeledLineConnectionFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(AbstractCompositeFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(GraphicalCompositeFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(EllipseFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(DiamondFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(RectangleFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(RoundRectangleFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(TriangleFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(TextFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(BezierFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(TextAreaFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(NodeFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(SVGImage, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(SVGPath, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(DependencyFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
Movemethodwithdelegate(LineConnectionFigure, [], RemoveNotifyVisitor, removeNotifyTmpVC, [RemoveNotifyVisitor], Void, visit) ;
ExtractSuperClassWithoutPullup([BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], Visitor) ;
PullupWithGenerics(Visitor, BasicTransformVisitor, [], visit, Void) ;
PullupWithGenerics(Visitor, ContainsVisitor, [], visit, Boolean) ;
PullupWithGenerics(Visitor, SetAttributeVisitor, [], visit, Void) ;
PullupWithGenerics(Visitor, FindFigureInsideVisitor, [], visit, Figure) ;
PullupWithGenerics(Visitor, AddNotifyVisitor, [], visit, Void) ;
PullupWithGenerics(Visitor, RemoveNotifyVisitor, [], visit, Void) ;
GeneraliseParameter(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], basicTransformTmpVC, v, BasicTransformVisitor, Visitor) ;
GeneraliseParameter(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], containsTmpVC, v, ContainsVisitor, Visitor) ;
GeneraliseParameter(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], setAttributeTmpVC, v, SetAttributeVisitor, Visitor) ;
GeneraliseParameter(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], findFigureInsideTmpVC, v, FindFigureInsideVisitor, Visitor) ;
GeneraliseParameter(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], addNotifyTmpVC, v, AddNotifyVisitor, Visitor) ;
GeneraliseParameter(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], removeNotifyTmpVC, v, RemoveNotifyVisitor, Visitor) ;
ReplaceMethodCodeDuplicatesInverters(LabeledLineConnectionFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(AbstractCompositeFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(GraphicalCompositeFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(EllipseFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(DiamondFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(RectangleFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(RoundRectangleFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(TriangleFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(TextFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;

ReplaceMethodCodeDuplicatesInverters(BezierFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(TextAreaFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(NodeFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(SVGImage, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(SVGPath, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(DependencyFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
ReplaceMethodCodeDuplicatesInverters(LineConnectionFigure, basicTransformTmpVC, [containsTmpVC; setAttributeTmpVC; findFigureInsideTmpVC; addNotifyTmpVC; removeNotifyTmpVC], Visitor, [Void; Boolean; Void; Figure; Void; Void]) ;
PullUpImplementation(LabeledLineConnectionFigure, [], containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(AbstractCompositeFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(GraphicalCompositeFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(EllipseFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(DiamondFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(RectangleFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(RoundRectangleFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TriangleFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TextFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(BezierFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TextAreaFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(NodeFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(SVGImage, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(SVGPath, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(DependencyFigure, containsTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(LineConnectionFigure, containsTmpVC, AbstractFigure) ;
PullUpImplementation(LabeledLineConnectionFigure, [], setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(AbstractCompositeFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(GraphicalCompositeFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(EllipseFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(DiamondFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(RectangleFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(RoundRectangleFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TriangleFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TextFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(BezierFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TextAreaFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(NodeFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(SVGImage, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(SVGPath, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(DependencyFigure, setAttributeTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(LineConnectionFigure, setAttributeTmpVC, AbstractFigure) ;
PullUpImplementation(LabeledLineConnectionFigure, [], findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(AbstractCompositeFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(GraphicalCompositeFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(EllipseFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(DiamondFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(RectangleFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(RoundRectangleFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TriangleFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TextFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(BezierFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TextAreaFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(NodeFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(SVGImage, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(SVGPath, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(DependencyFigure, findFigureInsideTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(LineConnectionFigure, findFigureInsideTmpVC, AbstractFigure) ;
PullUpImplementation(LabeledLineConnectionFigure, [], addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(AbstractCompositeFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(GraphicalCompositeFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(EllipseFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(DiamondFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(RectangleFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(RoundRectangleFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TriangleFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TextFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(BezierFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TextAreaFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(NodeFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(SVGImage, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(SVGPath, addNotifyTmpVC, AbstractFigure) ;

SafeDeleteDelegatorOverriding(DependencyFigure, addNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(LineConnectionFigure, addNotifyTmpVC, AbstractFigure) ;
PullUpImplementation(LabeledLineConnectionFigure, [], removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(AbstractCompositeFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(GraphicalCompositeFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(EllipseFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(DiamondFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(RectangleFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(RoundRectangleFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TriangleFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TextFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(BezierFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(TextAreaFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(NodeFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(SVGImage, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(SVGPath, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(DependencyFigure, removeNotifyTmpVC, AbstractFigure) ;
SafeDeleteDelegatorOverriding(LineConnectionFigure, removeNotifyTmpVC, AbstractFigure) ;
InlineAndDelete (AbstractFigure, containsTmpVC, [Visitor], contains, [setAttribute; findFigureInside; addNotify; removeNotify], [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure]) ;
InlineAndDelete (AbstractFigure, setAttributeTmpVC, [Visitor], setAttribute, [contains; findFigureInside; addNotify; removeNotify], [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure]) ;
InlineAndDelete (AbstractFigure, findFigureInsideTmpVC, [Visitor], findFigureInside, [contains; setAttribute; addNotify; removeNotify], [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure]) ;
InlineAndDelete (AbstractFigure, addNotifyTmpVC, [Visitor], addNotify, [contains; setAttribute; findFigureInside; removeNotify], [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure]) ;
InlineAndDelete (AbstractFigure, removeNotifyTmpVC, [Visitor], removeNotify, [contains; setAttribute; findFigureInside; addNotify], [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure]) ;
RenameinhierarchyNoOverloading(AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure], basicTransformTmpVC, [Visitor], accept) ;
DuplicateMethodInHierarchyGen (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [Void; Boolean; Void; Figure; Void; Void], [visit], [basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify], accept_BasicTransformVisitor_addspecializedMethod_tmp, [Visitor]) ;
SpecialiseParameter (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept_BasicTransformVisitor_addspecializedMethod_tmp, Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], BasicTransformVisitor) ;
RenameDelegatorWithOverloading(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept_BasicTransformVisitor_addspecializedMethod_tmp, BasicTransformVisitor, v, Visitor, accept) ;
DuplicateMethodInHierarchyGen (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [Void; Boolean; Void; Figure; Void; Void], [visit], [basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify], accept_ContainsVisitor_addspecializedMethod_tmp, [Visitor]) ;
SpecialiseParameter (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept_ContainsVisitor_addspecializedMethod_tmp, Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], ContainsVisitor) ;
RenameDelegatorWithOverloading(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept_ContainsVisitor_addspecializedMethod_tmp, ContainsVisitor, v, Visitor, accept) ;
DuplicateMethodInHierarchyGen (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [Void; Boolean; Void; Figure; Void; Void], [visit], [basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify], accept_SetAttributeVisitor_addspecializedMethod_tmp, [Visitor]) ;
SpecialiseParameter (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept_SetAttributeVisitor_addspecializedMethod_tmp, Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], SetAttributeVisitor) ;
RenameDelegatorWithOverloading(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept_SetAttributeVisitor_addspecializedMethod_tmp, SetAttributeVisitor, v, Visitor, accept) ;
DuplicateMethodInHierarchyGen (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [Void; Boolean; Void; Figure; Void; Void], [visit], [basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify], accept_FindFigureInsideVisitor_addspecializedMethod_tmp, [Visitor]) ;
SpecialiseParameter (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCom-

positeFigure; GraphicalCompositeFigure], accept_FindFigureInsideVisitor_addspecializedMethod_tmp, Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], FindFigureInsideVisitor) ;
RenameDelegatorWithOverloading(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept_FindFigureInsideVisitor_addspecializedMethod_tmp, FindFigureInsideVisitor, v, Visitor, accept) ;
DuplicateMethodInHierarchyGen (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [Void; Boolean; Void; Figure; Void; Void], [visit], [basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify], accept_AddNotifyVisitor_addspecializedMethod_tmp, [Visitor]) ;
SpecialiseParameter (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept_AddNotifyVisitor_addspecializedMethod_tmp, Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], AddNotifyVisitor) ;
RenameDelegatorWithOverloading(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept_AddNotifyVisitor_addspecializedMethod_tmp, AddNotifyVisitor, v, Visitor, accept) ;
DuplicateMethodInHierarchyGen (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [Void; Boolean; Void; Figure; Void; Void], [visit], [basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify], accept_RemoveNotifyVisitor_addspecializedMethod_tmp, [Visitor]) ;
SpecialiseParameter (AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept_RemoveNotifyVisitor_addspecializedMethod_tmp, Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], RemoveNotifyVisitor) ;
RenameDelegatorWithOverloading(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept_RemoveNotifyVisitor_addspecializedMethod_tmp, RemoveNotifyVisitor, v, Visitor, accept) ;
DeleteMethodInHierarchy(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [visit], Visitor) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [EllipseFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [DiamondFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [RectangleFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [RoundRectangleFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [TriangleFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [TextFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [BezierFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [TextAreaFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [NodeFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [SVGImage]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [SVGPath]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [DependencyFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [LineConnectionFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [LabeledLineConnectionFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [AbstractCompositeFigure]) ;
PushDownAll(Visitor, [BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor], visit, [GraphicalCompositeFigure]) ;
InlineMethod(EllipseFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(DiamondFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(RectangleFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(RoundRectangleFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(TriangleFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(TextFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(BezierFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(TextAreaFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(NodeFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(SVGImage, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(SVGPath, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(DependencyFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;

InlineMethod(LineConnectionFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(LabeledLineConnectionFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(AbstractCompositeFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(GraphicalCompositeFigure, visit, BasicTransformVisitor, accept, [AffineTransform tx]) ;
InlineMethod(EllipseFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(DiamondFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(RectangleFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(RoundRectangleFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(TriangleFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(TextFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(BezierFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(TextAreaFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(NodeFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(SVGImage, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(SVGPath, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(DependencyFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(LineConnectionFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(LabeledLineConnectionFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(AbstractCompositeFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(GraphicalCompositeFigure, visit, ContainsVisitor, accept, [Point2D.Double p]) ;
InlineMethod(EllipseFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(DiamondFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(RectangleFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(RoundRectangleFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(TriangleFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(TextFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(BezierFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(TextAreaFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(NodeFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(SVGImage, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(SVGPath, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(DependencyFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(LineConnectionFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(LabeledLineConnectionFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(AbstractCompositeFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(GraphicalCompositeFigure, visit, SetAttributeVisitor, accept, [AttributeKey key; Object value]) ;
InlineMethod(EllipseFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(DiamondFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(RectangleFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(RoundRectangleFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(TriangleFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(TextFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(BezierFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(TextAreaFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(NodeFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(SVGImage, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(SVGPath, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(DependencyFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(LineConnectionFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(LabeledLineConnectionFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(AbstractCompositeFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(GraphicalCompositeFigure, visit, FindFigureInsideVisitor, accept, [Point2D.Double p]) ;
InlineMethod(EllipseFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(DiamondFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(RectangleFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(RoundRectangleFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(TriangleFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(TextFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(BezierFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(TextAreaFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(NodeFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(SVGImage, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(SVGPath, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(DependencyFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(LineConnectionFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(LabeledLineConnectionFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(AbstractCompositeFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(GraphicalCompositeFigure, visit, AddNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(EllipseFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(DiamondFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(RectangleFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(RoundRectangleFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(TriangleFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(TextFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(BezierFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(TextAreaFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;

InlineMethod(NodeFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(SVGImage, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(SVGPath, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(DependencyFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(LineConnectionFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(LabeledLineConnectionFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(AbstractCompositeFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
InlineMethod(GraphicalCompositeFigure, visit, RemoveNotifyVisitor, accept, [Drawing d]) ;
RenameOverloadedMethodInHierarchy(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [BasicTransformVisitor], basicTransformTmpVC, [BasicTransformVisitor]) ;
RenameOverloadedMethodInHierarchy(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [ContainsVisitor], containsTmpVC, [ContainsVisitor]) ;
RenameOverloadedMethodInHierarchy(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [SetAttributeVisitor], setAttributeTmpVC, [SetAttributeVisitor]) ;
RenameOverloadedMethodInHierarchy(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [FindFigureInsideVisitor], findFigureInsideTmpVC, [FindFigureInsideVisitor]) ;
RenameOverloadedMethodInHierarchy(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [AddNotifyVisitor], addNotifyTmpVC, [AddNotifyVisitor]) ;
RenameOverloadedMethodInHierarchy(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], accept, [RemoveNotifyVisitor], removeNotifyTmpVC, [RemoveNotifyVisitor]) ;
Replace-method-duplication(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], basicTransform, basicTransformTmpVC, [AffineTransform tx]) ;
Replace-method-duplication(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], contains, containsTmpVC, [Point2D.Double p]) ;
Replace-method-duplication(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], setAttribute, setAttributeTmpVC, [AttributeKey key; Object value]) ;
Replace-method-duplication(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], findFigureInside, findFigureInsideTmpVC, [Point2D.Double p]) ;
Replace-method-duplication(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], addNotify, addNotifyTmpVC, [Drawing d]) ;
Replace-method-duplication(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], removeNotify, removeNotifyTmpVC, [Drawing d]) ;
PushDownImplementation(AbstractFigure, [], [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], basicTransform, [AffineTransform tx]) ;
PushDownImplementation(AbstractFigure, [], [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], contains, [Point2D.Double p]) ;
PushDownImplementation(AbstractFigure, [], [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], setAttribute, [AttributeKey key; Object value]) ;
PushDownImplementation(AbstractFigure, [], [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], findFigureInside, [Point2D.Double p]) ;
PushDownImplementation(AbstractFigure, [], [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], addNotify, [Drawing d]) ;
PushDownImplementation(AbstractFigure, [], [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], removeNotify, [Drawing d]) ;
PushDownAll(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], basicTransformTmpVC, [BasicTransformVisitor]) ;
PushDownAll(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], containsTmpVC, [ContainsVisitor]) ;
PushDownAll(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], setAttributeTmpVC, [SetAttributeVisitor]) ;
PushDownAll(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], findFigureInsideTmpVC, [FindFigureInsideVisitor]) ;
PushDownAll(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFig-

ure; GraphicalCompositeFigure], addNotifyTmpVC, [AddNotifyVisitor]) ;
PushDownAll(AbstractFigure, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure], removeNotifyTmpVC, [RemoveNotifyVisitor]) ;
InlineAndDelete (EllipseFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DiamondFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RectangleFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RoundRectangleFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TriangleFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (BezierFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextAreaFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (NodeFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGImage, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGPath, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DependencyFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LineConnectionFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LabeledLineConnectionFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (AbstractCompositeFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNoti-

fyVisitor]) ;
InlineAndDelete (GraphicalCompositeFigure, basicTransformTmpVC, [BasicTransformVisitor], basicTransform, [contains; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (EllipseFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DiamondFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RectangleFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RoundRectangleFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TriangleFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (BezierFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextAreaFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (NodeFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGImage, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGPath, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DependencyFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LineConnectionFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LabeledLineConnectionFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (AbstractCompositeFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (GraphicalCompositeFigure, containsTmpVC, [ContainsVisitor], contains, [basicTransform; setAttribute; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (EllipseFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DiamondFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFig-

ure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RectangleFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RoundRectangleFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TriangleFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (BezierFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextAreaFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (NodeFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGImage, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGPath, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DependencyFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LineConnectionFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LabeledLineConnectionFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (AbstractCompositeFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (GraphicalCompositeFigure, setAttributeTmpVC, [SetAttributeVisitor], setAttribute, [basicTransform; contains; findFigureInside; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (EllipseFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DiamondFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RectangleFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; Graphi-

calCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RoundRectangleFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TriangleFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (BezierFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextAreaFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (NodeFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGImage, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGPath, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DependencyFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LineConnectionFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LabeledLineConnectionFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (AbstractCompositeFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (GraphicalCompositeFigure, findFigureInsideTmpVC, [FindFigureInsideVisitor], findFigureInside, [basicTransform; contains; setAttribute; addNotify; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (EllipseFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DiamondFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RectangleFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; Graphi-

calCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RoundRectangleFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TriangleFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (BezierFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextAreaFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (NodeFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGImage, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGPath, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DependencyFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LineConnectionFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LabeledLineConnectionFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (AbstractCompositeFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (GraphicalCompositeFigure, addNotifyTmpVC, [AddNotifyVisitor], addNotify, [basicTransform; contains; setAttribute; findFigureInside; removeNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (EllipseFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DiamondFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RectangleFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (RoundRectangleFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TriangleFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureIn-

side; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (BezierFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (TextAreaFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (NodeFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGImage, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (SVGPath, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (DependencyFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LineConnectionFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (LabeledLineConnectionFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (AbstractCompositeFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineAndDelete (GraphicalCompositeFigure, removeNotifyTmpVC, [RemoveNotifyVisitor], removeNotify, [basicTransform; contains; setAttribute; findFigureInside; addNotify], [Visitor; AbstractFigure; EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor]) ;
InlineConstructor(EllipseFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(EllipseFigure, basicTransform, tx) ;
InlineAndDelete (EllipseFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(EllipseFigure, basicTransform, txvar) ;
InlineConstructor(DiamondFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(DiamondFigure, basicTransform, tx) ;
InlineAndDelete (DiamondFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(DiamondFigure, basicTransform, txvar) ;
InlineConstructor(RectangleFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(RectangleFigure, basicTransform, tx) ;
InlineAndDelete (RectangleFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(RectangleFigure, basicTransform, txvar) ;
InlineConstructor(RoundRectangleFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(RoundRectangleFigure, basicTransform, tx) ;
InlineAndDelete (RoundRectangleFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(RoundRectangleFigure, basicTransform, txvar) ;
InlineConstructor(TriangleFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(TriangleFigure, basicTransform, tx) ;
InlineAndDelete (TriangleFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(TriangleFigure, basicTransform, txvar) ;
InlineConstructor(TextFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(TextFigure, basicTransform, tx) ;
InlineAndDelete (TextFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(TextFigure, basicTransform, txvar) ;
InlineConstructor(BezierFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(BezierFigure, basicTransform, tx) ;

InlineAndDelete (BezierFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(BezierFigure, basicTransform, txvar) ;
InlineConstructor(TextAreaFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(TextAreaFigure, basicTransform, tx) ;
InlineAndDelete (TextAreaFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(TextAreaFigure, basicTransform, txvar) ;
InlineConstructor(NodeFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(NodeFigure, basicTransform, tx) ;
InlineAndDelete (NodeFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(NodeFigure, basicTransform, txvar) ;
InlineConstructor(SVGImage, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(SVGImage, basicTransform, tx) ;
InlineAndDelete (SVGImage, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(SVGImage, basicTransform, txvar) ;
InlineConstructor(SVGPath, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(SVGPath, basicTransform, tx) ;
InlineAndDelete (SVGPath, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(SVGPath, basicTransform, txvar) ;
InlineConstructor(DependencyFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(DependencyFigure, basicTransform, tx) ;
InlineAndDelete (DependencyFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(DependencyFigure, basicTransform, txvar) ;
InlineConstructor(LineConnectionFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(LineConnectionFigure, basicTransform, tx) ;
InlineAndDelete (LineConnectionFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(LineConnectionFigure, basicTransform, txvar) ;
InlineConstructor(LabeledLineConnectionFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(LabeledLineConnectionFigure, basicTransform, tx) ;
InlineAndDelete (LabeledLineConnectionFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(LabeledLineConnectionFigure, basicTransform, txvar) ;
InlineConstructor(AbstractCompositeFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(AbstractCompositeFigure, basicTransform, tx) ;
InlineAndDelete (AbstractCompositeFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(AbstractCompositeFigure, basicTransform, txvar) ;
InlineConstructor(GraphicalCompositeFigure, basicTransform, BasicTransformVisitor, [tx], [gettx]) ;
InlineLocalField(GraphicalCompositeFigure, basicTransform, tx) ;
InlineAndDelete (GraphicalCompositeFigure, gettx, [], basicTransform, [], []) ;
InlinelocalVariable(GraphicalCompositeFigure, basicTransform, txvar) ;
InlineConstructor(EllipseFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(EllipseFigure, contains, p) ;
InlineAndDelete (EllipseFigure, getp, [], contains, [], []) ;
InlinelocalVariable(EllipseFigure, contains, pvar) ;
InlineConstructor(DiamondFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(DiamondFigure, contains, p) ;
InlineAndDelete (DiamondFigure, getp, [], contains, [], []) ;
InlinelocalVariable(DiamondFigure, contains, pvar) ;
InlineConstructor(RectangleFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(RectangleFigure, contains, p) ;
InlineAndDelete (RectangleFigure, getp, [], contains, [], []) ;
InlinelocalVariable(RectangleFigure, contains, pvar) ;
InlineConstructor(RoundRectangleFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(RoundRectangleFigure, contains, p) ;
InlineAndDelete (RoundRectangleFigure, getp, [], contains, [], []) ;
InlinelocalVariable(RoundRectangleFigure, contains, pvar) ;
InlineConstructor(TriangleFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(TriangleFigure, contains, p) ;
InlineAndDelete (TriangleFigure, getp, [], contains, [], []) ;
InlinelocalVariable(TriangleFigure, contains, pvar) ;
InlineConstructor(TextFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(TextFigure, contains, p) ;
InlineAndDelete (TextFigure, getp, [], contains, [], []) ;
InlinelocalVariable(TextFigure, contains, pvar) ;
InlineConstructor(BezierFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(BezierFigure, contains, p) ;
InlineAndDelete (BezierFigure, getp, [], contains, [], []) ;
InlinelocalVariable(BezierFigure, contains, pvar) ;
InlineConstructor(TextAreaFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(TextAreaFigure, contains, p) ;
InlineAndDelete (TextAreaFigure, getp, [], contains, [], []) ;
InlinelocalVariable(TextAreaFigure, contains, pvar) ;
InlineConstructor(NodeFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(NodeFigure, contains, p) ;
InlineAndDelete (NodeFigure, getp, [], contains, [], []) ;
InlinelocalVariable(NodeFigure, contains, pvar) ;
InlineConstructor(SVGImage, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(SVGImage, contains, p) ;

InlineAndDelete (SVGImage, getp, [], contains, [], []) ;
InlinelocalVariable(SVGImage, contains, pvar) ;
InlineConstructor(SVGPath, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(SVGPath, contains, p) ;
InlineAndDelete (SVGPath, getp, [], contains, [], []) ;
InlinelocalVariable(SVGPath, contains, pvar) ;
InlineConstructor(DependencyFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(DependencyFigure, contains, p) ;
InlineAndDelete (DependencyFigure, getp, [], contains, [], []) ;
InlinelocalVariable(DependencyFigure, contains, pvar) ;
InlineConstructor(LineConnectionFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(LineConnectionFigure, contains, p) ;
InlineAndDelete (LineConnectionFigure, getp, [], contains, [], []) ;
InlinelocalVariable(LineConnectionFigure, contains, pvar) ;
InlineConstructor(LabeledLineConnectionFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(LabeledLineConnectionFigure, contains, p) ;
InlineAndDelete (LabeledLineConnectionFigure, getp, [], contains, [], []) ;
InlinelocalVariable(LabeledLineConnectionFigure, contains, pvar) ;
InlineConstructor(AbstractCompositeFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(AbstractCompositeFigure, contains, p) ;
InlineAndDelete (AbstractCompositeFigure, getp, [], contains, [], []) ;
InlinelocalVariable(AbstractCompositeFigure, contains, pvar) ;
InlineConstructor(GraphicalCompositeFigure, contains, ContainsVisitor, [p], [getp]) ;
InlineLocalField(GraphicalCompositeFigure, contains, p) ;
InlineAndDelete (GraphicalCompositeFigure, getp, [], contains, [], []) ;
InlinelocalVariable(GraphicalCompositeFigure, contains, pvar) ;
InlineConstructor(EllipseFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(EllipseFigure, setAttribute, key) ;
InlineLocalField(EllipseFigure, setAttribute, value) ;
InlineAndDelete (EllipseFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (EllipseFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(EllipseFigure, setAttribute, keyvar) ;
InlinelocalVariable(EllipseFigure, setAttribute, valuevar) ;
InlineConstructor(DiamondFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(DiamondFigure, setAttribute, key) ;
InlineLocalField(DiamondFigure, setAttribute, value) ;
InlineAndDelete (DiamondFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (DiamondFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(DiamondFigure, setAttribute, keyvar) ;
InlinelocalVariable(DiamondFigure, setAttribute, valuevar) ;
InlineConstructor(RectangleFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(RectangleFigure, setAttribute, key) ;
InlineLocalField(RectangleFigure, setAttribute, value) ;
InlineAndDelete (RectangleFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (RectangleFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(RectangleFigure, setAttribute, keyvar) ;
InlinelocalVariable(RectangleFigure, setAttribute, valuevar) ;
InlineConstructor(RoundRectangleFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(RoundRectangleFigure, setAttribute, key) ;
InlineLocalField(RoundRectangleFigure, setAttribute, value) ;
InlineAndDelete (RoundRectangleFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (RoundRectangleFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(RoundRectangleFigure, setAttribute, keyvar) ;
InlinelocalVariable(RoundRectangleFigure, setAttribute, valuevar) ;
InlineConstructor(TriangleFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(TriangleFigure, setAttribute, key) ;
InlineLocalField(TriangleFigure, setAttribute, value) ;
InlineAndDelete (TriangleFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (TriangleFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(TriangleFigure, setAttribute, keyvar) ;
InlinelocalVariable(TriangleFigure, setAttribute, valuevar) ;
InlineConstructor(TextFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(TextFigure, setAttribute, key) ;
InlineLocalField(TextFigure, setAttribute, value) ;
InlineAndDelete (TextFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (TextFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(TextFigure, setAttribute, keyvar) ;
InlinelocalVariable(TextFigure, setAttribute, valuevar) ;
InlineConstructor(BezierFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(BezierFigure, setAttribute, key) ;
InlineLocalField(BezierFigure, setAttribute, value) ;
InlineAndDelete (BezierFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (BezierFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(BezierFigure, setAttribute, keyvar) ;
InlinelocalVariable(BezierFigure, setAttribute, valuevar) ;
InlineConstructor(TextAreaFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;

InlineLocalField(TextAreaFigure, setAttribute, key) ;
InlineLocalField(TextAreaFigure, setAttribute, value) ;
InlineAndDelete (TextAreaFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (TextAreaFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(TextAreaFigure, setAttribute, keyvar) ;
InlinelocalVariable(TextAreaFigure, setAttribute, valuevar) ;
InlineConstructor(NodeFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(NodeFigure, setAttribute, key) ;
InlineLocalField(NodeFigure, setAttribute, value) ;
InlineAndDelete (NodeFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (NodeFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(NodeFigure, setAttribute, keyvar) ;
InlinelocalVariable(NodeFigure, setAttribute, valuevar) ;
InlineConstructor(SVGImage, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(SVGImage, setAttribute, key) ;
InlineLocalField(SVGImage, setAttribute, value) ;
InlineAndDelete (SVGImage, getkey, [], setAttribute, [], []) ;
InlineAndDelete (SVGImage, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(SVGImage, setAttribute, keyvar) ;
InlinelocalVariable(SVGImage, setAttribute, valuevar) ;
InlineConstructor(SVGPath, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(SVGPath, setAttribute, key) ;
InlineLocalField(SVGPath, setAttribute, value) ;
InlineAndDelete (SVGPath, getkey, [], setAttribute, [], []) ;
InlineAndDelete (SVGPath, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(SVGPath, setAttribute, keyvar) ;
InlinelocalVariable(SVGPath, setAttribute, valuevar) ;
InlineConstructor(DependencyFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(DependencyFigure, setAttribute, key) ;
InlineLocalField(DependencyFigure, setAttribute, value) ;
InlineAndDelete (DependencyFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (DependencyFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(DependencyFigure, setAttribute, keyvar) ;
InlinelocalVariable(DependencyFigure, setAttribute, valuevar) ;
InlineConstructor(LineConnectionFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(LineConnectionFigure, setAttribute, key) ;
InlineLocalField(LineConnectionFigure, setAttribute, value) ;
InlineAndDelete (LineConnectionFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (LineConnectionFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(LineConnectionFigure, setAttribute, keyvar) ;
InlinelocalVariable(LineConnectionFigure, setAttribute, valuevar) ;
InlineConstructor(LabeledLineConnectionFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(LabeledLineConnectionFigure, setAttribute, key) ;
InlineLocalField(LabeledLineConnectionFigure, setAttribute, value) ;
InlineAndDelete (LabeledLineConnectionFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (LabeledLineConnectionFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(LabeledLineConnectionFigure, setAttribute, keyvar) ;
InlinelocalVariable(LabeledLineConnectionFigure, setAttribute, valuevar) ;
InlineConstructor(AbstractCompositeFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(AbstractCompositeFigure, setAttribute, key) ;
InlineLocalField(AbstractCompositeFigure, setAttribute, value) ;
InlineAndDelete (AbstractCompositeFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (AbstractCompositeFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(AbstractCompositeFigure, setAttribute, keyvar) ;
InlinelocalVariable(AbstractCompositeFigure, setAttribute, valuevar) ;
InlineConstructor(GraphicalCompositeFigure, setAttribute, SetAttributeVisitor, [key; value], [getkey; getvalue]) ;
InlineLocalField(GraphicalCompositeFigure, setAttribute, key) ;
InlineLocalField(GraphicalCompositeFigure, setAttribute, value) ;
InlineAndDelete (GraphicalCompositeFigure, getkey, [], setAttribute, [], []) ;
InlineAndDelete (GraphicalCompositeFigure, getvalue, [], setAttribute, [], []) ;
InlinelocalVariable(GraphicalCompositeFigure, setAttribute, keyvar) ;
InlinelocalVariable(GraphicalCompositeFigure, setAttribute, valuevar) ;
InlineConstructor(EllipseFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(EllipseFigure, findFigureInside, p) ;
InlineAndDelete (EllipseFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(EllipseFigure, findFigureInside, pvar) ;
InlineConstructor(DiamondFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(DiamondFigure, findFigureInside, p) ;
InlineAndDelete (DiamondFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(DiamondFigure, findFigureInside, pvar) ;
InlineConstructor(RectangleFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(RectangleFigure, findFigureInside, p) ;
InlineAndDelete (RectangleFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(RectangleFigure, findFigureInside, pvar) ;
InlineConstructor(RoundRectangleFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(RoundRectangleFigure, findFigureInside, p) ;

InlineAndDelete (RoundRectangleFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(RoundRectangleFigure, findFigureInside, pvar) ;
InlineConstructor(TriangleFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(TriangleFigure, findFigureInside, p) ;
InlineAndDelete (TriangleFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(TriangleFigure, findFigureInside, pvar) ;
InlineConstructor(TextFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(TextFigure, findFigureInside, p) ;
InlineAndDelete (TextFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(TextFigure, findFigureInside, pvar) ;
InlineConstructor(BezierFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(BezierFigure, findFigureInside, p) ;
InlineAndDelete (BezierFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(BezierFigure, findFigureInside, pvar) ;
InlineConstructor(TextAreaFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(TextAreaFigure, findFigureInside, p) ;
InlineAndDelete (TextAreaFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(TextAreaFigure, findFigureInside, pvar) ;
InlineConstructor(NodeFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(NodeFigure, findFigureInside, p) ;
InlineAndDelete (NodeFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(NodeFigure, findFigureInside, pvar) ;
InlineConstructor(SVGImage, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(SVGImage, findFigureInside, p) ;
InlineAndDelete (SVGImage, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(SVGImage, findFigureInside, pvar) ;
InlineConstructor(SVGPath, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(SVGPath, findFigureInside, p) ;
InlineAndDelete (SVGPath, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(SVGPath, findFigureInside, pvar) ;
InlineConstructor(DependencyFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(DependencyFigure, findFigureInside, p) ;
InlineAndDelete (DependencyFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(DependencyFigure, findFigureInside, pvar) ;
InlineConstructor(LineConnectionFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(LineConnectionFigure, findFigureInside, p) ;
InlineAndDelete (LineConnectionFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(LineConnectionFigure, findFigureInside, pvar) ;
InlineConstructor(LabeledLineConnectionFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(LabeledLineConnectionFigure, findFigureInside, p) ;
InlineAndDelete (LabeledLineConnectionFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(LabeledLineConnectionFigure, findFigureInside, pvar) ;
InlineConstructor(AbstractCompositeFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(AbstractCompositeFigure, findFigureInside, p) ;
InlineAndDelete (AbstractCompositeFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(AbstractCompositeFigure, findFigureInside, pvar) ;
InlineConstructor(GraphicalCompositeFigure, findFigureInside, FindFigureInsideVisitor, [p], [getp]) ;
InlineLocalField(GraphicalCompositeFigure, findFigureInside, p) ;
InlineAndDelete (GraphicalCompositeFigure, getp, [], findFigureInside, [], []) ;
InlinelocalVariable(GraphicalCompositeFigure, findFigureInside, pvar) ;
InlineConstructor(EllipseFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(EllipseFigure, addNotify, d) ;
InlineAndDelete (EllipseFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(EllipseFigure, addNotify, dvar) ;
InlineConstructor(DiamondFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(DiamondFigure, addNotify, d) ;
InlineAndDelete (DiamondFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(DiamondFigure, addNotify, dvar) ;
InlineConstructor(RectangleFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(RectangleFigure, addNotify, d) ;
InlineAndDelete (RectangleFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(RectangleFigure, addNotify, dvar) ;
InlineConstructor(RoundRectangleFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(RoundRectangleFigure, addNotify, d) ;
InlineAndDelete (RoundRectangleFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(RoundRectangleFigure, addNotify, dvar) ;
InlineConstructor(TriangleFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(TriangleFigure, addNotify, d) ;
InlineAndDelete (TriangleFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(TriangleFigure, addNotify, dvar) ;
InlineConstructor(TextFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(TextFigure, addNotify, d) ;
InlineAndDelete (TextFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(TextFigure, addNotify, dvar) ;
InlineConstructor(BezierFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(BezierFigure, addNotify, d) ;

InlineAndDelete (BezierFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(BezierFigure, addNotify, dvar) ;
InlineConstructor(TextAreaFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(TextAreaFigure, addNotify, d) ;
InlineAndDelete (TextAreaFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(TextAreaFigure, addNotify, dvar) ;
InlineConstructor(NodeFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(NodeFigure, addNotify, d) ;
InlineAndDelete (NodeFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(NodeFigure, addNotify, dvar) ;
InlineConstructor(SVGImage, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(SVGImage, addNotify, d) ;
InlineAndDelete (SVGImage, getd, [], addNotify, [], []) ;
InlinelocalVariable(SVGImage, addNotify, dvar) ;
InlineConstructor(SVGPath, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(SVGPath, addNotify, d) ;
InlineAndDelete (SVGPath, getd, [], addNotify, [], []) ;
InlinelocalVariable(SVGPath, addNotify, dvar) ;
InlineConstructor(DependencyFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(DependencyFigure, addNotify, d) ;
InlineAndDelete (DependencyFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(DependencyFigure, addNotify, dvar) ;
InlineConstructor(LineConnectionFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(LineConnectionFigure, addNotify, d) ;
InlineAndDelete (LineConnectionFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(LineConnectionFigure, addNotify, dvar) ;
InlineConstructor(LabeledLineConnectionFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(LabeledLineConnectionFigure, addNotify, d) ;
InlineAndDelete (LabeledLineConnectionFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(LabeledLineConnectionFigure, addNotify, dvar) ;
InlineConstructor(AbstractCompositeFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(AbstractCompositeFigure, addNotify, d) ;
InlineAndDelete (AbstractCompositeFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(AbstractCompositeFigure, addNotify, dvar) ;
InlineConstructor(GraphicalCompositeFigure, addNotify, AddNotifyVisitor, [d], [getd]) ;
InlineLocalField(GraphicalCompositeFigure, addNotify, d) ;
InlineAndDelete (GraphicalCompositeFigure, getd, [], addNotify, [], []) ;
InlinelocalVariable(GraphicalCompositeFigure, addNotify, dvar) ;
InlineConstructor(EllipseFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(EllipseFigure, removeNotify, d) ;
InlineAndDelete (EllipseFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(EllipseFigure, removeNotify, dvar) ;
InlineConstructor(DiamondFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(DiamondFigure, removeNotify, d) ;
InlineAndDelete (DiamondFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(DiamondFigure, removeNotify, dvar) ;
InlineConstructor(RectangleFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(RectangleFigure, removeNotify, d) ;
InlineAndDelete (RectangleFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(RectangleFigure, removeNotify, dvar) ;
InlineConstructor(RoundRectangleFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(RoundRectangleFigure, removeNotify, d) ;
InlineAndDelete (RoundRectangleFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(RoundRectangleFigure, removeNotify, dvar) ;
InlineConstructor(TriangleFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(TriangleFigure, removeNotify, d) ;
InlineAndDelete (TriangleFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(TriangleFigure, removeNotify, dvar) ;
InlineConstructor(TextFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(TextFigure, removeNotify, d) ;
InlineAndDelete (TextFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(TextFigure, removeNotify, dvar) ;
InlineConstructor(BezierFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(BezierFigure, removeNotify, d) ;
InlineAndDelete (BezierFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(BezierFigure, removeNotify, dvar) ;
InlineConstructor(TextAreaFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(TextAreaFigure, removeNotify, d) ;
InlineAndDelete (TextAreaFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(TextAreaFigure, removeNotify, dvar) ;
InlineConstructor(NodeFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(NodeFigure, removeNotify, d) ;
InlineAndDelete (NodeFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(NodeFigure, removeNotify, dvar) ;
InlineConstructor(SVGImage, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(SVGImage, removeNotify, d) ;

InlineAndDelete (SVGImage, getd, [], removeNotify, [], []) ;
InlinelocalVariable(SVGImage, removeNotify, dvar) ;
InlineConstructor(SVGPath, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(SVGPath, removeNotify, d) ;
InlineAndDelete (SVGPath, getd, [], removeNotify, [], []) ;
InlinelocalVariable(SVGPath, removeNotify, dvar) ;
InlineConstructor(DependencyFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(DependencyFigure, removeNotify, d) ;
InlineAndDelete (DependencyFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(DependencyFigure, removeNotify, dvar) ;
InlineConstructor(LineConnectionFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(LineConnectionFigure, removeNotify, d) ;
InlineAndDelete (LineConnectionFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(LineConnectionFigure, removeNotify, dvar) ;
InlineConstructor(LabeledLineConnectionFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(LabeledLineConnectionFigure, removeNotify, d) ;
InlineAndDelete (LabeledLineConnectionFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(LabeledLineConnectionFigure, removeNotify, dvar) ;
InlineConstructor(AbstractCompositeFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(AbstractCompositeFigure, removeNotify, d) ;
InlineAndDelete (AbstractCompositeFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(AbstractCompositeFigure, removeNotify, dvar) ;
InlineConstructor(GraphicalCompositeFigure, removeNotify, RemoveNotifyVisitor, [d], [getd]) ;
InlineLocalField(GraphicalCompositeFigure, removeNotify, d) ;
InlineAndDelete (GraphicalCompositeFigure, getd, [], removeNotify, [], []) ;
InlinelocalVariable(GraphicalCompositeFigure, removeNotify, dvar) ;
DeleteClass(BasicTransformVisitor, Visitor, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor; Visitor; AbstractFigure;], [visit;], [accept; basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify])
;
DeleteClass(ContainsVisitor, Visitor, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor; Visitor; AbstractFigure], [visit], [accept; basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify]) ;
DeleteClass(SetAttributeVisitor, Visitor, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor; Visitor; AbstractFigure], [visit], [accept; basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify]) ;
DeleteClass(FindFigureInsideVisitor, Visitor, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor; Visitor; AbstractFigure], [visit], [accept; basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify]) ;
DeleteClass(AddNotifyVisitor, Visitor, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor; Visitor; AbstractFigure], [visit], [accept; basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify]) ;
DeleteClass(RemoveNotifyVisitor, Visitor, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor; Visitor; AbstractFigure], [visit], [accept; basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify]) ;
DeleteClass(Visitor, java.lang.Object, [EllipseFigure; DiamondFigure; RectangleFigure; RoundRectangleFigure; TriangleFigure; TextFigure; BezierFigure; TextAreaFigure; NodeFigure; SVGImage; SVGPath; DependencyFigure; LineConnectionFigure; LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; BasicTransformVisitor; ContainsVisitor; SetAttributeVisitor; FindFigureInsideVisitor; AddNotifyVisitor; RemoveNotifyVisitor; Visitor; AbstractFigure], [visit], [accept; basicTransform; contains; setAttribute; findFigureInside; addNotify; removeNotify]) ;
DeleteMethod(LineConnectionFigure, findFigureInside, [Point2D.Double p], BezierFigure, findFigureInside) ;
DeleteMethod(LineConnectionFigure, setAttribute, [AttributeKey key; Object value], BezierFigure, setAttribute) ;
DeleteMethod(LineConnectionFigure, contains, [Point2D.Double p], BezierFigure, contains) ;
DeleteMethod(SVGPath, addNotify, [Drawing d], AbstractCompositeFigure, addNotify) ;
DeleteMethod(SVGPath, removeNotify, [Drawing d], AbstractCompositeFigure, removeNotify) ;
DeleteMethod(SVGPath, findFigureInside, [Point2D.Double p], AbstractCompositeFigure, findFigureInside) ;
DeleteMethod(SVGPath, contains, [Point2D.Double p], AbstractCompositeFigure, contains) ;
DeleteMethod(LabeledLineConnectionFigure, basicTransform, [AffineTransform tx], BezierFigure, basicTransform) ;
DeleteMethod(LabeledLineConnectionFigure, setAttribute, [AttributeKey key; Object value], BezierFigure, setAttribute) ;
DeleteMethod(LabeledLineConnectionFigure, findFigureInside, [Point2D.Double p], BezierFigure, findFigureInside) ;
DeleteMethod(LabeledLineConnectionFigure, contains, [Point2D.Double p], BezierFigure, contains) ;
DeleteMethod(DependencyFigure, addNotify, [Drawing d], LineConnectionFigure, addNotify) ;
DeleteMethod(DependencyFigure, basicTransform, [AffineTransform tx], LineConnectionFigure, basicTransform) ;
DeleteMethod(DependencyFigure, setAttribute, [AttributeKey key; Object value], LineConnectionFigure, setAttribute) ;
DeleteMethod(DependencyFigure, findFigureInside, [Point2D.Double p], LineConnectionFigure, findFigureInside) ;
DeleteMethod(DependencyFigure, contains, [Point2D.Double p], LineConnectionFigure, contains) ;
DeleteMethod(NodeFigure, addNotify, [Drawing d], TextFigure, addNotify) ;
DeleteMethod(NodeFigure, basicTransform, [AffineTransform tx], TextFigure, basicTransform) ;
DeleteMethod(NodeFigure, setAttribute, [AttributeKey key; Object value], TextFigure, setAttribute) ;
DeleteMethod(NodeFigure, findFigureInside, [Point2D.Double p], TextFigure, findFigureInside) ;
DeleteMethod(NodeFigure, contains, [Point2D.Double p], TextFigure, contains) ;

DeleteMethod(GraphicalCompositeFigure, findFigureInside, [Point2D.Double p], AbstractCompositeFigure, findFigureInside)

C.2 Computed precondition for a round-trip transformation of JHotDraw

(Conjunction of 1852 propositions)
IsInheritedMethod($GraphicalCompositeFigure, findFigureInside$)
$\wedge$ IsInheritedMethod($NodeFigure, contains$)
$\wedge$ IsInheritedMethod($NodeFigure, findFigureInside$)
$\wedge$ IsInheritedMethod($NodeFigure, setAttribute$)
$\wedge$ IsInheritedMethod($NodeFigure, basicTransform$)
$\wedge$ IsInheritedMethod($NodeFigure, addNotify$)
$\wedge$ IsInheritedMethod($DependencyFigure, contains$)
$\wedge$ IsInheritedMethod($DependencyFigure, findFigureInside$)
$\wedge$ IsInheritedMethod($DependencyFigure, setAttribute$)
$\wedge$ IsInheritedMethod($DependencyFigure, basicTransform$)
$\wedge$ IsInheritedMethod($DependencyFigure, addNotify$)
$\wedge$ IsInheritedMethod($LabeledLineConnectionFigure, contains$)
$\wedge$ IsInheritedMethod($LabeledLineConnectionFigure, findFigureInside$)
$\wedge$ IsInheritedMethod($LabeledLineConnectionFigure, setAttribute$)
$\wedge$ IsInheritedMethod($LabeledLineConnectionFigure, basicTransform$)
$\wedge$ IsInheritedMethod($SVGPath, contains$)
$\wedge$ IsInheritedMethod($SVGPath, findFigureInside$)
$\wedge$ IsInheritedMethod($SVGPath, removeNotify$)
$\wedge$ IsInheritedMethod($SVGPath, addNotify$)
$\wedge$ IsInheritedMethod($LineConnectionFigure, contains$)
$\wedge$ IsInheritedMethod($LineConnectionFigure, setAttribute$)
$\wedge$ IsInheritedMethod($LineConnectionFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, EllipseFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, EllipseFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, EllipseFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, EllipseFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, EllipseFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, DiamondFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, DiamondFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, DiamondFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, DiamondFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, DiamondFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, RectangleFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, RectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, RectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, RectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, RectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, RoundRectangleFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, RoundRectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, RoundRectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, RoundRectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, RoundRectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TriangleFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TriangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TriangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TriangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TriangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TextFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TextFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TextFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TextFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TextFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, BezierFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, BezierFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, BezierFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, BezierFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, BezierFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TextAreaFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TextAreaFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TextAreaFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TextAreaFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, TextAreaFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, NodeFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, NodeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, NodeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, NodeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, NodeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, SVGImage, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, SVGImage, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, SVGImage, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, SVGImage, findFigureInside$)

$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, SVGImage, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, SVGPath, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, SVGPath, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, SVGPath, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, SVGPath, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, SVGPath, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, DependencyFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, DependencyFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, DependencyFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, DependencyFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, DependencyFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, LineConnectionFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, LineConnectionFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, LineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, LineConnectionFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, LineConnectionFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, LabeledLineConnectionFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, LabeledLineConnectionFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, LabeledLineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, LabeledLineConnectionFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, LabeledLineConnectionFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, AbstractCompositeFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, AbstractCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, AbstractCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, AbstractCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, AbstractCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, GraphicalCompositeFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, GraphicalCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, GraphicalCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, GraphicalCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, GraphicalCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, AbstractFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, AbstractFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, AbstractFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, AbstractFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, AbstractFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($RemoveNotifyVisitor, AbstractFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, EllipseFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, EllipseFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, EllipseFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, EllipseFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, EllipseFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, EllipseFigure, addNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DiamondFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DiamondFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DiamondFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DiamondFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DiamondFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DiamondFigure, addNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RectangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RectangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RectangleFigure, addNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RoundRectangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RoundRectangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RoundRectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RoundRectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RoundRectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, RoundRectangleFigure, addNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TriangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TriangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TriangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TriangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TriangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TriangleFigure, addNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextFigure, addNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, BezierFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, BezierFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, BezierFigure, contains$)

$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, BezierFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, BezierFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, BezierFigure, addNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextAreaFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextAreaFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextAreaFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextAreaFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextAreaFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, TextAreaFigure, addNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, NodeFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, NodeFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, NodeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, NodeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, NodeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, NodeFigure, addNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGImage, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGImage, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGImage, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGImage, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGImage, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGImage, addNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGPath, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGPath, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGPath, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGPath, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGPath, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, SVGPath, addNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DependencyFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DependencyFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DependencyFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DependencyFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DependencyFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, DependencyFigure, addNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LineConnectionFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LineConnectionFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LineConnectionFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LineConnectionFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LineConnectionFigure, addNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LabeledLineConnectionFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LabeledLineConnectionFigure, basicTransform$))
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LabeledLineConnectionFigure, containsTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LabeledLineConnectionFigure, contains$))
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LabeledLineConnectionFigure, setAttributeTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LabeledLineConnectionFigure, setAttribute$))
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LabeledLineConnectionFigure, findFigureInsideTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LabeledLineConnectionFigure, findFigureInside$))
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LabeledLineConnectionFigure, addNotifyTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, LabeledLineConnectionFigure, addNotify$))
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, AbstractCompositeFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, AbstractCompositeFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, AbstractCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, AbstractCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, AbstractCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, AbstractCompositeFigure, addNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, GraphicalCompositeFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, GraphicalCompositeFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, GraphicalCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, GraphicalCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, GraphicalCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($RemoveNotifyVisitor, GraphicalCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, EllipseFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, EllipseFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, EllipseFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, EllipseFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, EllipseFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, DiamondFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, DiamondFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, DiamondFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, DiamondFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, DiamondFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, RectangleFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, RectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, RectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, RectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, RectangleFigure, removeNotify$)

$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, RoundRectangleFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, RoundRectangleFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, RoundRectangleFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, RoundRectangleFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, RoundRectangleFigure, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TriangleFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TriangleFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TriangleFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TriangleFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TriangleFigure, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TextFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TextFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TextFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TextFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TextFigure, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, BezierFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, BezierFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, BezierFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, BezierFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, BezierFigure, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TextAreaFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TextAreaFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TextAreaFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TextAreaFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, TextAreaFigure, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, NodeFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, NodeFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, NodeFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, NodeFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, NodeFigure, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, SVGImage, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, SVGImage, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, SVGImage, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, SVGImage, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, SVGImage, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, SVGPath, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, SVGPath, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, SVGPath, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, SVGPath, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, SVGPath, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, DependencyFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, DependencyFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, DependencyFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, DependencyFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, DependencyFigure, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, LineConnectionFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, LineConnectionFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, LineConnectionFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, LineConnectionFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, LineConnectionFigure, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, LabeledLineConnectionFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, LabeledLineConnectionFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, LabeledLineConnectionFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, LabeledLineConnectionFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, LabeledLineConnectionFigure, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, AbstractCompositeFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, AbstractCompositeFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, AbstractCompositeFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, AbstractCompositeFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, AbstractCompositeFigure, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, GraphicalCompositeFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, GraphicalCompositeFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, GraphicalCompositeFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, GraphicalCompositeFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, GraphicalCompositeFigure, removeNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, AbstractFigure, basicTransform$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, AbstractFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, AbstractFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, AbstractFigure, findFigureInside$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, AbstractFigure, addNotify$)
$\wedge$ $\neg$IsUsedConstructorAsMethodParameter($AddNotifyVisitor, AbstractFigure, removeNotify$)
$\wedge$ $\neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, EllipseFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, EllipseFigure, basicTransform$))
$\wedge$ $\neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, EllipseFigure, contains$)
$\wedge$ $\neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, EllipseFigure, setAttribute$)
$\wedge$ $\neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, EllipseFigure, findFigureInside$)

$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, EllipseFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DiamondFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DiamondFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DiamondFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DiamondFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DiamondFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DiamondFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RectangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RectangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RectangleFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RoundRectangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RoundRectangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RoundRectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RoundRectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RoundRectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, RoundRectangleFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TriangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TriangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TriangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TriangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TriangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TriangleFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, BezierFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, BezierFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, BezierFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, BezierFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, BezierFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, BezierFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextAreaFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextAreaFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextAreaFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextAreaFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextAreaFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, TextAreaFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, NodeFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, NodeFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, NodeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, NodeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, NodeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, NodeFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGImage, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGImage, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGImage, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGImage, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGImage, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGImage, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGPath, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGPath, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGPath, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGPath, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGPath, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, SVGPath, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DependencyFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DependencyFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DependencyFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DependencyFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DependencyFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, DependencyFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, LineConnectionFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, LineConnectionFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, LineConnectionFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, LineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, LineConnectionFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, LineConnectionFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, LabeledLineConnectionFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, LabeledLineConnectionFigure, basicTransform$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor, LabeledLineConnectionFigure, containsTmpVC$)

$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $LabeledLineConnectionFigure$, $contains$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $LabeledLineConnectionFigure$, $setAttributeTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $LabeledLineConnectionFigure$, $setAttribute$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $LabeledLineConnectionFigure$, $findFigureInsideTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $LabeledLineConnectionFigure$, $findFigureInside$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $LabeledLineConnectionFigure$, $removeNotifyTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $LabeledLineConnectionFigure$, $removeNotify$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $AbstractCompositeFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $AbstractCompositeFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $AbstractCompositeFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $AbstractCompositeFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $AbstractCompositeFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $AbstractCompositeFigure$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $GraphicalCompositeFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $GraphicalCompositeFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $GraphicalCompositeFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $GraphicalCompositeFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $GraphicalCompositeFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($AddNotifyVisitor$, $GraphicalCompositeFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $EllipseFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $EllipseFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $EllipseFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $EllipseFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $EllipseFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $DiamondFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $DiamondFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $DiamondFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $DiamondFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $DiamondFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $RectangleFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $RectangleFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $RectangleFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $RectangleFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $RectangleFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $RoundRectangleFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $RoundRectangleFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $RoundRectangleFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $RoundRectangleFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $RoundRectangleFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TriangleFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TriangleFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TriangleFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TriangleFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TriangleFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TextFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TextFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TextFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TextFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TextFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $BezierFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $BezierFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $BezierFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $BezierFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $BezierFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TextAreaFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TextAreaFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TextAreaFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TextAreaFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $TextAreaFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $NodeFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $NodeFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $NodeFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $NodeFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $NodeFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $SVGImage$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $SVGImage$, $contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $SVGImage$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $SVGImage$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $SVGImage$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $SVGPath$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $SVGPath$, $contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $SVGPath$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $SVGPath$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $SVGPath$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $DependencyFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor$, $DependencyFigure$, $contains$)

$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, DependencyFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, DependencyFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, DependencyFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, LineConnectionFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, LineConnectionFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, LineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, LineConnectionFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, LineConnectionFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, LabeledLineConnectionFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, LabeledLineConnectionFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, LabeledLineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, LabeledLineConnectionFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, LabeledLineConnectionFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, AbstractCompositeFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, AbstractCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, AbstractCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, AbstractCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, AbstractCompositeFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, GraphicalCompositeFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, GraphicalCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, GraphicalCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, GraphicalCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, GraphicalCompositeFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, AbstractFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, AbstractFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, AbstractFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, AbstractFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, AbstractFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($FindFigureInsideVisitor, AbstractFigure, removeNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, EllipseFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, EllipseFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, EllipseFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, EllipseFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, EllipseFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, EllipseFigure, removeNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DiamondFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DiamondFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DiamondFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DiamondFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DiamondFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DiamondFigure, removeNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RectangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RectangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RectangleFigure, removeNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RoundRectangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RoundRectangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RoundRectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RoundRectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RoundRectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, RoundRectangleFigure, removeNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TriangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TriangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TriangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TriangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TriangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TriangleFigure, removeNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextFigure, removeNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, BezierFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, BezierFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, BezierFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, BezierFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, BezierFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, BezierFigure, removeNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextAreaFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextAreaFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextAreaFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextAreaFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextAreaFigure, addNotify$)

$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, TextAreaFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, NodeFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, NodeFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, NodeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, NodeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, NodeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, NodeFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGImage, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGImage, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGImage, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGImage, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGImage, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGImage, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGPath, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGPath, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGPath, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGPath, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGPath, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, SVGPath, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DependencyFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DependencyFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DependencyFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DependencyFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DependencyFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, DependencyFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LineConnectionFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LineConnectionFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LineConnectionFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LineConnectionFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LineConnectionFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LabeledLineConnectionFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LabeledLineConnectionFigure, basicTransform$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LabeledLineConnectionFigure, containsTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LabeledLineConnectionFigure, contains$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LabeledLineConnectionFigure, setAttributeTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LabeledLineConnectionFigure, setAttribute$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LabeledLineConnectionFigure, addNotifyTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LabeledLineConnectionFigure, addNotify$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LabeledLineConnectionFigure, removeNotifyTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, LabeledLineConnectionFigure, removeNotify$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, AbstractCompositeFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, AbstractCompositeFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, AbstractCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, AbstractCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, AbstractCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, AbstractCompositeFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, GraphicalCompositeFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, GraphicalCompositeFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, GraphicalCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, GraphicalCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, GraphicalCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($FindFigureInsideVisitor, GraphicalCompositeFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, EllipseFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, EllipseFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, EllipseFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, EllipseFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, EllipseFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, DiamondFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, DiamondFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, DiamondFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, DiamondFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, DiamondFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, RectangleFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, RectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, RectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, RectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, RectangleFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, RoundRectangleFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, RoundRectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, RoundRectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, RoundRectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, RoundRectangleFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TriangleFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TriangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TriangleFigure, findFigureInside$)

$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TriangleFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TriangleFigure, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TextFigure, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TextFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TextFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TextFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TextFigure, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, BezierFigure, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, BezierFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, BezierFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, BezierFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, BezierFigure, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TextAreaFigure, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TextAreaFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TextAreaFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TextAreaFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, TextAreaFigure, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, NodeFigure, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, NodeFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, NodeFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, NodeFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, NodeFigure, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, SVGImage, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, SVGImage, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, SVGImage, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, SVGImage, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, SVGImage, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, SVGPath, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, SVGPath, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, SVGPath, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, SVGPath, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, SVGPath, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, DependencyFigure, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, DependencyFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, DependencyFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, DependencyFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, DependencyFigure, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, LineConnectionFigure, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, LineConnectionFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, LineConnectionFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, LineConnectionFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, LineConnectionFigure, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, LabeledLineConnectionFigure, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, LabeledLineConnectionFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, LabeledLineConnectionFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, LabeledLineConnectionFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, LabeledLineConnectionFigure, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, AbstractCompositeFigure, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, AbstractCompositeFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, AbstractCompositeFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, AbstractCompositeFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, AbstractCompositeFigure, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, GraphicalCompositeFigure, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, GraphicalCompositeFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, GraphicalCompositeFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, GraphicalCompositeFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, GraphicalCompositeFigure, removeNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, AbstractFigure, basicTransform$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, AbstractFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, AbstractFigure, setAttribute$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, AbstractFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, AbstractFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsMethodParameter($SetAttributeVisitor, AbstractFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, EllipseFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, EllipseFigure, basicTransform$))
$\wedge \neg$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, EllipseFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, EllipseFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, EllipseFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, EllipseFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, DiamondFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, DiamondFigure, basicTransform$))
$\wedge \neg$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, DiamondFigure, contains$)
$\wedge \neg$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, DiamondFigure, findFigureInside$)
$\wedge \neg$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, DiamondFigure, addNotify$)
$\wedge \neg$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, DiamondFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor, RectangleFigure, basicTransformTmpVC$)

$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $RectangleFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $RectangleFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $RectangleFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $RectangleFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $RectangleFigure$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $RoundRectangleFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $RoundRectangleFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $RoundRectangleFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $RoundRectangleFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $RoundRectangleFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $RoundRectangleFigure$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TriangleFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TriangleFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TriangleFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TriangleFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TriangleFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TriangleFigure$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextFigure$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $BezierFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $BezierFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $BezierFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $BezierFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $BezierFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $BezierFigure$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextAreaFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextAreaFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextAreaFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextAreaFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextAreaFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $TextAreaFigure$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $NodeFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $NodeFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $NodeFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $NodeFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $NodeFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $NodeFigure$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGImage$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGImage$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGImage$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGImage$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGImage$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGImage$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGPath$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGPath$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGPath$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGPath$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGPath$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $SVGPath$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $DependencyFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $DependencyFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $DependencyFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $DependencyFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $DependencyFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $DependencyFigure$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LineConnectionFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LineConnectionFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LineConnectionFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LineConnectionFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LineConnectionFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LineConnectionFigure$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LabeledLineConnectionFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LabeledLineConnectionFigure$, $basicTransform$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LabeledLineConnectionFigure$, $containsTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LabeledLineConnectionFigure$, $contains$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LabeledLineConnectionFigure$, $findFigureInsideTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LabeledLineConnectionFigure$, $findFigureInside$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LabeledLineConnectionFigure$, $addNotifyTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LabeledLineConnectionFigure$, $addNotify$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LabeledLineConnectionFigure$, $removeNotifyTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $LabeledLineConnectionFigure$, $removeNotify$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $AbstractCompositeFigure$, $basicTransformTmpVC$)

$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $AbstractCompositeFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $AbstractCompositeFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $AbstractCompositeFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $AbstractCompositeFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $AbstractCompositeFigure$, $removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $GraphicalCompositeFigure$, $basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $GraphicalCompositeFigure$, $basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $GraphicalCompositeFigure$, $contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $GraphicalCompositeFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $GraphicalCompositeFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($SetAttributeVisitor$, $GraphicalCompositeFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $EllipseFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $EllipseFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $EllipseFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $EllipseFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $EllipseFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $DiamondFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $DiamondFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $DiamondFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $DiamondFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $DiamondFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $RectangleFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $RectangleFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $RectangleFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $RectangleFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $RectangleFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $RoundRectangleFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $RoundRectangleFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $RoundRectangleFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $RoundRectangleFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $RoundRectangleFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TriangleFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TriangleFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TriangleFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TriangleFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TriangleFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TextFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TextFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TextFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TextFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TextFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $BezierFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $BezierFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $BezierFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $BezierFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $BezierFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TextAreaFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TextAreaFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TextAreaFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TextAreaFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $TextAreaFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $NodeFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $NodeFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $NodeFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $NodeFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $NodeFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $SVGImage$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $SVGImage$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $SVGImage$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $SVGImage$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $SVGImage$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $SVGPath$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $SVGPath$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $SVGPath$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $SVGPath$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $SVGPath$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $DependencyFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $DependencyFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $DependencyFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $DependencyFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $DependencyFigure$, $removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $LineConnectionFigure$, $basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $LineConnectionFigure$, $setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $LineConnectionFigure$, $findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $LineConnectionFigure$, $addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor$, $LineConnectionFigure$, $removeNotify$)

$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, LabeledLineConnectionFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, LabeledLineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, LabeledLineConnectionFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, LabeledLineConnectionFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, LabeledLineConnectionFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, AbstractCompositeFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, AbstractCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, AbstractCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, AbstractCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, AbstractCompositeFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, GraphicalCompositeFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, GraphicalCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, GraphicalCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, GraphicalCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, GraphicalCompositeFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, AbstractFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, AbstractFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, AbstractFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, AbstractFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, AbstractFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($ContainsVisitor, AbstractFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, EllipseFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, EllipseFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, EllipseFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, EllipseFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, EllipseFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, EllipseFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, DiamondFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, DiamondFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, DiamondFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, DiamondFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, DiamondFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, DiamondFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, RectangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, RectangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, RectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, RectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, RectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, RectangleFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, RoundRectangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, RoundRectangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, RoundRectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, RoundRectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, RoundRectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, RoundRectangleFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, TriangleFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, TriangleFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TriangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TriangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TriangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TriangleFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, BezierFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, BezierFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, BezierFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, BezierFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, BezierFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, BezierFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextAreaFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextAreaFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextAreaFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextAreaFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextAreaFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, TextAreaFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, NodeFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, NodeFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, NodeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, NodeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, NodeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, NodeFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGImage, basicTransformTmpVC$)

$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGImage, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGImage, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGImage, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGImage, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGImage, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGPath, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGPath, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGPath, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGPath, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGPath, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, SVGPath, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, DependencyFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, DependencyFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, DependencyFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, DependencyFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, DependencyFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, DependencyFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, LineConnectionFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, LineConnectionFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, LineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, LineConnectionFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, LineConnectionFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, LineConnectionFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, LabeledLineConnectionFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, LabeledLineConnectionFigure, basicTransform$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, LabeledLineConnectionFigure, setAttributeTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, LabeledLineConnectionFigure, setAttribute$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, LabeledLineConnectionFigure, findFigureInsideTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, LabeledLineConnectionFigure, findFigureInside$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, LabeledLineConnectionFigure, addNotifyTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, LabeledLineConnectionFigure, addNotify$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, LabeledLineConnectionFigure, removeNotifyTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, LabeledLineConnectionFigure, removeNotify$))
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, AbstractCompositeFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, AbstractCompositeFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, AbstractCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, AbstractCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, AbstractCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, AbstractCompositeFigure, removeNotify$)
$\wedge \neg$(IsUsedConstructorAsObjectReceiver($ContainsVisitor, GraphicalCompositeFigure, basicTransformTmpVC$)
$\vee$ IsUsedConstructorAsObjectReceiver($ContainsVisitor, GraphicalCompositeFigure, basicTransform$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, GraphicalCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, GraphicalCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, GraphicalCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($ContainsVisitor, GraphicalCompositeFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, EllipseFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, EllipseFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, EllipseFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, EllipseFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, EllipseFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, DiamondFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, DiamondFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, DiamondFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, DiamondFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, DiamondFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, RectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, RectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, RectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, RectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, RectangleFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, RoundRectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, RoundRectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, RoundRectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, RoundRectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, RoundRectangleFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TriangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TriangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TriangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TriangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TriangleFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TextFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TextFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TextFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TextFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TextFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, BezierFigure, contains$)

$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, BezierFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, BezierFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, BezierFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, BezierFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TextAreaFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TextAreaFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TextAreaFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TextAreaFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, TextAreaFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, NodeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, NodeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, NodeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, NodeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, NodeFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, SVGImage, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, SVGImage, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, SVGImage, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, SVGImage, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, SVGImage, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, SVGPath, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, SVGPath, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, SVGPath, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, SVGPath, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, SVGPath, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, DependencyFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, DependencyFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, DependencyFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, DependencyFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, DependencyFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, LineConnectionFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, LineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, LineConnectionFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, LineConnectionFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, LineConnectionFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, LabeledLineConnectionFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, LabeledLineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, LabeledLineConnectionFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, LabeledLineConnectionFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, LabeledLineConnectionFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, AbstractCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, AbstractCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, AbstractCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, AbstractCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, AbstractCompositeFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, GraphicalCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, GraphicalCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, GraphicalCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, GraphicalCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, GraphicalCompositeFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, AbstractFigure, basicTransform$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, AbstractFigure, contains$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, AbstractFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, AbstractFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, AbstractFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsMethodParameter($BasicTransformVisitor, AbstractFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, EllipseFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, EllipseFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, EllipseFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, EllipseFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, EllipseFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, DiamondFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, DiamondFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, DiamondFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, DiamondFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, DiamondFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, RectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, RectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, RectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, RectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, RectangleFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, RoundRectangleFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, RoundRectangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, RoundRectangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, RoundRectangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, RoundRectangleFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TriangleFigure, contains$)

$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TriangleFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TriangleFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TriangleFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TriangleFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TextFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TextFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TextFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TextFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TextFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, BezierFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, BezierFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, BezierFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, BezierFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, BezierFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TextAreaFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TextAreaFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TextAreaFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TextAreaFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, TextAreaFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, NodeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, NodeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, NodeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, NodeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, NodeFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, SVGImage, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, SVGImage, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, SVGImage, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, SVGImage, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, SVGImage, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, SVGPath, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, SVGPath, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, SVGPath, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, SVGPath, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, SVGPath, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, DependencyFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, DependencyFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, DependencyFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, DependencyFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, DependencyFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LineConnectionFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LineConnectionFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LineConnectionFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LineConnectionFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LineConnectionFigure, removeNotify$)
$\wedge \neg($IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LabeledLineConnectionFigure, containsTmpVC$)$
$\quad \vee$ IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LabeledLineConnectionFigure, contains$))
$\wedge \neg($IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LabeledLineConnectionFigure, setAttributeTmpVC$)$
$\quad \vee$ IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LabeledLineConnectionFigure, setAttribute$))
$\wedge \neg($IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LabeledLineConnectionFigure, findFigureInsideTmpVC$)$
$\quad \vee$ IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LabeledLineConnectionFigure, findFigureInside$))
$\wedge \neg($IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LabeledLineConnectionFigure, addNotifyTmpVC$)$
$\quad \vee$ IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LabeledLineConnectionFigure, addNotify$))
$\wedge \neg($IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LabeledLineConnectionFigure, removeNotifyTmpVC$)$
$\quad \vee$ IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, LabeledLineConnectionFigure, removeNotify$))
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, AbstractCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, AbstractCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, AbstractCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, AbstractCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, AbstractCompositeFigure, removeNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, GraphicalCompositeFigure, contains$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, GraphicalCompositeFigure, setAttribute$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, GraphicalCompositeFigure, findFigureInside$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, GraphicalCompositeFigure, addNotify$)
$\wedge \neg$IsUsedConstructorAsObjectReceiver($BasicTransformVisitor, GraphicalCompositeFigure, removeNotify$)
$\wedge$ ExistsType($AbstractCompositeFigure$)
$\wedge$ ExistsType($SVGImage$)
$\wedge$ ExistsType($TextAreaFigure$)
$\wedge$ ExistsType($BezierFigure$)
$\wedge$ ExistsType($TextFigure$)
$\wedge$ ExistsType($TriangleFigure$)
$\wedge$ ExistsType($RoundRectangleFigure$)
$\wedge$ ExistsType($RectangleFigure$)
$\wedge$ ExistsType($DiamondFigure$)
$\wedge$ ExistsType($EllipseFigure$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, removeNotifyTmpVC, LabeledLineConnectionFigure, basicTransform$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, removeNotifyTmpVC, LabeledLineConnectionFigure, contains$)

$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, removeNotifyTmpVC, LabeledLineConnectionFigure, setAttribute$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, removeNotifyTmpVC, LabeledLineConnectionFigure, findFigureInside$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, removeNotifyTmpVC, LabeledLineConnectionFigure, addNotify$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, addNotifyTmpVC, LabeledLineConnectionFigure, basicTransform$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, addNotifyTmpVC, LabeledLineConnectionFigure, contains$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, addNotifyTmpVC, LabeledLineConnectionFigure, setAttribute$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, addNotifyTmpVC, LabeledLineConnectionFigure, findFigureInside$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, addNotifyTmpVC, LabeledLineConnectionFigure, removeNotify$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, findFigureInsideTmpVC, LabeledLineConnectionFigure, basicTransform$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, findFigureInsideTmpVC, LabeledLineConnectionFigure, contains$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, findFigureInsideTmpVC, LabeledLineConnectionFigure, setAttribute$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, findFigureInsideTmpVC, LabeledLineConnectionFigure, addNotify$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, findFigureInsideTmpVC, LabeledLineConnectionFigure, removeNotify$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, setAttributeTmpVC, LabeledLineConnectionFigure, basicTransform$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, setAttributeTmpVC, LabeledLineConnectionFigure, contains$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, setAttributeTmpVC, LabeledLineConnectionFigure, findFigureInside$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, setAttributeTmpVC, LabeledLineConnectionFigure, addNotify$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, setAttributeTmpVC, LabeledLineConnectionFigure, removeNotify$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, containsTmpVC, LabeledLineConnectionFigure, basicTransform$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, containsTmpVC, LabeledLineConnectionFigure, setAttribute$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, containsTmpVC, LabeledLineConnectionFigure, findFigureInside$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, containsTmpVC, LabeledLineConnectionFigure, addNotify$)
$\wedge \neg$ExistsMethodInvocation($LabeledLineConnectionFigure, containsTmpVC, LabeledLineConnectionFigure, removeNotify$)
$\wedge \neg$IsUsedMethod($AbstractFigure, accept\_RemoveNotifyVisitor\_addspecializedMethod\_tmp, [RemoveNotifyVisitor]$)
$\wedge \neg$IsUsedMethod($AbstractFigure, accept\_AddNotifyVisitor\_addspecializedMethod\_tmp, [AddNotifyVisitor]$)
$\wedge \neg$IsUsedMethod($AbstractFigure, accept\_FindFigureInsideVisitor\_addspecializedMethod\_tmp, [FindFigureInsideVisitor]$)
$\wedge \neg$IsUsedMethod($AbstractFigure, accept\_SetAttributeVisitor\_addspecializedMethod\_tmp, [SetAttributeVisitor]$)
$\wedge \neg$IsUsedMethod($AbstractFigure, accept\_ContainsVisitor\_addspecializedMethod\_tmp, [ContainsVisitor]$)
$\wedge \neg$IsUsedMethod($AbstractFigure, accept\_BasicTransformVisitor\_addspecializedMethod\_tmp, [BasicTransformVisitor]$)
$\wedge$AllSubclasses($AbstractFigure$, [$EllipseFigure$; $DiamondFigure$; $RectangleFigure$; $RoundRectangleFigure$; $TriangleFigure$; $TextFigure$; $BezierFi$
$\wedge \neg$ExistsMethodDefinition($AbstractFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($LabeledLineConnectionFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($AbstractCompositeFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($GraphicalCompositeFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($EllipseFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($DiamondFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($RectangleFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($RoundRectangleFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($TriangleFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($TextFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($BezierFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($TextAreaFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($NodeFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($SVGImage, accept$)
$\wedge \neg$ExistsMethodDefinition($SVGPath, accept$)
$\wedge \neg$ExistsMethodDefinition($DependencyFigure, accept$)
$\wedge \neg$ExistsMethodDefinition($LineConnectionFigure, accept$)
$\wedge \neg$IsInheritedMethod($AbstractFigure, accept$)
$\wedge \neg$ExistsMethodInvocation($AbstractFigure, removeNotifyTmpVC, AbstractFigure, contains$)
$\wedge \neg$ExistsMethodInvocation($AbstractFigure, removeNotifyTmpVC, AbstractFigure, setAttribute$)
$\wedge \neg$ExistsMethodInvocation($AbstractFigure, removeNotifyTmpVC, AbstractFigure, findFigureInside$)
$\wedge \neg$ExistsMethodInvocation($AbstractFigure, removeNotifyTmpVC, AbstractFigure, addNotify$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, LabeledLineConnectionFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, AbstractCompositeFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, GraphicalCompositeFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, EllipseFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, DiamondFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, RectangleFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, RoundRectangleFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, TriangleFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, TextFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, BezierFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, TextAreaFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, NodeFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, SVGImage$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, SVGPath$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, DependencyFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, removeNotifyTmpVC, LineConnectionFigure$)
$\wedge \neg$ExistsMethodInvocation($AbstractFigure, addNotifyTmpVC, AbstractFigure, contains$)
$\wedge \neg$ExistsMethodInvocation($AbstractFigure, addNotifyTmpVC, AbstractFigure, setAttribute$)
$\wedge \neg$ExistsMethodInvocation($AbstractFigure, addNotifyTmpVC, AbstractFigure, findFigureInside$)
$\wedge \neg$ExistsMethodInvocation($AbstractFigure, addNotifyTmpVC, AbstractFigure, removeNotify$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, LabeledLineConnectionFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, AbstractCompositeFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, GraphicalCompositeFigure$)
$\wedge \neg$IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, EllipseFigure$)

$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, DiamondFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, RectangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, RoundRectangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, TriangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, TextFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, BezierFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, TextAreaFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, NodeFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, SVGImage$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, SVGPath$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, DependencyFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, addNotifyTmpVC, LineConnectionFigure$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, findFigureInsideTmpVC, AbstractFigure, contains$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, findFigureInsideTmpVC, AbstractFigure, setAttribute$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, findFigureInsideTmpVC, AbstractFigure, addNotify$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, findFigureInsideTmpVC, AbstractFigure, removeNotify$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, LabeledLineConnectionFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, AbstractCompositeFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, GraphicalCompositeFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, EllipseFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, DiamondFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, RectangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, RoundRectangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, TriangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, TextFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, BezierFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, TextAreaFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, NodeFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, SVGImage$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, SVGPath$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, DependencyFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, findFigureInsideTmpVC, LineConnectionFigure$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, setAttributeTmpVC, AbstractFigure, contains$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, setAttributeTmpVC, AbstractFigure, findFigureInside$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, setAttributeTmpVC, AbstractFigure, addNotify$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, setAttributeTmpVC, AbstractFigure, removeNotify$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, LabeledLineConnectionFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, AbstractCompositeFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, GraphicalCompositeFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, EllipseFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, DiamondFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, RectangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, RoundRectangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, TriangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, TextFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, BezierFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, TextAreaFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, NodeFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, SVGImage$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, SVGPath$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, DependencyFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, setAttributeTmpVC, LineConnectionFigure$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, containsTmpVC, AbstractFigure, setAttribute$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, containsTmpVC, AbstractFigure, findFigureInside$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, containsTmpVC, AbstractFigure, addNotify$)
$\wedge \neg$ ExistsMethodInvocation($AbstractFigure, containsTmpVC, AbstractFigure, removeNotify$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, LabeledLineConnectionFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, AbstractCompositeFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, GraphicalCompositeFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, EllipseFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, DiamondFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, RectangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, RoundRectangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, TriangleFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, TextFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, BezierFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, TextAreaFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, NodeFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, SVGImage$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, SVGPath$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, DependencyFigure$)
$\wedge \neg$ IsUsedMethodIn($AbstractFigure, containsTmpVC, LineConnectionFigure$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LineConnectionFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DependencyFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGImage, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($NodeFigure, removeNotify, this$)

$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextAreaFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($BezierFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TriangleFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RoundRectangleFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RectangleFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DiamondFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($EllipseFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($GraphicalCompositeFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC($LabeledLineConnectionFigure, removeNotify, this, AbstractFigure$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LabeledLineConnectionFigure, removeNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGImage, addNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextAreaFigure, addNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($BezierFigure, addNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TriangleFigure, addNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RoundRectangleFigure, addNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RectangleFigure, addNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DiamondFigure, addNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($EllipseFigure, addNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($GraphicalCompositeFigure, addNotify, this$)
$\wedge$ AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC($LabeledLineConnectionFigure, addNotify, this, AbstractFigure$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LabeledLineConnectionFigure, addNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGImage, findFigureInside, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextAreaFigure, findFigureInside, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TriangleFigure, findFigureInside, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RoundRectangleFigure, findFigureInside, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RectangleFigure, findFigureInside, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DiamondFigure, findFigureInside, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($EllipseFigure, findFigureInside, this$)
$\wedge$ AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC($LabeledLineConnectionFigure, findFigureInside, this, AbstractFigure$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGPath, setAttribute, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGImage, setAttribute, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextAreaFigure, setAttribute, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TriangleFigure, setAttribute, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RoundRectangleFigure, setAttribute, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RectangleFigure, setAttribute, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DiamondFigure, setAttribute, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($EllipseFigure, setAttribute, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($GraphicalCompositeFigure, setAttribute, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractCompositeFigure, setAttribute, this$)
$\wedge$ AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC($LabeledLineConnectionFigure, setAttribute, this, AbstractFigure$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGImage, contains, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextAreaFigure, contains, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TriangleFigure, contains, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RoundRectangleFigure, contains, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RectangleFigure, contains, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DiamondFigure, contains, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($EllipseFigure, contains, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($GraphicalCompositeFigure, contains, this$)
$\wedge$ AllInvokedMethodsOnObjectOInBodyOfMAreDeclaredInC($LabeledLineConnectionFigure, contains, this, AbstractFigure$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LabeledLineConnectionFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractCompositeFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($GraphicalCompositeFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($EllipseFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DiamondFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RectangleFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RoundRectangleFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TriangleFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($BezierFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextAreaFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($NodeFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGImage, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGPath, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DependencyFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LineConnectionFigure, removeNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LabeledLineConnectionFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractCompositeFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($GraphicalCompositeFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($EllipseFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DiamondFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RectangleFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RoundRectangleFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TriangleFigure, addNotify, v$)

$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($BezierFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextAreaFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($NodeFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGImage, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGPath, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DependencyFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LineConnectionFigure, addNotify, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LabeledLineConnectionFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractCompositeFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($GraphicalCompositeFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($EllipseFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DiamondFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RectangleFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RoundRectangleFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TriangleFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($BezierFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextAreaFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($NodeFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGImage, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGPath, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DependencyFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LineConnectionFigure, findFigureInside, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LabeledLineConnectionFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractCompositeFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($GraphicalCompositeFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($EllipseFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DiamondFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RectangleFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RoundRectangleFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TriangleFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($BezierFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextAreaFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($NodeFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGImage, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGPath, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DependencyFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LineConnectionFigure, setAttribute, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LabeledLineConnectionFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractCompositeFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($GraphicalCompositeFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($EllipseFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DiamondFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RectangleFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RoundRectangleFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TriangleFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($BezierFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextAreaFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($NodeFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGImage, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGPath, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DependencyFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LineConnectionFigure, contains, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LabeledLineConnectionFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($AbstractCompositeFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($GraphicalCompositeFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($EllipseFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DiamondFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RectangleFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($RoundRectangleFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TriangleFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($BezierFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextAreaFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($NodeFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGImage, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($SVGPath, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($DependencyFigure, basicTransform, v$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LineConnectionFigure, basicTransform, v$)

$\wedge\ \neg$IsPrivate($RemoveNotifyVisitor, visit$)
$\wedge\ \neg$HasParameterType($RemoveNotifyVisitor, Void$)
$\wedge\ \neg$IsPrivate($AddNotifyVisitor, visit$)
$\wedge\ \neg$HasParameterType($AddNotifyVisitor, Void$)
$\wedge\ \neg$IsPrimitiveType($Figure$)
$\wedge\ \neg$IsPrivate($FindFigureInsideVisitor, visit$)
$\wedge\ \neg$HasParameterType($FindFigureInsideVisitor, Figure$)
$\wedge\ \neg$IsPrivate($SetAttributeVisitor, visit$)
$\wedge\ \neg$HasParameterType($SetAttributeVisitor, Void$)
$\wedge\ \neg$IsPrimitiveType($Boolean$)
$\wedge\ \neg$IsPrivate($ContainsVisitor, visit$)
$\wedge\ \neg$HasParameterType($ContainsVisitor, Boolean$)
$\wedge\ \neg$ExistsAbstractMethod($Visitor, visit$)
$\wedge\ \neg$IsPrimitiveType($Void$)
$\wedge\ \neg$IsPrivate($BasicTransformVisitor, visit$)
$\wedge\ \neg$HasParameterType($BasicTransformVisitor, Void$)
$\wedge\ \neg$ExistsType($Visitor$)
$\wedge$ ExistsClass($SVGImage$)
$\wedge$ ExistsClass($TextAreaFigure$)
$\wedge$ ExistsClass($BezierFigure$)
$\wedge$ ExistsClass($TextFigure$)
$\wedge$ ExistsClass($TriangleFigure$)
$\wedge$ ExistsClass($RoundRectangleFigure$)
$\wedge$ ExistsClass($RectangleFigure$)
$\wedge$ ExistsClass($DiamondFigure$)
$\wedge$ ExistsClass($EllipseFigure$)
$\wedge$ ExistsClass($AbstractCompositeFigure$)
$\wedge\ \neg$ExistsType($RemoveNotifyVisitor$)
$\wedge$ BoundVariableInMethodBody($LabeledLineConnectionFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($GraphicalCompositeFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($EllipseFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($DiamondFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($RectangleFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($RoundRectangleFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($TriangleFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($TextFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($BezierFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($TextAreaFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($NodeFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($SVGImage, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($AbstractCompositeFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($DependencyFigure, removeNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($LineConnectionFigure, removeNotify, Drawingd$)
$\wedge\ \neg$ExistsType($AddNotifyVisitor$)
$\wedge$ BoundVariableInMethodBody($LabeledLineConnectionFigure, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($GraphicalCompositeFigure, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($EllipseFigure, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($DiamondFigure, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($RectangleFigure, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($RoundRectangleFigure, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($TriangleFigure, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($BezierFigure, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($TextAreaFigure, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($TextFigure, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($SVGImage, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($AbstractCompositeFigure, addNotify, Drawingd$)
$\wedge$ BoundVariableInMethodBody($LineConnectionFigure, addNotify, Drawingd$)
$\wedge\ \neg$ExistsType($FindFigureInsideVisitor$)
$\wedge$ BoundVariableInMethodBody($EllipseFigure, findFigureInside, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($DiamondFigure, findFigureInside, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($RectangleFigure, findFigureInside, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($RoundRectangleFigure, findFigureInside, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($TriangleFigure, findFigureInside, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($TextAreaFigure, findFigureInside, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($TextFigure, findFigureInside, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($SVGImage, findFigureInside, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($AbstractCompositeFigure, findFigureInside, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($BezierFigure, findFigureInside, Point2D.Doublep$)
$\wedge\ \neg$ExistsType($SetAttributeVisitor$)
$\wedge$ BoundVariableInMethodBody($AbstractCompositeFigure, setAttribute, AttributeKeykey$)
$\wedge$ BoundVariableInMethodBody($AbstractCompositeFigure, setAttribute, Objectvalue$)
$\wedge$ BoundVariableInMethodBody($GraphicalCompositeFigure, setAttribute, AttributeKeykey$)
$\wedge$ BoundVariableInMethodBody($GraphicalCompositeFigure, setAttribute, Objectvalue$)
$\wedge$ BoundVariableInMethodBody($EllipseFigure, setAttribute, AttributeKeykey$)
$\wedge$ BoundVariableInMethodBody($EllipseFigure, setAttribute, Objectvalue$)
$\wedge$ BoundVariableInMethodBody($DiamondFigure, setAttribute, AttributeKeykey$)

$\wedge$ BoundVariableInMethodBody($DiamondFigure, setAttribute, Objectvalue$)
$\wedge$ BoundVariableInMethodBody($RectangleFigure, setAttribute, AttributeKeykey$)
$\wedge$ BoundVariableInMethodBody($RectangleFigure, setAttribute, Objectvalue$)
$\wedge$ BoundVariableInMethodBody($RoundRectangleFigure, setAttribute, AttributeKeykey$)
$\wedge$ BoundVariableInMethodBody($RoundRectangleFigure, setAttribute, Objectvalue$)
$\wedge$ BoundVariableInMethodBody($TriangleFigure, setAttribute, AttributeKeykey$)
$\wedge$ BoundVariableInMethodBody($TriangleFigure, setAttribute, Objectvalue$)
$\wedge$ BoundVariableInMethodBody($TextAreaFigure, setAttribute, AttributeKeykey$)
$\wedge$ BoundVariableInMethodBody($TextAreaFigure, setAttribute, Objectvalue$)
$\wedge$ BoundVariableInMethodBody($TextFigure, setAttribute, AttributeKeykey$)
$\wedge$ BoundVariableInMethodBody($TextFigure, setAttribute, Objectvalue$)
$\wedge$ BoundVariableInMethodBody($SVGImage, setAttribute, AttributeKeykey$)
$\wedge$ BoundVariableInMethodBody($SVGImage, setAttribute, Objectvalue$)
$\wedge$ BoundVariableInMethodBody($SVGPath, setAttribute, AttributeKeykey$)
$\wedge$ BoundVariableInMethodBody($SVGPath, setAttribute, Objectvalue$)
$\wedge$ BoundVariableInMethodBody($BezierFigure, setAttribute, AttributeKeykey$)
$\wedge$ BoundVariableInMethodBody($BezierFigure, setAttribute, Objectvalue$)
$\wedge \neg$ExistsType($ContainsVisitor$)
$\wedge$ BoundVariableInMethodBody($GraphicalCompositeFigure, contains, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($EllipseFigure, contains, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($DiamondFigure, contains, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($RectangleFigure, contains, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($RoundRectangleFigure, contains, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($TriangleFigure, contains, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($TextAreaFigure, contains, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($TextFigure, contains, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($SVGImage, contains, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($AbstractCompositeFigure, contains, Point2D.Doublep$)
$\wedge$ BoundVariableInMethodBody($BezierFigure, contains, Point2D.Doublep$)
$\wedge$ ExistsType($AbstractFigure$)
$\wedge \neg$ExistsType($BasicTransformVisitor$)
$\wedge$ BoundVariableInMethodBody($AbstractCompositeFigure, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($GraphicalCompositeFigure, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($EllipseFigure, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($DiamondFigure, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($RectangleFigure, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($RoundRectangleFigure, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($TriangleFigure, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($BezierFigure, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($TextAreaFigure, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($TextFigure, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($SVGImage, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($SVGPath, basicTransform, AffineTransformtx$)
$\wedge$ BoundVariableInMethodBody($LineConnectionFigure, basicTransform, AffineTransformtx$)
$\wedge$ ExistsMethodDefinitionWithParams($AbstractFigure, removeNotify, [Drawingd]$)
$\wedge$ ExistsAbstractMethod($AbstractFigure, removeNotify$)
$\wedge \neg$IsInheritedMethod($AbstractFigure, removeNotifyTmpVC$)
$\wedge \neg$IsInheritedMethodWithParams($AbstractFigure, removeNotifyTmpVC, [Drawingd]$)
$\wedge \neg$ExistsMethodDefinitionWithParams($AbstractFigure, removeNotifyTmpVC, [Drawingd]$)
$\wedge$ HasReturnType($AbstractFigure, removeNotify, Void$)
$\wedge$ ExistsMethodDefinition($AbstractFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($LabeledLineConnectionFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($GraphicalCompositeFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($EllipseFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($DiamondFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($RectangleFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($RoundRectangleFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($TriangleFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($TextFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($BezierFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($TextAreaFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($NodeFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($SVGImage, removeNotify$)
$\wedge$ ExistsMethodDefinition($DependencyFigure, removeNotify$)
$\wedge$ ExistsMethodDefinition($LineConnectionFigure, removeNotify$)
$\wedge \neg$ExistsMethodDefinition($AbstractFigure, removeNotifyTmpVC$)
$\wedge \neg$ExistsMethodDefinition($LabeledLineConnectionFigure, removeNotifyTmpVC$)
$\wedge \neg$ExistsMethodDefinition($AbstractCompositeFigure, removeNotifyTmpVC$)
$\wedge \neg$ExistsMethodDefinition($GraphicalCompositeFigure, removeNotifyTmpVC$)
$\wedge \neg$ExistsMethodDefinition($EllipseFigure, removeNotifyTmpVC$)
$\wedge \neg$ExistsMethodDefinition($DiamondFigure, removeNotifyTmpVC$)
$\wedge \neg$ExistsMethodDefinition($RectangleFigure, removeNotifyTmpVC$)
$\wedge \neg$ExistsMethodDefinition($RoundRectangleFigure, removeNotifyTmpVC$)
$\wedge \neg$ExistsMethodDefinition($TriangleFigure, removeNotifyTmpVC$)
$\wedge \neg$ExistsMethodDefinition($TextFigure, removeNotifyTmpVC$)
$\wedge \neg$ExistsMethodDefinition($BezierFigure, removeNotifyTmpVC$)

$\wedge$ $\neg$ExistsMethodDefinition($TextAreaFigure, removeNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($NodeFigure, removeNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($SVGImage, removeNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($SVGPath, removeNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($DependencyFigure, removeNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($LineConnectionFigure, removeNotifyTmpVC$)
$\wedge$ ExistsMethodDefinitionWithParams($AbstractFigure, addNotify, [Drawingd]$)
$\wedge$ ExistsAbstractMethod($AbstractFigure, addNotify$)
$\wedge$ $\neg$IsInheritedMethod($AbstractFigure, addNotifyTmpVC$)
$\wedge$ $\neg$IsInheritedMethodWithParams($AbstractFigure, addNotifyTmpVC, [Drawingd]$)
$\wedge$ $\neg$ExistsMethodDefinitionWithParams($AbstractFigure, addNotifyTmpVC, [Drawingd]$)
$\wedge$ HasReturnType($AbstractFigure, addNotify, Void$)
$\wedge$ ExistsMethodDefinition($AbstractFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($LabeledLineConnectionFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($GraphicalCompositeFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($EllipseFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($DiamondFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($RectangleFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($RoundRectangleFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($TriangleFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($BezierFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($TextAreaFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($SVGImage, addNotify$)
$\wedge$ $\neg$ExistsMethodDefinition($AbstractFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($LabeledLineConnectionFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($AbstractCompositeFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($GraphicalCompositeFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($EllipseFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($DiamondFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($RectangleFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($RoundRectangleFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TriangleFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TextFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($BezierFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TextAreaFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($NodeFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($SVGImage, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($SVGPath, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($DependencyFigure, addNotifyTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($LineConnectionFigure, addNotifyTmpVC$)
$\wedge$ ExistsMethodDefinitionWithParams($AbstractFigure, findFigureInside, [Point2D.Doublep]$)
$\wedge$ ExistsAbstractMethod($AbstractFigure, findFigureInside$)
$\wedge$ $\neg$IsInheritedMethod($AbstractFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$IsInheritedMethodWithParams($AbstractFigure, findFigureInsideTmpVC, [Point2D.Doublep]$)
$\wedge$ $\neg$ExistsMethodDefinitionWithParams($AbstractFigure, findFigureInsideTmpVC, [Point2D.Doublep]$)
$\wedge$ HasReturnType($AbstractFigure, findFigureInside, Figure$)
$\wedge$ ExistsMethodDefinition($AbstractFigure, findFigureInside$)
$\wedge$ ExistsMethodDefinition($EllipseFigure, findFigureInside$)
$\wedge$ ExistsMethodDefinition($DiamondFigure, findFigureInside$)
$\wedge$ ExistsMethodDefinition($RectangleFigure, findFigureInside$)
$\wedge$ ExistsMethodDefinition($RoundRectangleFigure, findFigureInside$)
$\wedge$ ExistsMethodDefinition($TriangleFigure, findFigureInside$)
$\wedge$ ExistsMethodDefinition($TextAreaFigure, findFigureInside$)
$\wedge$ ExistsMethodDefinition($SVGImage, findFigureInside$)
$\wedge$ $\neg$ExistsMethodDefinition($AbstractFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($LabeledLineConnectionFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($AbstractCompositeFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($GraphicalCompositeFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($EllipseFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($DiamondFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($RectangleFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($RoundRectangleFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TriangleFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TextFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($BezierFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TextAreaFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($NodeFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($SVGImage, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($SVGPath, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($DependencyFigure, findFigureInsideTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($LineConnectionFigure, findFigureInsideTmpVC$)
$\wedge$ ExistsMethodDefinitionWithParams($AbstractFigure, setAttribute, [AttributeKeykey; Objectvalue]$)
$\wedge$ ExistsAbstractMethod($AbstractFigure, setAttribute$)
$\wedge$ $\neg$IsInheritedMethod($AbstractFigure, setAttributeTmpVC$)
$\wedge$ $\neg$IsInheritedMethodWithParams($AbstractFigure, setAttributeTmpVC, [AttributeKeykey; Objectvalue]$)
$\wedge$ $\neg$ExistsMethodDefinitionWithParams($AbstractFigure, setAttributeTmpVC, [AttributeKeykey; Objectvalue]$)

$\wedge$ HasReturnType($AbstractFigure, setAttribute, Void$)
$\wedge$ ExistsMethodDefinition($AbstractFigure, setAttribute$)
$\wedge$ ExistsMethodDefinition($AbstractCompositeFigure, setAttribute$)
$\wedge$ ExistsMethodDefinition($GraphicalCompositeFigure, setAttribute$)
$\wedge$ ExistsMethodDefinition($EllipseFigure, setAttribute$)
$\wedge$ ExistsMethodDefinition($DiamondFigure, setAttribute$)
$\wedge$ ExistsMethodDefinition($RectangleFigure, setAttribute$)
$\wedge$ ExistsMethodDefinition($RoundRectangleFigure, setAttribute$)
$\wedge$ ExistsMethodDefinition($TriangleFigure, setAttribute$)
$\wedge$ ExistsMethodDefinition($TextAreaFigure, setAttribute$)
$\wedge$ ExistsMethodDefinition($SVGImage, setAttribute$)
$\wedge$ ExistsMethodDefinition($SVGPath, setAttribute$)
$\wedge$ $\neg$ExistsMethodDefinition($AbstractFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($LabeledLineConnectionFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($AbstractCompositeFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($GraphicalCompositeFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($EllipseFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($DiamondFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($RectangleFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($RoundRectangleFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TriangleFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TextFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($BezierFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TextAreaFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($NodeFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($SVGImage, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($SVGPath, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($DependencyFigure, setAttributeTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($LineConnectionFigure, setAttributeTmpVC$)
$\wedge$ ExistsMethodDefinitionWithParams($AbstractFigure, contains, [Point2D.Doublep]$)
$\wedge$ ExistsAbstractMethod($AbstractFigure, contains$)
$\wedge$ $\neg$IsInheritedMethod($AbstractFigure, containsTmpVC$)
$\wedge$ $\neg$IsInheritedMethodWithParams($AbstractFigure, containsTmpVC, [Point2D.Doublep]$)
$\wedge$ $\neg$ExistsMethodDefinitionWithParams($AbstractFigure, containsTmpVC, [Point2D.Doublep]$)
$\wedge$ HasReturnType($AbstractFigure, contains, Boolean$)
$\wedge$ ExistsMethodDefinition($AbstractFigure, contains$)
$\wedge$ ExistsMethodDefinition($GraphicalCompositeFigure, contains$)
$\wedge$ ExistsMethodDefinition($EllipseFigure, contains$)
$\wedge$ ExistsMethodDefinition($DiamondFigure, contains$)
$\wedge$ ExistsMethodDefinition($RectangleFigure, contains$)
$\wedge$ ExistsMethodDefinition($RoundRectangleFigure, contains$)
$\wedge$ ExistsMethodDefinition($TriangleFigure, contains$)
$\wedge$ ExistsMethodDefinition($TextAreaFigure, contains$)
$\wedge$ ExistsMethodDefinition($SVGImage, contains$)
$\wedge$ $\neg$ExistsMethodDefinition($AbstractFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($LabeledLineConnectionFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($AbstractCompositeFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($GraphicalCompositeFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($EllipseFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($DiamondFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($RectangleFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($RoundRectangleFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TriangleFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TextFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($BezierFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($TextAreaFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($NodeFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($SVGImage, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($SVGPath, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($DependencyFigure, containsTmpVC$)
$\wedge$ $\neg$ExistsMethodDefinition($LineConnectionFigure, containsTmpVC$)
$\wedge$ ExistsClass($AbstractFigure$)
$\wedge$ IsAbstractClass($AbstractFigure$)
$\wedge$ ExistsMethodDefinitionWithParams($AbstractFigure, basicTransform, [AffineTransformtx]$)
$\wedge$ ExistsAbstractMethod($AbstractFigure, basicTransform$)
$\wedge$ $\neg$IsInheritedMethod($AbstractFigure, basicTransformTmpVC$)
$\wedge$ $\neg$IsInheritedMethodWithParams($AbstractFigure, basicTransformTmpVC, [AffineTransformtx]$)
$\wedge$ $\neg$ExistsMethodDefinitionWithParams($AbstractFigure, basicTransformTmpVC, [AffineTransformtx]$)
$\wedge$AllSubclasses($AbstractFigure, [LabeledLineConnectionFigure; AbstractCompositeFigure; GraphicalCompositeFigure; EllipseFigure; DiamondF$
$\wedge$ HasReturnType($AbstractFigure, basicTransform, Void$)
$\wedge$ $\neg$IsPrivate($AbstractFigure, basicTransform$)
$\wedge$ $\neg$IsPrivate($LabeledLineConnectionFigure, basicTransform$)
$\wedge$ $\neg$IsPrivate($AbstractCompositeFigure, basicTransform$)
$\wedge$ $\neg$IsPrivate($GraphicalCompositeFigure, basicTransform$)
$\wedge$ $\neg$IsPrivate($EllipseFigure, basicTransform$)
$\wedge$ $\neg$IsPrivate($DiamondFigure, basicTransform$)

$\wedge \neg$ IsPrivate($RectangleFigure, basicTransform$)
$\wedge \neg$ IsPrivate($RoundRectangleFigure, basicTransform$)
$\wedge \neg$ IsPrivate($TriangleFigure, basicTransform$)
$\wedge \neg$ IsPrivate($TextFigure, basicTransform$)
$\wedge \neg$ IsPrivate($BezierFigure, basicTransform$)
$\wedge \neg$ IsPrivate($TextAreaFigure, basicTransform$)
$\wedge \neg$ IsPrivate($NodeFigure, basicTransform$)
$\wedge \neg$ IsPrivate($SVGImage, basicTransform$)
$\wedge \neg$ IsPrivate($SVGPath, basicTransform$)
$\wedge \neg$ IsPrivate($DependencyFigure, basicTransform$)
$\wedge \neg$ IsPrivate($LineConnectionFigure, basicTransform$)
$\wedge$ ExistsMethodDefinition($AbstractFigure, basicTransform$)
$\wedge$ ExistsMethodDefinition($AbstractCompositeFigure, basicTransform$)
$\wedge$ ExistsMethodDefinition($GraphicalCompositeFigure, basicTransform$)
$\wedge$ ExistsMethodDefinition($EllipseFigure, basicTransform$)
$\wedge$ ExistsMethodDefinition($DiamondFigure, basicTransform$)
$\wedge$ ExistsMethodDefinition($RectangleFigure, basicTransform$)
$\wedge$ ExistsMethodDefinition($RoundRectangleFigure, basicTransform$)
$\wedge$ ExistsMethodDefinition($TriangleFigure, basicTransform$)
$\wedge$ ExistsMethodDefinition($TextAreaFigure, basicTransform$)
$\wedge$ ExistsMethodDefinition($SVGImage, basicTransform$)
$\wedge$ ExistsMethodDefinition($SVGPath, basicTransform$)
$\wedge \neg$ ExistsMethodDefinition($AbstractFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($LabeledLineConnectionFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($AbstractCompositeFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($GraphicalCompositeFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($EllipseFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($DiamondFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($RectangleFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($RoundRectangleFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($TriangleFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($TextFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($BezierFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($TextAreaFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($NodeFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($SVGImage, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($SVGPath, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($DependencyFigure, basicTransformTmpVC$)
$\wedge \neg$ ExistsMethodDefinition($LineConnectionFigure, basicTransformTmpVC$)
$\wedge$ ExistsType($GraphicalCompositeFigure$)
$\wedge$ ExistsClass($GraphicalCompositeFigure$)
$\wedge$ IsSubType($GraphicalCompositeFigure, AbstractCompositeFigure$)
$\wedge \neg$ ExistsMethodDefinition($GraphicalCompositeFigure, findFigureInside$)
$\wedge$ ExistsType($NodeFigure$)
$\wedge$ ExistsClass($NodeFigure$)
$\wedge$ IsSubType($NodeFigure, TextFigure$)
$\wedge \neg$ ExistsMethodDefinition($NodeFigure, addNotify$)
$\wedge \neg$ ExistsMethodDefinition($NodeFigure, basicTransform$)
$\wedge \neg$ ExistsMethodDefinition($NodeFigure, setAttribute$)
$\wedge \neg$ ExistsMethodDefinition($NodeFigure, findFigureInside$)
$\wedge \neg$ ExistsMethodDefinition($NodeFigure, contains$)
$\wedge$ ExistsMethodDefinition($TextFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($TextFigure, basicTransform$)
$\wedge$ ExistsMethodDefinition($TextFigure, setAttribute$)
$\wedge$ ExistsMethodDefinition($TextFigure, findFigureInside$)
$\wedge$ ExistsMethodDefinition($TextFigure, contains$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, addNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, basicTransform, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, setAttribute, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, findFigureInside, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($TextFigure, contains, this$)
$\wedge$ ExistsType($DependencyFigure$)
$\wedge$ ExistsClass($DependencyFigure$)
$\wedge$ IsSubType($DependencyFigure, LineConnectionFigure$)
$\wedge \neg$ ExistsMethodDefinition($DependencyFigure, addNotify$)
$\wedge \neg$ ExistsMethodDefinition($DependencyFigure, basicTransform$)
$\wedge \neg$ ExistsMethodDefinition($DependencyFigure, setAttribute$)
$\wedge \neg$ ExistsMethodDefinition($DependencyFigure, findFigureInside$)
$\wedge \neg$ ExistsMethodDefinition($DependencyFigure, contains$)
$\wedge$ ExistsMethodDefinition($LineConnectionFigure, addNotify$)
$\wedge$ ExistsMethodDefinition($LineConnectionFigure, basicTransform$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LineConnectionFigure, addNotify, this$)
$\wedge$ AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded($LineConnectionFigure, basicTransform, this$)
$\wedge$ ExistsType($LabeledLineConnectionFigure$)
$\wedge$ ExistsClass($LabeledLineConnectionFigure$)
$\wedge$ IsSubType($LabeledLineConnectionFigure, BezierFigure$)

$\wedge$ $\neg$`ExistsMethodDefinition`($LabeledLineConnectionFigure, basicTransform$)
$\wedge$ $\neg$`ExistsMethodDefinition`($LabeledLineConnectionFigure, setAttribute$)
$\wedge$ $\neg$`ExistsMethodDefinition`($LabeledLineConnectionFigure, findFigureInside$)
$\wedge$ $\neg$`ExistsMethodDefinition`($LabeledLineConnectionFigure, contains$)
$\wedge$ `ExistsMethodDefinition`($BezierFigure, basicTransform$)
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`($BezierFigure, basicTransform, this$)
$\wedge$ `ExistsType`($SVGPath$)
$\wedge$ `ExistsClass`($SVGPath$)
$\wedge$ `IsSubType`($SVGPath, AbstractCompositeFigure$)
$\wedge$ $\neg$`ExistsMethodDefinition`($SVGPath, addNotify$)
$\wedge$ $\neg$`ExistsMethodDefinition`($SVGPath, removeNotify$)
$\wedge$ $\neg$`ExistsMethodDefinition`($SVGPath, findFigureInside$)
$\wedge$ $\neg$`ExistsMethodDefinition`($SVGPath, contains$)
$\wedge$ `ExistsMethodDefinition`($AbstractCompositeFigure, addNotify$)
$\wedge$ `ExistsMethodDefinition`($AbstractCompositeFigure, removeNotify$)
$\wedge$ `ExistsMethodDefinition`($AbstractCompositeFigure, findFigureInside$)
$\wedge$ `ExistsMethodDefinition`($AbstractCompositeFigure, contains$)
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`($AbstractCompositeFigure, addNotify, this$)
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`($AbstractCompositeFigure, removeNotify, this$)
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`($AbstractCompositeFigure, findFigureInside, this$)
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`($AbstractCompositeFigure, contains, this$)
$\wedge$ `ExistsType`($LineConnectionFigure$)
$\wedge$ `ExistsClass`($LineConnectionFigure$)
$\wedge$ `IsSubType`($LineConnectionFigure, BezierFigure$)
$\wedge$ $\neg$`ExistsMethodDefinition`($LineConnectionFigure, findFigureInside$)
$\wedge$ $\neg$`ExistsMethodDefinition`($LineConnectionFigure, setAttribute$)
$\wedge$ $\neg$`ExistsMethodDefinition`($LineConnectionFigure, contains$)
$\wedge$ `ExistsMethodDefinition`($BezierFigure, findFigureInside$)
$\wedge$ `ExistsMethodDefinition`($BezierFigure, setAttribute$)
$\wedge$ `ExistsMethodDefinition`($BezierFigure, contains$)
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`($BezierFigure, findFigureInside, this$)
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`($BezierFigure, setAttribute, this$)
$\wedge$ `AllInvokedMethodsWithParameterOInBodyOfMAreNotOverloaded`($BezierFigure, contains, this$)