CSE 5382 Secure Programming Fall 2022

Programming Assignment 1 - Introduction to Static Analysis

Prem Atul Jethwa | paj1810| 1001861810

Tool Versions

The tools I have used to test the code are:

- FindBugs – 3.0.1, findsecbugs-plugin-1.8.0
- SonarQube

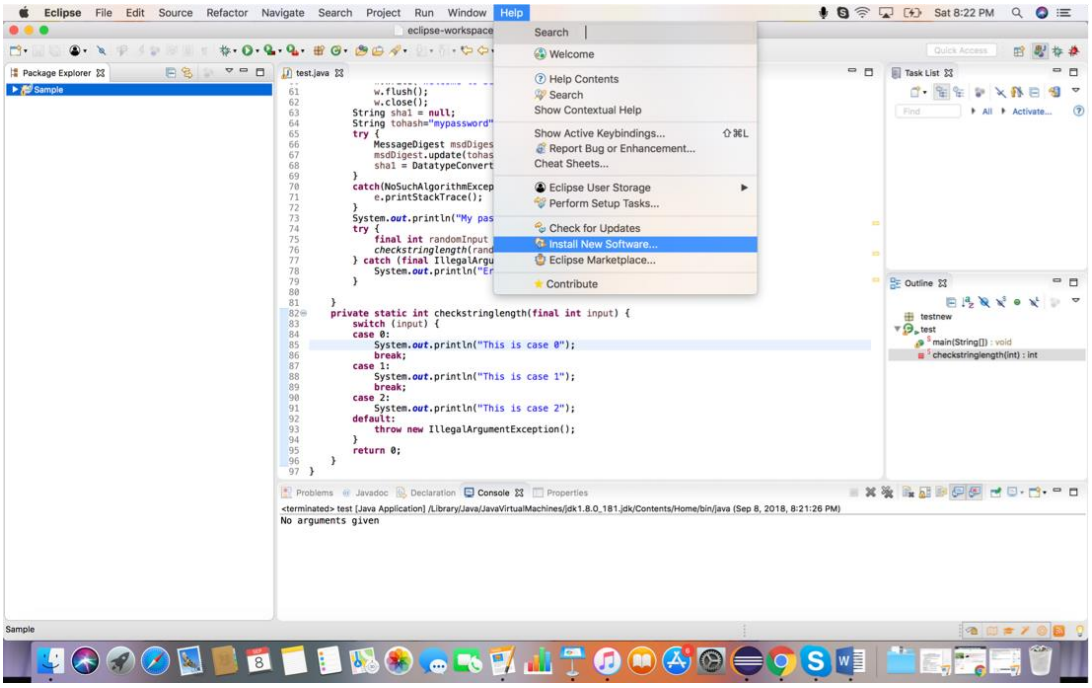
FindBugs – 3.0.1, findsecbugs-plugin-1.8.0

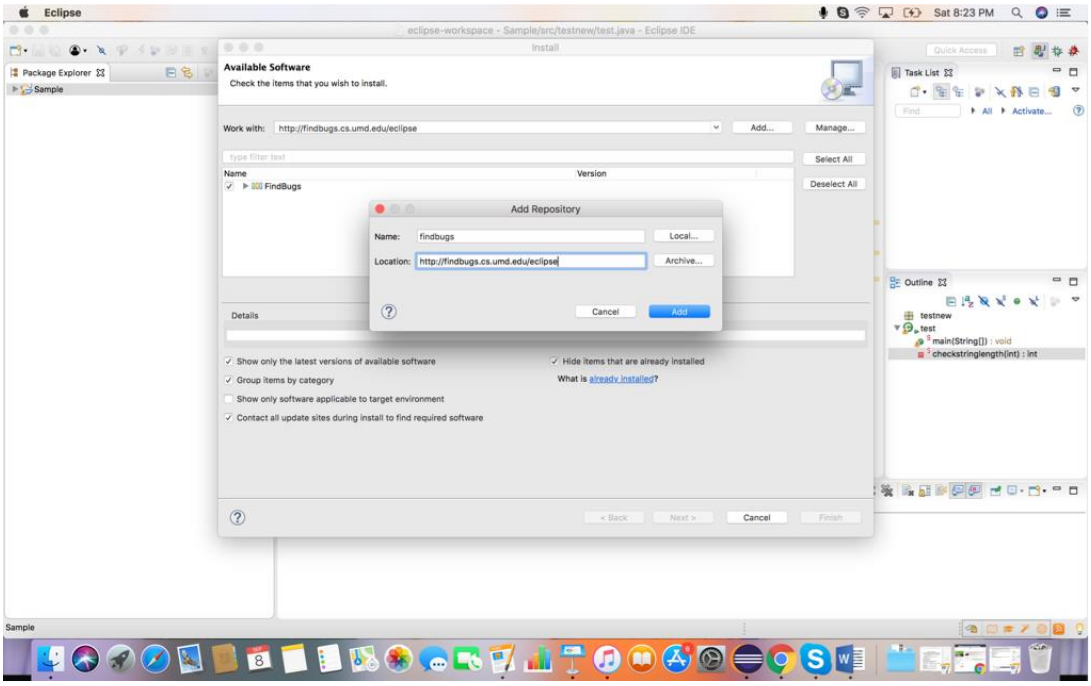
Tool Invocation Process

Eclipse is installed, From Help tab, go to “Install New Software” to add the FindBugs plugin as shown below:

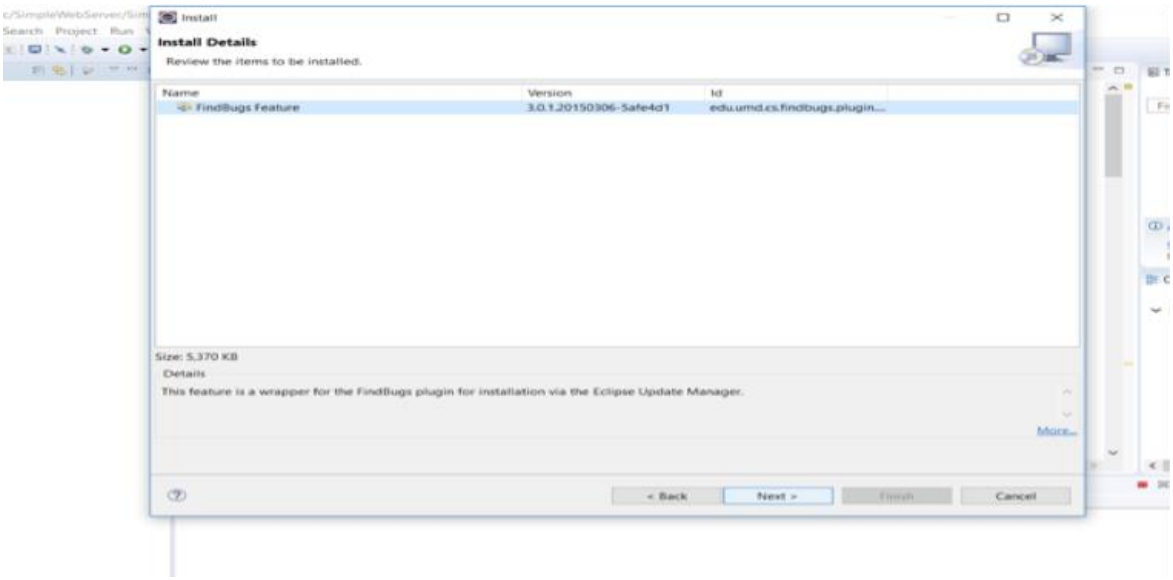
1. Click on “Install New Software
2. Give the url to get the FindBugs plugin for eclipse as: http://findbugs.cs.umd.edu/eclipse and click on “Add”

   Reference: **http://findbugs.sourceforge.net/**

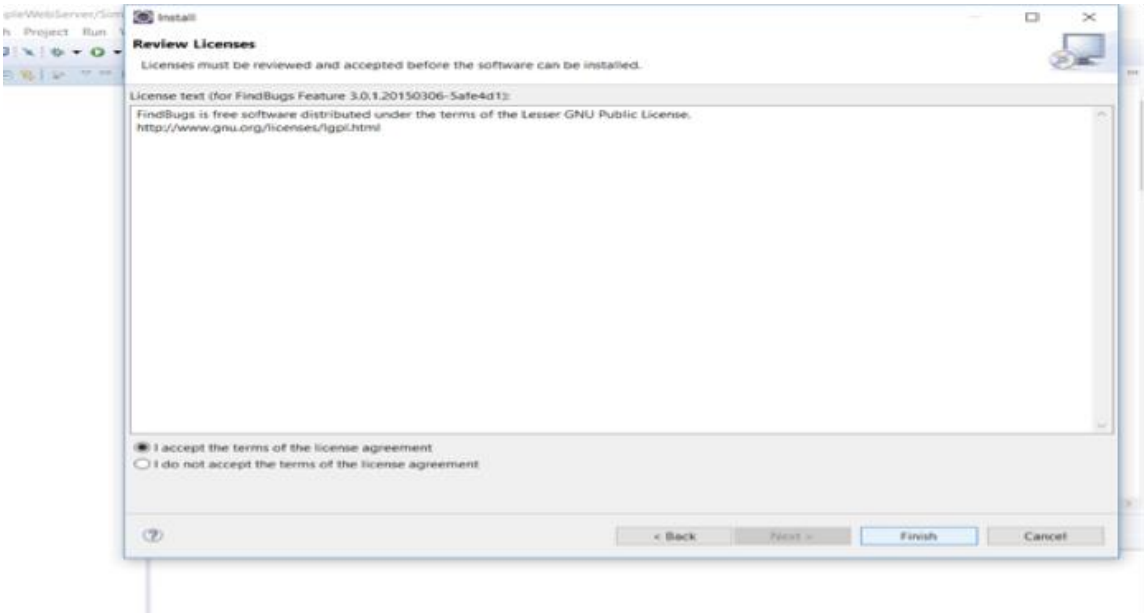




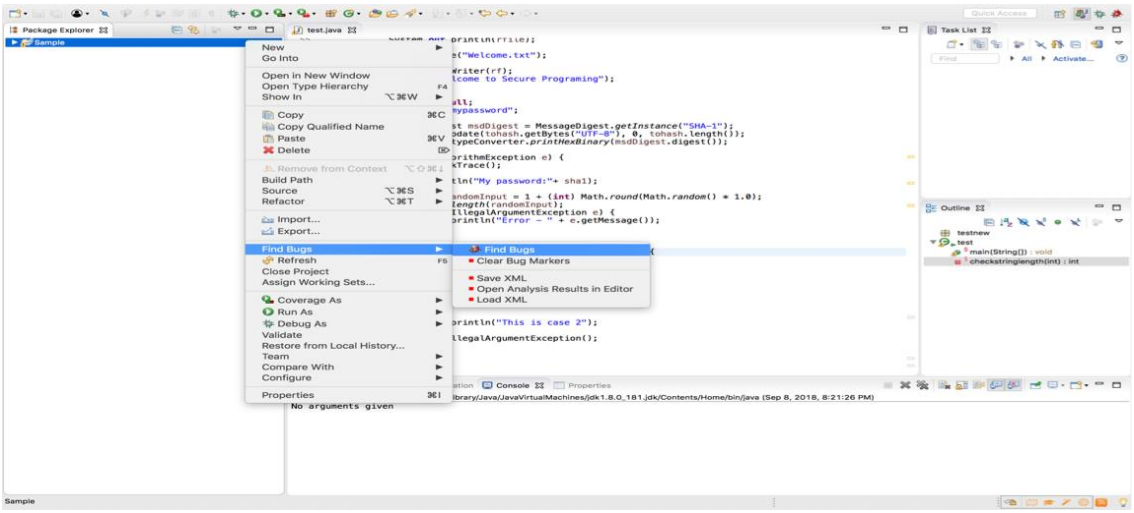
3. Click ‘Next’ again as shown below:



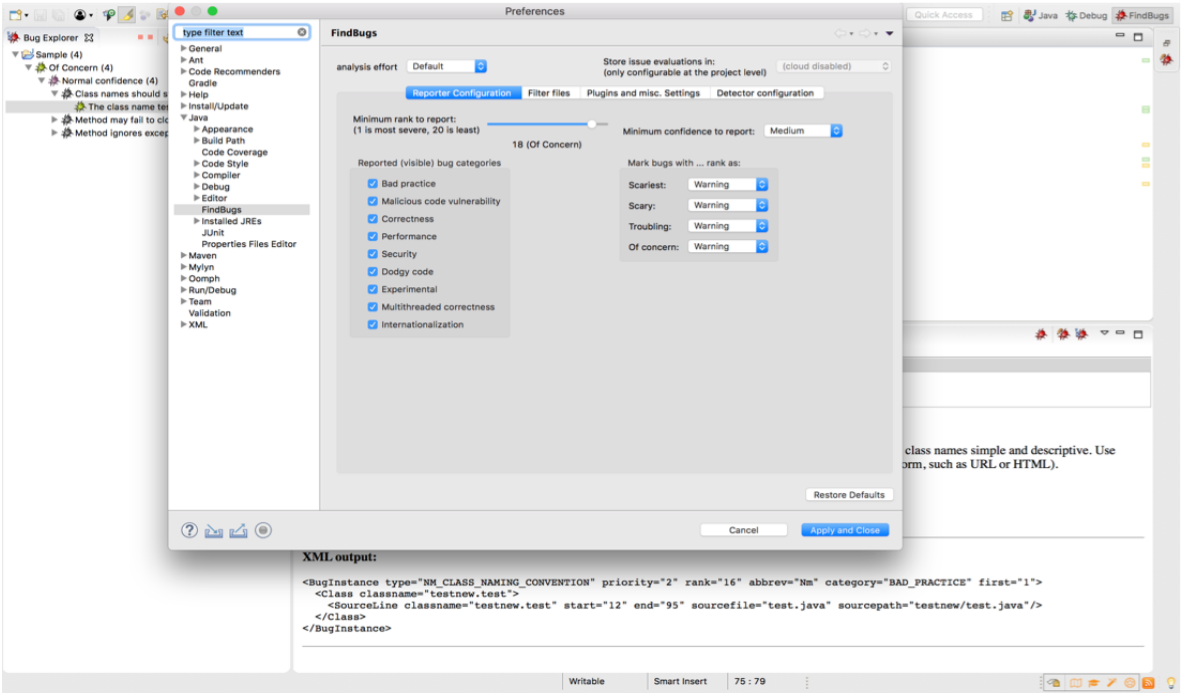
4. Select ‘I accept’ and click Finish to complete the installation FindBugs plugin for eclipse:



5. It will ask you to restart eclipse, click ok to restart eclipse.
6. To run the tool, select the project in the package explorer and select Find Bugs and in the sub menu, select ‘Find Bugs’ again.
7. Added the included example in the eclipse



The below Find bugs explorer will display the bugs present in the code. The recommended configuration to use with Find Security Bugs is to limit the scan to Security only bug detectors. Go to Eclipse -> Preferences (Mac) or Window -> Preferences (Windows). Then go to Java -> FindBugs, and make sure only "Security" is checked on the "Reporting configuration" tab's "Reported (visible) bug categories" list.



Find Bugs categorizes the bugs as follows:

1. Scariest: This type of bug is ranked between 1 & 4

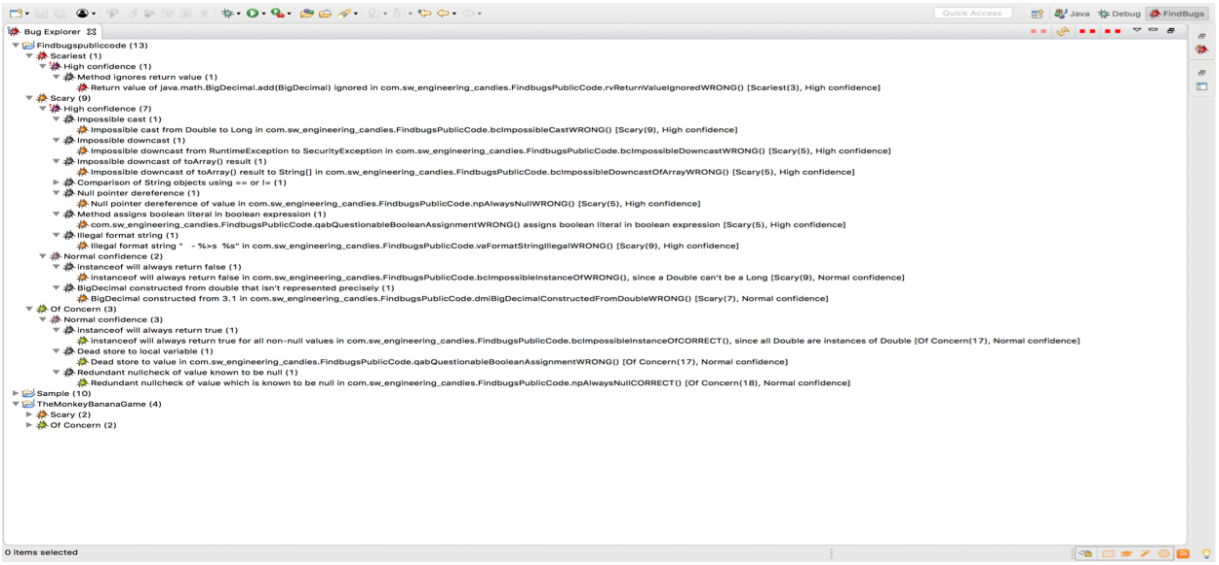
2. Scary: This type of bug is ranked between 5 & 9

3. Troubling: This type of bug is ranked between 10 & 14

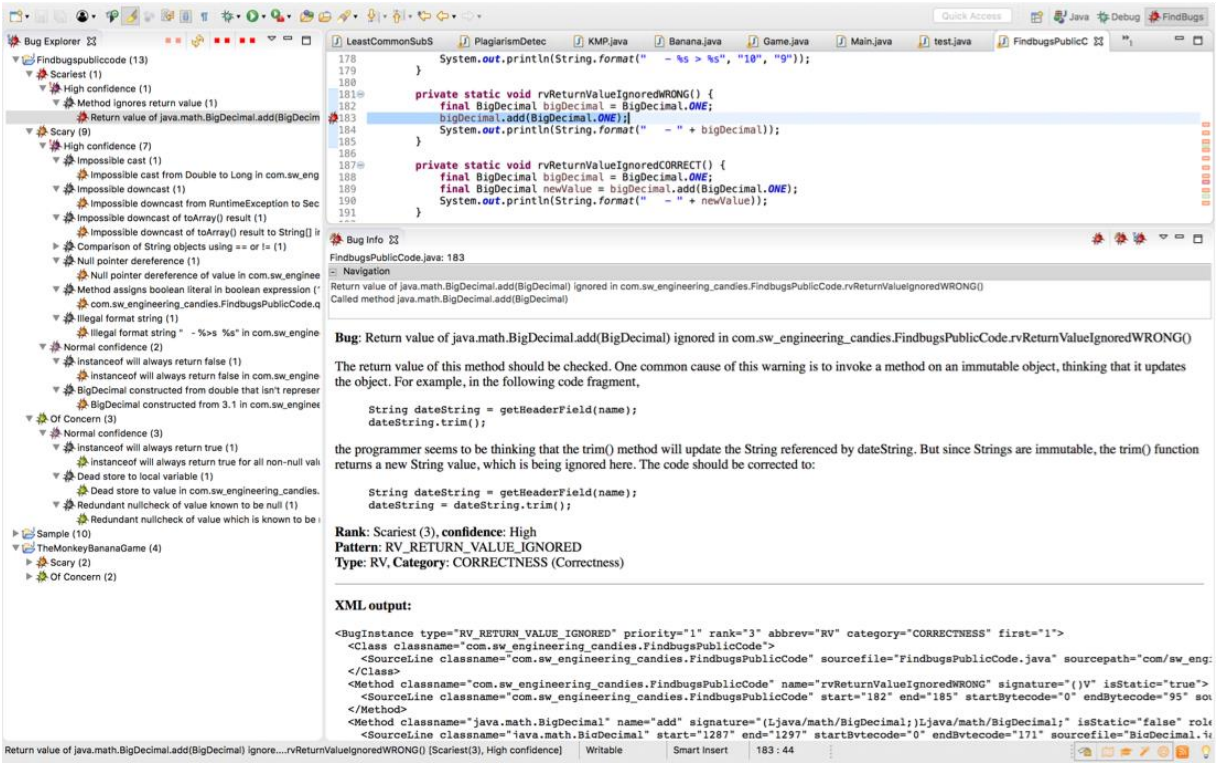
4. Of concern: This type of bug is ranked between 15 & 20

With the FindBugs the below code is giving 13 bugs.

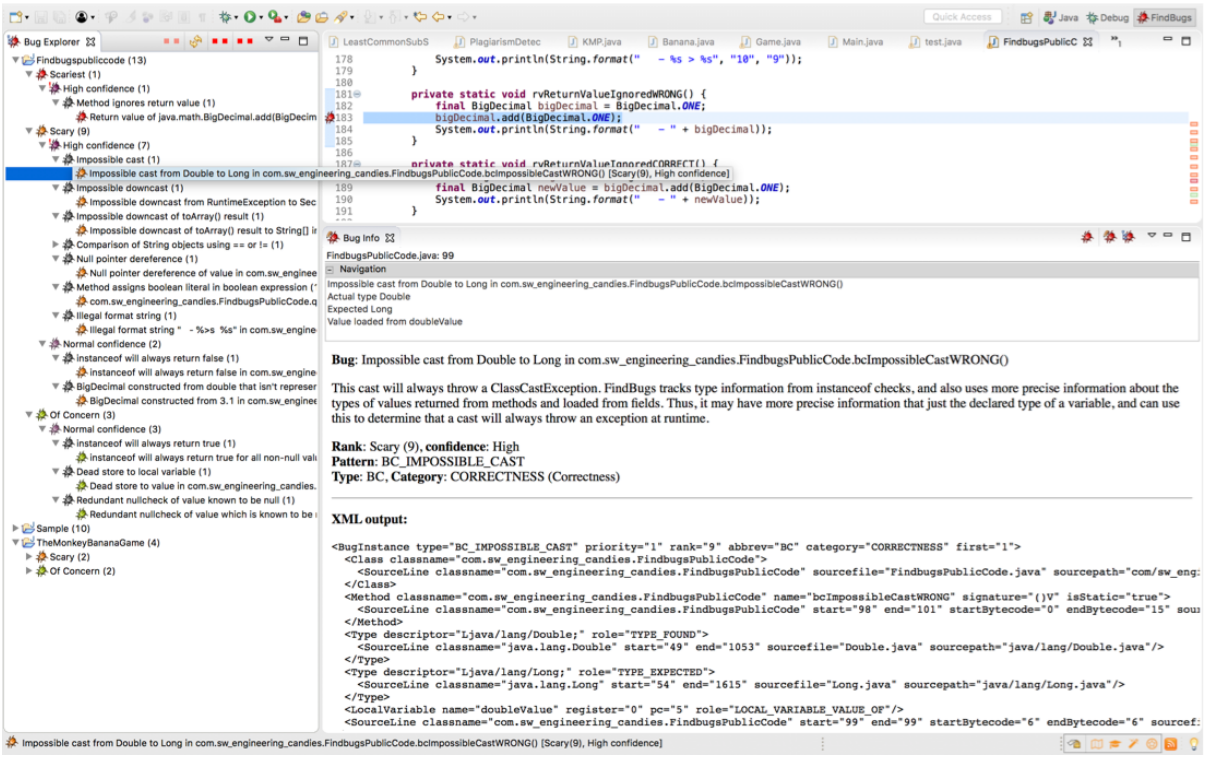
Reference: http://www.sw-engineering-candies.com/blog-1/findbugstmwarningsbysample-parti



Bug 1



Bug 2



Bug: Impossible cast from Double to Long in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleCastWRONG()

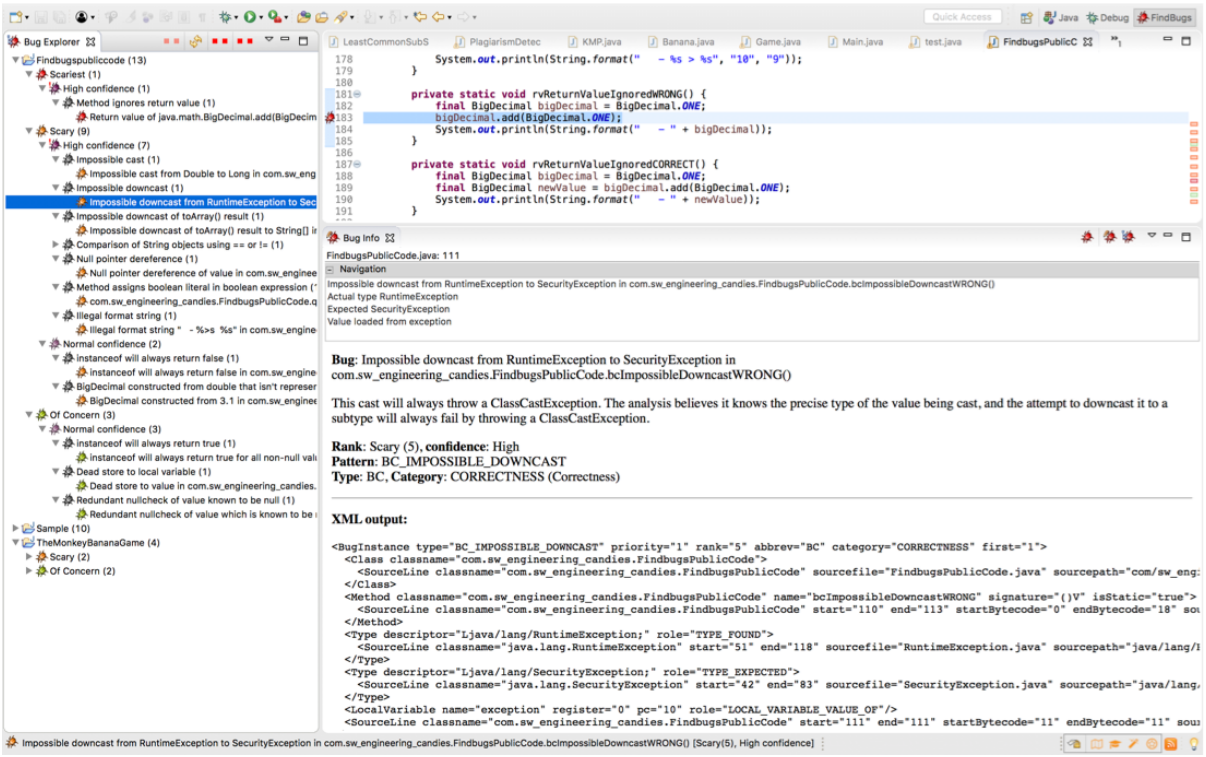
This cast will always throw a ClassCastException. FindBugs tracks type information from instanceof checks, and also uses more precise information about the types of values returned from methods and loaded from fields. Thus, it may have more precise information that just the declared type of a variable, and can use this to determine that a cast will always throw an exception at runtime.

Rank: Scary (9), **confidence**: High
Pattern: BC_IMPOSSIBLE_CAST
Type: BC, **Category**: CORRECTNESS (Correctness)

XML output:

```
<BugInstance type="BC_IMPOSSIBLE_CAST" priority="1" rank="9" abbrev="BC" category="CORRECTNESS" first="1">
  <Class classname="com.sw_engineering_candies.FindbugsPublicCode">
    <SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" sourcefile="FindbugsPublicCode.java" sourcepath="com/sw_eng:
  </Class>
  <Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="bcImpossibleCastWRONG" signature="()V" isStatic="true">
    <SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="98" end="101" startBytecode="0" endBytecode="15" sou
  </Method>
  <Type descriptor="Ljava/lang/Double;" role="TYPE_FOUND">
    <SourceLine classname="java.lang.Double" start="49" end="1053" sourcefile="Double.java" sourcepath="java/lang/Double.java"/>
  </Type>
  <Type descriptor="Ljava/lang/Long;" role="TYPE_EXPECTED">
    <SourceLine classname="java.lang.Long" start="54" end="1615" sourcefile="Long.java" sourcepath="java/lang/Long.java"/>
  </Type>
  <LocalVariable name="doubleValue" register="0" pc="5" role="LOCAL_VARIABLE_VALUE_OF"/>
  <SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="99" end="99" startBytecode="6" endBytecode="6" sourcef:
```

Bug 3



Bug: Impossible downcast from RuntimeException to SecurityException in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleDowncastWRONG()

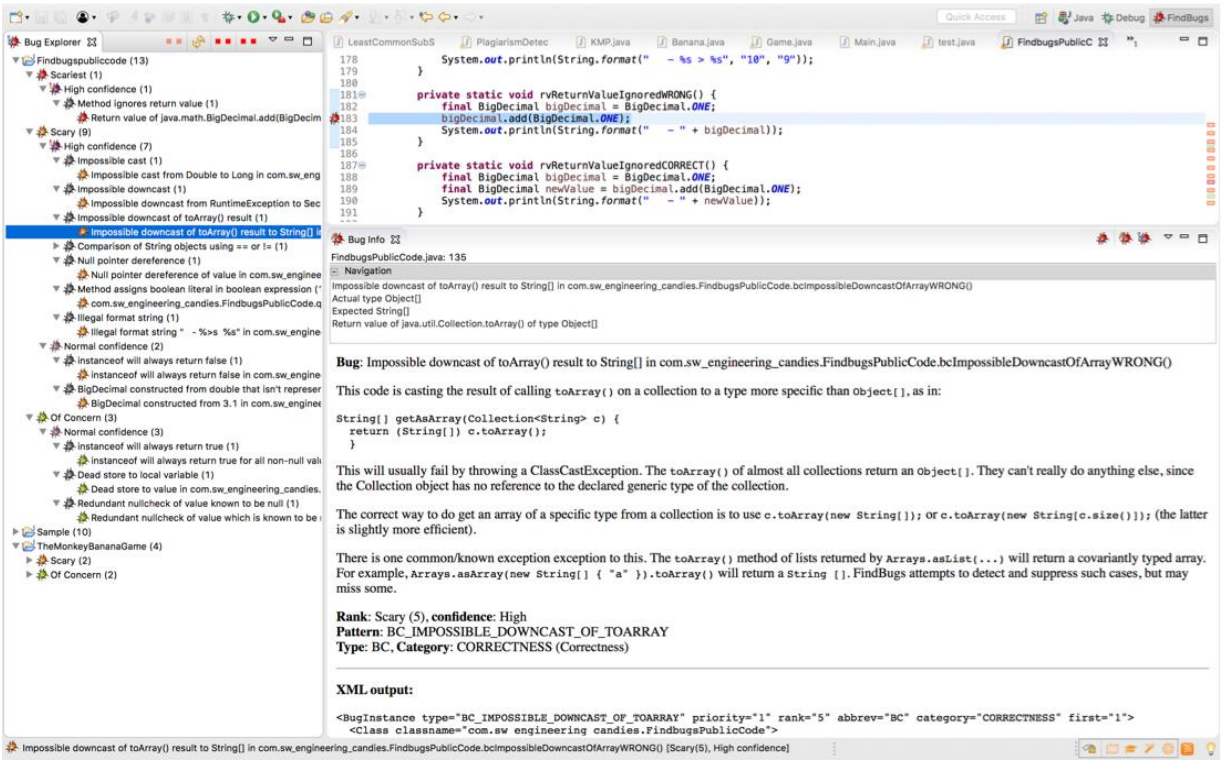
This cast will always throw a ClassCastException. The analysis believes it knows the precise type of the value being cast, and the attempt to downcast it to a subtype will always fail by throwing a ClassCastException.

Rank: Scary (5), **confidence**: High
Pattern: BC_IMPOSSIBLE_DOWNCAST
Type: BC, **Category**: CORRECTNESS (Correctness)

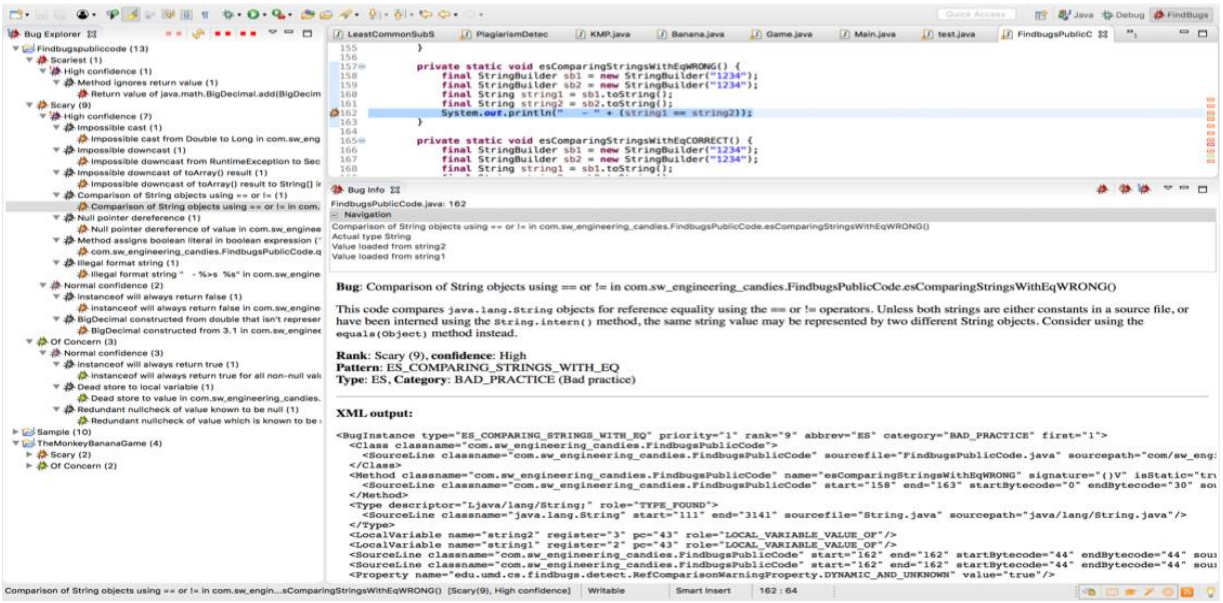
XML output:

```
<BugInstance type="BC_IMPOSSIBLE_DOWNCAST" priority="1" rank="5" abbrev="BC" category="CORRECTNESS" first="1">
  <Class classname="com.sw_engineering_candies.FindbugsPublicCode">
    <SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" sourcefile="FindbugsPublicCode.java" sourcepath="com/sw_eng:
  </Class>
  <Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="bcImpossibleDowncastWRONG" signature="()V" isStatic="true">
    <SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="110" end="113" startBytecode="0" endBytecode="18" sou
  </Method>
  <Type descriptor="Ljava/lang/RuntimeException;" role="TYPE_FOUND">
    <SourceLine classname="java.lang.RuntimeException" start="51" end="118" sourcefile="RuntimeException.java" sourcepath="java/lang/I
  </Type>
  <Type descriptor="Ljava/lang/SecurityException;" role="TYPE_EXPECTED">
    <SourceLine classname="java.lang.SecurityException" start="42" end="83" sourcefile="SecurityException.java" sourcepath="java/lang/
  </Type>
  <LocalVariable name="exception" register="0" pc="10" role="LOCAL_VARIABLE_VALUE_OF"/>
  <SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="111" end="111" startBytecode="11" endBytecode="11" sou:
```

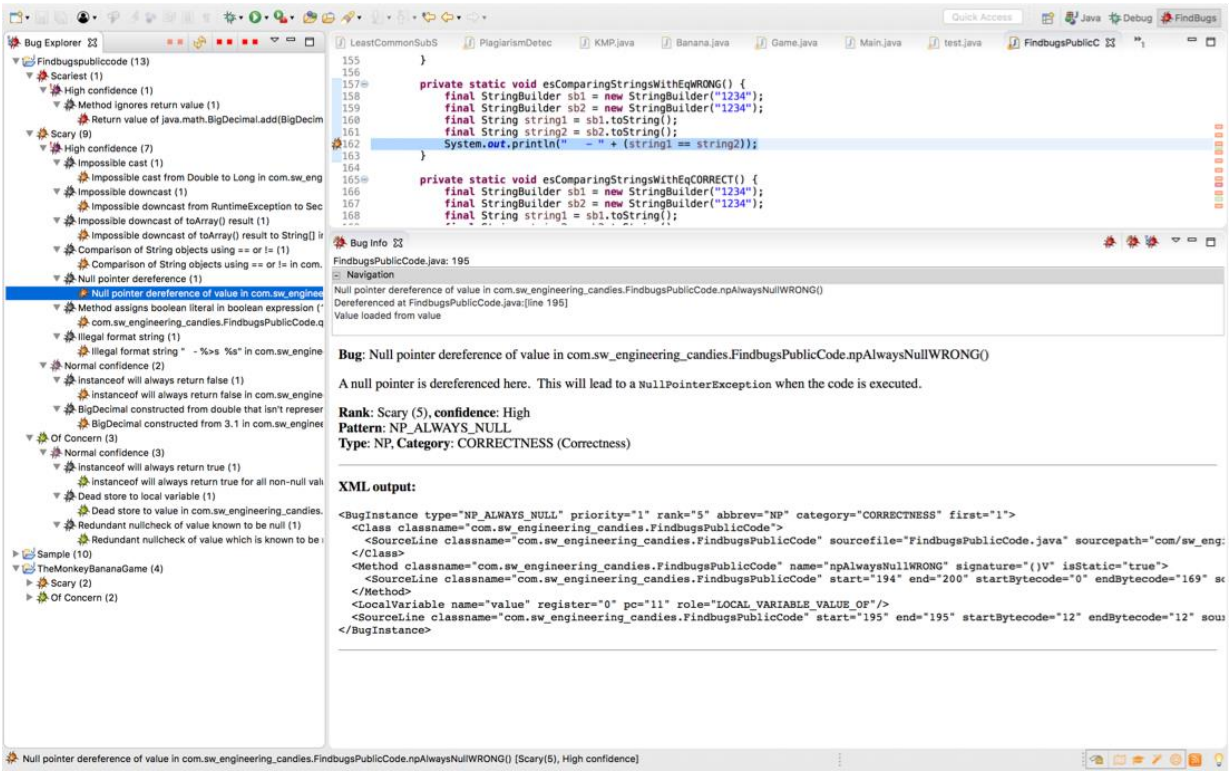
Bug 4


Bug Explorer
Findbugspubliccode (13)
Scariest (1)
High confidence (1)
Method ignores return value (1)
Return value of java.math.BigDecimal.add(BigDecim
Scary (9)
High confidence (7)
Impossible cast (1)
Impossible cast from Double to Long in com.sw_eng
Impossible downcast (1)
Impossible downcast from RuntimeException to Sec
Impossible downcast of toArray() result (1)
Impossible downcast of toArray() result to String[]
Comparison of String objects using == or != (1)
Null pointer dereference (1)
Null pointer dereference of value in com.sw_enginee
Method assigns boolean literal in boolean expression
com.sw_engineering_candies.FindbugsPublicCode.q
Illegal format string (1)
Illegal format string " - %>s %s" in com.sw_engine
Normal confidence (2)
instanceof will always return false (1)
instanceof will always return false in com.sw_engine
BigDecimal constructed from double that isn't represer
BigDecimal constructed from 3.1 in com.sw_enginee
Of Concern (3)
Normal confidence (3)
instanceof will always return true (1)
instanceof will always return true for all non-null vali
Dead store to local variable (1)
Dead store to value in com.sw_engineering_candies.
Redundant nullcheck of value known to be null (1)
Redundant nullcheck of value which is known to be
Sample (10)
TheMonkeyBananaGame (4)
Scary (2)
Of Concern (2)
LeastCommonSubS
PlagiarismDetec
KMP.java
Banana.java
Game.java
Main.java
test.java
FindbugsPublicC
178 System.out.println(String.format(" - %s > %s", "10", "9"));
179 }
180
181 private static void rvReturnValueIgnoredWRONG() {
182 final BigDecimal bigDecimal = BigDecimal.ONE;
183 bigDecimal.add(BigDecimal.ONE);
184 System.out.println(String.format(" - " + bigDecimal));
185 }
186
187 private static void rvReturnValueIgnoredCORRECT() {
188 final BigDecimal bigDecimal = BigDecimal.ONE;
189 final BigDecimal newValue = bigDecimal.add(BigDecimal.ONE);
190 System.out.println(String.format(" - " + newValue));
191 }
Bug Info
FindbugsPublicCode.java: 135
Navigation
Impossible downcast of toArray() result to String[] in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleDowncastOfArrayWRONG()
Actual type Object[]
Expected String[]
Return value of java.util.Collection.toArray() of type Object[]
Bug: Impossible downcast of toArray() result to String[] in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleDowncastOfArrayWRONG()
This code is casting the result of calling toArray() on a collection to a type more specific than Object[], as in:
String[] getAsArray(Collection<String> c) {
return (String[]) c.toArray();
}
This will usually fail by throwing a ClassCastException. The toArray() of almost all collections return an Object[]. They can't really do anything else, since the Collection object has no reference to the declared generic type of the collection.
The correct way to do get an array of a specific type from a collection is to use c.toArray(new String[]); or c.toArray(new String[c.size()]); (the latter is slightly more efficient).
There is one common/known exception exception to this. The toArray() method of lists returned by Arrays.asList(...) will return a covariantly typed array. For example, Arrays.asArray(new String[] { "a" }).toArray() will return a String []. FindBugs attempts to detect and suppress such cases, but may miss some.
Rank: Scary (5), confidence: High
Pattern: BC_IMPOSSIBLE_DOWNCAST_OF_TOARRAY
Type: BC, Category: CORRECTNESS (Correctness)
XML output:
<BugInstance type="BC_IMPOSSIBLE_DOWNCAST_OF_TOARRAY" priority="1" rank="5" abbrev="BC" category="CORRECTNESS" first="1">
<Class classname="com.sw engineering candies.FindbugsPublicCode">
Impossible downcast of toArray() result to String[] in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleDowncastOfArrayWRONG() [Scary(5), High confidence]

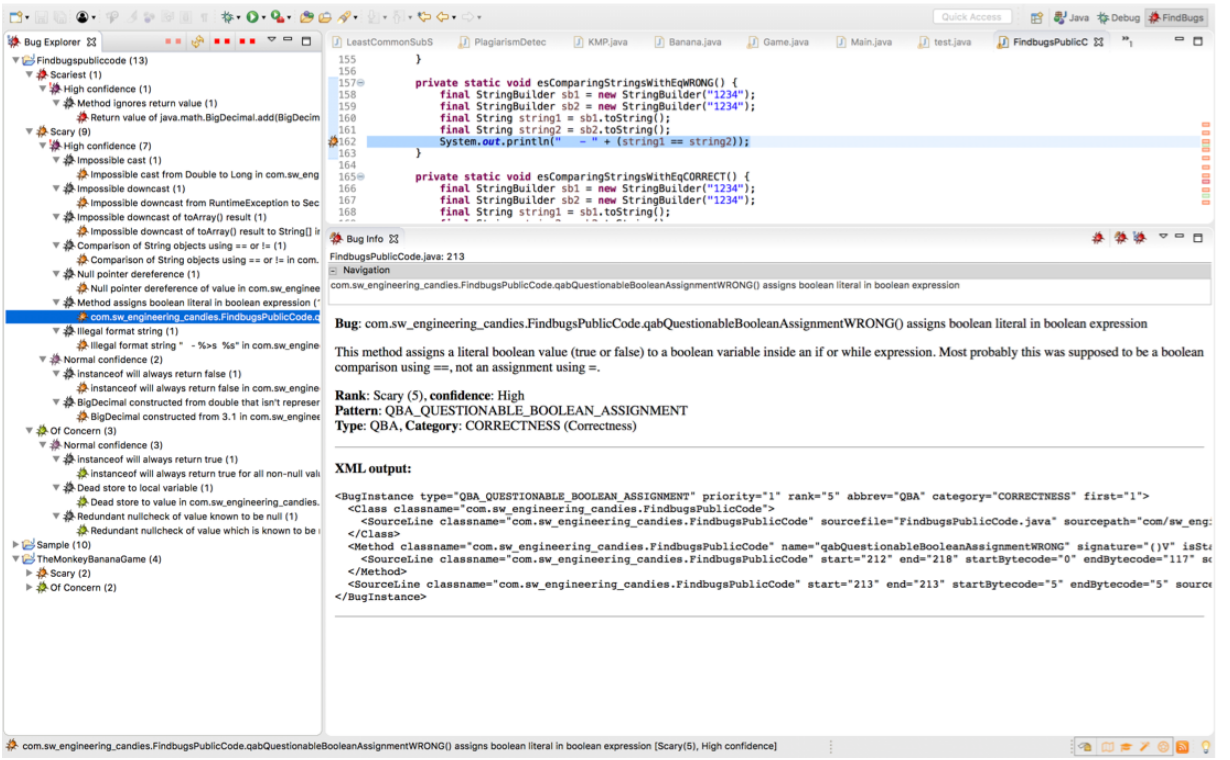
Bug 5


Bug Explorer
Findbugspubliccode (13)
Scariest (1)
High confidence (1)
Method ignores return value (1)
Return value of java.math.BigDecimal.add(BigDecim
Scary (9)
High confidence (7)
Impossible cast (1)
Impossible cast from Double to Long in com.sw_eng
Impossible downcast (1)
Impossible downcast from RuntimeException to Sec
Impossible downcast of toArray() result (1)
Impossible downcast of toArray() result to String[] ir
Comparison of String objects using == or != (1)
Comparison of String objects using == or != in com.
Null pointer dereference (1)
Null pointer dereference of value in com.sw_enginee
Method assigns boolean literal in boolean expression
com.sw_engineering_candies.FindbugsPublicCode.q
Illegal format string (1)
Illegal format string " - %>s %s" in com.sw_engine
Normal confidence (2)
instanceof will always return false (1)
instanceof will always return false in com.sw_engine
BigDecimal constructed from double that isn't represer
BigDecimal constructed from 3.1 in com.sw_enginee
Of Concern (3)
Normal confidence (3)
instanceof will always return true (1)
instanceof will always return true for all non-null vali
Dead store to local variable (1)
Dead store to value in com.sw_engineering_candies.
Redundant nullcheck of value known to be null (1)
Redundant nullcheck of value which is known to be
Sample (10)
TheMonkeyBananaGame (4)
Scary (2)
Of Concern (2)
LeastCommonSubS
PlagiarismDetec
KMP.java
Banana.java
Game.java
Main.java
test.java
FindbugsPublicC
155 }
156
157 private static void esComparingStringsWithEqWRONG() {
158 final StringBuilder sb1 = new StringBuilder("1234");
159 final StringBuilder sb2 = new StringBuilder("1234");
160 final String string1 = sb1.toString();
161 final String string2 = sb2.toString();
162 System.out.println(" - " + (string1 == string2));
163 }
164
165 private static void esComparingStringsWithEqCORRECT() {
166 final StringBuilder sb1 = new StringBuilder("1234");
167 final StringBuilder sb2 = new StringBuilder("1234");
168 final String string1 = sb1.toString();
Bug Info
FindbugsPublicCode.java: 162
Navigation
Comparison of String objects using == or != in com.sw_engineering_candies.FindbugsPublicCode.esComparingStringsWithEqWRONG()
Actual type String
Value loaded from string2
Value loaded from string1
Bug: Comparison of String objects using == or != in com.sw_engineering_candies.FindbugsPublicCode.esComparingStringsWithEqWRONG()
This code compares java.lang.String objects for reference equality using the == or != operators. Unless both strings are either constants in a source file, or have been interned using the String.intern() method, the same string value may be represented by two different String objects. Consider using the equals(Object) method instead.
Rank: Scary (9), confidence: High
Pattern: ES_COMPARING_STRINGS_WITH_EQ
Type: ES, Category: BAD_PRACTICE (Bad practice)
XML output:
<BugInstance type="ES_COMPARING_STRINGS_WITH_EQ" priority="1" rank="9" abbrev="ES" category="BAD_PRACTICE" first="1">
<Class classname="com.sw_engineering_candies.FindbugsPublicCode">
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" sourcefile="FindbugsPublicCode.java" sourcepath="com/sw_eng
</Class>
<Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="esComparingStringsWithEqWRONG" signature="()V" isStatic="tru
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="158" end="163" startBytecode="0" endBytecode="30" sou
</Method>
<Type descriptor="Ljava/lang/String;" role="TYPE_FOUND">
<SourceLine classname="java.lang.String" start="111" end="3141" sourcefile="String.java" sourcepath="java/lang/String.java"/>
</Type>
<LocalVariable name="string2" register="3" pc="43" role="LOCAL_VARIABLE_VALUE_OF"/>
<LocalVariable name="string1" register="2" pc="43" role="LOCAL_VARIABLE_VALUE_OF"/>
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="162" end="162" startBytecode="44" endBytecode="44" sou
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="162" end="162" startBytecode="44" endBytecode="44" sou
<Property name="edu.umd.cs.findbugs.detect.RefComparisonWarningProperty.DYNAMIC_AND_UNKNOWN" value="true"/>
Comparison of String objects using == or != in com.sw_engin...sComparingStringsWithEqWRONG() [Scary(9), High confidence]
Writable
Smart Insert
162 : 64

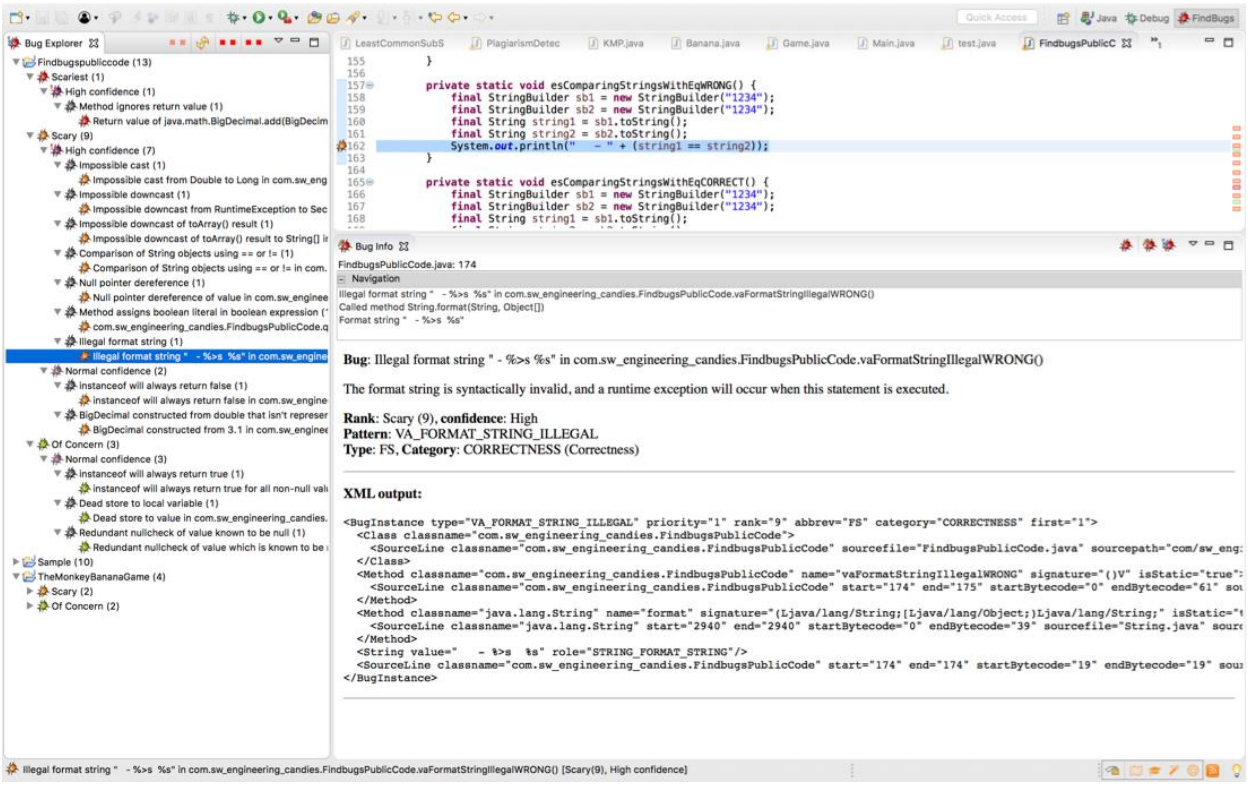
Bug 6


Bug Explorer
Findbugspubliccode (13)
Scariest (1)
High confidence (1)
Method ignores return value (1)
Return value of java.math.BigDecimal.add(BigDecim
Scary (9)
High confidence (7)
Impossible cast (1)
Impossible cast from Double to Long in com.sw_eng
Impossible downcast (1)
Impossible downcast from RuntimeException to Sec
Impossible downcast of toArray() result (1)
Impossible downcast of toArray() result to String[] i
Comparison of String objects using == or != (1)
Comparison of String objects using == or != in com.
Null pointer dereference (1)
Null pointer dereference of value in com.sw_enginee
Method assigns boolean literal in boolean expression (
com.sw_engineering_candies.FindbugsPublicCode.q
Illegal format string (1)
Illegal format string " - %>s %s" in com.sw_engine
Normal confidence (2)
instanceof will always return false (1)
instanceof will always return false in com.sw_engine
BigDecimal constructed from double that isn't represer
BigDecimal constructed from 3.1 in com.sw_enginee
Of Concern (3)
Normal confidence (3)
instanceof will always return true (1)
instanceof will always return true for all non-null vali
Dead store to local variable (1)
Dead store to value in com.sw_engineering_candies.
Redundant nullcheck of value known to be null (1)
Redundant nullcheck of value which is known to be
Sample (10)
TheMonkeyBananaGame (4)
Scary (2)
Of Concern (2)
LeastCommonSubS
PlagiarismDetec
KMP.java
Banana.java
Game.java
Main.java
test.java
FindbugsPublicC
155 }
156
157 private static void esComparingStringsWithEqWRONG() {
158 final StringBuilder sb1 = new StringBuilder("1234");
159 final StringBuilder sb2 = new StringBuilder("1234");
160 final String string1 = sb1.toString();
161 final String string2 = sb2.toString();
162 System.out.println(" - " + (string1 == string2));
163 }
164
165 private static void esComparingStringsWithEqCORRECT() {
166 final StringBuilder sb1 = new StringBuilder("1234");
167 final StringBuilder sb2 = new StringBuilder("1234");
168 final String string1 = sb1.toString();
Bug Info
FindbugsPublicCode.java: 195
Navigation
Null pointer dereference of value in com.sw_engineering_candies.FindbugsPublicCode.npAlwaysNullWRONG()
Dereferenced at FindbugsPublicCode.java:[line 195]
Value loaded from value
Bug: Null pointer dereference of value in com.sw_engineering_candies.FindbugsPublicCode.npAlwaysNullWRONG()
A null pointer is dereferenced here. This will lead to a NullPointerException when the code is executed.
Rank: Scary (5), confidence: High
Pattern: NP_ALWAYS_NULL
Type: NP, Category: CORRECTNESS (Correctness)
XML output:
<BugInstance type="NP_ALWAYS_NULL" priority="1" rank="5" abbrev="NP" category="CORRECTNESS" first="1">
<Class classname="com.sw_engineering_candies.FindbugsPublicCode">
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" sourcefile="FindbugsPublicCode.java" sourcepath="com/sw_eng
</Class>
<Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="npAlwaysNullWRONG" signature="()V" isStatic="true">
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="194" end="200" startBytecode="0" endBytecode="169" s
</Method>
<LocalVariable name="value" register="0" pc="11" role="LOCAL_VARIABLE_VALUE_OF"/>
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="195" end="195" startBytecode="12" endBytecode="12" sou
</BugInstance>
Null pointer dereference of value in com.sw_engineering_candies.FindbugsPublicCode.npAlwaysNullWRONG() [Scary(5), High confidence]

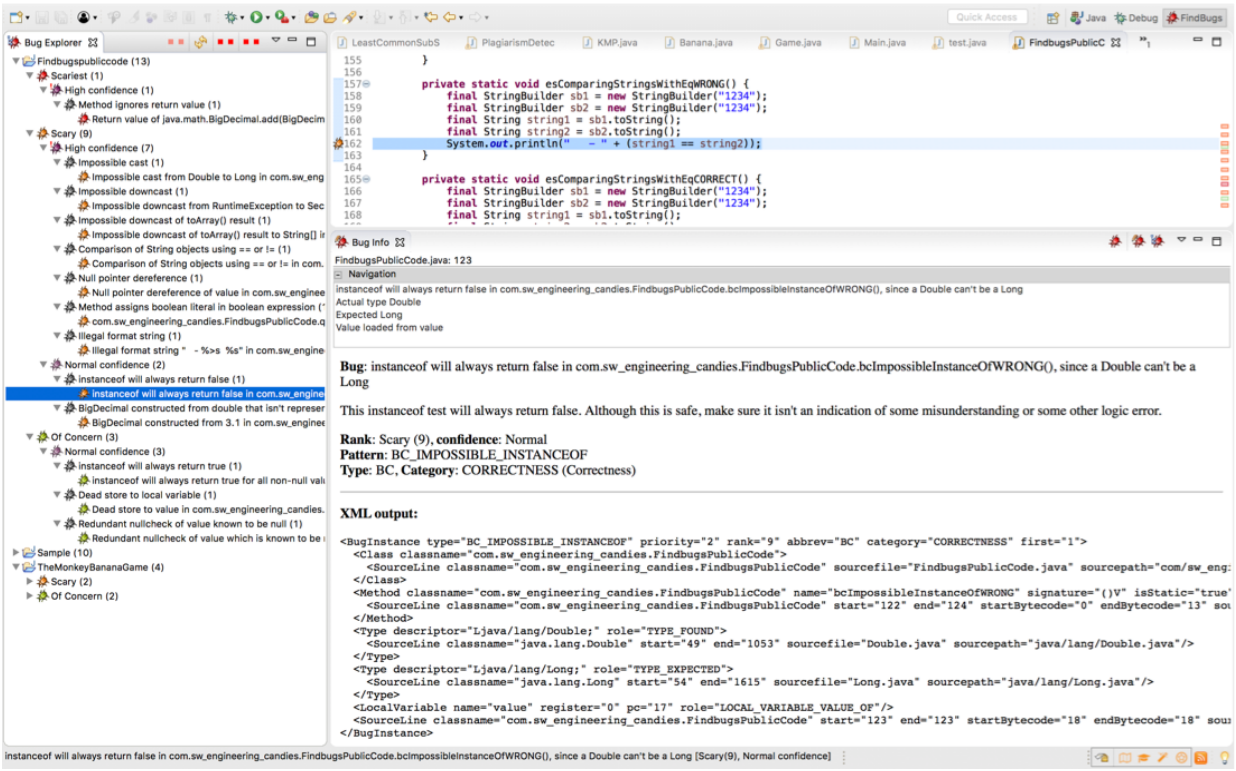
Bug 7


Bug Explorer
Findbugspubliccode (13)
Scariest (1)
High confidence (1)
Method ignores return value (1)
Return value of java.math.BigDecimal.add(BigDecim
Scary (9)
High confidence (7)
Impossible cast (1)
Impossible cast from Double to Long in com.sw_eng
Impossible downcast (1)
Impossible downcast from RuntimeException to Sec
Impossible downcast of toArray() result (1)
Impossible downcast of toArray() result to String[] i
Comparison of String objects using == or != (1)
Comparison of String objects using == or != in com.
Null pointer dereference (1)
Null pointer dereference of value in com.sw_enginee
Method assigns boolean literal in boolean expression (
com.sw_engineering_candies.FindbugsPublicCode.q
Illegal format string (1)
Illegal format string " - %>s %s" in com.sw_engine
Normal confidence (2)
instanceof will always return false (1)
instanceof will always return false in com.sw_engine
BigDecimal constructed from double that isn't represer
BigDecimal constructed from 3.1 in com.sw_enginee
Of Concern (3)
Normal confidence (3)
instanceof will always return true (1)
instanceof will always return true for all non-null vali
Dead store to local variable (1)
Dead store to value in com.sw_engineering_candies.
Redundant nullcheck of value known to be null (1)
Redundant nullcheck of value which is known to be
Sample (10)
TheMonkeyBananaGame (4)
Scary (2)
Of Concern (2)
LeastCommonSubS
PlagiarismDetec
KMP.java
Banana.java
Game.java
Main.java
test.java
FindbugsPublicC
155 }
156
157 private static void esComparingStringsWithEqWRONG() {
158 final StringBuilder sb1 = new StringBuilder("1234");
159 final StringBuilder sb2 = new StringBuilder("1234");
160 final String string1 = sb1.toString();
161 final String string2 = sb2.toString();
162 System.out.println(" - " + (string1 == string2));
163 }
164
165 private static void esComparingStringsWithEqCORRECT() {
166 final StringBuilder sb1 = new StringBuilder("1234");
167 final StringBuilder sb2 = new StringBuilder("1234");
168 final String string1 = sb1.toString();
Bug Info
FindbugsPublicCode.java: 213
Navigation
com.sw_engineering_candies.FindbugsPublicCode.qabQuestionableBooleanAssignmentWRONG() assigns boolean literal in boolean expression
Bug: com.sw_engineering_candies.FindbugsPublicCode.qabQuestionableBooleanAssignmentWRONG() assigns boolean literal in boolean expression
This method assigns a literal boolean value (true or false) to a boolean variable inside an if or while expression. Most probably this was supposed to be a boolean comparison using ==, not an assignment using =.
Rank: Scary (5), confidence: High
Pattern: QBA_QUESTIONABLE_BOOLEAN_ASSIGNMENT
Type: QBA, Category: CORRECTNESS (Correctness)
XML output:
<BugInstance type="QBA_QUESTIONABLE_BOOLEAN_ASSIGNMENT" priority="1" rank="5" abbrev="QBA" category="CORRECTNESS" first="1">
<Class classname="com.sw_engineering_candies.FindbugsPublicCode">
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" sourcefile="FindbugsPublicCode.java" sourcepath="com/sw_eng
</Class>
<Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="qabQuestionableBooleanAssignmentWRONG" signature="()V" isSt
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="212" end="218" startBytecode="0" endBytecode="117" s
</Method>
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="213" end="213" startBytecode="5" endBytecode="5" sourc
</BugInstance>
com.sw_engineering_candies.FindbugsPublicCode.qabQuestionableBooleanAssignmentWRONG() assigns boolean literal in boolean expression [Scary(5), High confidence]

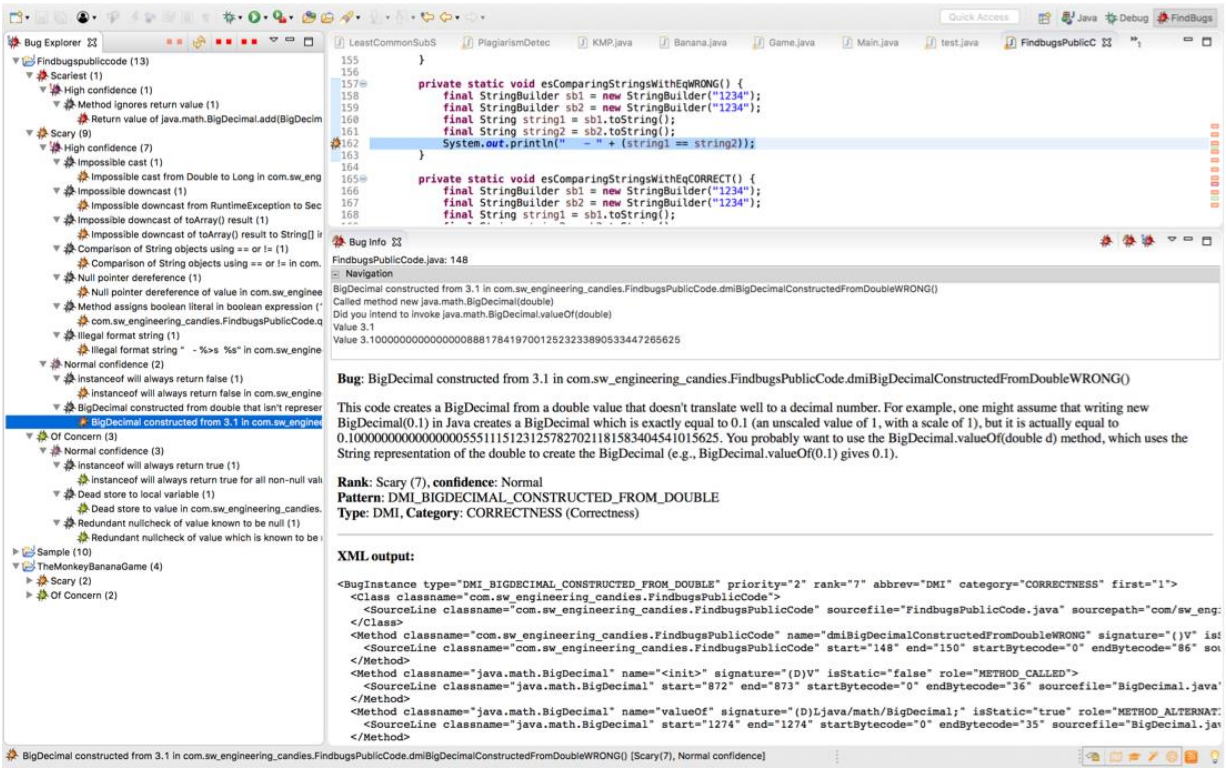
Bug 8


Bug Explorer
Findbugspubliccode (13)
Scariest (1)
High confidence (1)
Method ignores return value (1)
Return value of java.math.BigDecimal.add(BigDecim
Scary (9)
High confidence (7)
Impossible cast (1)
Impossible cast from Double to Long in com.sw_eng
Impossible downcast (1)
Impossible downcast from RuntimeException to Sec
Impossible downcast of toArray() result (1)
Impossible downcast of toArray() result to String[] i
Comparison of String objects using == or != (1)
Comparison of String objects using == or != in com.
Null pointer dereference (1)
Null pointer dereference of value in com.sw_enginee
Method assigns boolean literal in boolean expression
com.sw_engineering_candies.FindbugsPublicCode.q
Illegal format string (1)
Illegal format string " - %>s %s" in com.sw_engine
Normal confidence (2)
instanceof will always return false (1)
instanceof will always return false in com.sw_engine
BigDecimal constructed from double that isn't represer
BigDecimal constructed from 3.1 in com.sw_enginee
Of Concern (3)
Normal confidence (3)
instanceof will always return true (1)
instanceof will always return true for all non-null vali
Dead store to local variable (1)
Dead store to value in com.sw_engineering_candies.
Redundant nullcheck of value known to be null (1)
Redundant nullcheck of value which is known to be
Sample (10)
TheMonkeyBananaGame (4)
Scary (2)
Of Concern (2)
LeastCommonSubS
PlagiarismDetec
KMP.java
Banana.java
Game.java
Main.java
test.java
FindbugsPublicC
155 }
156
157 private static void esComparingStringsWithEqWRONG() {
158 final StringBuilder sb1 = new StringBuilder("1234");
159 final StringBuilder sb2 = new StringBuilder("1234");
160 final String string1 = sb1.toString();
161 final String string2 = sb2.toString();
162 System.out.println(" - " + (string1 == string2));
163 }
164
165 private static void esComparingStringsWithEqCORRECT() {
166 final StringBuilder sb1 = new StringBuilder("1234");
167 final StringBuilder sb2 = new StringBuilder("1234");
168 final String string1 = sb1.toString();
Bug Info
FindbugsPublicCode.java: 174
Navigation
Illegal format string " - %>s %s" in com.sw_engineering_candies.FindbugsPublicCode.vaFormatStringIllegalWRONG()
Called method String.format(String, Object[])
Format string " - %>s %s"
Bug: Illegal format string " - %>s %s" in com.sw_engineering_candies.FindbugsPublicCode.vaFormatStringIllegalWRONG()
The format string is syntactically invalid, and a runtime exception will occur when this statement is executed.
Rank: Scary (9), confidence: High
Pattern: VA_FORMAT_STRING_ILLEGAL
Type: FS, Category: CORRECTNESS (Correctness)
XML output:
<BugInstance type="VA_FORMAT_STRING_ILLEGAL" priority="1" rank="9" abbrev="FS" category="CORRECTNESS" first="1">
<Class classname="com.sw_engineering_candies.FindbugsPublicCode">
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" sourcefile="FindbugsPublicCode.java" sourcepath="com/sw_eng
</Class>
<Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="vaFormatStringIllegalWRONG" signature="()V" isStatic="true"
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="174" end="175" startBytecode="0" endBytecode="61" sou
</Method>
<Method classname="java.lang.String" name="format" signature="(Ljava/lang/String;[Ljava/lang/Object;)Ljava/lang/String;" isStatic="t
<SourceLine classname="java.lang.String" start="2940" end="2940" startBytecode="0" endBytecode="39" sourcefile="String.java" sourc
</Method>
<String value=" - %>s %s" role="STRING_FORMAT_STRING"/>
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="174" end="174" startBytecode="19" endBytecode="19" sou
</BugInstance>
Illegal format string " - %>s %s" in com.sw_engineering_candies.FindbugsPublicCode.vaFormatStringIllegalWRONG() [Scary(9), High confidence]

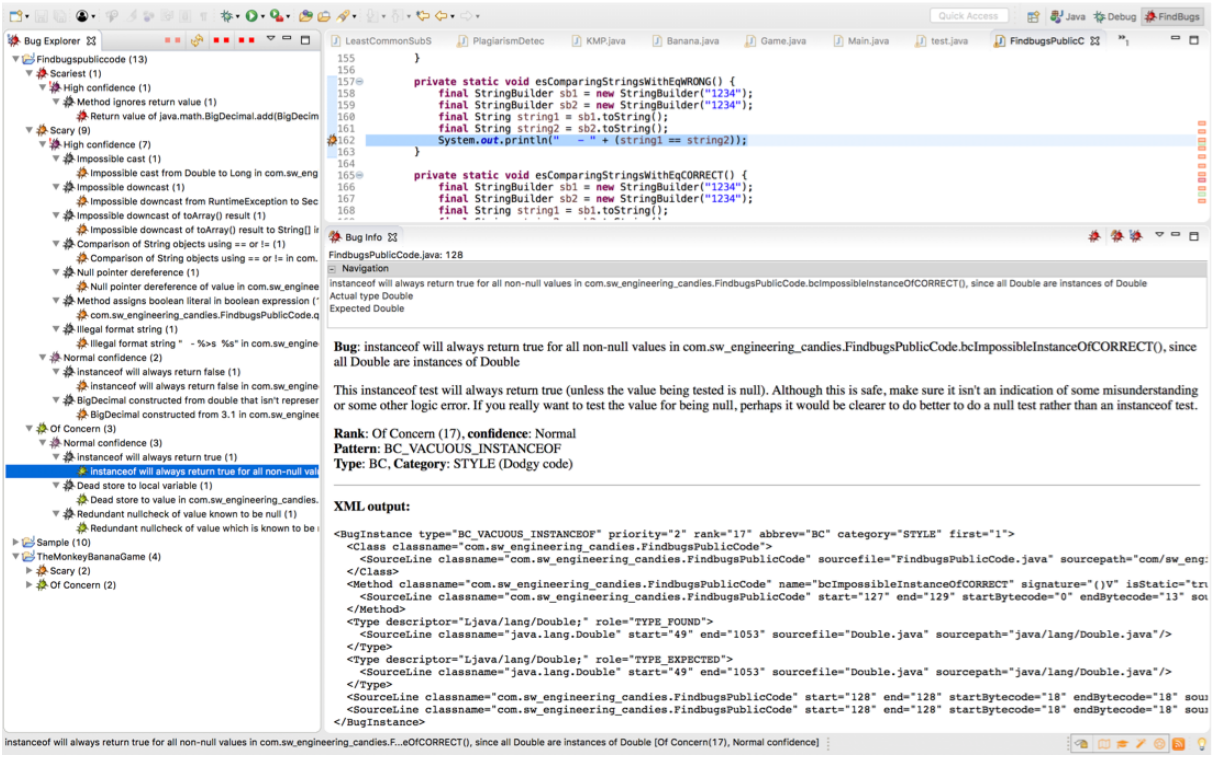
Bug 9


Bug Explorer
Findbugspubliccode (13)
Scariest (1)
High confidence (1)
Method ignores return value (1)
Return value of java.math.BigDecimal.add(BigDecim
Scary (9)
High confidence (7)
Impossible cast (1)
Impossible cast from Double to Long in com.sw_eng
Impossible downcast (1)
Impossible downcast from RuntimeException to Sec
Impossible downcast of toArray() result (1)
Impossible downcast of toArray() result to String[] i
Comparison of String objects using == or != (1)
Comparison of String objects using == or != in com.
Null pointer dereference (1)
Null pointer dereference of value in com.sw_enginee
Method assigns boolean literal in boolean expression
com.sw_engineering_candies.FindbugsPublicCode.q
Illegal format string (1)
Illegal format string " - %>s %s" in com.sw_engine
Normal confidence (2)
instanceof will always return false (1)
instanceof will always return false in com.sw_engine
BigDecimal constructed from double that isn't represer
BigDecimal constructed from 3.1 in com.sw_enginee
Of Concern (3)
Normal confidence (3)
instanceof will always return true (1)
instanceof will always return true for all non-null vali
Dead store to local variable (1)
Dead store to value in com.sw_engineering_candies.
Redundant nullcheck of value known to be null (1)
Redundant nullcheck of value which is known to be
Sample (10)
TheMonkeyBananaGame (4)
Scary (2)
Of Concern (2)
LeastCommonSubS
PlagiarismDetec
KMP.java
Banana.java
Game.java
Main.java
test.java
FindbugsPublicC
155 }
156
157 private static void esComparingStringsWithEqWRONG() {
158 final StringBuilder sb1 = new StringBuilder("1234");
159 final StringBuilder sb2 = new StringBuilder("1234");
160 final String string1 = sb1.toString();
161 final String string2 = sb2.toString();
162 System.out.println(" - " + (string1 == string2));
163 }
164
165 private static void esComparingStringsWithEqCORRECT() {
166 final StringBuilder sb1 = new StringBuilder("1234");
167 final StringBuilder sb2 = new StringBuilder("1234");
168 final String string1 = sb1.toString();
Bug Info
FindbugsPublicCode.java: 123
Navigation
instanceof will always return false in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleInstanceOfWRONG(), since a Double can't be a Long
Actual type Double
Expected Long
Value loaded from value
Bug: instanceof will always return false in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleInstanceOfWRONG(), since a Double can't be a Long
This instanceof test will always return false. Although this is safe, make sure it isn't an indication of some misunderstanding or some other logic error.
Rank: Scary (9), confidence: Normal
Pattern: BC_IMPOSSIBLE_INSTANCEOF
Type: BC, Category: CORRECTNESS (Correctness)
XML output:
<BugInstance type="BC_IMPOSSIBLE_INSTANCEOF" priority="2" rank="9" abbrev="BC" category="CORRECTNESS" first="1">
<Class classname="com.sw_engineering_candies.FindbugsPublicCode">
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" sourcefile="FindbugsPublicCode.java" sourcepath="com/sw_eng
</Class>
<Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="bcImpossibleInstanceOfWRONG" signature="()V" isStatic="true"
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="122" end="124" startBytecode="0" endBytecode="13" sou
</Method>
<Type descriptor="Ljava/lang/Double;" role="TYPE_FOUND">
<SourceLine classname="java.lang.Double" start="49" end="1053" sourcefile="Double.java" sourcepath="java/lang/Double.java"/>
</Type>
<Type descriptor="Ljava/lang/Long;" role="TYPE_EXPECTED">
<SourceLine classname="java.lang.Long" start="54" end="1615" sourcefile="Long.java" sourcepath="java/lang/Long.java"/>
</Type>
<LocalVariable name="value" register="0" pc="17" role="LOCAL_VARIABLE_VALUE_OF"/>
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="123" end="123" startBytecode="18" endBytecode="18" sou
</BugInstance>
instanceof will always return false in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleInstanceOfWRONG(), since a Double can't be a Long [Scary(9), Normal confidence]

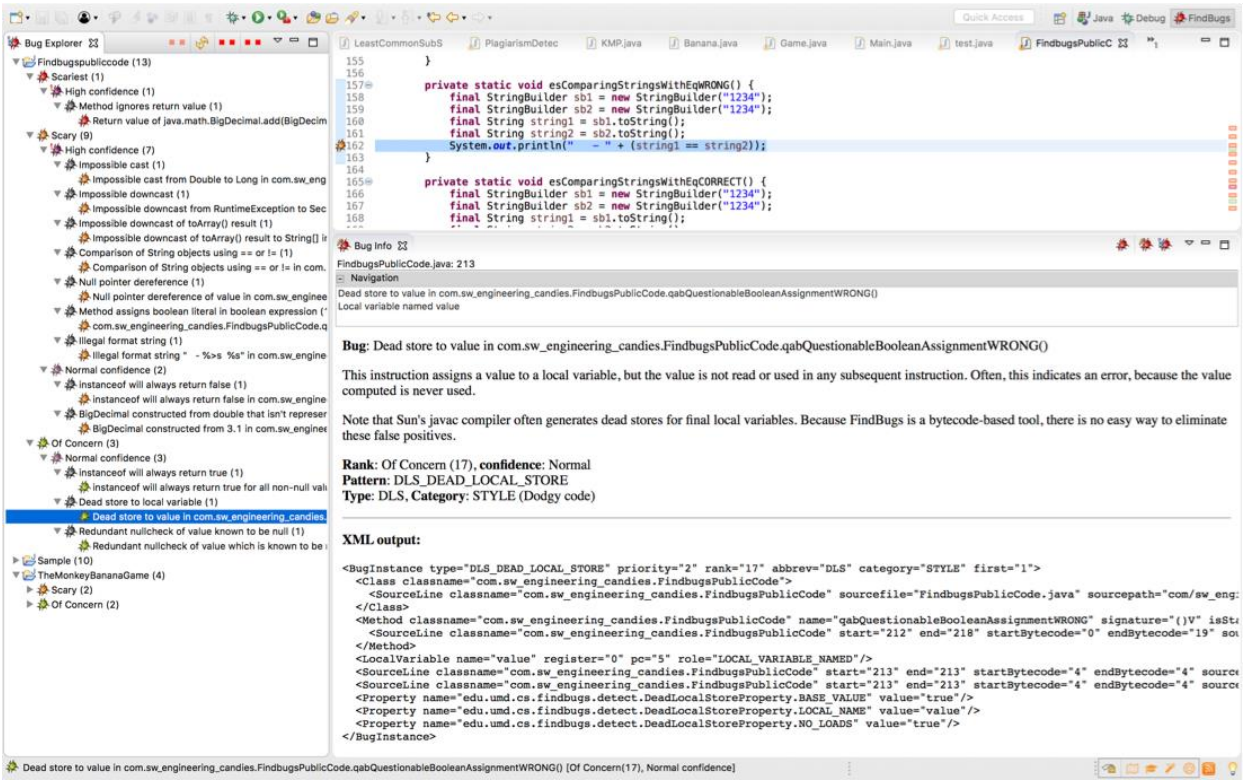
Bug 10


Bug Explorer
Findbugspubliccode (13)
Scariest (1)
High confidence (1)
Method ignores return value (1)
Return value of java.math.BigDecimal.add(BigDecim
Scary (9)
High confidence (7)
Impossible cast (1)
Impossible cast from Double to Long in com.sw_eng
Impossible downcast (1)
Impossible downcast from RuntimeException to Sec
Impossible downcast of toArray() result (1)
Impossible downcast of toArray() result to String[]
Comparison of String objects using == or != (1)
Comparison of String objects using == or != in com.
Null pointer dereference (1)
Null pointer dereference of value in com.sw_enginee
Method assigns boolean literal in boolean expression
com.sw_engineering_candies.FindbugsPublicCode.q
Illegal format string (1)
Illegal format string " - %>s %s" in com.sw_engine
Normal confidence (2)
instanceof will always return false (1)
instanceof will always return false in com.sw_engine
BigDecimal constructed from double that isn't represer
BigDecimal constructed from 3.1 in com.sw_enginee
Of Concern (3)
Normal confidence (3)
instanceof will always return true (1)
instanceof will always return true for all non-null vali
Dead store to local variable (1)
Dead store to value in com.sw_engineering_candies.
Redundant nullcheck of value known to be null (1)
Redundant nullcheck of value which is known to be
Sample (10)
TheMonkeyBananaGame (4)
Scary (2)
Of Concern (2)
LeastCommonSubS
PlagiarismDetec
KMP.java
Banana.java
Game.java
Main.java
test.java
FindbugsPublicC
155 }
156
157 private static void esComparingStringsWithEqWRONG() {
158 final StringBuilder sb1 = new StringBuilder("1234");
159 final StringBuilder sb2 = new StringBuilder("1234");
160 final String string1 = sb1.toString();
161 final String string2 = sb2.toString();
162 System.out.println(" - " + (string1 == string2));
163 }
164
165 private static void esComparingStringsWithEqCORRECT() {
166 final StringBuilder sb1 = new StringBuilder("1234");
167 final StringBuilder sb2 = new StringBuilder("1234");
168 final String string1 = sb1.toString();
Bug Info
FindbugsPublicCode.java: 148
Navigation
BigDecimal constructed from 3.1 in com.sw_engineering_candies.FindbugsPublicCode.dmiBigDecimalConstructedFromDoubleWRONG()
Called method new java.math.BigDecimal(double)
Did you intend to invoke java.math.BigDecimal.valueOf(double)
Value 3.1
Value 3.1000000000000000888178419700125232338905334472656 25
Bug: BigDecimal constructed from 3.1 in com.sw_engineering_candies.FindbugsPublicCode.dmiBigDecimalConstructedFromDoubleWRONG()
This code creates a BigDecimal from a double value that doesn't translate well to a decimal number. For example, one might assume that writing new BigDecimal(0.1) in Java creates a BigDecimal which is exactly equal to 0.1 (an unscaled value of 1, with a scale of 1), but it is actually equal to 0.1000000000000000055511151231257827021181583404541015625. You probably want to use the BigDecimal.valueOf(double d) method, which uses the String representation of the double to create the BigDecimal (e.g., BigDecimal.valueOf(0.1) gives 0.1).
Rank: Scary (7), confidence: Normal
Pattern: DMI_BIGDECIMAL_CONSTRUCTED_FROM_DOUBLE
Type: DMI, Category: CORRECTNESS (Correctness)
XML output:
<BugInstance type="DMI_BIGDECIMAL_CONSTRUCTED_FROM_DOUBLE" priority="2" rank="7" abbrev="DMI" category="CORRECTNESS" first="1">
<Class classname="com.sw_engineering_candies.FindbugsPublicCode">
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" sourcefile="FindbugsPublicCode.java" sourcepath="com/sw_eng
</Class>
<Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="dmiBigDecimalConstructedFromDoubleWRONG" signature="()V" is
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="148" end="150" startBytecode="0" endBytecode="86" so
</Method>
<Method classname="java.math.BigDecimal" name="<init>" signature="(D)V" isStatic="false" role="METHOD_CALLED">
<SourceLine classname="java.math.BigDecimal" start="872" end="873" startBytecode="0" endBytecode="36" sourcefile="BigDecimal.java"
</Method>
<Method classname="java.math.BigDecimal" name="valueOf" signature="(D)Ljava/math/BigDecimal;" isStatic="true" role="METHOD_ALTERNAT
<SourceLine classname="java.math.BigDecimal" start="1274" end="1274" startBytecode="0" endBytecode="35" sourcefile="BigDecimal.ja
</Method>
BigDecimal constructed from 3.1 in com.sw_engineering_candies.FindbugsPublicCode.dmiBigDecimalConstructedFromDoubleWRONG() [Scary(7), Normal confidence]

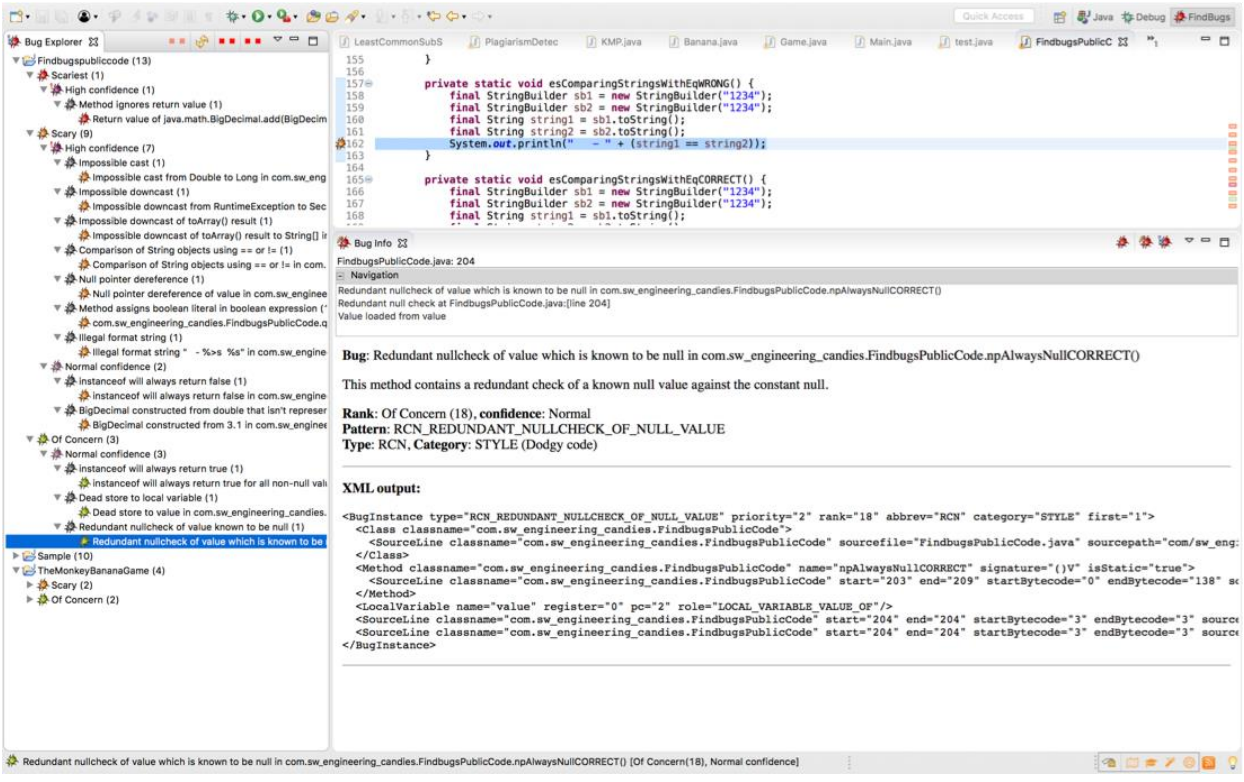
Bug 11


Bug Explorer
Findbugspubliccode (13)
Scariest (1)
High confidence (1)
Method ignores return value (1)
Return value of java.math.BigDecimal.add(BigDecim
Scary (9)
High confidence (7)
Impossible cast (1)
Impossible cast from Double to Long in com.sw_eng
Impossible downcast (1)
Impossible downcast from RuntimeException to Sec
Impossible downcast of toArray() result (1)
Impossible downcast of toArray() result to String[]
Comparison of String objects using == or != (1)
Comparison of String objects using == or != in com.
Null pointer dereference (1)
Null pointer dereference of value in com.sw_enginee
Method assigns boolean literal in boolean expression
com.sw_engineering_candies.FindbugsPublicCode.q
Illegal format string (1)
Illegal format string " - %>s %s" in com.sw_engine
Normal confidence (2)
instanceof will always return false (1)
instanceof will always return false in com.sw_engine
BigDecimal constructed from double that isn't represer
BigDecimal constructed from 3.1 in com.sw_enginee
Of Concern (3)
Normal confidence (3)
instanceof will always return true (1)
instanceof will always return true for all non-null val
Dead store to local variable (1)
Dead store to value in com.sw_engineering_candies.
Redundant nullcheck of value known to be null (1)
Redundant nullcheck of value which is known to be
Sample (10)
TheMonkeyBananaGame (4)
Scary (2)
Of Concern (2)
LeastCommonSubS
PlagiarismDetec
KMP.java
Banana.java
Game.java
Main.java
test.java
FindbugsPublicC
155 }
156
157 private static void esComparingStringsWithEqWRONG() {
158 final StringBuilder sb1 = new StringBuilder("1234");
159 final StringBuilder sb2 = new StringBuilder("1234");
160 final String string1 = sb1.toString();
161 final String string2 = sb2.toString();
162 System.out.println(" - " + (string1 == string2));
163 }
164
165 private static void esComparingStringsWithEqCORRECT() {
166 final StringBuilder sb1 = new StringBuilder("1234");
167 final StringBuilder sb2 = new StringBuilder("1234");
168 final String string1 = sb1.toString();
Bug Info
FindbugsPublicCode.java: 128
Navigation
instanceof will always return true for all non-null values in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleInstanceOfCORRECT(), since all Double are instances of Double
Actual type Double
Expected Double
Bug: instanceof will always return true for all non-null values in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleInstanceOfCORRECT(), since all Double are instances of Double
This instanceof test will always return true (unless the value being tested is null). Although this is safe, make sure it isn't an indication of some misunderstanding or some other logic error. If you really want to test the value for being null, perhaps it would be clearer to do better to do a null test rather than an instanceof test.
Rank: Of Concern (17), confidence: Normal
Pattern: BC_VACUOUS_INSTANCEOF
Type: BC, Category: STYLE (Dodgy code)
XML output:
<BugInstance type="BC_VACUOUS_INSTANCEOF" priority="2" rank="17" abbrev="BC" category="STYLE" first="1">
<Class classname="com.sw_engineering_candies.FindbugsPublicCode">
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" sourcefile="FindbugsPublicCode.java" sourcepath="com/sw_eng
</Class>
<Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="bcImpossibleInstanceOfCORRECT" signature="()V" isStatic="tru
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="127" end="129" startBytecode="0" endBytecode="13" so
</Method>
<Type descriptor="Ljava/lang/Double;" role="TYPE_FOUND">
<SourceLine classname="java.lang.Double" start="49" end="1053" sourcefile="Double.java" sourcepath="java/lang/Double.java"/>
</Type>
<Type descriptor="Ljava/lang/Double;" role="TYPE_EXPECTED">
<SourceLine classname="java.lang.Double" start="49" end="1053" sourcefile="Double.java" sourcepath="java/lang/Double.java"/>
</Type>
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="128" end="128" startBytecode="18" endBytecode="18" sou
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="128" end="128" startBytecode="18" endBytecode="18" sou
</BugInstance>
instanceof will always return true for all non-null values in com.sw_engineering_candies.F...eOfCORRECT(), since all Double are instances of Double [Of Concern(17), Normal confidence]

Bug 12


Bug Explorer
Findbugspubliccode (13)
Scariest (1)
High confidence (1)
Method ignores return value (1)
Return value of java.math.BigDecimal.add(BigDecim
Scary (9)
High confidence (7)
Impossible cast (1)
Impossible cast from Double to Long in com.sw_eng
Impossible downcast (1)
Impossible downcast from RuntimeException to Sec
Impossible downcast of toArray() result (1)
Impossible downcast of toArray() result to String[] ir
Comparison of String objects using == or != (1)
Comparison of String objects using == or != in com.
Null pointer dereference (1)
Null pointer dereference of value in com.sw_enginee
Method assigns boolean literal in boolean expression ("
com.sw_engineering_candies.FindbugsPublicCode.q
Illegal format string (1)
Illegal format string " - %>s %s" in com.sw_engine
Normal confidence (2)
instanceof will always return false (1)
instanceof will always return false in com.sw_engine
BigDecimal constructed from double that isn't represer
BigDecimal constructed from 3.1 in com.sw_enginee
Of Concern (3)
Normal confidence (3)
instanceof will always return true (1)
instanceof will always return true for all non-null vali
Dead store to local variable (1)
Dead store to value in com.sw_engineering_candies.
Redundant nullcheck of value known to be null (1)
Redundant nullcheck of value which is known to be
Sample (10)
TheMonkeyBananaGame (4)
Scary (2)
Of Concern (2)
LeastCommonSubS
PlagiarismDetec
KMP.java
Banana.java
Game.java
Main.java
test.java
FindbugsPublicC
155 }
156
157 private static void esComparingStringsWithEqWRONG() {
158 final StringBuilder sb1 = new StringBuilder("1234");
159 final StringBuilder sb2 = new StringBuilder("1234");
160 final String string1 = sb1.toString();
161 final String string2 = sb2.toString();
162 System.out.println(" - " + (string1 == string2));
163 }
164
165 private static void esComparingStringsWithEqCORRECT() {
166 final StringBuilder sb1 = new StringBuilder("1234");
167 final StringBuilder sb2 = new StringBuilder("1234");
168 final String string1 = sb1.toString();
Bug Info
FindbugsPublicCode.java: 213
Navigation
Dead store to value in com.sw_engineering_candies.FindbugsPublicCode.qabQuestionableBooleanAssignmentWRONG()
Local variable named value
Bug: Dead store to value in com.sw_engineering_candies.FindbugsPublicCode.qabQuestionableBooleanAssignmentWRONG()
This instruction assigns a value to a local variable, but the value is not read or used in any subsequent instruction. Often, this indicates an error, because the value computed is never used.
Note that Sun's javac compiler often generates dead stores for final local variables. Because FindBugs is a bytecode-based tool, there is no easy way to eliminate these false positives.
Rank: Of Concern (17), confidence: Normal
Pattern: DLS_DEAD_LOCAL_STORE
Type: DLS, Category: STYLE (Dodgy code)
XML output:
<BugInstance type="DLS_DEAD_LOCAL_STORE" priority="2" rank="17" abbrev="DLS" category="STYLE" first="1">
<Class classname="com.sw_engineering_candies.FindbugsPublicCode">
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" sourcefile="FindbugsPublicCode.java" sourcepath="com/sw_eng
</Class>
<Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="qabQuestionableBooleanAssignmentWRONG" signature="()V" isSt
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="212" end="218" startBytecode="0" endBytecode="19" so
</Method>
<LocalVariable name="value" register="0" pc="5" role="LOCAL_VARIABLE_NAMED"/>
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="213" end="213" startBytecode="4" endBytecode="4" source
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="213" end="213" startBytecode="4" endBytecode="4" source
<Property name="edu.umd.cs.findbugs.detect.DeadLocalStoreProperty.BASE_VALUE" value="true"/>
<Property name="edu.umd.cs.findbugs.detect.DeadLocalStoreProperty.LOCAL_NAME" value="value"/>
<Property name="edu.umd.cs.findbugs.detect.DeadLocalStoreProperty.NO_LOADS" value="true"/>
</BugInstance>
Dead store to value in com.sw_engineering_candies.FindbugsPublicCode.qabQuestionableBooleanAssignmentWRONG() [Of Concern(17), Normal confidence]

Bug 13


Bug Explorer
Findbugspubliccode (13)
Scariest (1)
High confidence (1)
Method ignores return value (1)
Return value of java.math.BigDecimal.add(BigDecim
Scary (9)
High confidence (7)
Impossible cast (1)
Impossible cast from Double to Long in com.sw_eng
Impossible downcast (1)
Impossible downcast from RuntimeException to Sec
Impossible downcast of toArray() result (1)
Impossible downcast of toArray() result to String[] ir
Comparison of String objects using == or != (1)
Comparison of String objects using == or != in com.
Null pointer dereference (1)
Null pointer dereference of value in com.sw_enginee
Method assigns boolean literal in boolean expression ("
com.sw_engineering_candies.FindbugsPublicCode.q
Illegal format string (1)
Illegal format string " - %>s %s" in com.sw_engine
Normal confidence (2)
instanceof will always return false (1)
instanceof will always return false in com.sw_engine
BigDecimal constructed from double that isn't represer
BigDecimal constructed from 3.1 in com.sw_enginee
Of Concern (3)
Normal confidence (3)
instanceof will always return true (1)
instanceof will always return true for all non-null vali
Dead store to local variable (1)
Dead store to value in com.sw_engineering_candies.
Redundant nullcheck of value known to be null (1)
Redundant nullcheck of value which is known to be
Sample (10)
TheMonkeyBananaGame (4)
Scary (2)
Of Concern (2)
LeastCommonSubS
PlagiarismDetec
KMP.java
Banana.java
Game.java
Main.java
test.java
FindbugsPublicC
155 }
156
157 private static void esComparingStringsWithEqWRONG() {
158 final StringBuilder sb1 = new StringBuilder("1234");
159 final StringBuilder sb2 = new StringBuilder("1234");
160 final String string1 = sb1.toString();
161 final String string2 = sb2.toString();
162 System.out.println(" - " + (string1 == string2));
163 }
164
165 private static void esComparingStringsWithEqCORRECT() {
166 final StringBuilder sb1 = new StringBuilder("1234");
167 final StringBuilder sb2 = new StringBuilder("1234");
168 final String string1 = sb1.toString();
Bug Info
FindbugsPublicCode.java: 204
Navigation
Redundant nullcheck of value which is known to be null in com.sw_engineering_candies.FindbugsPublicCode.npAlwaysNullCORRECT()
Redundant null check at FindbugsPublicCode.java:[line 204]
Value loaded from value
Bug: Redundant nullcheck of value which is known to be null in com.sw_engineering_candies.FindbugsPublicCode.npAlwaysNullCORRECT()
This method contains a redundant check of a known null value against the constant null.
Rank: Of Concern (18), confidence: Normal
Pattern: RCN_REDUNDANT_NULLCHECK_OF_NULL_VALUE
Type: RCN, Category: STYLE (Dodgy code)
XML output:
<BugInstance type="RCN_REDUNDANT_NULLCHECK_OF_NULL_VALUE" priority="2" rank="18" abbrev="RCN" category="STYLE" first="1">
<Class classname="com.sw_engineering_candies.FindbugsPublicCode">
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" sourcefile="FindbugsPublicCode.java" sourcepath="com/sw_eng
</Class>
<Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="npAlwaysNullCORRECT" signature="()V" isStatic="true">
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="203" end="209" startBytecode="0" endBytecode="138" s
</Method>
<LocalVariable name="value" register="0" pc="2" role="LOCAL_VARIABLE_VALUE_OF"/>
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="204" end="204" startBytecode="3" endBytecode="3" source
<SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="204" end="204" startBytecode="3" endBytecode="3" source
</BugInstance>
Redundant nullcheck of value which is known to be null in com.sw_engineering_candies.FindbugsPublicCode.npAlwaysNullCORRECT() [Of Concern(18), Normal confidence]

SonarQube

SonarQube, is a self-managed, automatic code review tool that systematically helps you deliver Clean Code. As a core element of our Sonar solution, SonarQube integrates into your existing workflow and detects issues in your code to help you perform continuous code inspections of your projects.

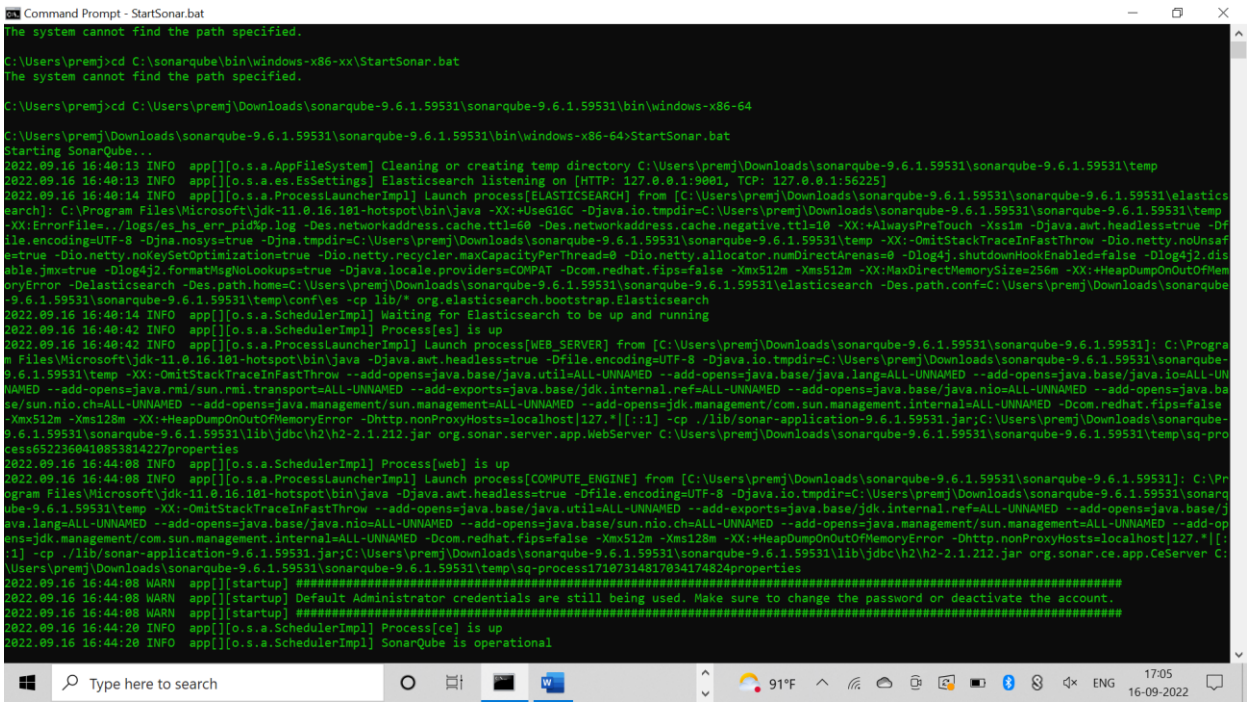
SonarQube is a Static Analysis tool which collects and analyses source code and provides Code Quality Assurance by detecting code vulnerabilities and issues.

Website to download the SonarQube: - https://www.sonarqube.org/successdownload-community-edition/

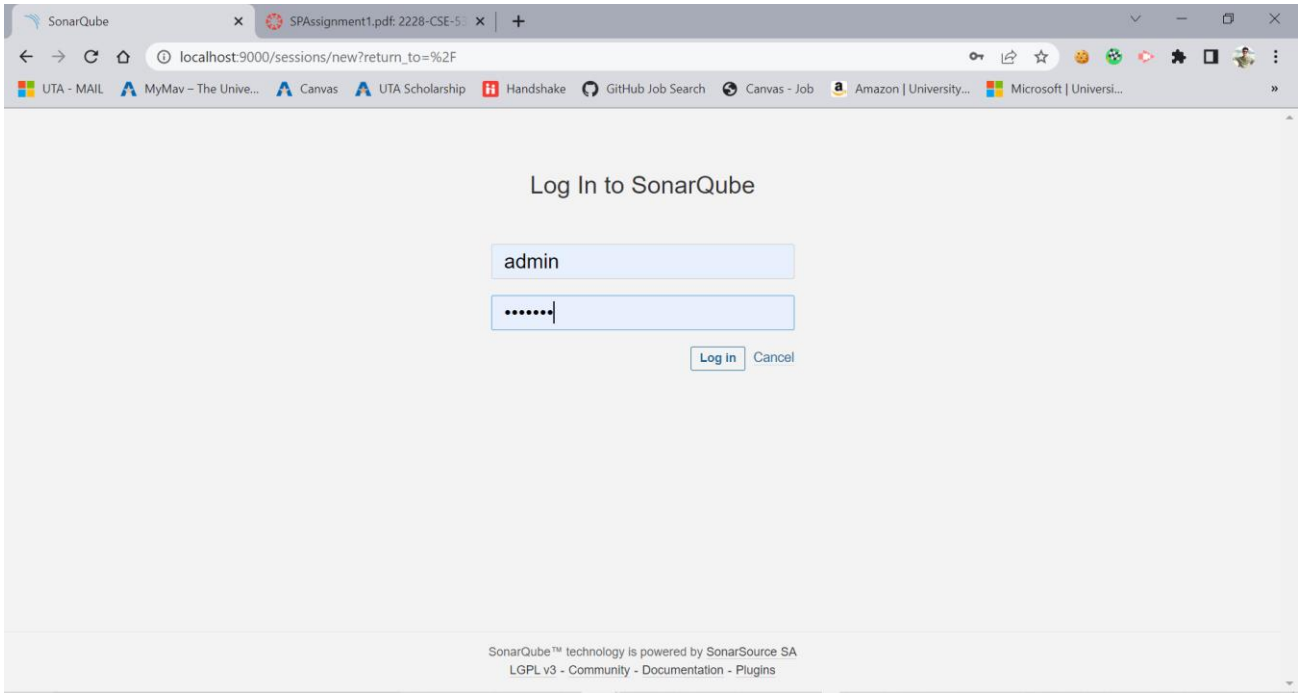
https://docs.sonarqube.org/latest/

Inputs: SonarQube takes entire code (Java) as an input. It will go through each line of the source code to detect bugs, vulnerabilities, and defects.

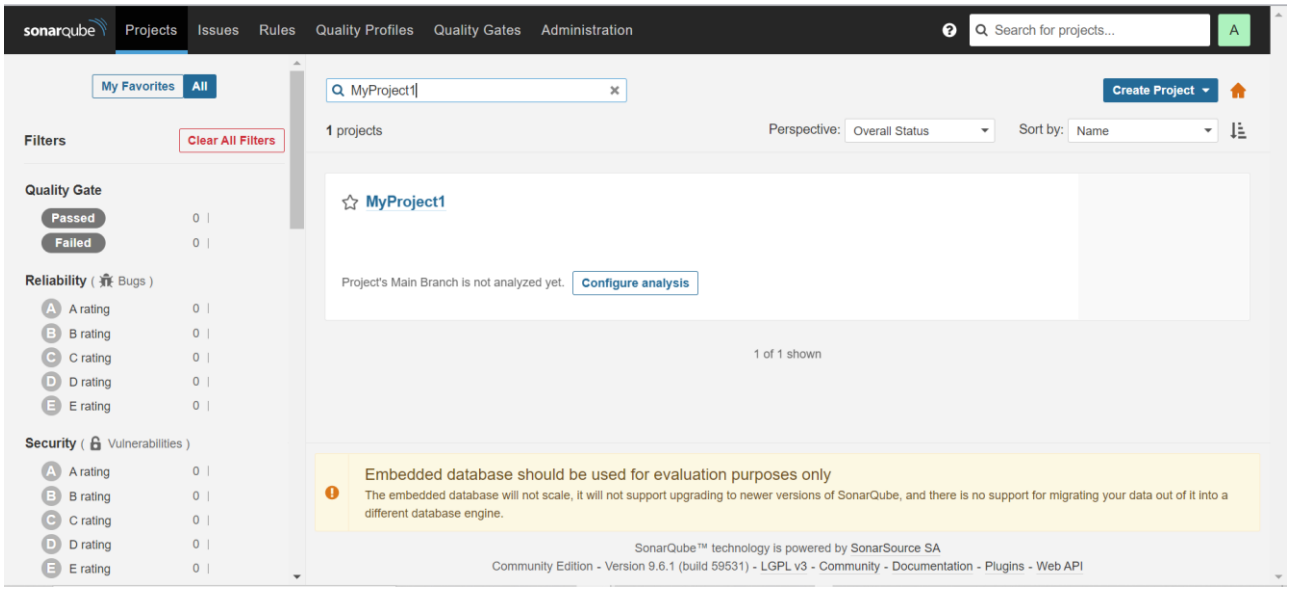
In the below screenshot I have started the SonarQube using terminal (windows)

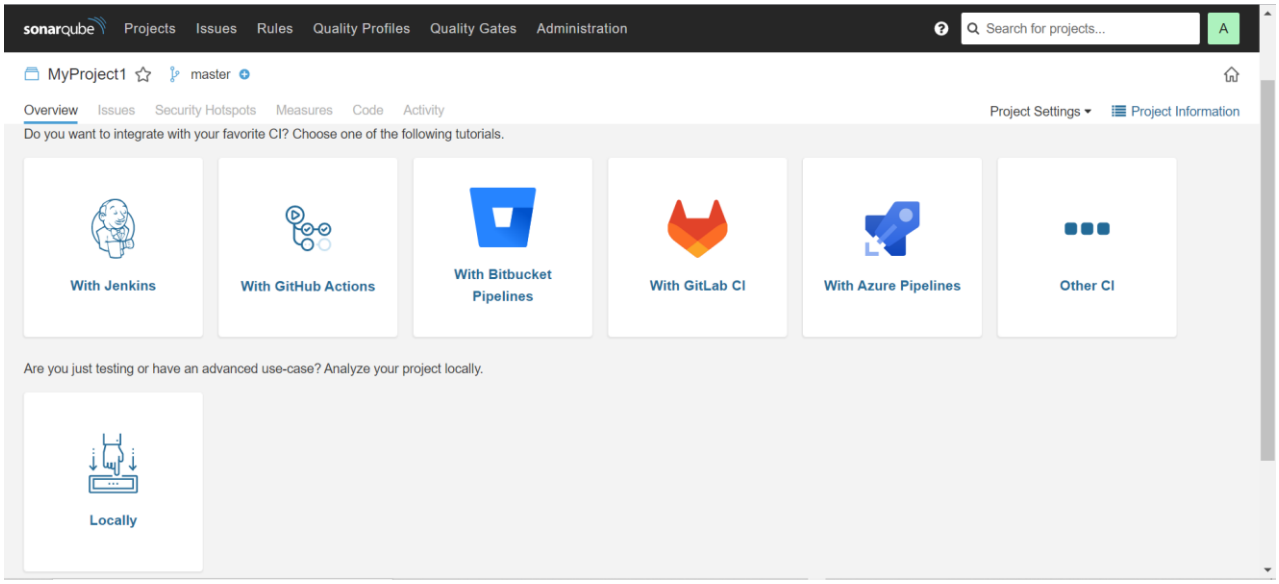


Once the SonarQube is up, the following login page appears at **localhost:9000**

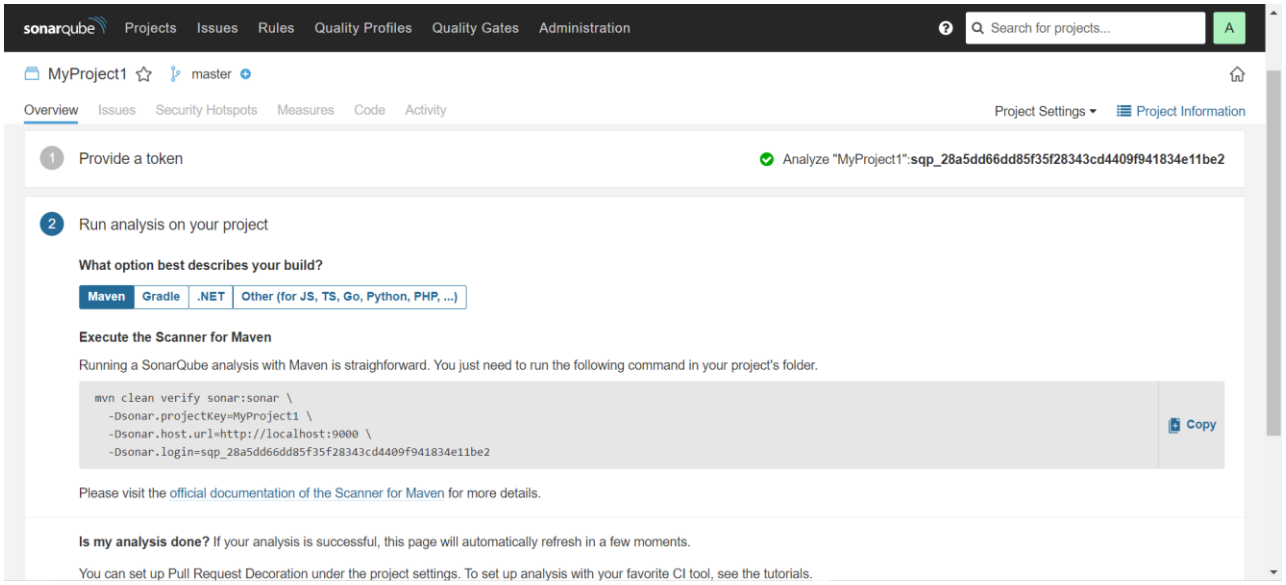


After logging in I created the project, I chose to analyse the project locally.
To analyze it locally I decided to use Maven tool.

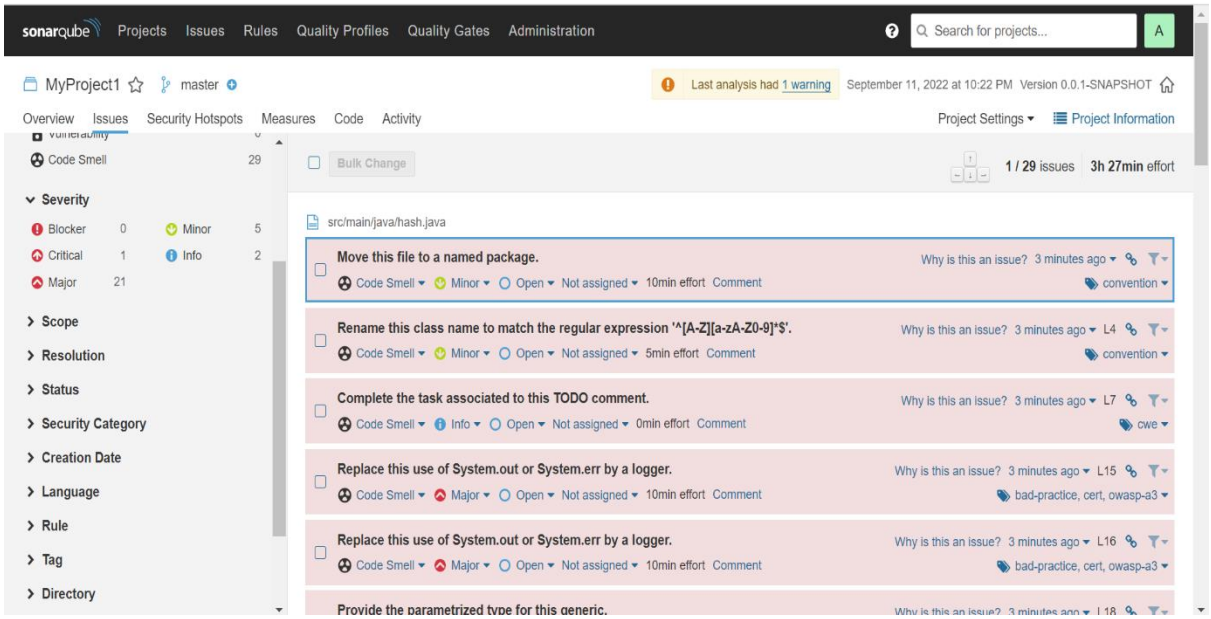




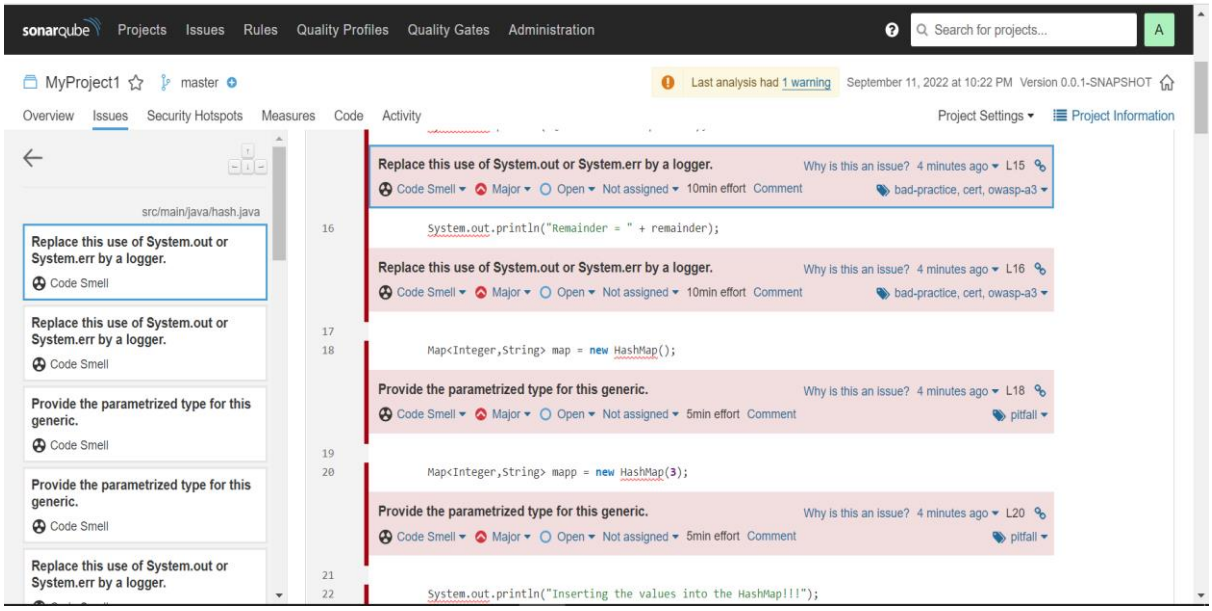
Manually uploaded the code using Maven 3.8.6

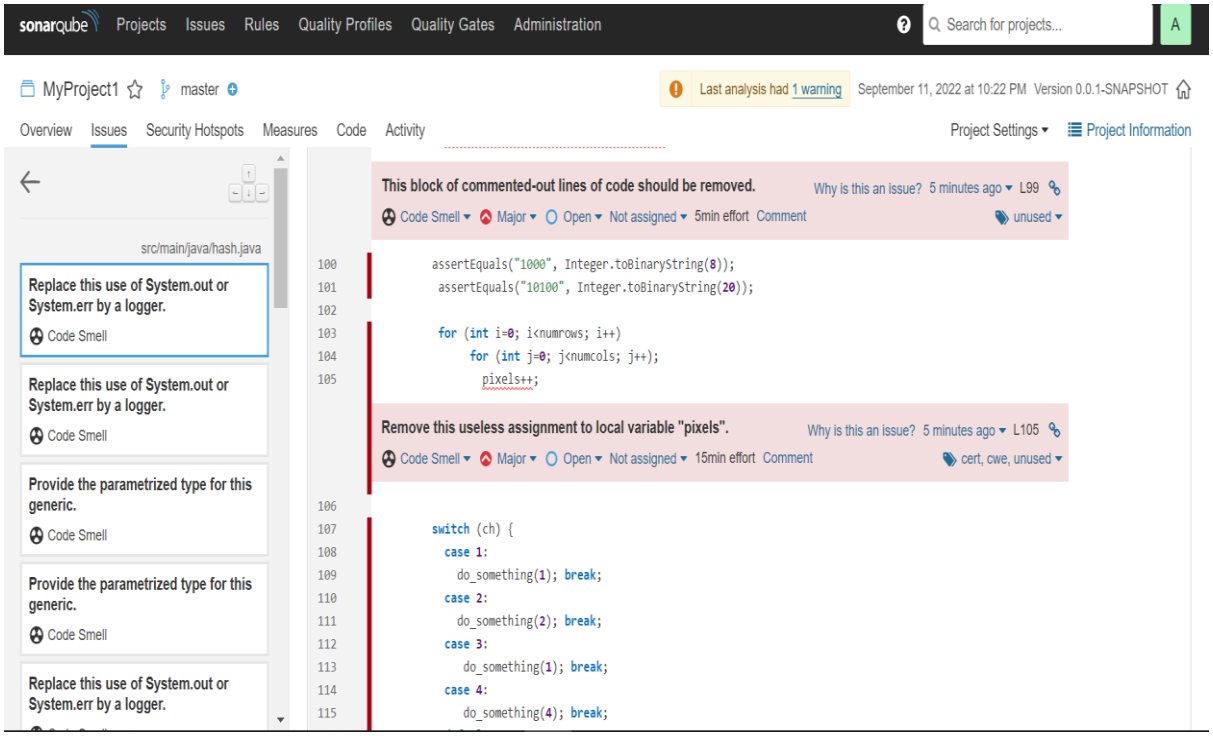


Once the analysis is completed the SonarQube launches an analysis report. It lists the issues as well as the severity of the issues.

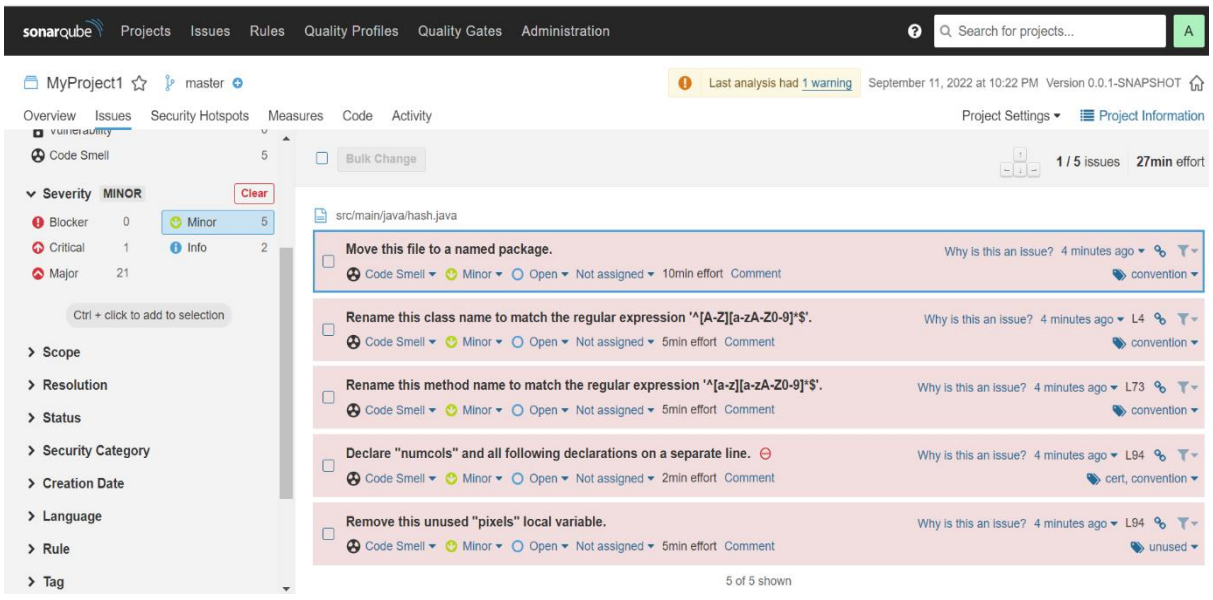


SonarQube gives suggestions about how the issue can be avoided





In the following image, SonarQube identifies minor bugs and gives the description



Conclusion:

- Both the tools take Java Byte code as an input.

- Following bugs can be found:
  FindBugs: Performance, Bad practice, Dodgy , Internationalization, Malicious code, Vulnerability, Bogus random, Noise, Correctness, Multithreaded, Style.
- FindBugs cannot find Security category, so we have FindSecurityBugs with the security mode enabled.
  FindSecurityBugs: Security Bugs is to limit the scan to Security only bug detectors on, i.e. when the security category is enabled it is FindSecurityBugs Security.s

- FindBugs/FindSecurityBugs will find based on the suspicious patterns in the byte code and when it hits such byte strings in our byte code, it flags it to vulnerability.

- As **SonarQube** can work as both client and server. So, it is more efficient than **FindBugs**.