Match algorithm description

Bruno Crotman

18/04/2020

1 Introduction

This document is part of a bigger project. The project is a research about software degradation caused by careless developers' behavior and strategies to deal with it. The strategies to deal with the problem will possibly be inspired by concepts from game theory. At this moment, we think that software degradation can be measured by the number and the types of kludges made by software developers in the code. So, one of the goals of this project at this moment is to study how software projects evolve in terms of number of kludges and kinds of kludges.

Right now we are trying to identify kludges looking at alerts generated by PMD source code analyzer. This tool receives a programming code as input and generates a list of bad programming practices contained in the code.

In order to evaluate how the number of alerts is evolving along the history of a software project, we must be able to analyze two different versions of a code (an old version and a new version) and cateborize each alert as **new**, **fixed** or **not fixed**.

- each PMD alert generated for the old version is either **not fixed** or **fixed** in the new version. When an alert is **not fixed**, it means that it remains in the new version of the code. When an alert is **fixed**, it means that it does not exist anymore in the new version.
- each PMD alert generated for the new version is **not fixed** or **new**. When an alert is **not fixed** it means that the same alert was identified in the old version. When an alert is **new**, it means that the same alert cannot be identified in the old version.

The alerts identified as *not fixed* are the equivalent in both new and old versions. And the intersection between the *fixed* alerts, the *new* alerts and the *not fixed* alerts is empty.

In order to decide if an alert is *not fixed*, *fixed* or *new*, we have to identify if an alert in the old version is equivalent to an alert in the new version.

I describe here two algorithms. The first one, described in Section 2, is a naive algorithm based on matches by lines of code. The second is a more sophisticated algorithm, based on matches by blocks of code.

2 Matches by line of code

In this first algorithm, I match the lines of code of the old version with the lines of code of the new version using information from the output of git's diff command. When an alert with the same features occurs in both matched lines, this alert is declared *not fixed.* The alerts that occur in a not matched line of the old version are declared *fixed* and the alerts located in a not matched line of the new version are declared *new*

These are the steps of the algorithm:

1. Generate a list of alerts from each version (old and new) using PMD Alert

2. Generate the git diff between the two versions
3. Using information from git diff, create a map between the lines

2.1 Generate a list of alerts for each version

The two codes presented in this Section, named “new version” and “old version”, are used in Sections 2.2, 2.3 and 2.4 to describe the algorithm.

The old version, with the alerts generated by PMI, is shown in Figure 1.

```
/*  1-                                  */package twitter4j;
/*  2-                                  */
/*  3-UnusedImports                     */import java.util.concurrent.ConcurrentHashMap;
/*  4-                                  */
/*  5-                                  */class TwitterImpl extends TwitterBaseImpl implements Twitter {
/*  6-                                  */    private static final long serialVersionUID = 9170943084096085770L;
/*  7-UnusedPrivateField                */    private static final Logger logger = Logger.getLogger(TwitterBaseImpl.class);
/*  8-                                  */
/*  9-                                  */    /*package*/
/* 10-                                  */    TwitterImpl(Configuration conf, Authorization auth) {
/* 11-                                  *//* ...  */
/* 45-                                  *//* ...  */
/* 46-                                  */            if (conf.isTweetModeExtended()) {
/* 47-                                  */                params.add(new HttpParameter("tweet_mode", "extended"));
/* 48-                                  */            }
/* 49-OptimizableToArrayCall            */            HttpParameter[] implicitParams = params.toArray(new HttpParameter[params.size()]);
/* 50-                                  */
/* 51-                                  */            // implicitParamsMap.containsKey() is evaluated in the above if clause.
/* 52-                                  */            // thus implicitParamsStrMap needs to be initialized first
/* 53-                                  *//* ...  */
/* 59-                                  *//* ...  */
/* 60-                                  */
/* 61-                                  */
/* 62-                                  */    @Override
/* 63-FormalParameterNamingConventions   */    public AccountSettings updateAccountSettings(Integer trend_locationWoeid,
/* 64-FormalParameterNamingConventions(2)*/                                                 Boolean sleep_timeEnabled, String start_sleepTime,
/* 65-FormalParameterNamingConventions(2)*/                                                 String end_sleepTime, String time_zone, String lang)
/* 66-                                  */            throws TwitterException {
/* 67-                                  */        List<HttpParameter> profile = new ArrayList<HttpParameter>(6);
/* 68-                                  */        if (trend_locationWoeid != null) {
/* 69-                                  *//* ...  */
/* 83-                                  *//* ...  */
/* 84-                                  */            profile.add(new HttpParameter("lang", lang));
/* 85-                                  */        }
/* 86-                                  */        return factory.createAccountSettings(post(conf.getRestBaseURL() + "account/settings.json"
/* 87-OptimizableToArrayCall            */                , profile.toArray(new HttpParameter[profile.size()])));
/* 88-                                  */
/* 89-                                  */    }
/* 90-                                  */
/* 91-                                  */}
```

Figure 1: Example: old version

Table 1 lists the alerts found in the old version.

Table 1: Alerts in the old version

id	beginline	ruleset	rule	package	class	method	variable
1	3	Best Practices	UnusedImports	twitter4j	TwitterImpl	No method	No variable
2	7	Best Practices	UnusedPrivateField	twitter4j	TwitterImpl	No method	logger
3	49	Performance	OptimizableToArrayCall	twitter4j	TwitterImpl	TwitterImpl	No variable
4	63	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	trend_locationWoeid
5	64	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	sleep_timeEnabled
6	64	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	start_sleepTime
7	65	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	end_sleepTime
8	65	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	time_zone
9	87	Performance	OptimizableToArrayCall	twitter4j	TwitterImpl	updateAccountSettings	No variable

The new version (Figure 2) has the following changes in relation to the old version:

- line 24 of the old version “import java.util.concurrent.ConcurrentHashMap”, that was a “Unused Import”, was removed. So the alert related to this line must be declared *fixed*.
- line 92 of the old version (91 of the new version) was fixed by changing the name of the parameter. This must be classified as another *fixed* alert.
- line 119 was included in the new version, with an unused private field. This must be categorized as a *new* alert

```
/*  1-                                   */package twitter4j;
/*  2-                                   */
/*  3-                                   */class TwitterImpl extends TwitterBaseImpl implements Twitter {
/*  4-                                   */    private static final long serialVersionUID = 9170943084096085770L;
/*  5-UnusedPrivateField                 */    private static final Logger logger = Logger.getLogger(TwitterBaseImpl.class);
/*  6-                                   */
/*  7-                                   */    /*package*/
/*  8-                                   */    TwitterImpl(Configuration conf, Authorization auth) {
/*  9-                                   *//* ...  */
/* 43-                                   *//* ...  */
/* 44-                                   */            if (conf.isTweetModeExtended()) {
/* 45-                                   */                params.add(new HttpParameter("tweet_mode", "extended"));
/* 46-                                   */            }
/* 47-OptimizableToArrayCall             */            HttpParameter[] implicitParams = params.toArray(new HttpParameter[params.size()]);
/* 48-                                   */
/* 49-                                   */            // implicitParamsMap.containsKey() is evaluated in the above if clause.
/* 50-                                   */            // thus implicitParamsStrMap needs to be initialized first
/* 51-                                   *//* ...  */
/* 58-                                   *//* ...  */
/* 59-                                   */
/* 60-                                   */    @Override
/* 61-                                   */    public AccountSettings updateAccountSettings(Integer trendlocationWoeid,
/* 62-FormalParameterNamingConventions(2)*/                                                 Boolean sleep_timeEnabled, String start_sleepTime,
/* 63-FormalParameterNamingConventions(2)*/                                                 String end_sleepTime, String time_zone, String lang)
/* 64-                                   */            throws TwitterException {
/* 65-                                   */        List<HttpParameter> profile = new ArrayList<HttpParameter>(6);
/* 66-                                   */        if (trendlocationWoeid != null) {
/* 67-                                   *//* ...  */
/* 81-                                   *//* ...  */
/* 82-                                   */            profile.add(new HttpParameter("lang", lang));
/* 83-                                   */        }
/* 84-                                   */        return factory.createAccountSettings(post(conf.getRestBaseURL() + "account/settings.json"
/* 85-OptimizableToArrayCall             */                , profile.toArray(new HttpParameter[profile.size()])));
/* 86-                                   */
/* 87-                                   */    }
/* 88-                                   */
/* 89-UnusedPrivateField                 */    private int not_used = 0;
/* 90-                                   */
/* 91-                                   */}
```

Figure 2: Example: new version

Table 2 lists the alerts found in the new version.

Table 2: Alerts in the new version

id	beginline	ruleset	rule	package	class	method	variable
1	5	Best Practices	UnusedPrivateField	twitter4j	TwitterImpl	No method	logger
2	47	Performance	OptimizableToArrayCall	twitter4j	TwitterImpl	TwitterImpl	No variable
3	62	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	sleep_timeEnabled
4	62	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	start_sleepTime
5	63	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	end_sleepTime
6	63	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	time_zone
7	85	Performance	OptimizableToArrayCall	twitter4j	TwitterImpl	updateAccountSettings	No variable
8	89	Best Practices	UnusedPrivateField	twitter4j	TwitterImpl	No method	not_used

2.2 Generate the git diff between the two versions

The command git diff is executed between the two versions with the option –patience.

The result of the git diff operation between this two versions is shown in Figure 3.

```
5   5   j/match_algorithm_description/{old => new}/code.java

diff --git a/j/match_algorithm_description/old/code.java b/j/match_algorithm_description/new/code.java
index cd61181..245a6e8 100644
--- a/j/match_algorithm_description/old/code.java
+++ b/j/match_algorithm_description/new/code.java
@@ -3,2 +2,0 @@ package twitter4j;
-import java.util.concurrent.ConcurrentHashMap;
-
@@ -63 +61 @@ class TwitterImpl extends TwitterBaseImpl implements Twitter {
-    public AccountSettings updateAccountSettings(Integer trend_locationWoeid,
+    public AccountSettings updateAccountSettings(Integer trendlocationWoeid,
@@ -68,2 +66,2 @@ class TwitterImpl extends TwitterBaseImpl implements Twitter {
-        if (trend_locationWoeid != null) {
-            profile.add(new HttpParameter("trend_location_woeid", trend_locationWoeid));
+        if (trendlocationWoeid != null) {
+            profile.add(new HttpParameter("trend_location_woeid", trendlocationWoeid));
@@ -90,0 +89,2 @@ class TwitterImpl extends TwitterBaseImpl implements Twitter {
+    private int not_used = 0;
+
```

Figure 3: Git diff between code in Figure 1 and code in Figure 2

2.3 Using information from git diff, create a map between the lines

Using this information from git's diff, it's possible to create a map between the lines of the old version and the new version.

For each difference stated in the git diff output (the sections of the diff file starting with "@@"), there is an indication of the number of lines removed from the old version and the number of lines added to the new version. The line in which the lines are removed from the old version and the line at which the lines are added is indicated, too. Following this information it´s possible to create a map between the lines of the old version and the equivalent lines in the new version.

For the new and old versions presented in Section 2.1, the map is shown in Table 3

Table 3: Map between lines of the old version and lines of the new version

old	new
1	1
2	2
3	NA
4	NA
5	3
6	4
...	...
...	...
60	58
61	59
NA	61
62	60
63	NA
64	62
65	63
66	64
NA	66
67	65
NA	67
68	NA
69	NA
70	68
71	69
...	...
...	...
88	86
89	87
NA	89
90	88
NA	90
91	91

Now we can connect the two versions and plugin the alerts as we see in Figure 4.

```
/*  1-                       */package twitter4j;                                                                        /*  1-                       */package twitter4j;
/*  2-                       */                                                                                          /*  2-                       */
/*  3-UnusedImports          */import java.util.concurrent.ConcurrentHashMap;                                            /*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*  4-                       */                                                                                          /*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*  5-                       */class TwitterImpl extends TwitterBaseImpl implements Twitter {                            /*  3-                       */class TwitterImpl extends TwitterBaseImpl implements Twitter {
/*  6-                       */    private static final long serialVersionUID = 9170943084096085770L;                    /*  4-                       */    private static final long serialVersionUID = 9170943084096085770L;
/*  7-UnusedPrivateField     */    private static final Logger logger = Logger.getLogger(TwitterBaseImpl.class);         /*  5-UnusedPrivateField     */    private static final Logger logger = Logger.getLogger(TwitterBaseImpl.class);
/*  8-                       */                                                                                          /*  6-                       */
/*  9-                       */    /*package*/                                                                           /*  7-                       */    /*package*/
/* 10-                       */    TwitterImpl(Configuration conf, Authorization auth) {                                 /*  8-                       */    TwitterImpl(Configuration conf, Authorization auth) {
/* 11-                       *//* ...  */                                                                                /*  9-                       *//* ...  */
/* 45-                       *//* ...  */                                                                                /* 43-                       *//* ...  */
/* 46-                       */            if (conf.isTweetModeExtended()) {                                             /* 44-                       */            if (conf.isTweetModeExtended()) {
/* 47-                       */                params.add(new HttpParameter("tweet_mode", "extended"));                  /* 45-                       */                params.add(new HttpParameter("tweet_mode", "extended"));
/* 48-                       */            }                                                                             /* 46-                       */            }
/* 49-OptimizableToArrayCa   */            HttpParameter[] implicitParams = params.toArray(new HttpParameter[params.size(/* 47-OptimizableToArrayCa   */            HttpParameter[] implicitParams = params.toArray(new HttpParameter[params.size(
/* 50-                       */                                                                                          /* 48-                       */
/* 51-                       */            // implicitParamsMap.containsKey() is evaluated in the above if clause.       /* 49-                       */            // implicitParamsMap.containsKey() is evaluated in the above if clause.
/* 52-                       */            // thus implicitParamsStrMap needs to be initialized first                    /* 50-                       */            // thus implicitParamsStrMap needs to be initialized first
/* 53-                       *//* ...  */                                                                                /* 51-                       *//* ...  */
/* 60-                       *//* ...  */                                                                                /* 58-                       *//* ...  */
/* 61-                       */                                                                                          /* 59-                       */
/*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                                                /* 61-                       */    public AccountSettings updateAccountSettings(Integer trendlocationWoeid,
/* 62-                       */    @Override                                                                             /* 60-                       */    @Override
/* 63-FormalParameterNamin   */    public AccountSettings updateAccountSettings(Integer trend_locationWoeid,             /*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 64-FormalParameterNamin(2)*/                                                 Boolean sleep_timeEnabled, String start_s/* 62-FormalParameterNamin(2)*/                                                 Boolean sleep_timeEnabled, String start_s
/* 65-FormalParameterNamin(2)*/                                                 String end_sleepTime, String time_zone, S/* 63-FormalParameterNamin(2)*/                                                 String end_sleepTime, String time_zone, S
/* 66-                       */            throws TwitterException {                                                     /* 64-                       */            throws TwitterException {
/*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                                                /* 66-                       */        if (trendlocationWoeid != null) {
/* 67-                       */        List<HttpParameter> profile = new ArrayList<HttpParameter>(6);                    /* 65-                       */        List<HttpParameter> profile = new ArrayList<HttpParameter>(6);
/*   -                       *//* ...  */                                                                                /* 67-                       *//* ...  */
/* 83-                       *//* ...  */                                                                                /* 81-                       *//* ...  */
/* 84-                       */            profile.add(new HttpParameter("lang", lang));                                 /* 82-                       */            profile.add(new HttpParameter("lang", lang));
/* 85-                       */        }                                                                                 /* 83-                       */        }
/* 86-                       */        return factory.createAccountSettings(post(conf.getRestBaseURL() + "account/setting/* 84-                       */        return factory.createAccountSettings(post(conf.getRestBaseURL() + "account/setting
/* 87-OptimizableToArrayCa   */                , profile.toArray(new HttpParameter[profile.size()])));                   /* 85-OptimizableToArrayCa   */                , profile.toArray(new HttpParameter[profile.size()])));
/* 88-                       */                                                                                          /* 86-                       */
/* 89-                       */    }                                                                                     /* 87-                       */    }
/*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                                                /* 89-UnusedPrivateField     */    private int not_used = 0;
/* 90-                       */                                                                                          /* 88-                       */
/*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                                                /* 90-                       */
/* 91-                       */}                                                                                         /* 91-                       */}
```

Figure 4: Comparison between old and new version

2.4 Categorize the alerts

The alert in the old version is classified as **not fixed** if there is an alert in the new version in the corresponding line with the same rule, same method name and same variable name. Otherwise, the alert is categorised as **fixed**

Table 4 shows the classification of the alerts in the old version. For alerts 1 and 4, it is not possible to find an alert with the same rule, same method name and same variable name in a new version's line that is mapped to the line in the old version. So these alerts are classified as **fixed**.

For each other alert, it is possible to find an alert with the same rule, same method name and same variable name in the new version's line that is mapped to the original line. So these alerts are classified as **not fixed**.

Table 4: Classifications of the alerts in the old version

id	line	rule	class	method	variable	idnew	linenew	category
1	3	UnusedImports	TwitterImpl	No method	No variable	NA	NA	Fixed
2	7	UnusedPrivateField	TwitterImpl	No method	logger	1	5	Not fixed
3	49	OptimizableToArrayCall	TwitterImpl	TwitterImpl	No variable	2	47	Not fixed
4	63	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	trend_locationWoeid	NA	NA	Fixed
5	64	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	sleep_timeEnabled	3	62	Not fixed
6	64	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	start_sleepTime	4	62	Not fixed
7	65	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	end_sleepTime	5	63	Not fixed
8	65	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	time_zone	6	63	Not fixed
9	87	OptimizableToArrayCall	TwitterImpl	updateAccountSettings	No variable	7	85	Not fixed

Table 5 shows the classification of the alerts in the new version. The alert 8 is the only one for which there is no alert with the same characteristics in a line of the old version that is mapped to the original line in the new version. So it is the only **new** alert.

Table 5: Classifications of the alerts in the new version

id	line	rule	class	method	variable	idold	lineold	category
1	5	UnusedPrivateField	TwitterImpl	No method	logger	2	7	Not fixed
2	47	OptimizableToArrayCall	TwitterImpl	TwitterImpl	No variable	3	49	Not fixed
3	62	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	sleep_timeEnabled	5	64	Not fixed
4	62	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	start_sleepTime	6	64	Not fixed
5	63	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	end_sleepTime	7	65	Not fixed
6	63	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	time_zone	8	65	Not fixed
7	85	OptimizableToArrayCall	TwitterImpl	updateAccountSettings	No variable	9	87	Not fixed
8	89	UnusedPrivateField	TwitterImpl	No method	not_used	NA	NA	New

3 Matches using the Abstract Syntax Tree and the mapping of lines of code

3.1 Creating an Abstract Syntax Tree

The Abstract Syntax Tree (AST) of a programming code contains the elements represented in the code, such as class declarations, method declarations, statements. The AST can be used, in conjunction with the mapping lines we saw in Section 2.3, to understand the location of an alert in a version of a code.

The PMD Alert tool lets us configure our own rules. In section 2.1 we generated alerts using the default configuration of alerts.

The alerts that PMD generates are elements that the tool captures as it is traversing the code, visiting all the elements of the AST. An element is captured and becomes an alert when it is matched with a rule. The rules are defined in a XML file.

In Figure 5, we can see an example of a simple code and the alerts that were generated by the default ruleset of PMD alerts tool.

```
/*  1-                                   */package pack_x;
/*  2-                                   */
/*  3-                                   */import importX.function;
/*  4-                                   */
/*  5-                                   */class ClassX extends ClassY implements InterfX {
/*  6-                                   */    private long fieldX;
/*  7-                                   */
/*  8-                                   */    ClassX(int paramX, double paramY) {
/*  9-                                   */        int varX = function(paramX, paramY);
/* 10-                                   */        if (varX == 0)
/* 11-ControlStatementBraces             */            this.fieldX = 1;
/* 12-                                   */        else{
/* 13-                                   */            this.fieldX = 0;
/* 14-                                   */     }
/* 15-                                   */    }
/* 16-                                   */    @Override
/* 17-                                   */    public int methodX(int paramW, Boolean paramZ)
/* 18-                                   */    {
/* 19-                                   */        if (paramZ)
/* 20-ControlStatementBraces             */            fieldX = paramW;
/* 21-                                   */        else{
/* 22-                                   */            fieldX = 0;
/* 23-                                   */     }
/* 24-                                   */        return paramW + this.fieldX;
/* 25-                                   */     }
/* 26-                                   */}
```

Figure 5: Simple code with its alerts

It's possible to create custom rules for PMD. Figure 6 shows the designer tool that helps a user to create custom rules.

We can see that the tool traverses the programming code visiting many different kinds of elements. If we build our own simple rules, aimed only to capture some kinds of elements, we will generate list of "alerts" that will contain all the elements of the chosen kinds.

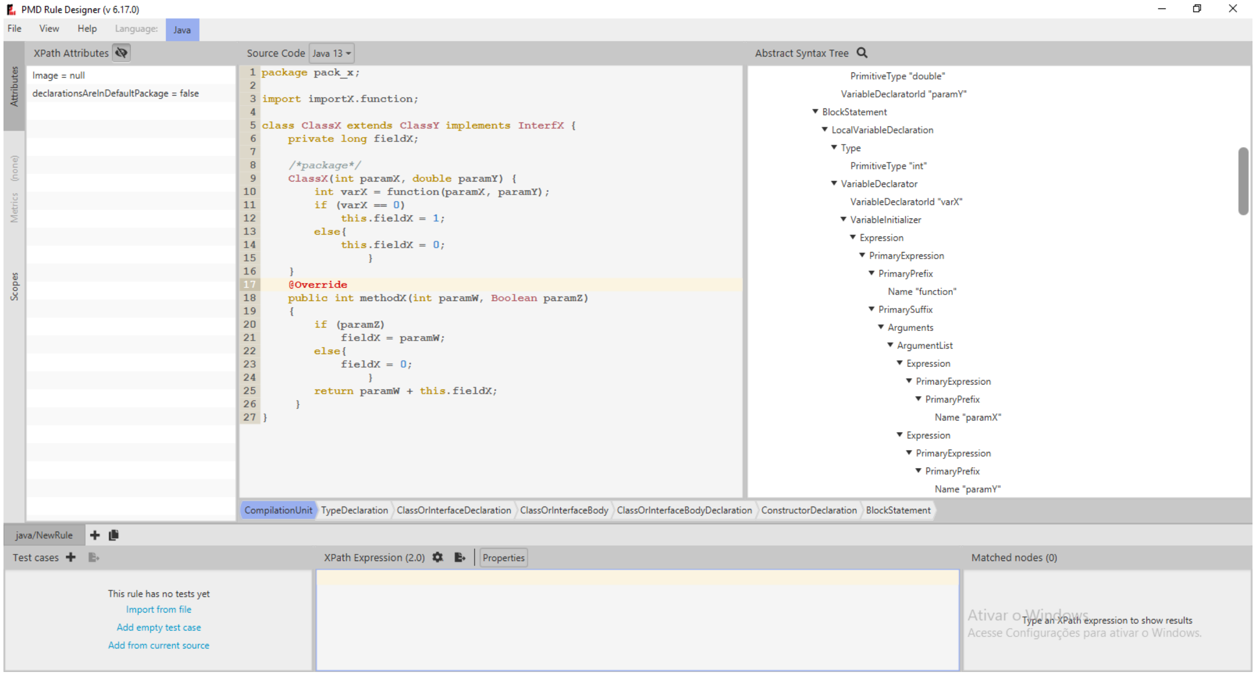

Image = null
declarationsAreInDefaultPackage = false

```java
package pack_x;

import importX.function;

class ClassX extends ClassY implements InterfX {
    private long fieldX;

    /*package*/
    ClassX(int paramX, double paramY) {
        int varX = function(paramX, paramY);
        if (varX == 0)
            this.fieldX = 1;
        else{
            this.fieldX = 0;
                }
    }
    @Override
    public int methodX(int paramW, Boolean paramZ)
    {
        if (paramZ)
            fieldX = paramW;
        else{
            fieldX = 0;
                }
        return paramW + this.fieldX;
     }
}
```

```
        PrimitiveType "double"
      VariableDeclaratorId "paramY"
▼ BlockStatement
  ▼ LocalVariableDeclaration
    ▼ Type
        PrimitiveType "int"
    ▼ VariableDeclarator
        VariableDeclaratorId "varX"
      ▼ VariableInitializer
        ▼ Expression
          ▼ PrimaryExpression
            ▼ PrimaryPrefix
                Name "function"
            ▼ PrimarySuffix
              ▼ Arguments
                ▼ ArgumentList
                  ▼ Expression
                    ▼ PrimaryExpression
                      ▼ PrimaryPrefix
                          Name "paramX"
                  ▼ Expression
                    ▼ PrimaryExpression
                      ▼ PrimaryPrefix
                          Name "paramY"
```

Figure 6: PMD Designer tool

In Figure 7, we show an example of a ruleset that captures all the method declarations.

```xml
<?xml version="1.0"?>
<ruleset name="complete"
         xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0 https://pmd.sourceforge.io/ruleset_2_0_0.xsd"
    <description>Test.</description>

    <rule name="method"
      language="java"
      message="method"
      class="net.sourceforge.pmd.lang.rule.XPathRule" >
    <description>
TODO
    </description>
    <priority>3</priority>
    <properties>
        <property name="xpath">
            <value>
<![CDATA[
//MethodDeclaration
]]>
            </value>
        </property>
    </properties>
    </rule>
]

</ruleset>
```

Figure 7: Custom ruleset for PMD alerts tool

Table 6 shows the kinds of elements that were select for the creation of an AST for the code in Figure 5.

Table 6: Kinds of elements selected

rule_number	rule
1	annotation
2	block
3	class_or_interface_body
4	class_or_interface_declaration
5	class_or_interface_type
6	compilation_unit
7	constructor_declaration
8	extends_list
9	field_declaration
10	formal_parameter
11	formal_parameters
12	if_statement
13	implements_list
14	import_declaration
15	method
16	name
17	package
18	statement
19	type_declaration
20	variable_id

If we select the list in Table 6, the simple code shown in this Section captures the elements shown in Table 7

Table 6 contains the list and location of the elements of the AST. In order to recreate the AST, we must follow three steps:

1. Link each element a to the set of elements X that are fully located between the begin line / begin column and end line / end column of element a. We can construct a directed graph in which the elements are the nodes and the links are the edges. This is not a tree yet, because each node will have edges directed to all its descendents and not only its children in the AST.
2. Sort the nodes in the decreasing order of its number of childs. The objective is to establish that, in a search through this graph, the first child chosen will be the one that is a child in the AST, and not only on this graph.
3. Proceed a deep-first search starting from the compilation unit node.

Table 7: Elements captured in code

line	endline	col	endcol	rule	method	code
1	26	1	3	compilation_unit	No method	package pack_x; import i...
1	1	1	15	package	No method	package pack_x;
1	1	9	14	name	No method	pack_x
3	3	1	24	import_declaration	No method	import importX.function;
3	3	8	23	name	No method	importX.function
5	26	1	1	class_or_interface_declaration	No method	class ClassX extends ClassY...
5	5	14	27	extends_list	No method	extends ClassY
5	5	22	27	class_or_interface_type	No method	ClassY
5	5	29	46	implements_list	No method	implements InterfX
5	5	40	46	class_or_interface_type	No method	InterfX
5	26	48	1	class_or_interface_body	No method	{ private long fieldX; ...
6	6	13	24	field_declaration	No method	long fieldX;
6	6	18	23	variable_id	No method	fieldX
8	15	5	5	constructor_declaration	ClassX	ClassX(int paramX, double p...
8	8	11	37	formal_parameters	ClassX	(int paramX, double paramY)
8	8	12	21	formal_parameter	ClassX	int paramX
8	8	16	21	variable_id	ClassX	paramX
8	8	24	36	formal_parameter	ClassX	double paramY
8	8	31	36	variable_id	ClassX	paramY
9	9	13	16	variable_id	ClassX	varX
9	9	20	27	name	ClassX	function
9	9	29	34	name	ClassX	paramX
9	9	37	42	name	ClassX	paramY
10	14	9	17	statement	ClassX	if (varX == 0) ...
10	14	9	17	if_statement	ClassX	if (varX == 0) ...
10	10	13	16	name	ClassX	varX
11	11	13	28	statement	ClassX	this.fieldX = 1;
12	14	13	17	block	ClassX	{ this.fieldX =...
12	14	13	17	statement	ClassX	{ this.fieldX =...
13	13	13	28	statement	ClassX	this.fieldX = 0;
16	16	5	13	annotation	No method	@Override
16	16	6	13	name	No method	Override
17	25	12	6	method	methodX	int methodX(int paramW, Boo...
17	17	23	50	formal_parameters	methodX	(int paramW, Boolean paramZ)
17	17	24	33	formal_parameter	methodX	int paramW
17	17	28	33	variable_id	methodX	paramW
17	17	36	42	class_or_interface_type	methodX	Boolean
17	17	36	49	formal_parameter	methodX	Boolean paramZ
17	17	44	49	variable_id	methodX	paramZ
18	25	5	6	block	methodX	{ if (paramZ) ...
19	23	9	17	statement	methodX	if (paramZ) fie...
19	23	9	17	if_statement	methodX	if (paramZ) fie...
19	19	13	18	name	methodX	paramZ
20	20	13	28	statement	methodX	fieldX = paramW;
20	20	13	18	name	methodX	fieldX
20	20	22	27	name	methodX	paramW
21	23	13	17	block	methodX	{ fieldX = 0; }
21	23	13	17	statement	methodX	{ fieldX = 0; }
22	22	13	23	statement	methodX	fieldX = 0;
22	22	13	18	name	methodX	fieldX
24	24	9	36	statement	methodX	return paramW + this.fieldX;
24	24	16	21	name	methodX	paramW

After we follow these steps, we come up with the AST as we se in Figure 8.

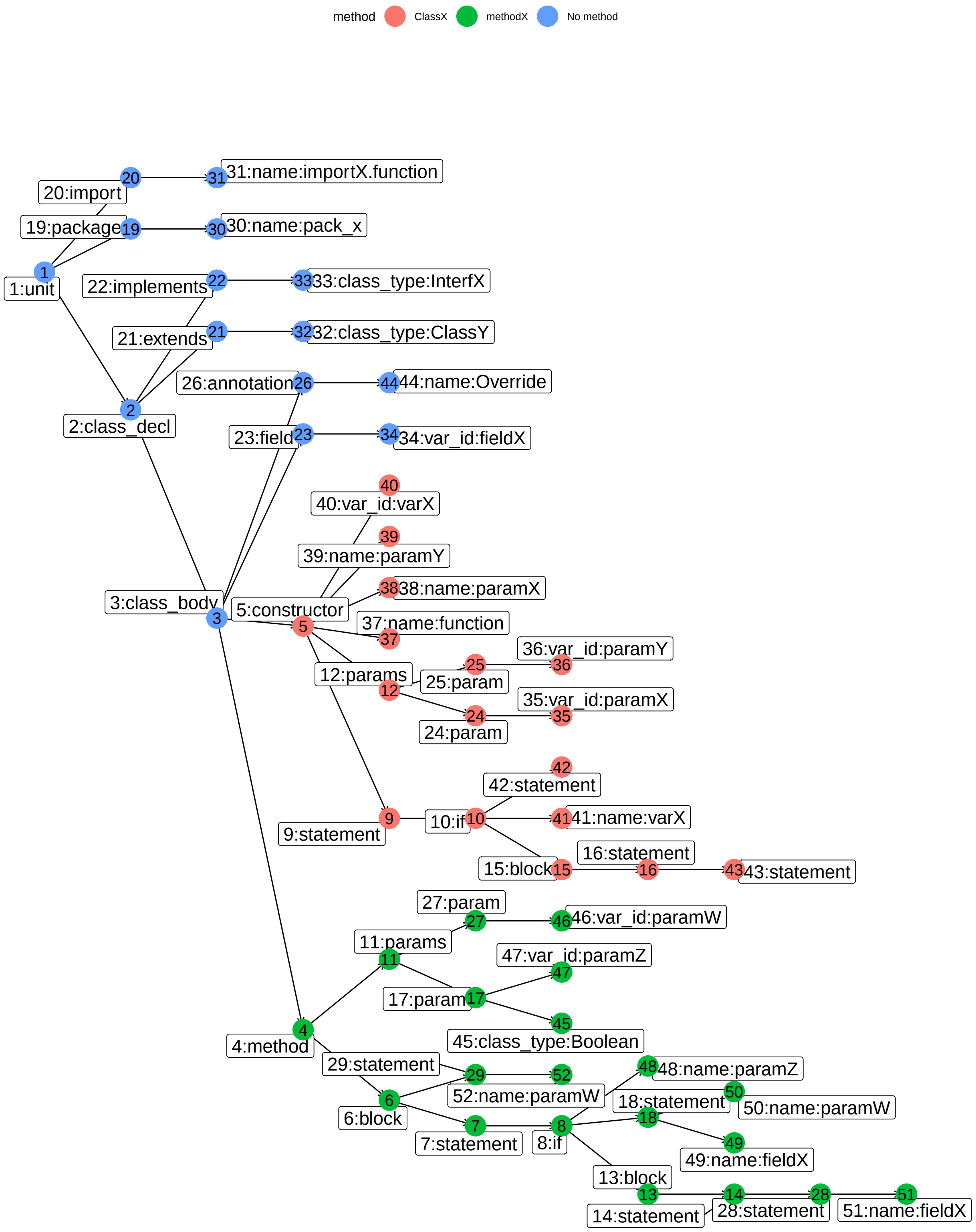

Figure 8: Abstract Syntax Tree