Source Code Differencing for Software Evolution Research

Jean-Rémy Falleri

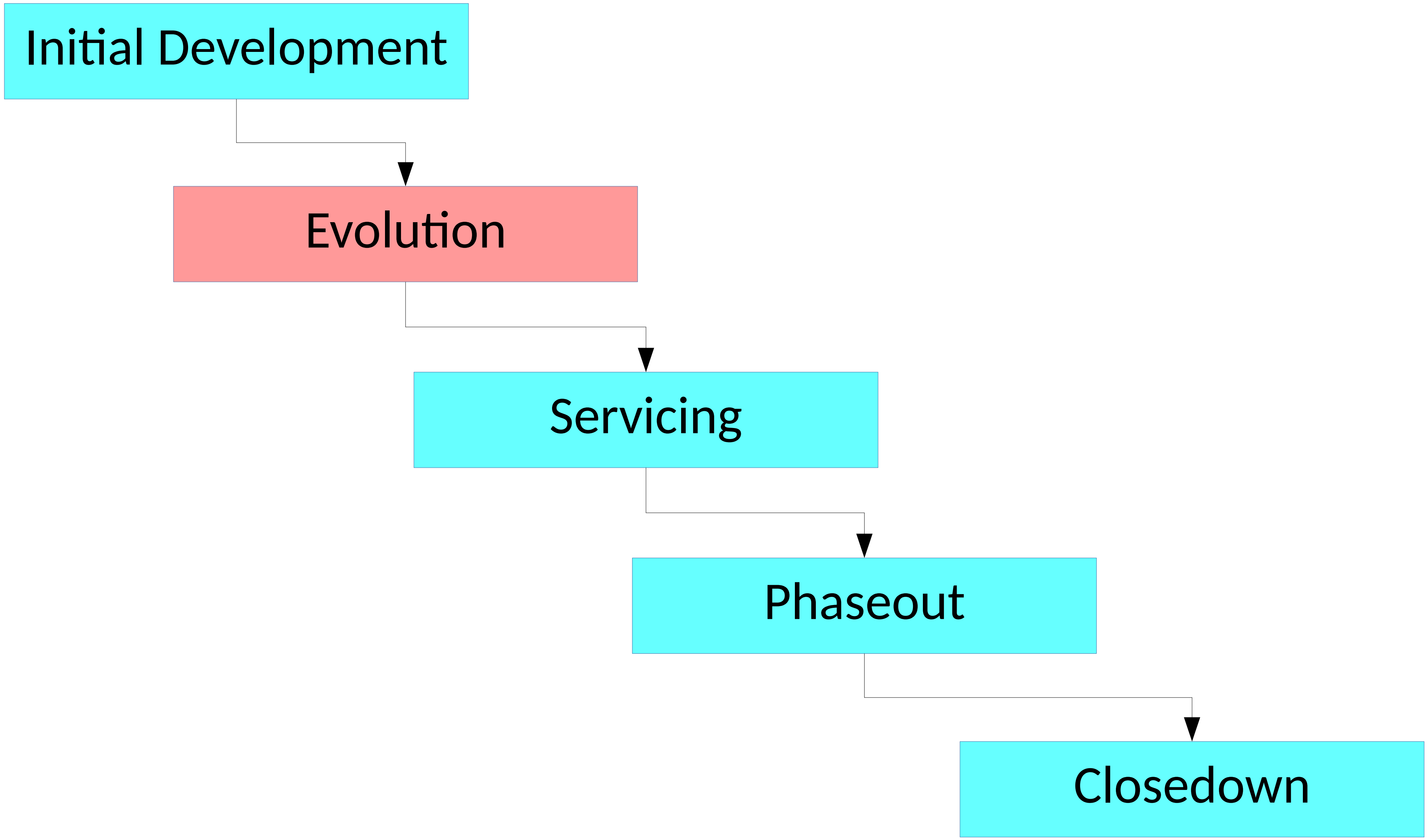
Habilitation à Diriger des Recherches

Software Engineering@LaBRI, Bordeaux, France

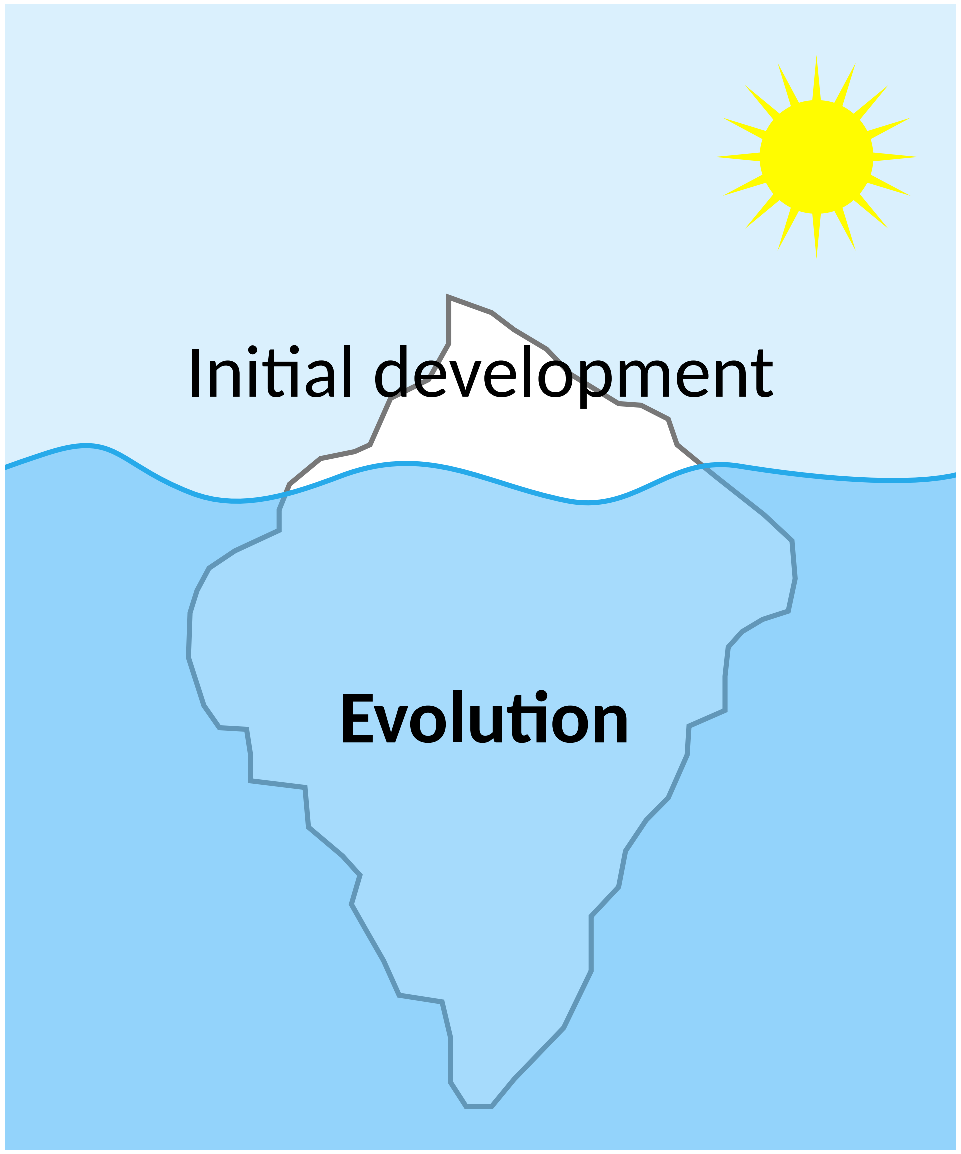
Outline

Introduction

Software evolution



Software development costs



Software evolution research

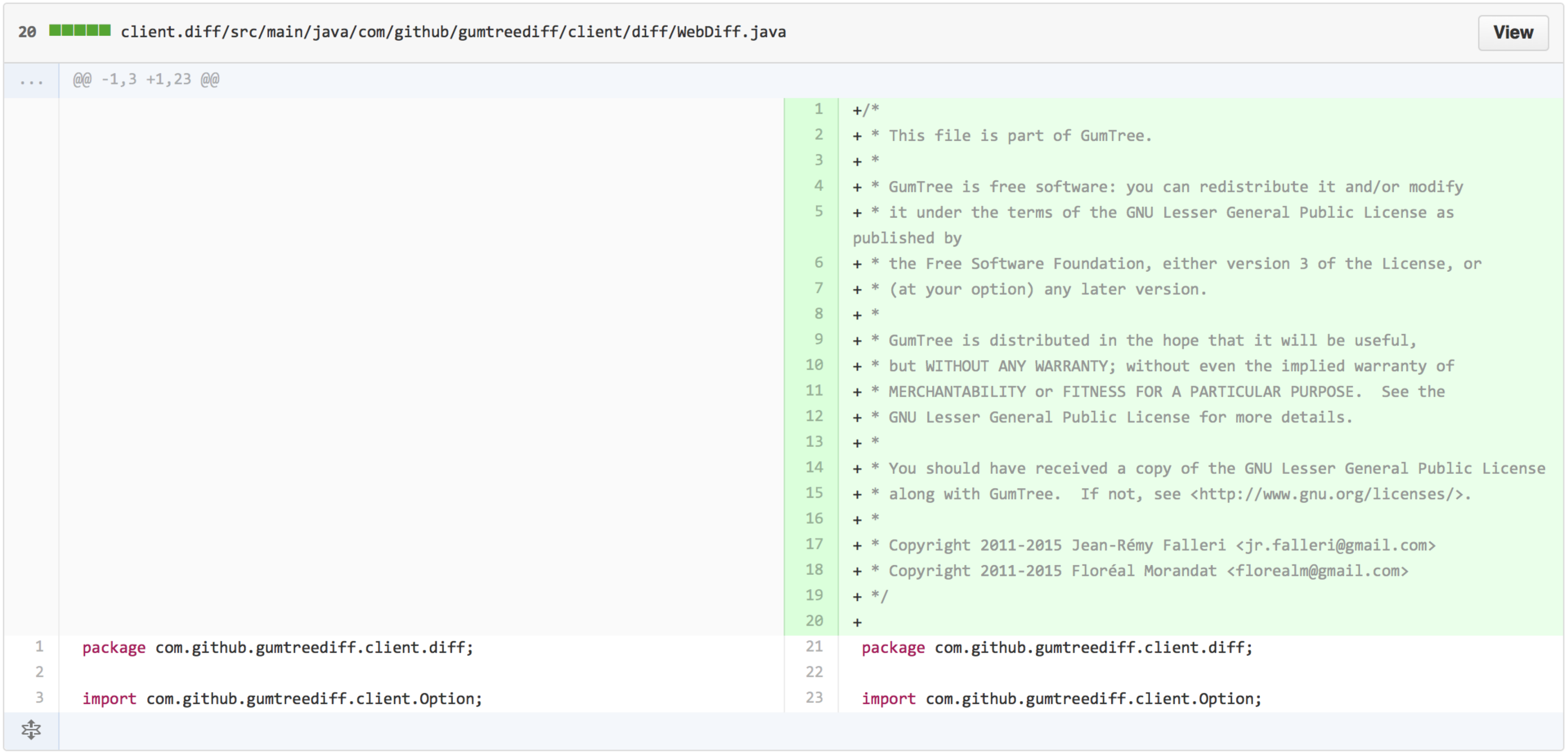
- Objectives
  - Understanding (How)?
    - Better understand what software evolution is about
  - Support (What and why)?
    - Furnish tools to practitioners to face software evolution
- My research
  - Contributing to both objectives
  - Mining open-source software systems repositories a.k.a. MSR
  - Using the empirical method

A step back

- Most of my work
  - Analyze source code evolution to understand and support software evolution
- Most MSR work
  - Analyze source code evolution to understand and support software evolution
    - 46% of surveyed approaches analyze source code evolution

→ but approaches to analyze source code evolution are limited

Currently

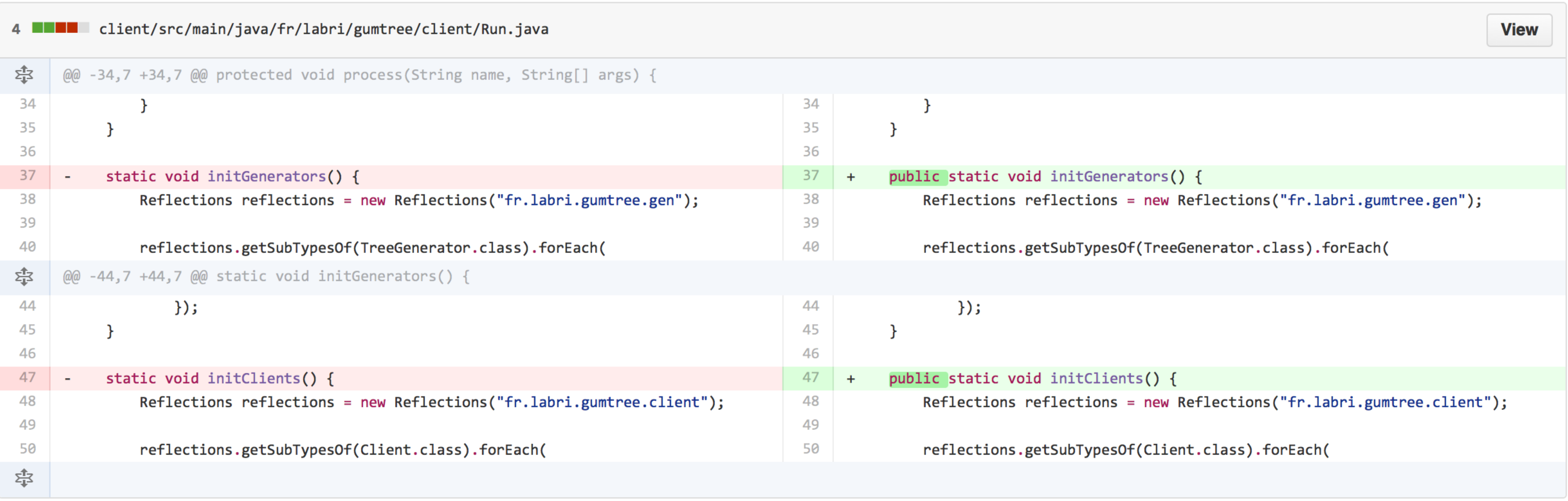


20 client.diff/src/main/java/com/github/gumtreediff/client/diff/WebDiff.java View

```
@@ -1,3 +1,23 @@
+/*
+ * This file is part of GumTree.
+ *
+ * GumTree is free software: you can redistribute it and/or modify
+ * it under the terms of the GNU Lesser General Public License as published by
+ * the Free Software Foundation, either version 3 of the License, or
+ * (at your option) any later version.
+ *
+ * GumTree is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ * GNU Lesser General Public License for more details.
+ *
+ * You should have received a copy of the GNU Lesser General Public License
+ * along with GumTree.  If not, see <http://www.gnu.org/licenses/>.
+ *
+ * Copyright 2011-2015 Jean-Rémy Falleri <jr.falleri@gmail.com>
+ * Copyright 2011-2015 Floréal Morandat <florealm@gmail.com>
+ */
+
 package com.github.gumtreediff.client.diff;

 import com.github.gumtreediff.client.Option;
```

Find big modifications

Other example



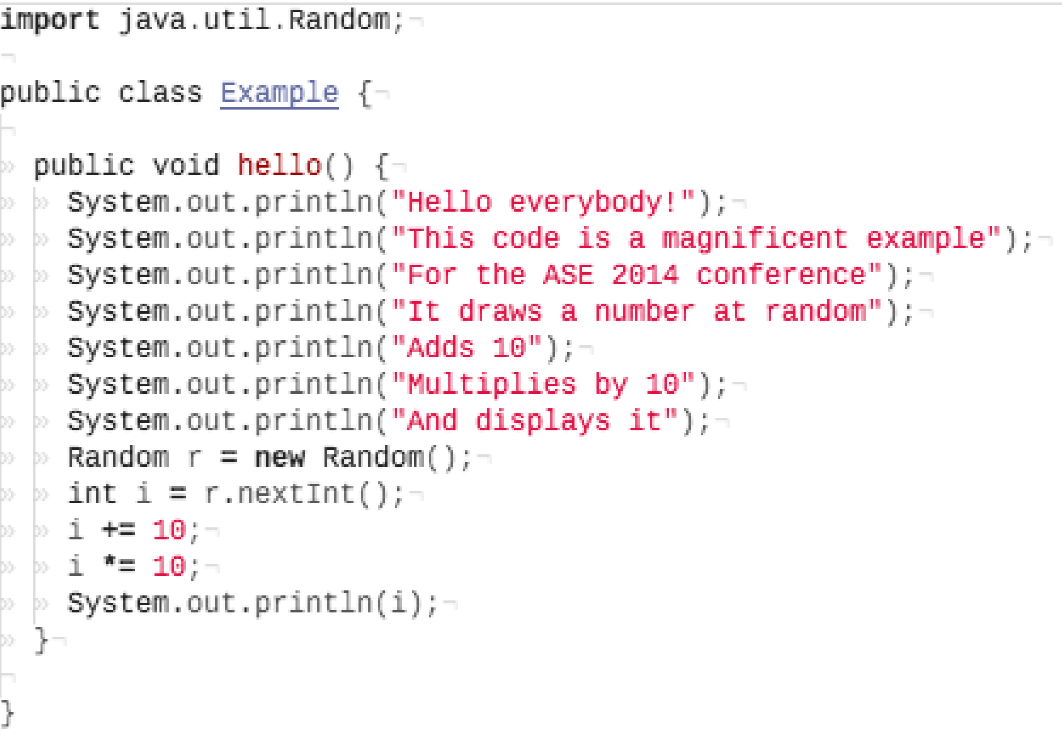
4 client/src/main/java/fr/labri/gumtree/client/Run.java View

```
@@ -34,7 +34,7 @@ protected void process(String name, String[] args) {
            }                                                           }
        }                                                           }

-       static void initGenerators() {                          +   public static void initGenerators() {
            Reflections reflections = new Reflections("fr.labri.gumtree.gen");      Reflections reflections = new Reflections("fr.labri.gumtree.gen");

            reflections.getSubTypesOf(TreeGenerator.class).forEach(             reflections.getSubTypesOf(TreeGenerator.class).forEach(
@@ -44,7 +44,7 @@ static void initGenerators() {
                });                                                         });
        }                                                           }

-       static void initClients() {                             +   public static void initClients() {
            Reflections reflections = new Reflections("fr.labri.gumtree.client");   Reflections reflections = new Reflections("fr.labri.gumtree.client");

            reflections.getSubTypesOf(Client.class).forEach(                    reflections.getSubTypesOf(Client.class).forEach(
```

Find added methods

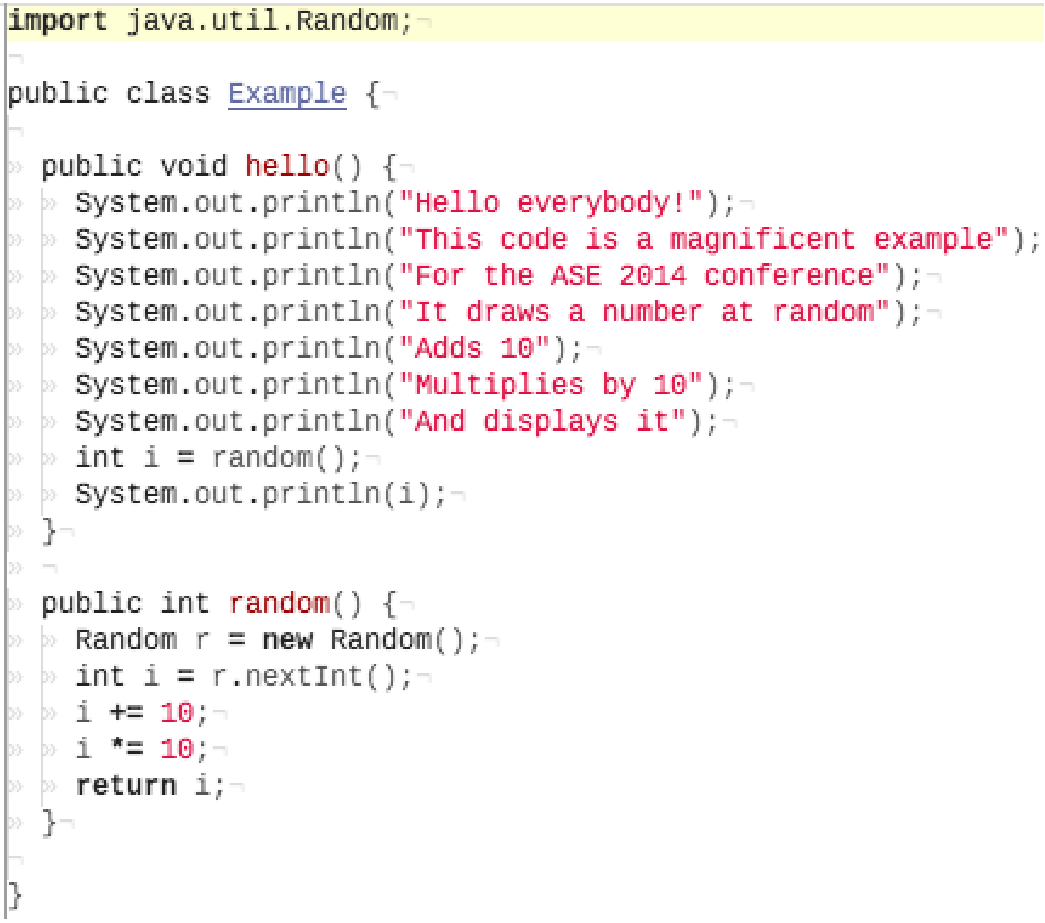
Source code differencing

Example



```
import java.util.Random;

public class Example {

    public void hello() {
        System.out.println("Hello everybody!");
        System.out.println("This code is a magnificent example");
        System.out.println("For the ASE 2014 conference");
        System.out.println("It draws a number at random");
        System.out.println("Adds 10");
        System.out.println("Multiplies by 10");
        System.out.println("And displays it");
        Random r = new Random();
        int i = r.nextInt();
        i += 10;
        i *= 10;
        System.out.println(i);
    }

}
```

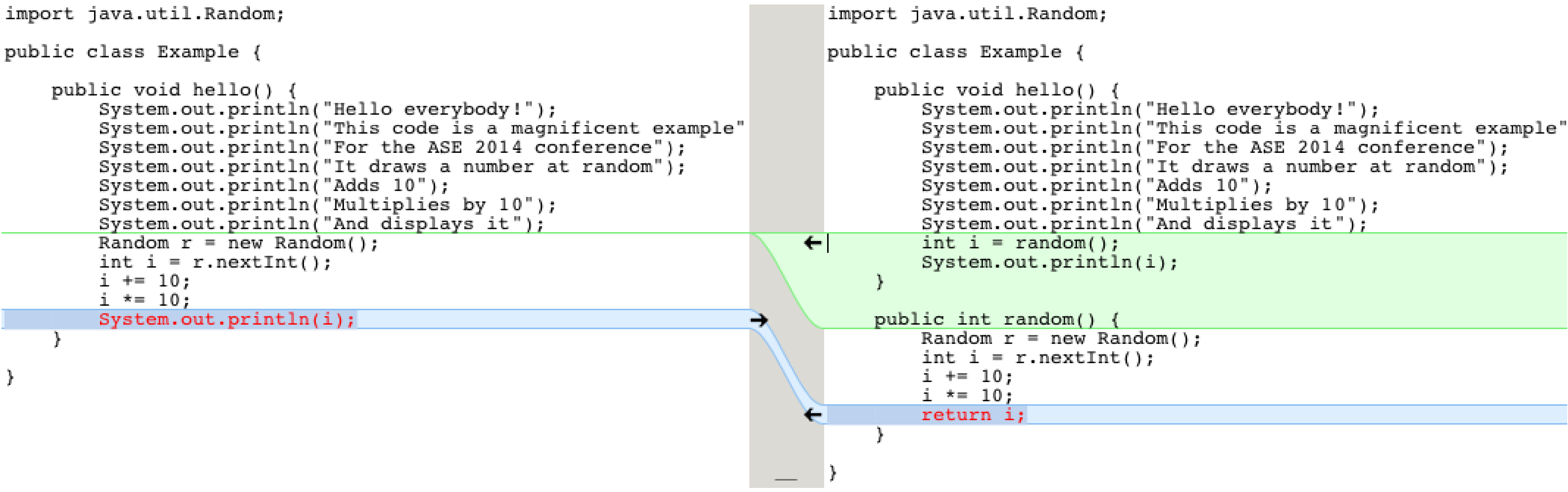
Previous version



```
import java.util.Random;

public class Example {

    public void hello() {
        System.out.println("Hello everybody!");
        System.out.println("This code is a magnificent example");
        System.out.println("For the ASE 2014 conference");
        System.out.println("It draws a number at random");
        System.out.println("Adds 10");
        System.out.println("Multiplies by 10");
        System.out.println("And displays it");
        int i = random();
        System.out.println(i);
    }

    public int random() {
        Random r = new Random();
        int i = r.nextInt();
        i += 10;
        i *= 10;
        return i;
    }

}
```

Current version

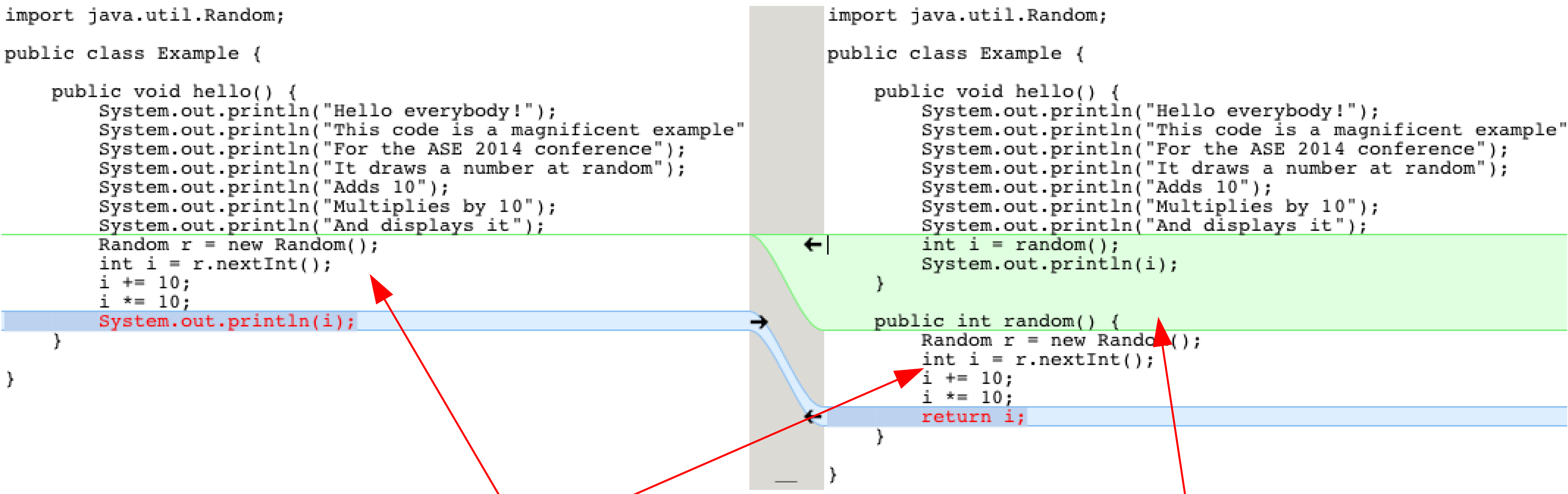
Current approaches



```
import java.util.Random;

public class Example {

    public void hello() {
        System.out.println("Hello everybody!");
        System.out.println("This code is a magnificent example"
        System.out.println("For the ASE 2014 conference");
        System.out.println("It draws a number at random");
        System.out.println("Adds 10");
        System.out.println("Multiplies by 10");
        System.out.println("And displays it");
        Random r = new Random();
        int i = r.nextInt();
        i += 10;
        i *= 10;
        System.out.println(i);
    }

}
```

```
import java.util.Random;

public class Example {

    public void hello() {
        System.out.println("Hello everybody!");
        System.out.println("This code is a magnificent example"
        System.out.println("For the ASE 2014 conference");
        System.out.println("It draws a number at random");
        System.out.println("Adds 10");
        System.out.println("Multiplies by 10");
        System.out.println("And displays it");
        int i = random();
        System.out.println(i);
    }

    public int random() {
        Random r = new Random();
        int i = r.nextInt();
        i += 10;
        i *= 10;
        return i;
    }

}
```

Current approaches



```
import java.util.Random;

public class Example {

    public void hello() {
        System.out.println("Hello everybody!");
        System.out.println("This code is a magnificent example"
        System.out.println("For the ASE 2014 conference");
        System.out.println("It draws a number at random");
        System.out.println("Adds 10");
        System.out.println("Multiplies by 10");
        System.out.println("And displays it");
        Random r = new Random();
        int i = r.nextInt();
        i += 10;
        i *= 10;
        System.out.println(i);
    }

}
```

```
import java.util.Random;

public class Example {

    public void hello() {
        System.out.println("Hello everybody!");
        System.out.println("This code is a magnificent example"
        System.out.println("For the ASE 2014 conference");
        System.out.println("It draws a number at random");
        System.out.println("Adds 10");
        System.out.println("Multiplies by 10");
        System.out.println("And displays it");
        int i = random();
        System.out.println(i);
    }

    public int random() {
        Random r = new Random();
        int i = r.nextInt();
        i += 10;
        i *= 10;
        return i;
    }

}
```

Don't detect moves

Not aligned on the code

→ Don't represent the developer intent

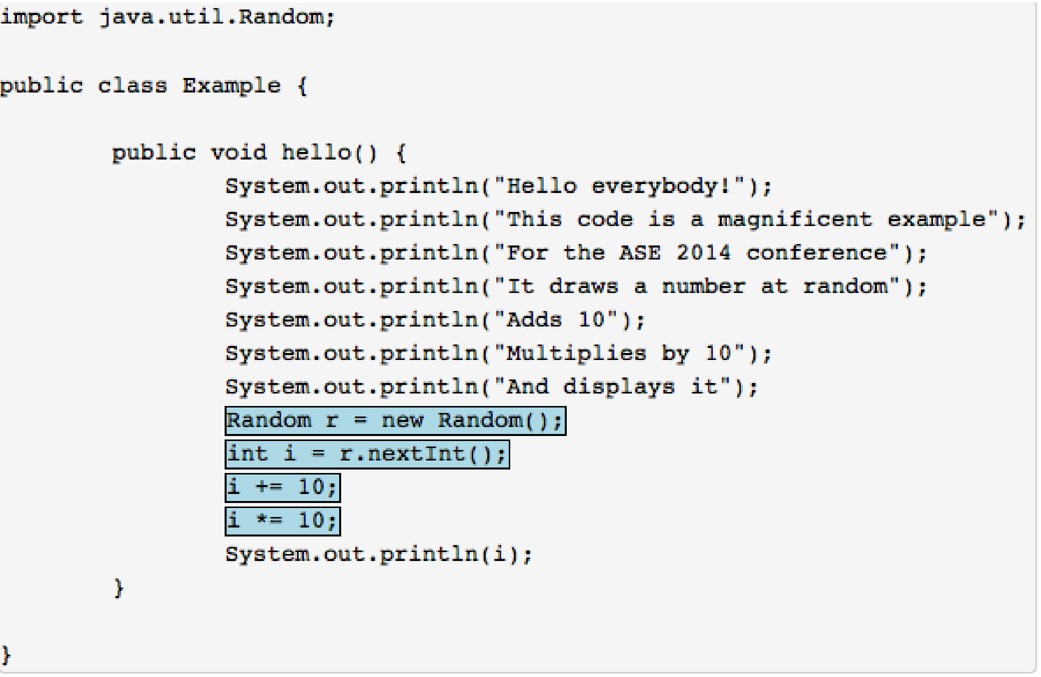
Textual differencing

- Source code model is a sequence of text lines
- Possible actions are
  - Delete a text line
  - Insert a text line
- Problem solved: find one shortest sequence of actions
  - Transforming the previous source code model
  - Into the current source code model

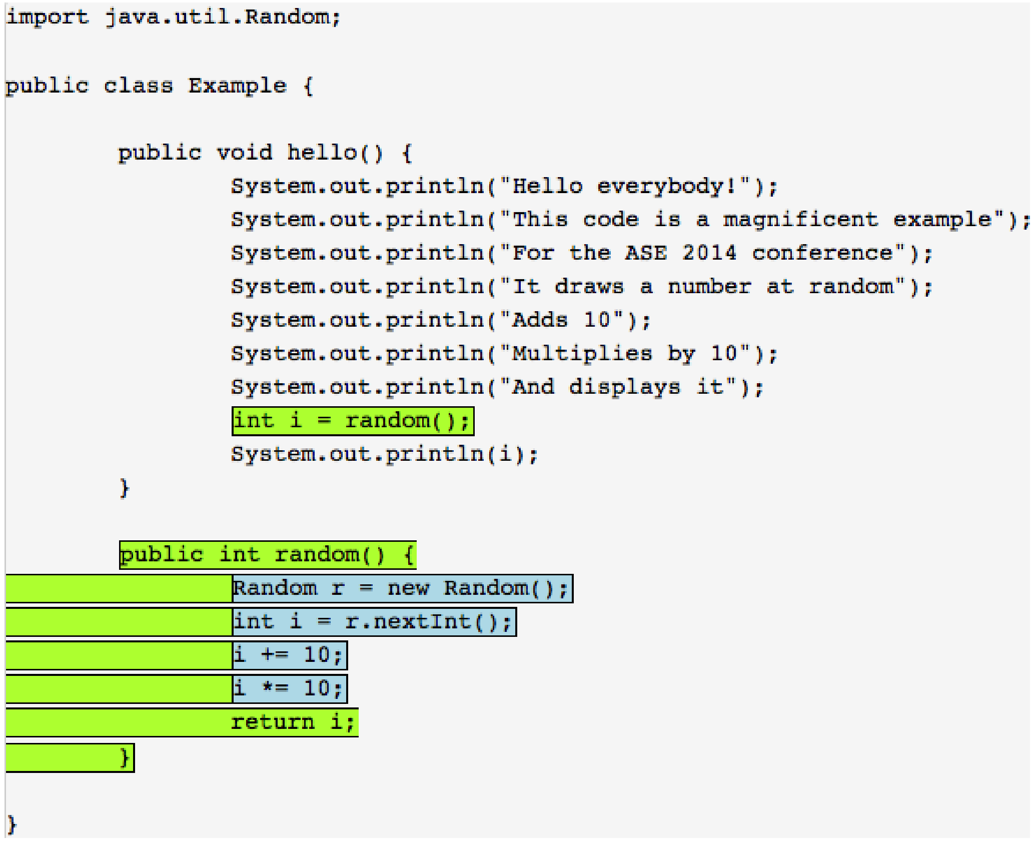
Source code differencing is hard

- It combines three implementation choices
  - Source code model
  - Set of possible actions
  - Problem solved (usually shortest sequence of actions)
- Many combinations lead to NP-hard problems
  - Textual differencing including move a subsequence of lines is NP-hard
  - Labeled graph differencing is NP-hard, even with only basic actions
  - Unordered tree differencing is NP-hard, even with only basic actions

GumTree



```
import java.util.Random;

public class Example {

        public void hello() {
                System.out.println("Hello everybody!");
                System.out.println("This code is a magnificent example");
                System.out.println("For the ASE 2014 conference");
                System.out.println("It draws a number at random");
                System.out.println("Adds 10");
                System.out.println("Multiplies by 10");
                System.out.println("And displays it");
                Random r = new Random();
                int i = r.nextInt();
                i += 10;
                i *= 10;
                System.out.println(i);
        }

}
```



```
import java.util.Random;

public class Example {

        public void hello() {
                System.out.println("Hello everybody!");
                System.out.println("This code is a magnificent example");
                System.out.println("For the ASE 2014 conference");
                System.out.println("It draws a number at random");
                System.out.println("Adds 10");
                System.out.println("Multiplies by 10");
                System.out.println("And displays it");
                int i = random();
                System.out.println(i);
        }

        public int random() {
                Random r = new Random();
                int i = r.nextInt();
                i += 10;
                i *= 10;
                return i;
        }

}
```

Our choices

- Source code model
  - a labeled ordered rooted tree
- Possible actions
  - Node insertion
  - Node deletion
  - Node relabel
  - Subtree move
- Problem solved
  - short sequence that corresponds to a developer intent

The process

1. Parse code files to our code agnostic tree structure
2. **Find mappings between nodes**
   - *Like developers manually proceed*
3. Deduce code edition actions (as in [6])
4. Output code differences

Finding mappings

1. Top-down phase
   - Find biggest chunks of unmodified code
2. Bottom-up phase
   - Propagate mappings to the containers of these chunks (functions, classes, …)
   - Extend mappings in the left-over code of these containers

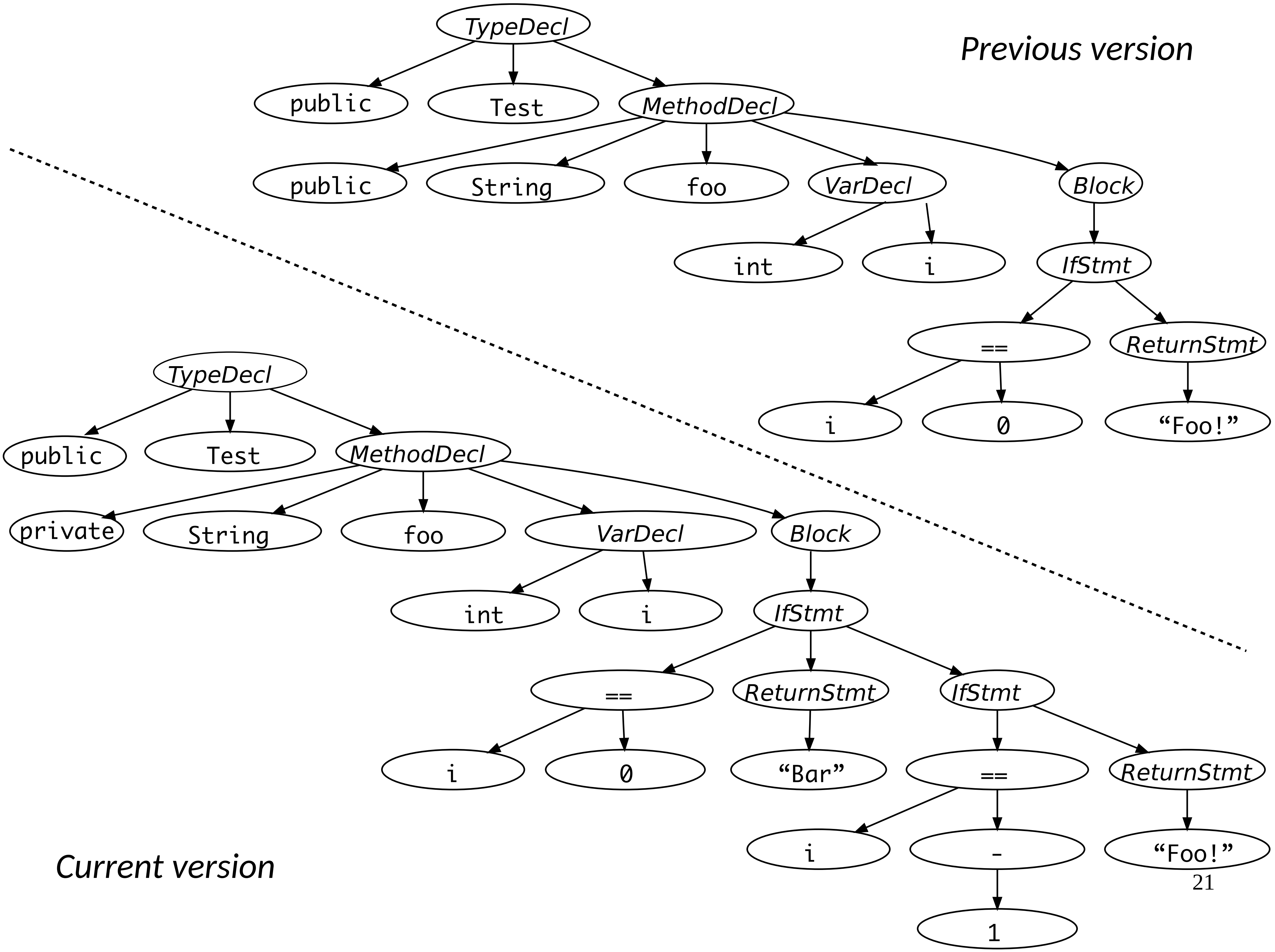
Example

```java
public class Test {
  public String foo(int i) {
    if (i == 0)
      return "Foo!";
  }
}
```

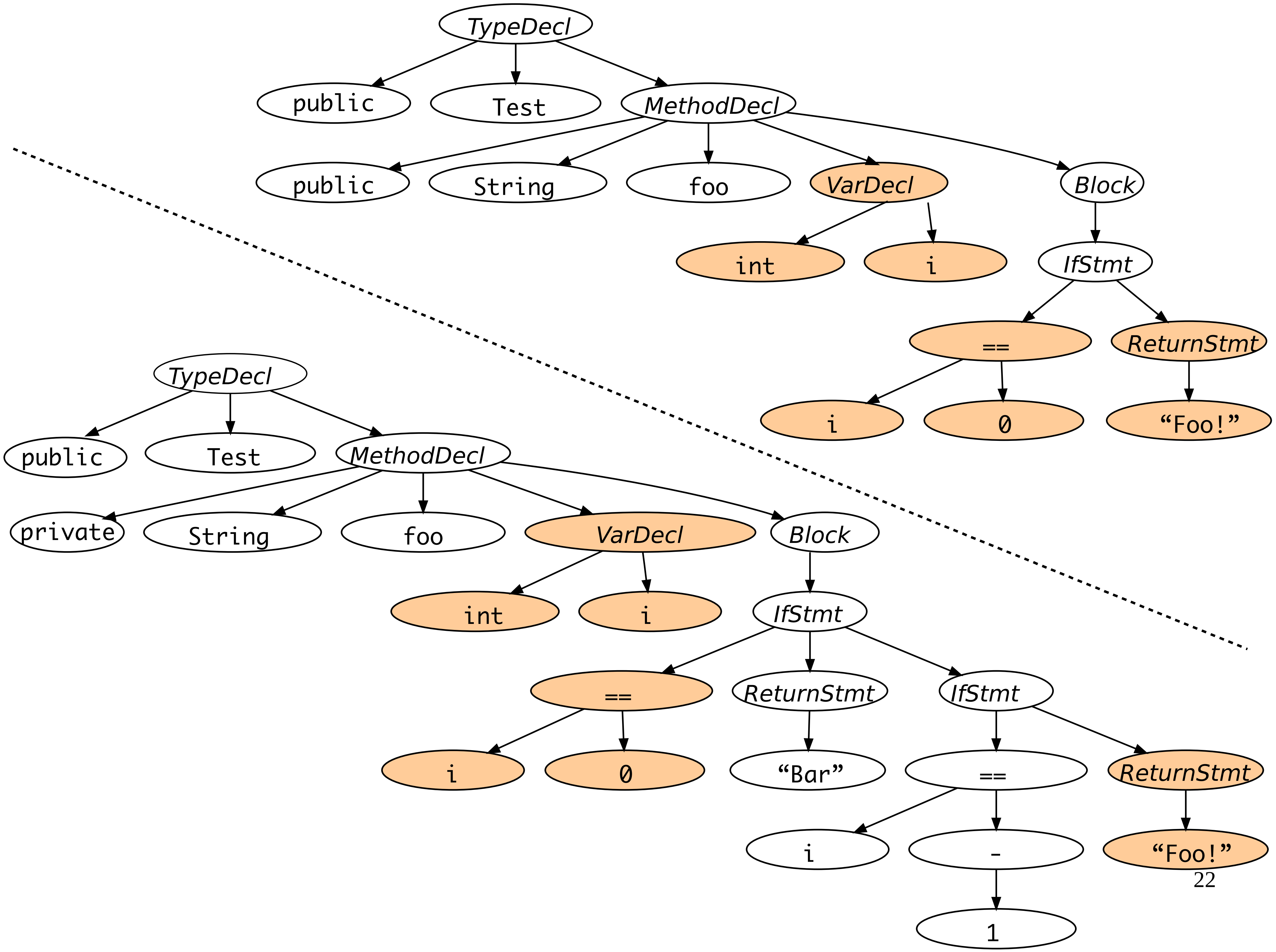
Previous version

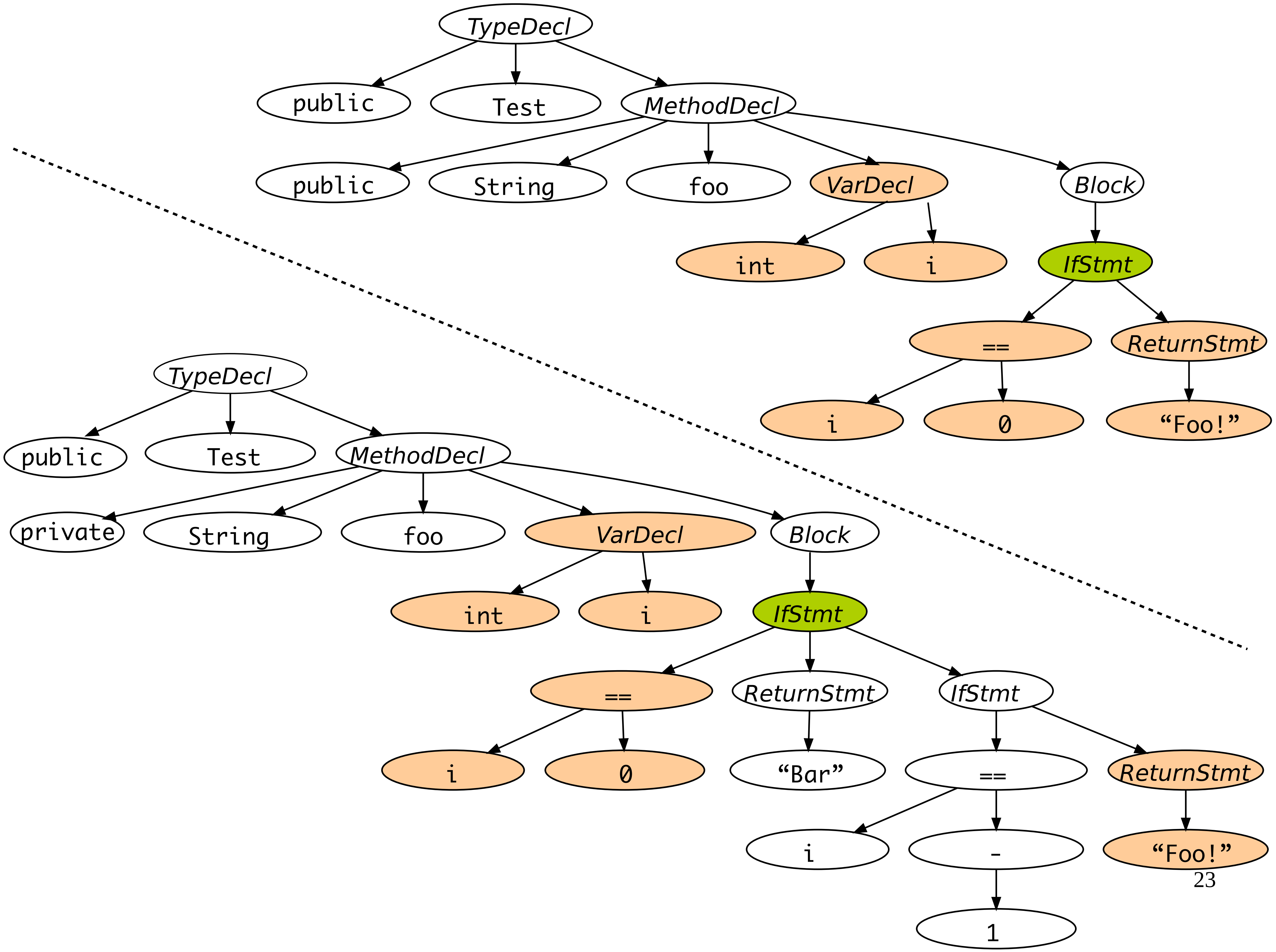
```java
public class Test {
  private String foo(int i) {
    if (i == 0)
      return "Bar";
    else if (i == -1)
      return "Foo!";
  }
}
```

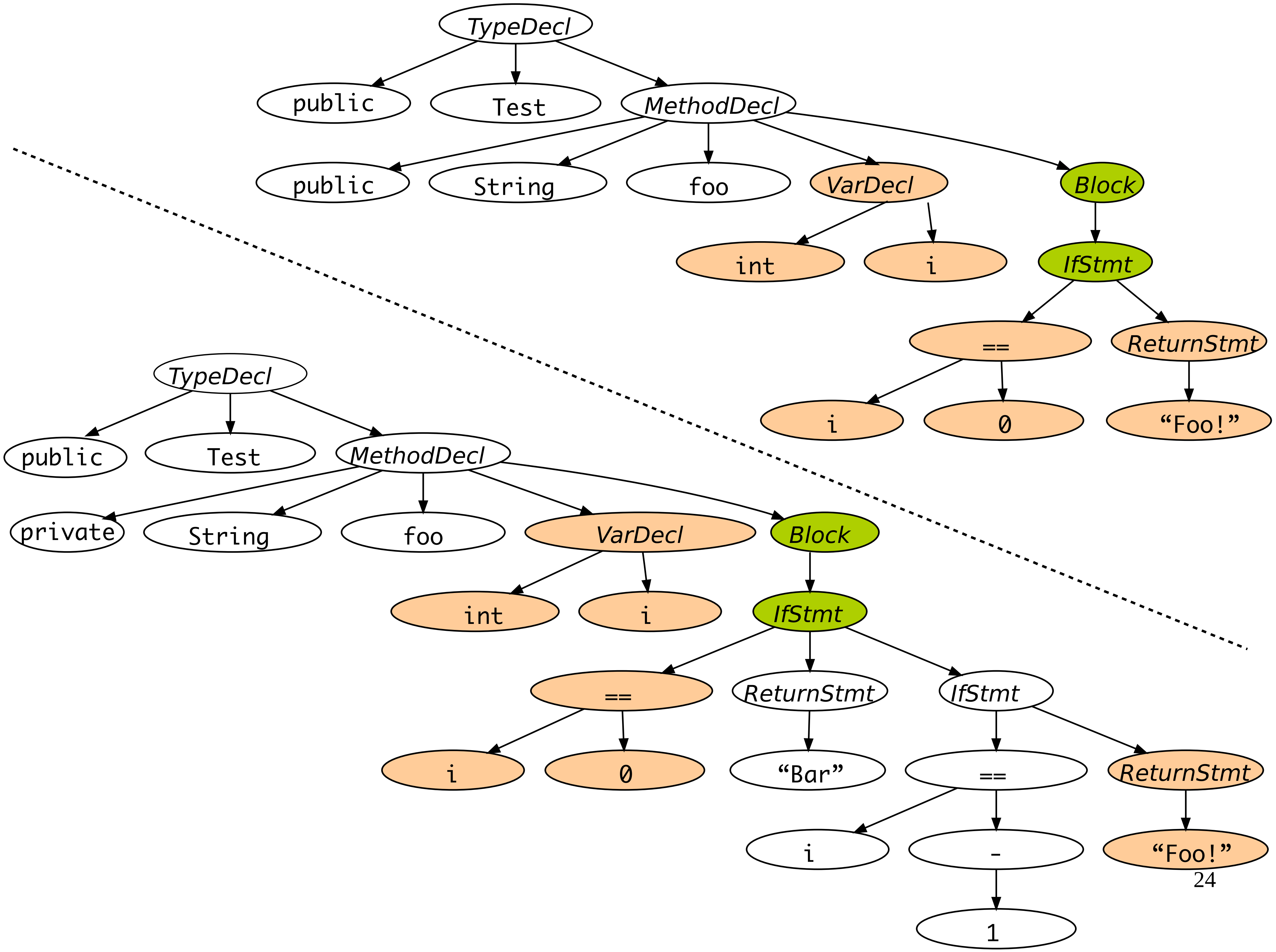
Current version

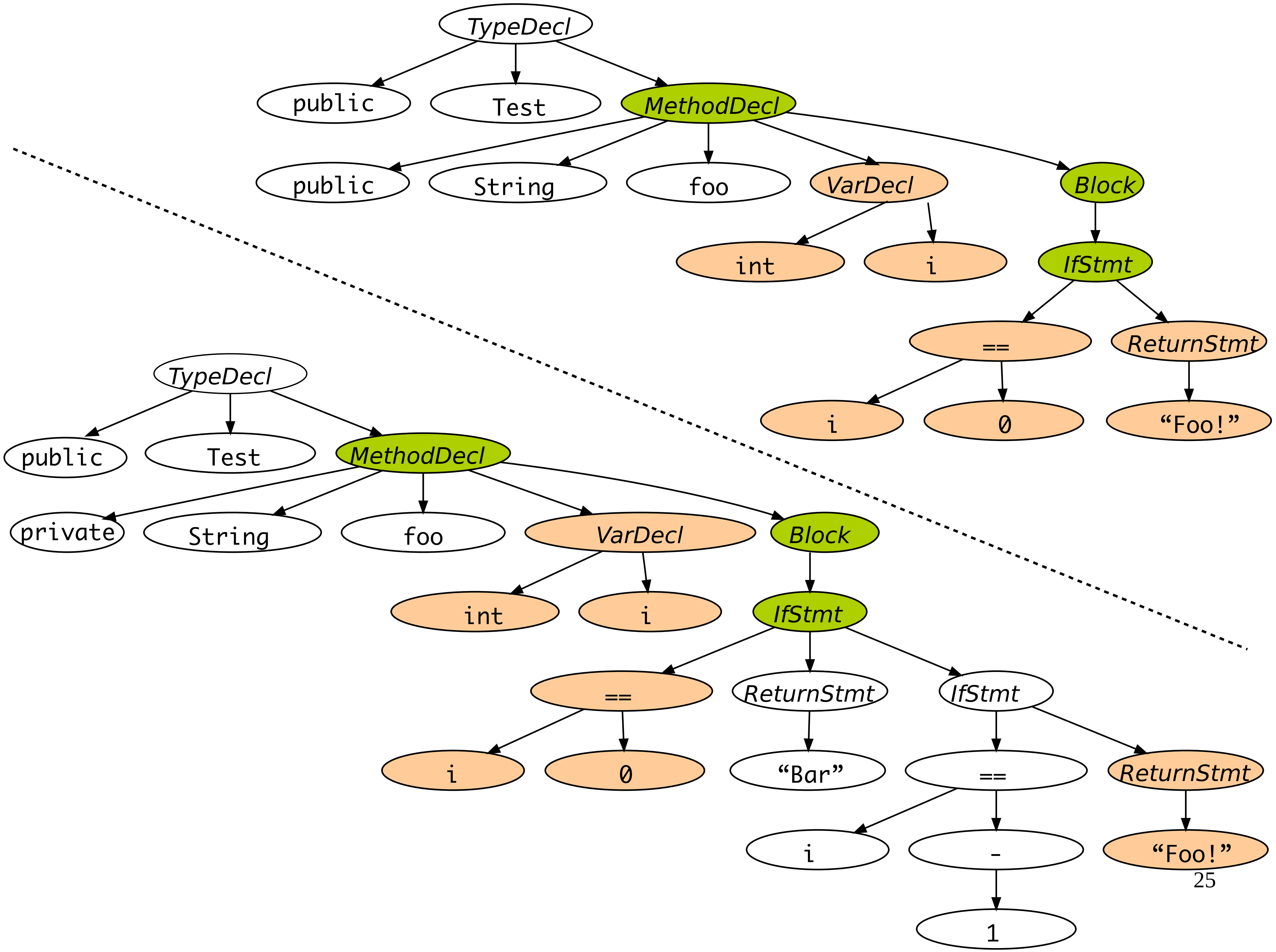


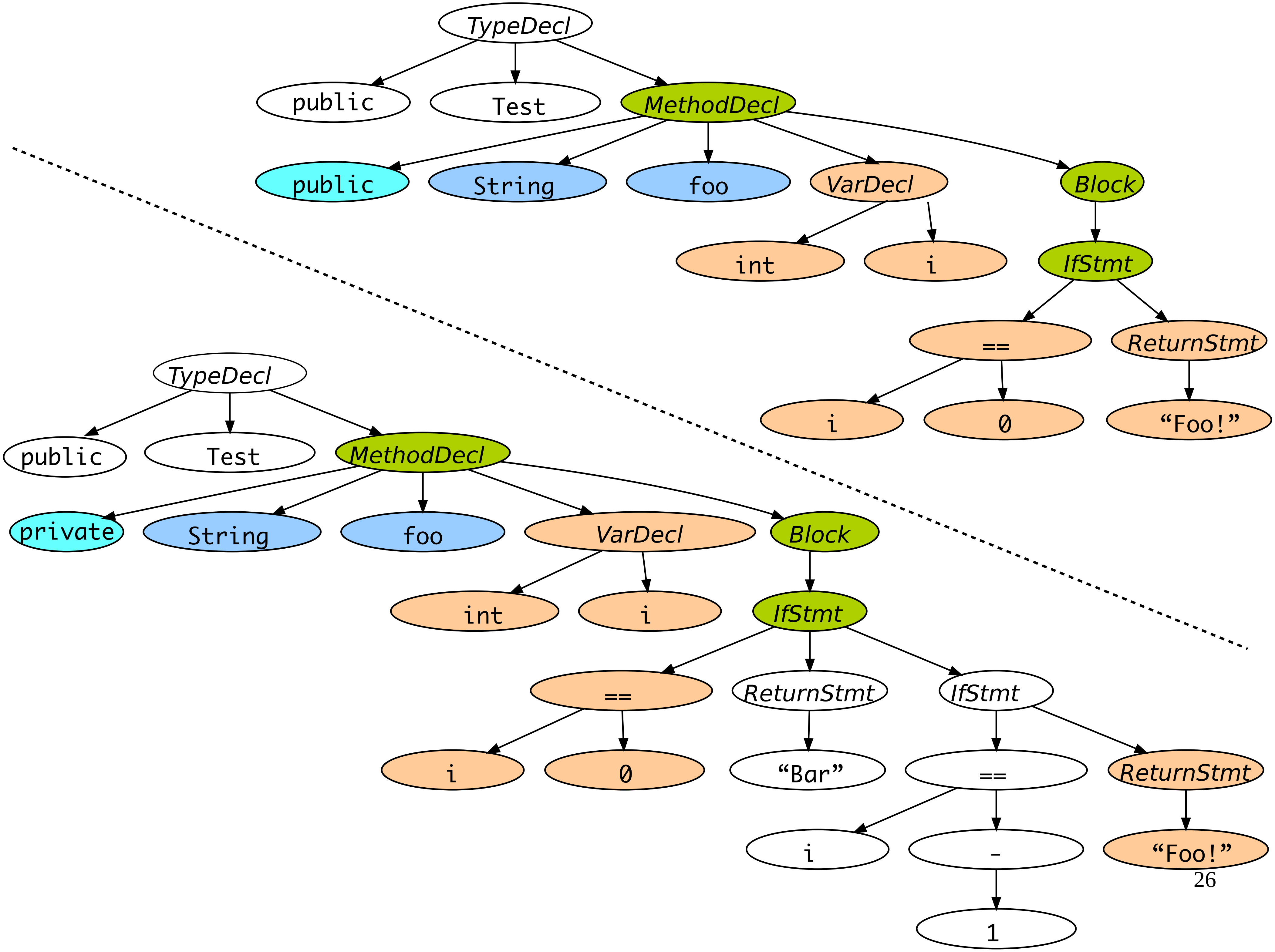
Previous version
TypeDecl
public
Test
MethodDecl
public
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
i
0
“Foo!”
TypeDecl
public
Test
MethodDecl
private
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
IfStmt
i
0
“Bar”
==
ReturnStmt
i
-
“Foo!”
1
Current version

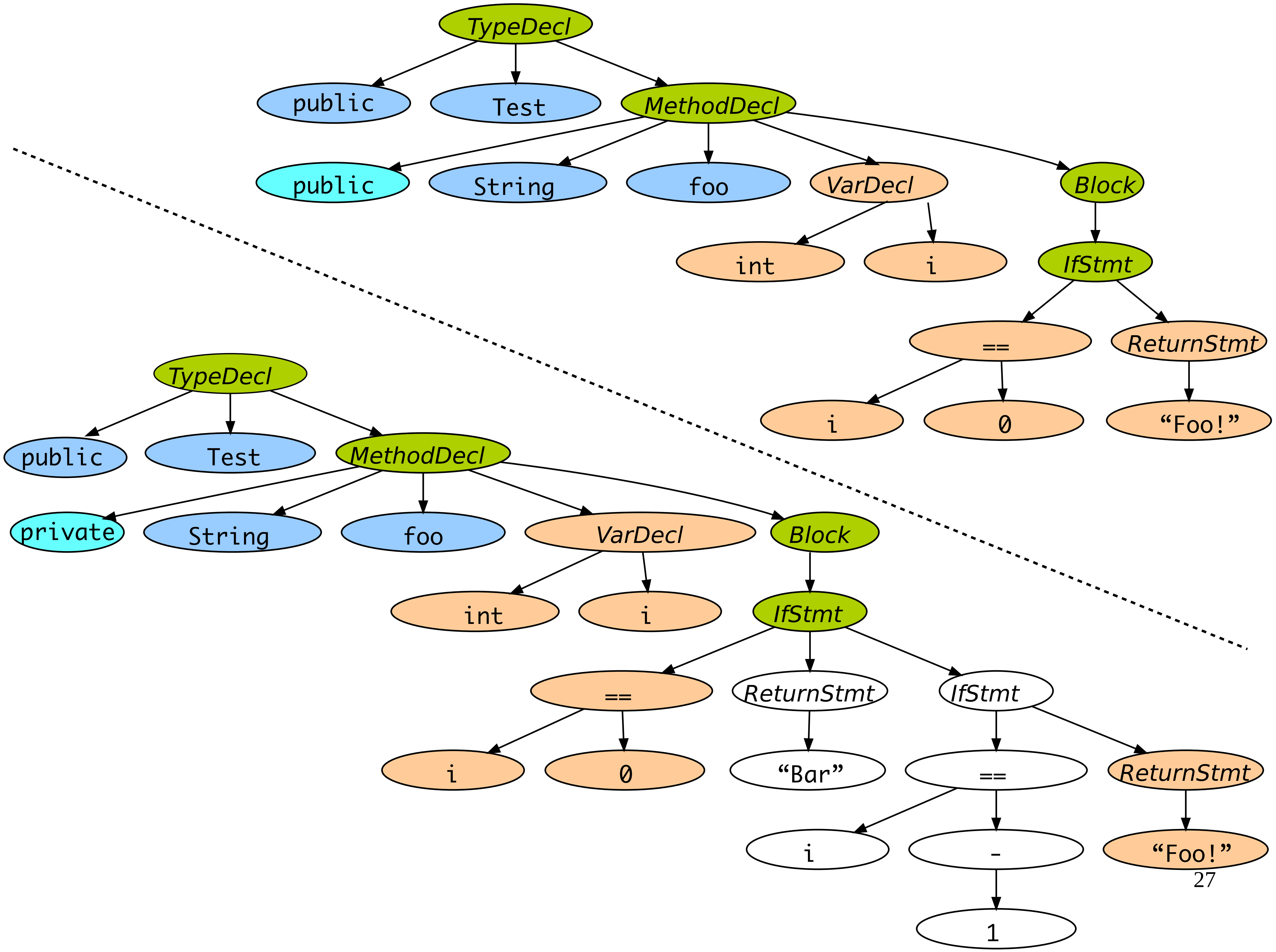

TypeDecl
public
Test
MethodDecl
public
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
i
0
“Foo!”
TypeDecl
public
Test
MethodDecl
private
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
IfStmt
i
0
“Bar”
==
ReturnStmt
i
-
“Foo!”
1


TypeDecl
public
Test
MethodDecl
public
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
i
0
“Foo!”
TypeDecl
public
Test
MethodDecl
private
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
IfStmt
i
0
“Bar”
==
ReturnStmt
i
-
“Foo!”
1


TypeDecl
public
Test
MethodDecl
public
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
i
0
“Foo!”
TypeDecl
public
Test
MethodDecl
private
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
IfStmt
i
0
“Bar”
==
ReturnStmt
i
-
“Foo!”
1


TypeDecl
public
Test
MethodDecl
public
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
i
0
“Foo!”
TypeDecl
public
Test
MethodDecl
private
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
IfStmt
i
0
“Bar”
==
ReturnStmt
i
-
“Foo!”
1


TypeDecl
public
Test
MethodDecl
public
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
i
0
“Foo!”
TypeDecl
public
Test
MethodDecl
private
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
IfStmt
i
0
“Bar”
==
ReturnStmt
i
-
“Foo!”
1


TypeDecl
public
Test
MethodDecl
public
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
i
0
“Foo!”
TypeDecl
public
Test
MethodDecl
private
String
foo
VarDecl
Block
int
i
IfStmt
==
ReturnStmt
IfStmt
i
0
“Bar”
==
ReturnStmt
i
-
“Foo!”
1

Output

```
public class Test {
  public String foo(int i) {
    if (i == 0)
      return "Foo!";
  }
}
```

Previous version

```
public class Test {
  private String foo(int i) {
    if (i == 0)
      return "Bar";
    else if (i == -1)
      return "Foo!";
  }
}
```

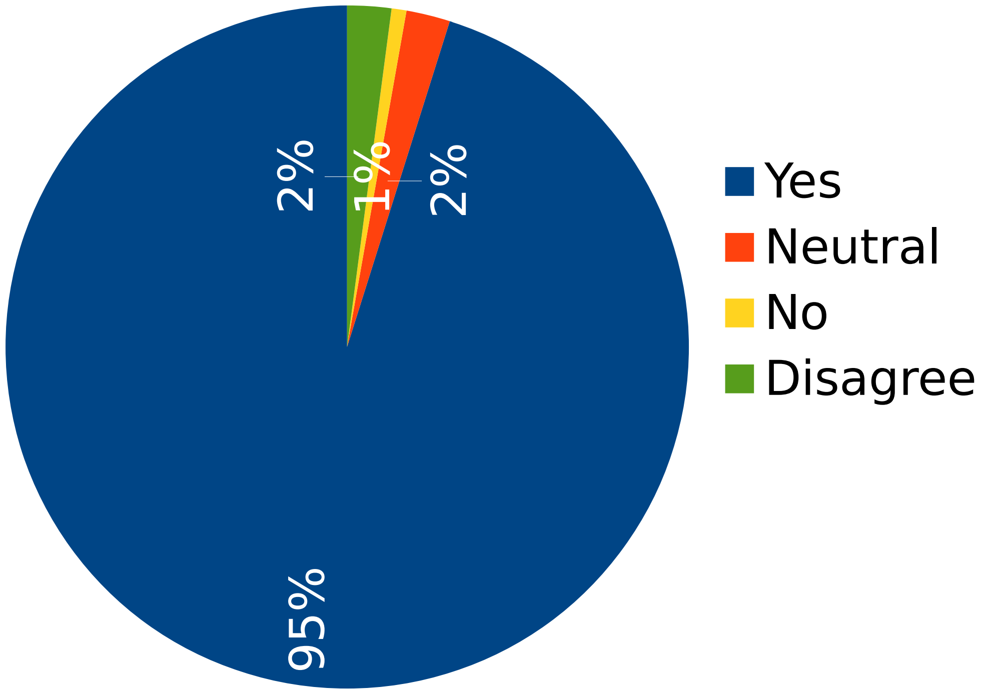
Current version

Human evaluation

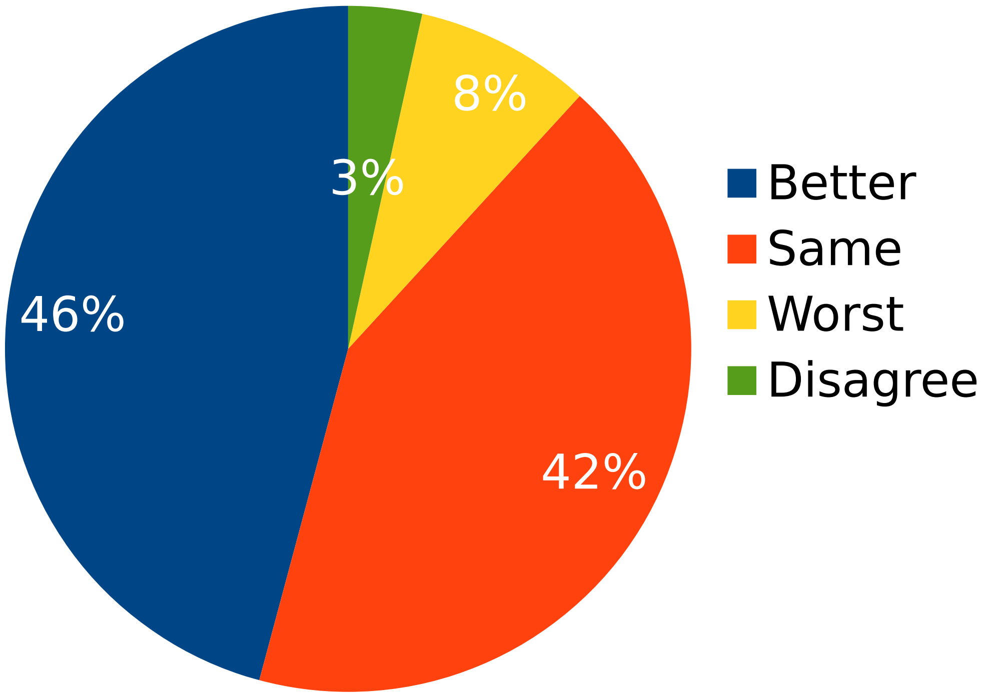
- 144 file modifications from 16 Java projects
- Two tools
  - GumTree
  - A visual text diff tool
- Two questions
  - GumTree does a good job? (*Yes*, *Neutral or No*)
  - GumTree is better than text diff? (*Yes*, *No* or *Same*)
- Three raters
- I report the opinion of the majority

Results

GumTree does a good job?



GumTree vs text diff?



Support application

Find experts

- Software development involves large teams of developers who have their own set of skills
- Experts are developers with a strong knowledge in a technology (API, language, ...)
- Experts are often searched for
  - Fix issue with a particular library
  - Problem with a language construct

→ how to know what are the technical skills of developers?

Self and peer evaluation

- Very subjective
- Time consuming
- Hard to keep up to date

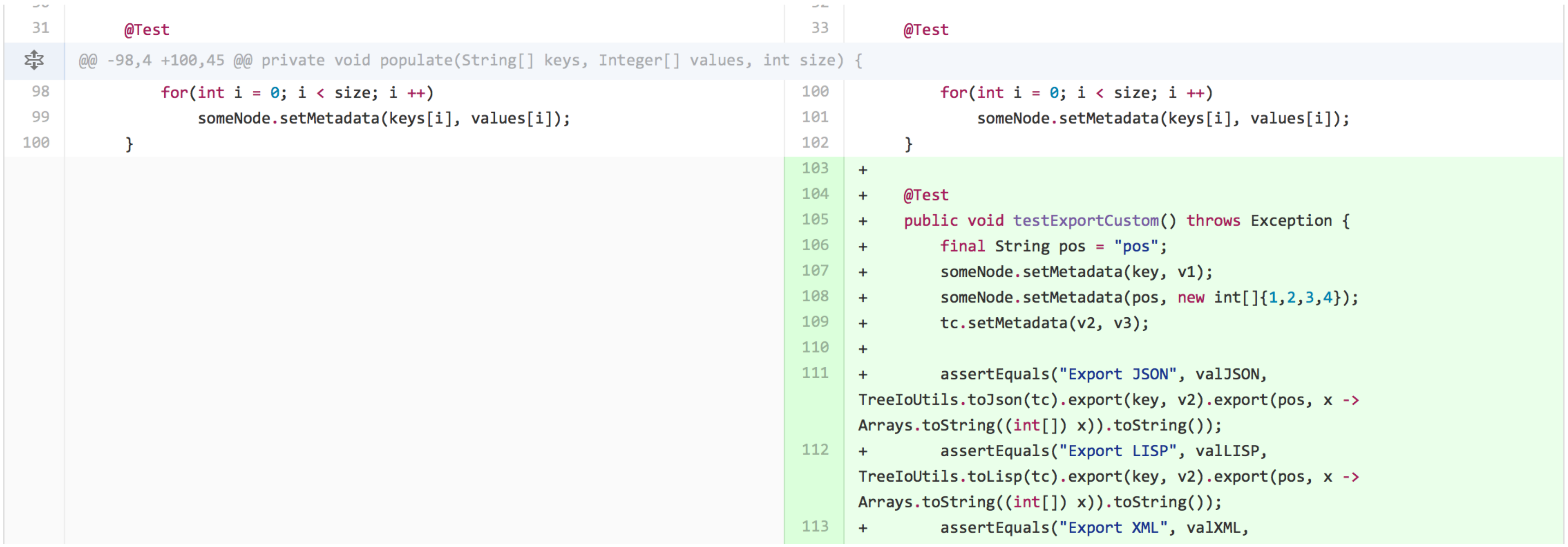
What is an expert?

- What is a Java language expert?
  - Someone who knows inheritance
  - Someone who knows how to compile Java code
  - Someone who knows how to test in Java
  - …

Our approach

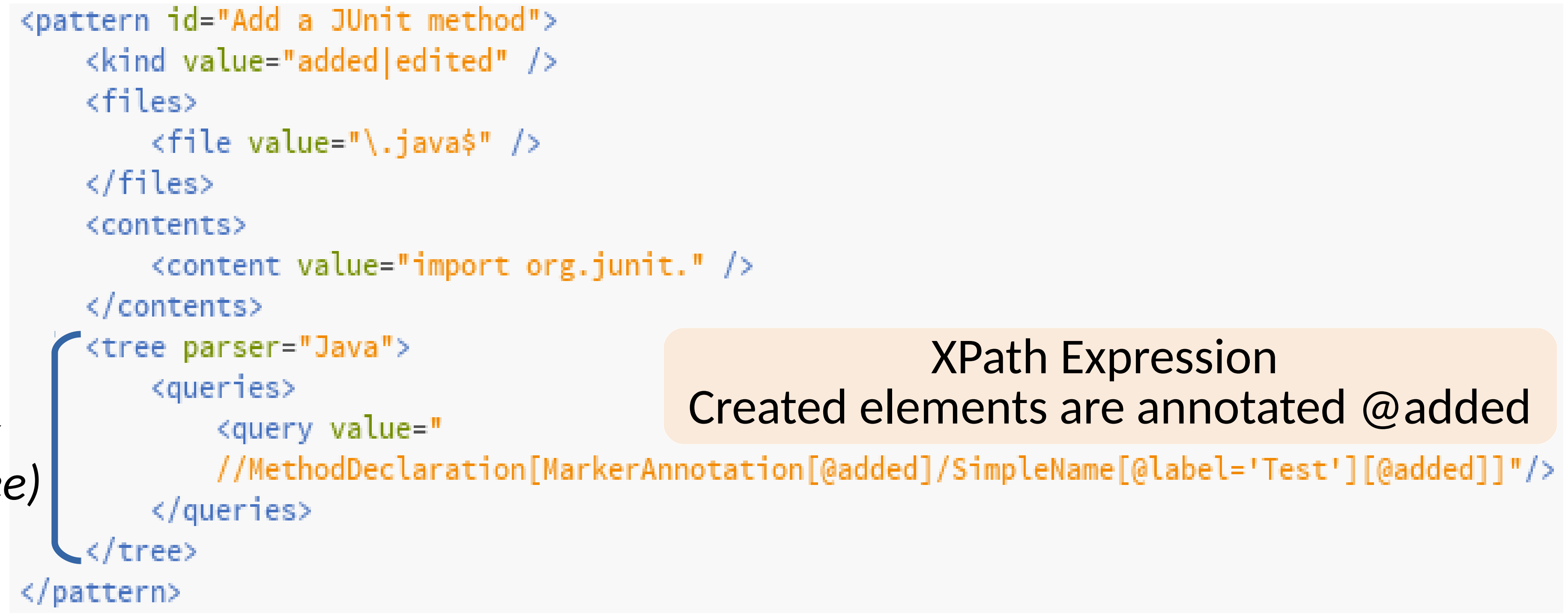
- Thomas the apostle: “I believe what I observe”
- Domain specific language to define how to observe skills from code modifications
- And count skill observations in all code modifications
  - Less subjective
  - Easy to keep up-to-date

Example



```
    @Test                                                        @Test
@@ -98,4 +100,45 @@ private void populate(String[] keys, Integer[] values, int size) {
        for(int i = 0; i < size; i ++)                               for(int i = 0; i < size; i ++)
            someNode.setMetadata(keys[i], values[i]);                    someNode.setMetadata(keys[i], values[i]);
    }                                                            }
                                                             +
                                                             +    @Test
                                                             +    public void testExportCustom() throws Exception {
                                                             +        final String pos = "pos";
                                                             +        someNode.setMetadata(key, v1);
                                                             +        someNode.setMetadata(pos, new int[]{1,2,3,4});
                                                             +        tc.setMetadata(v2, v3);
                                                             +
                                                             +        assertEquals("Export JSON", valJSON,
                                                             TreeIoUtils.toJson(tc).export(key, v2).export(pos, x ->
                                                             Arrays.toString((int[]) x)).toString());
                                                             +        assertEquals("Export LISP", valLISP,
                                                             TreeIoUtils.toLisp(tc).export(key, v2).export(pos, x ->
                                                             Arrays.toString((int[]) x)).toString());
                                                             +        assertEquals("Export XML", valXML,
```

Technical skills DSL



```
<pattern id="Add a JUnit method">
    <kind value="added|edited" />
    <files>
        <file value="\.java$" />
    </files>
    <contents>
        <content value="import org.junit." />
    </contents>
    <tree parser="Java">
        <queries>
            <query value="
            //MethodDeclaration[MarkerAnnotation[@added]/SimpleName[@label='Test'][@added]]"/>
        </queries>
    </tree>
</pattern>
```

XPath Expression
Created elements are annotated @added

Syntax (GumTree)

Other applications

- Automated study on the evolution of faults in Linux
- JavaScript differencing in web applications
- Study how developers document source code
- Improve software activity metrics
- ...

Lessons learned

My other contributions

- Automated extraction of developers' skills
- Relation between developer's organization and bugs
- Benchmarks to validate clone detectors and clone detection in CSS
- Assistance to library replacement
- Study and tools to understand and remove cycles in software systems

Is it *really* a problem?

- We (researchers) are usually not industrial software developers
- The problem we are working on might be irrelevant
- Solution 1: mining to find problem occurrence
  - Advantages: easy to do
  - Caveats: lack of observations does not imply irrelevance
- Solution 2: ask real developers
  - Advantages: the best way to ensure relevance
  - Caveats: hard to find developers, legal issues

Collecting data

- In a perfect world
  - Processes and tools are used perfectly
  - Data is well structured

- In the software world
  - Processes and tools are misused (or ignored)
  - Information is hidden among weird conventions

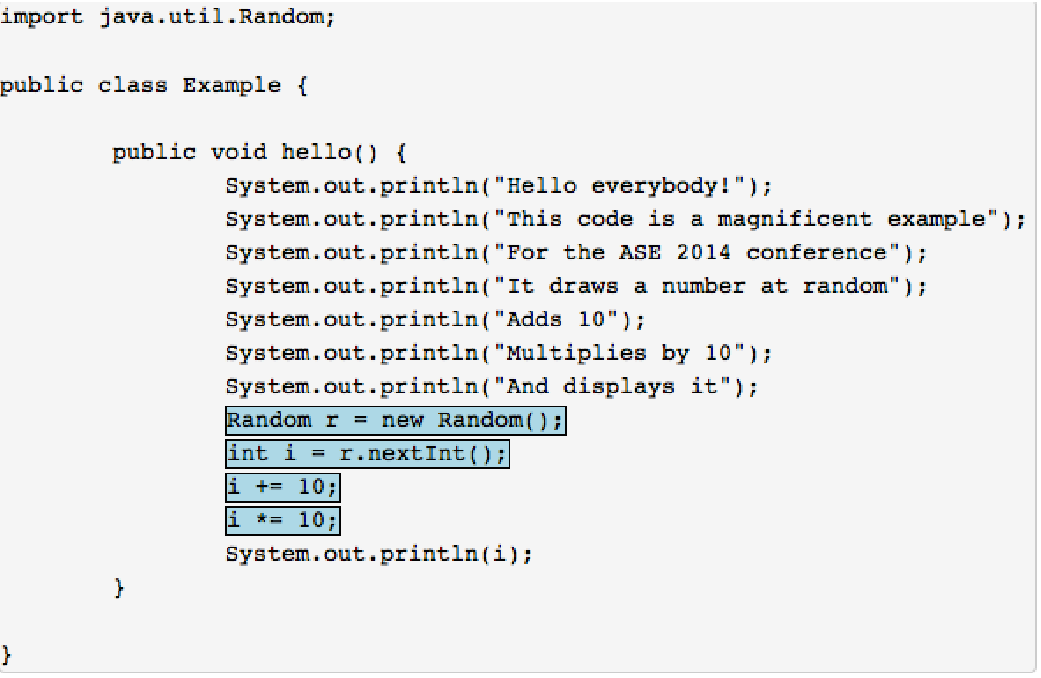
- Solution 1 : automatic cleaning approaches
  - Advantages : quick and easy to apply
  - Caveats : usually bad precision and/or recall
- Solution 2 : manual cleaning
  - Advantages : much more efficient
  - Caveats : long, tedious and researchers cannot judge everything

Validation

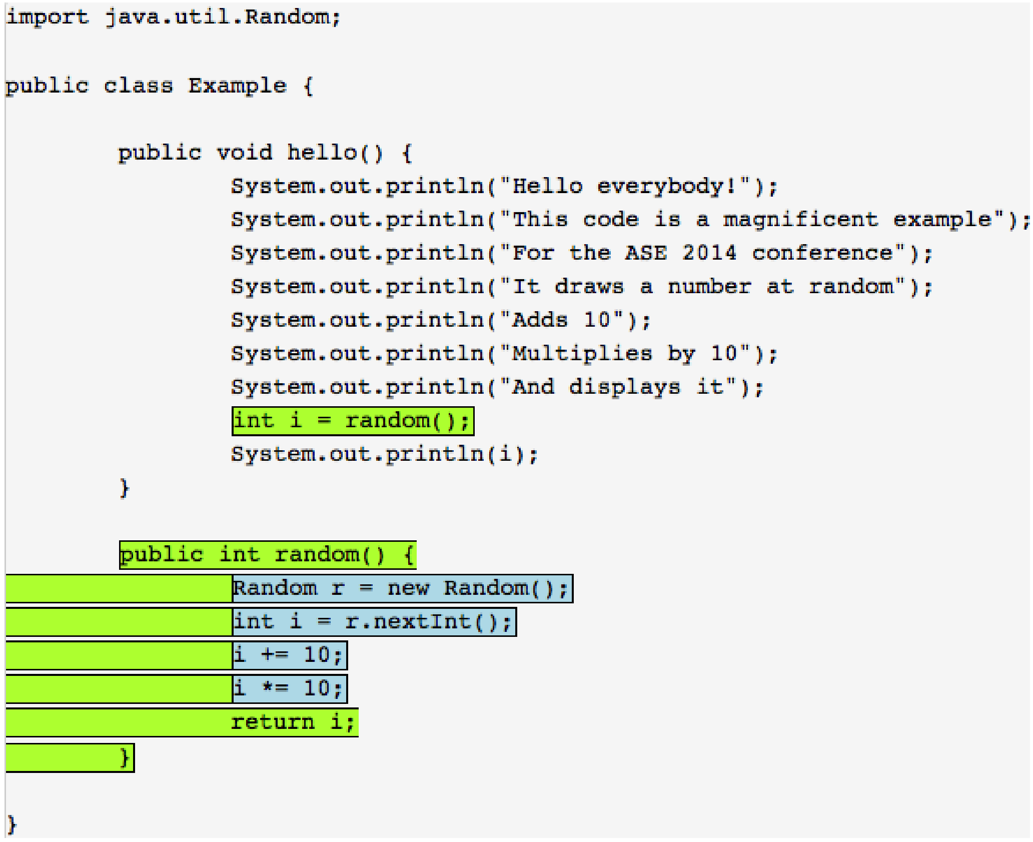
- Studies should have a good internal validity
- Studies should have a good external validity
- Having both at the same time is a nightmare
- Solution : replication
  - Replicating a study is hard
  - Tools have to be available
  - Datasets have to be available
  - We ensure that our tools and datasets are available

Future work & conclusion

Change clusters



```
import java.util.Random;

public class Example {

        public void hello() {
                System.out.println("Hello everybody!");
                System.out.println("This code is a magnificent example");
                System.out.println("For the ASE 2014 conference");
                System.out.println("It draws a number at random");
                System.out.println("Adds 10");
                System.out.println("Multiplies by 10");
                System.out.println("And displays it");
                Random r = new Random();
                int i = r.nextInt();
                i += 10;
                i *= 10;
                System.out.println(i);
        }

}
```



```
import java.util.Random;

public class Example {

        public void hello() {
                System.out.println("Hello everybody!");
                System.out.println("This code is a magnificent example");
                System.out.println("For the ASE 2014 conference");
                System.out.println("It draws a number at random");
                System.out.println("Adds 10");
                System.out.println("Multiplies by 10");
                System.out.println("And displays it");
                int i = random();
                System.out.println(i);
        }

        public int random() {
                Random r = new Random();
                int i = r.nextInt();
                i += 10;
                i *= 10;
                return i;
        }

}
```

Change clusters

```
01 Insert MethodDeclaration (84) into TypeDeclaration(85) at 3
02 Insert Modifier:public(55) into MethodDeclaration (84) at 0
03 Insert PrimitiveType:int(56) into MethodDeclaration(84) at 1
04 Insert SimpleName:random(57) into MethodDeclaration(84) at 2
05 Insert Block(83) into MethodDeclaration(84) at 3
06 Insert VariableDeclarationStatement(47) into Block(53) at 7
07 Move VariableDeclarationStatement(49) into Block(83) at 0
08 Move VariableDeclarationStatement(56) into Block(83) at 1
09 Move ExpressionStatement(60) into Block(83) at 2
10 Move ExpressionStatement(64) into Block(83) at 3
11 Insert ReturnStatement(82) into Block(83) at 4
12 Insert PrimitiveType:int(42) into VariableDeclarationStatement(47) at 0
13 Insert VariableDeclarationFragment(46) into VariableDeclarationStatement(47) at 1
14 Insert SimpleName:i(81) into ReturnStatement(82) at 0
15 Insert SimpleName:i(43) into VariableDeclarationFragment(46) at 0
16 Insert MethodInvocation(45) into VariableDeclarationFragment(46) at 1
17 Insert SimpleName:random(44) into MethodInvocation(45) at 0
```

Change clusters

```
01 Insert MethodDeclaration (84) into TypeDeclaration(85) at 3
02 Insert Modifier:public(55) into MethodDeclaration (84) at 0
03 Insert PrimitiveType:int(56) into MethodDeclaration(84) at 1
04 Insert SimpleName:random(57) into MethodDeclaration(84) at 2
05 Insert Block(83) into MethodDeclaration(84) at 3
06 Insert VariableDeclarationStatement(47) into Block(53) at 7
```

New method

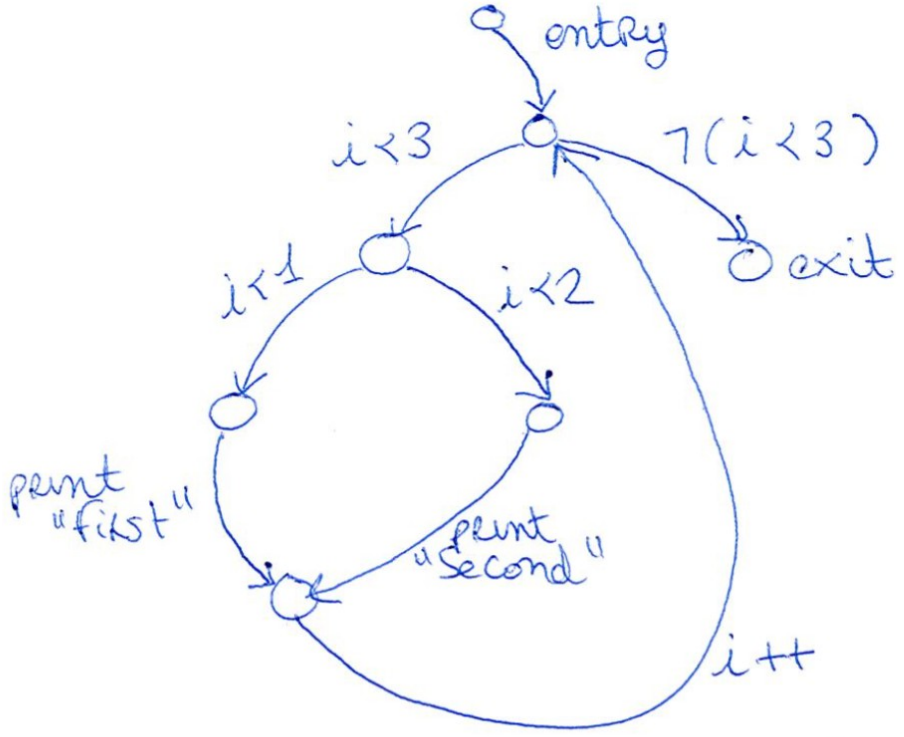
```
07 Move VariableDeclarationStatement(49) into Block(83) at 0
08 Move VariableDeclarationStatement(56) into Block(83) at 1
09 Move ExpressionStatement(60) into Block(83) at 2
10 Move ExpressionStatement(64) into Block(83) at 3
```

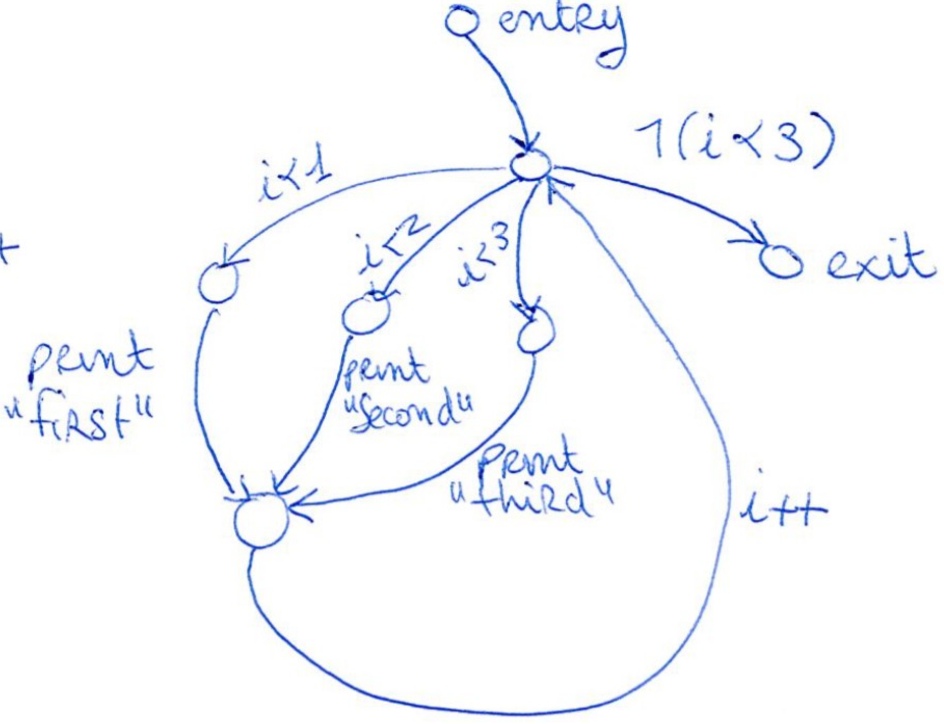
Move statements

```
11 Insert ReturnStatement(82) into Block(83) at 4
12 Insert PrimitiveType:int(42) into VariableDeclarationStatement(47) at 0
13 Insert VariableDeclarationFragment(46) into VariableDeclarationStatement(47) at 1
14 Insert SimpleName:i(81) into ReturnStatement(82) at 0
15 Insert SimpleName:i(43) into VariableDeclarationFragment(46) at 0
16 Insert MethodInvocation(45) into VariableDeclarationFragment(46) at 1
17 Insert SimpleName:random(44) into MethodInvocation(45) at 0
```

Call to new method

Programs as graphs





Graph differencing

- Find the right model
  - labeled digraph?
- Find the right set of actions
  - Insert node ?
  - Disconnect edge ?
  - ...
- Find heuristics to compute solutions
  - That make sense to software developers!

Applications to software evolution

- Empirical studies on how developers use syntactic constructs during evolution
  - Do Java developers use the new lambdas?
  - How code documentation is realized in practice?
- Better tools to assess impact of changes
  - On syntax
  - On execution
  - On tests
  - …

Wrapping up

- Source code differencing is essential to software evolution research
- Existing approaches to differentiate code are limited
- GumTree, an improved source code differencing approach
- Successfully applied in several software evolution research scenarios