CSE 5382 Secure Programming Fall 2018, Programming Assignment 1
Introduction to Static Analysis
Swetha, Vijaya Raghavan | sxv1229| 1001551229

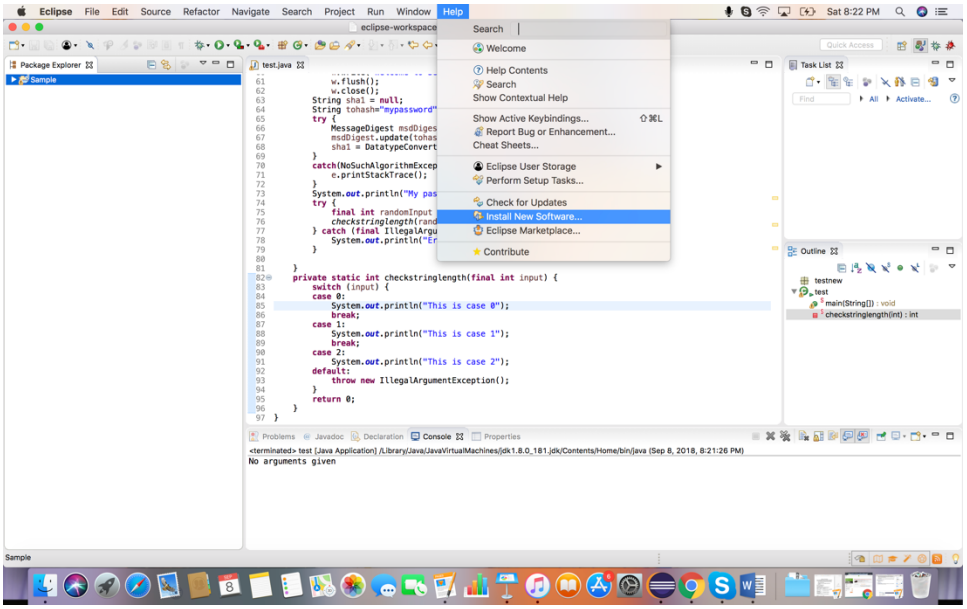
Tool Versions

The tools I have used to test the code are:

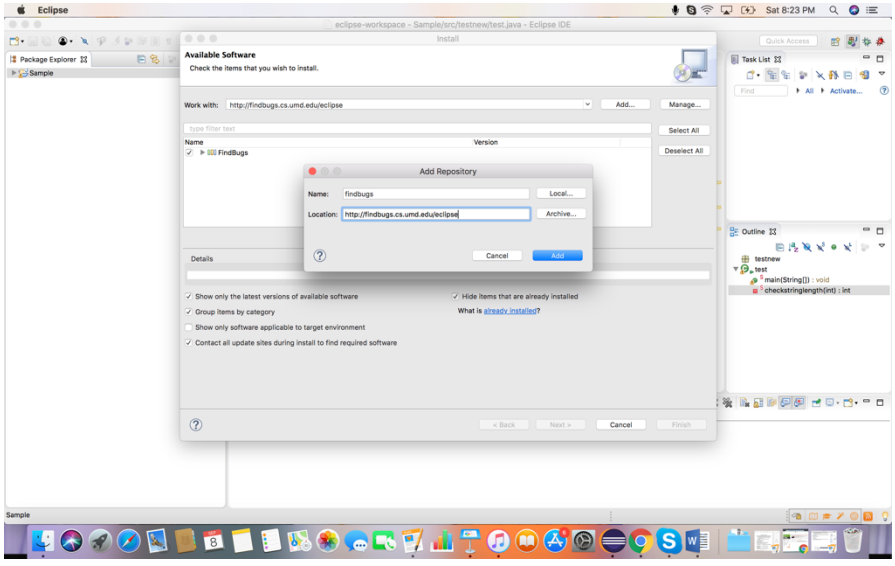
- Eclipse IDE for Java Developers, Java JDK 8
- Findbugs – 3.0.1, findsecbugs-plugin-1.8.0

Tool Invocation Process

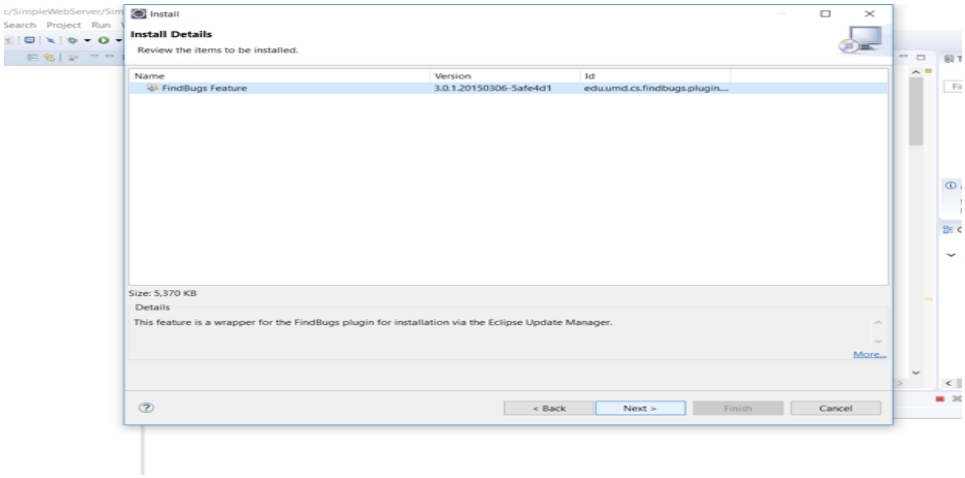
1. As Eclipse is installed, From Help tab, go to "Install New Software" to add the Findbugs plugin as shown below:
   a. Click on "Install New Software



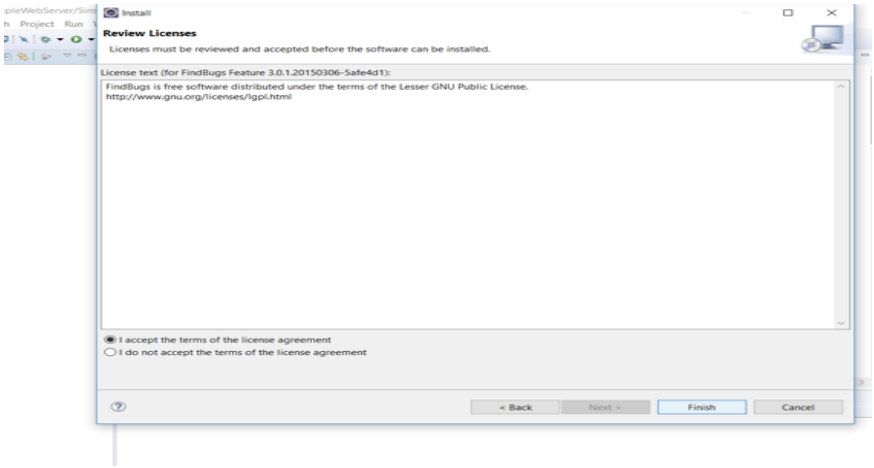
   b. Give the url to get the findbugs plugin for eclipse as: http://findbugs.cs.umd.edu/eclipse and click on "Add"



c. Click ‘Next’ again as shown below:



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

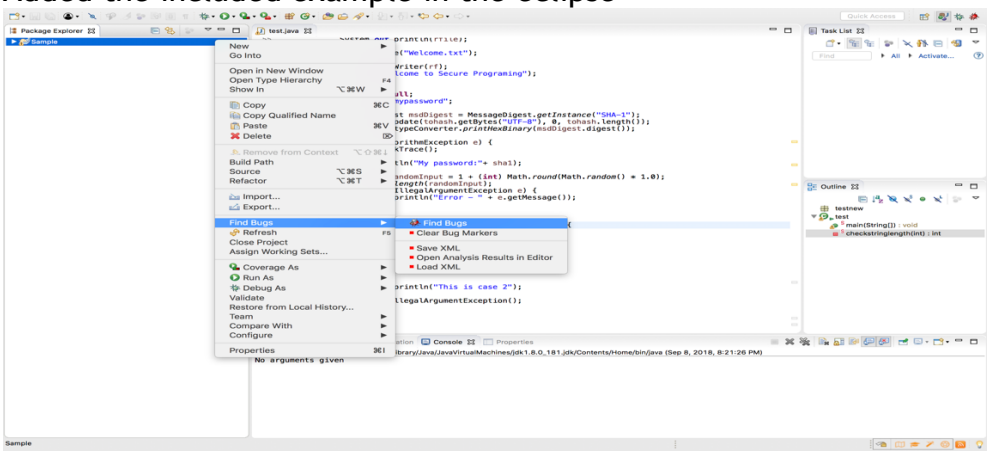


e. It will ask you to restart eclipse, click ok to restart eclipse.

f. To run the tool, select the project in the package explorer and select Find Bugs and in the sub menu, select ‘Find Bugs’ again.

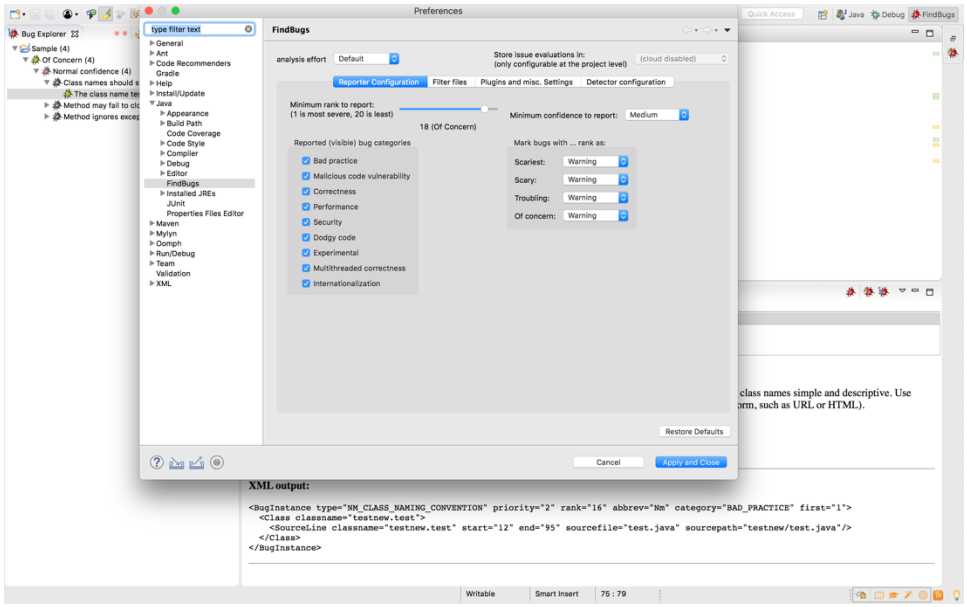
g. Added the included example in the eclipse

h.



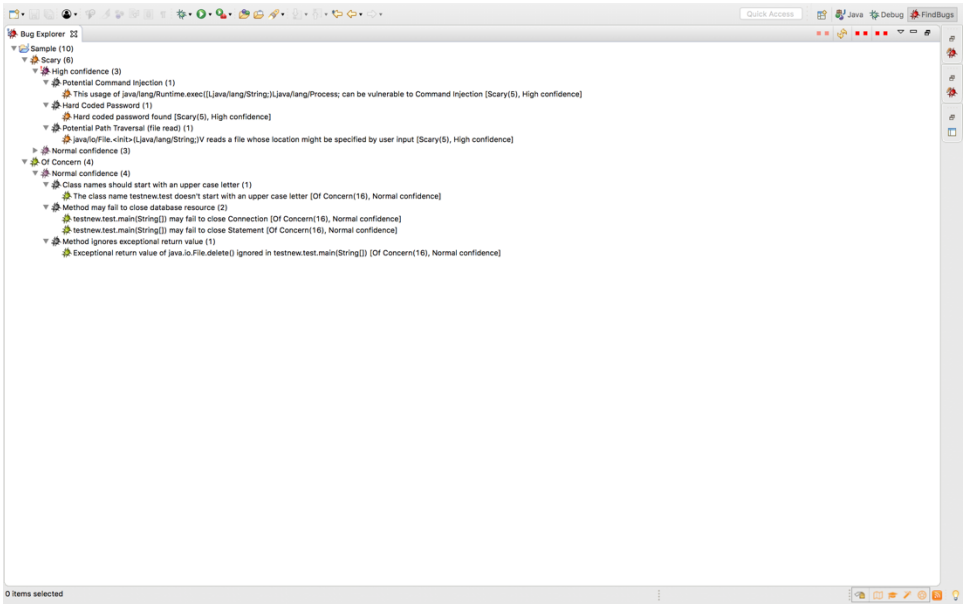
i. The below Find bugs explorer will display the bugs present in the code

j. The recommended configuration to use with Find Security Bugs is to limit the scan to Securityonly 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.

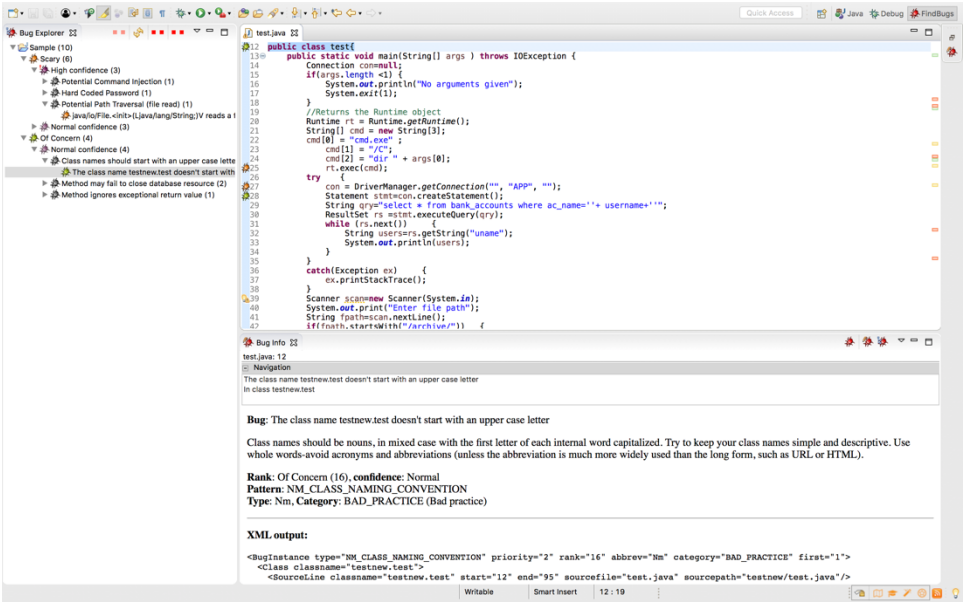


k. This is enabled for the following screenshots. Able to find 10 bugs(6 scary, 4 of concern)

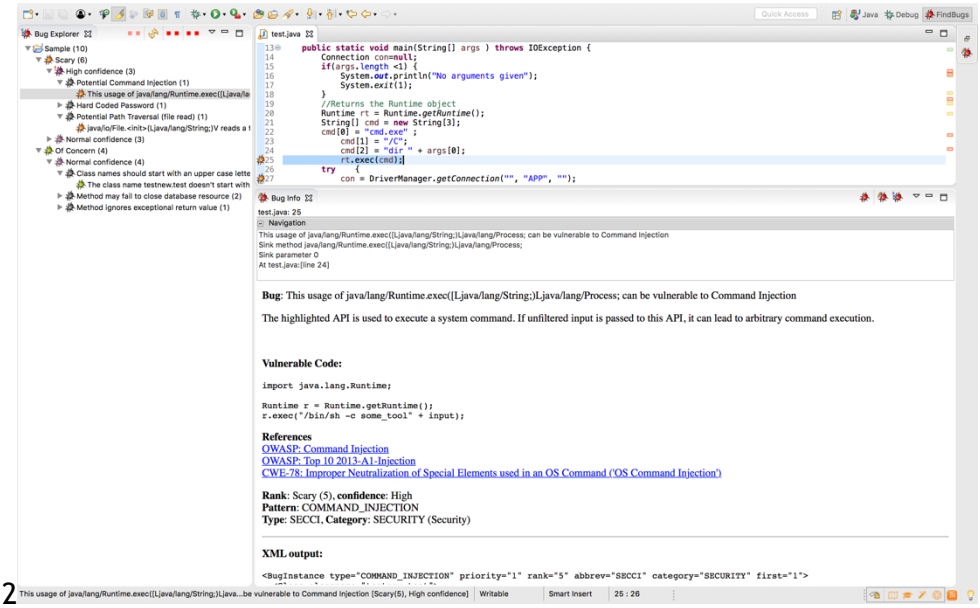
l.



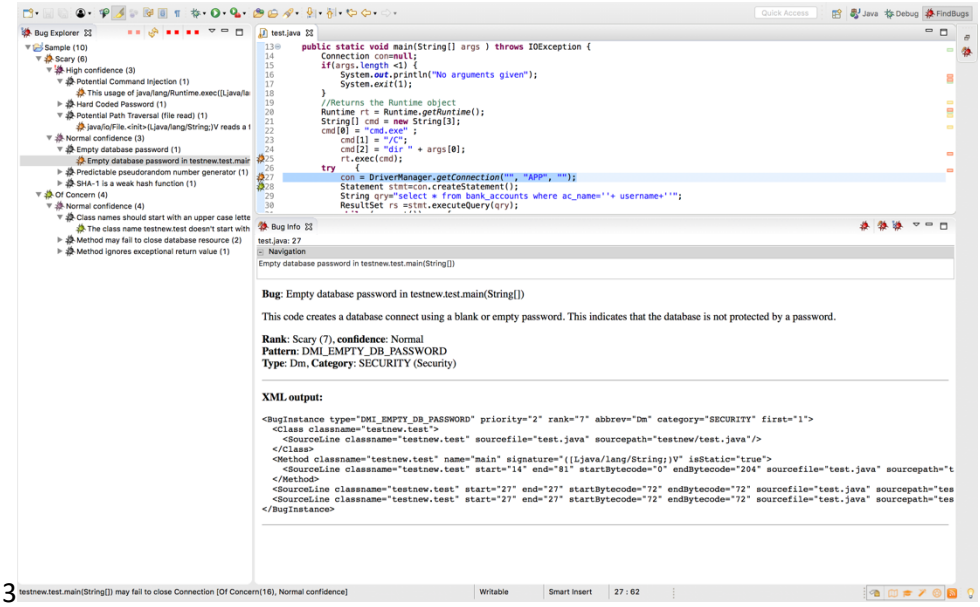
m. Bug 1


Bug Explorer
Sample (10)
Scary (6)
High confidence (3)
Potential Command Injection (1)
Hard Coded Password (1)
Potential Path Traversal (file read) (1)
Normal confidence (3)
Of Concern (4)
Normal confidence (4)
Class names should start with an upper case lette
The class name testnew.test doesn't start with
Method may fail to close database resource (2)
Method ignores exceptional return value (1)
test.java
public class test{
public static void main(String[] args) throws IOException {
Connection con=null;
if(args.length <1) {
System.out.println("No arguments given");
System.exit(1);
}
//Returns the Runtime object
Runtime rt = Runtime.getRuntime();
String[] cmd = new String[3];
cmd[0] = "cmd.exe" ;
cmd[1] = "/C";
cmd[2] = "dir " + args[0];
rt.exec(cmd);
try {
con = DriverManager.getConnection("", "APP", "");
Statement stmt=con.createStatement();
String qry="select * from bank_accounts where ac_name='"+ username+"'";
ResultSet rs =stmt.executeQuery(qry);
while (rs.next()) {
String users=rs.getString("uname");
System.out.println(users);
}
}
catch(Exception ex) {
ex.printStackTrace();
}
Scanner scan=new Scanner(System.in);
System.out.print("Enter file path");
String fpath=scan.nextLine();
Bug Info
test.java: 12
Navigation
The class name testnew.test doesn't start with an upper case letter
In class testnew.test
Bug: The class name testnew.test doesn't start with an upper case letter
Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).
Rank: Of Concern (16), confidence: Normal
Pattern: NM_CLASS_NAMING_CONVENTION
Type: Nm, Category: BAD_PRACTICE (Bad practice)
XML output:
<BugInstance type="NM_CLASS_NAMING_CONVENTION" priority="2" rank="16" abbrev="Nm" category="BAD_PRACTICE" first="1">
<Class classname="testnew.test">
<SourceLine classname="testnew.test" start="12" end="95" sourcefile="test.java" sourcepath="testnew/test.java"/>
Writable
Smart Insert
12 : 19

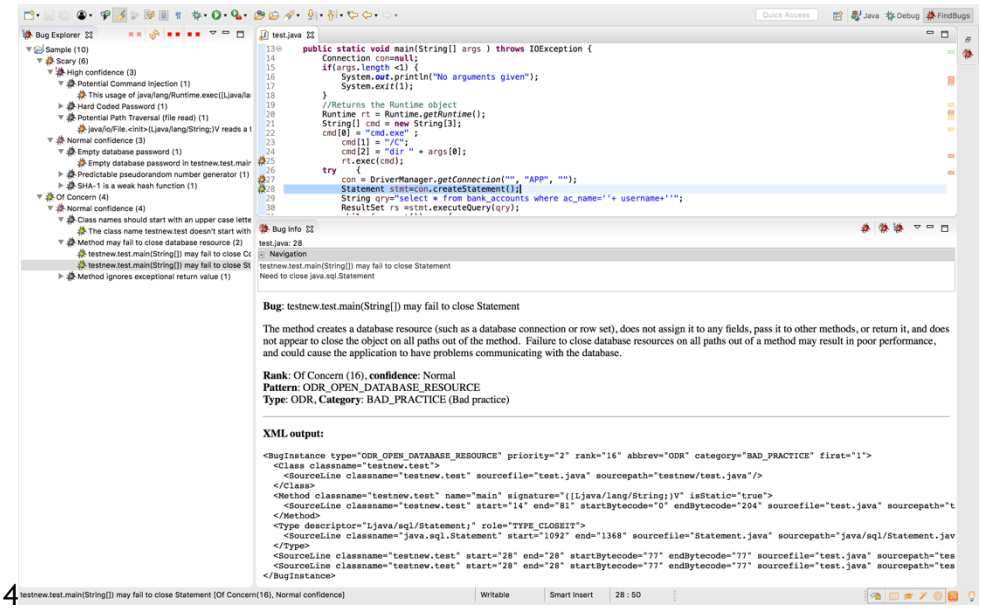
n. Bug 2


Bug Explorer
Sample (10)
Scary (6)
High confidence (3)
Potential Command Injection (1)
This usage of java/lang/Runtime.exec([Ljava/la
Hard Coded Password (1)
Potential Path Traversal (file read) (1)
Normal confidence (3)
Of Concern (4)
Normal confidence (4)
Class names should start with an upper case lette
The class name testnew.test doesn't start with
Method may fail to close database resource (2)
Method ignores exceptional return value (1)
test.java
public static void main(String[] args) throws IOException {
Connection con=null;
if(args.length <1) {
System.out.println("No arguments given");
System.exit(1);
}
//Returns the Runtime object
Runtime rt = Runtime.getRuntime();
String[] cmd = new String[3];
cmd[0] = "cmd.exe" ;
cmd[1] = "/C";
cmd[2] = "dir " + args[0];
rt.exec(cmd);
try {
con = DriverManager.getConnection("", "APP", "");
Bug Info
test.java: 25
Navigation
This usage of java/lang/Runtime.exec([Ljava/lang/String;)Ljava/lang/Process; can be vulnerable to Command Injection
Sink method java/lang/Runtime.exec([Ljava/lang/String;)Ljava/lang/Process;
Sink parameter 0
At test.java:[line 24]
Bug: This usage of java/lang/Runtime.exec([Ljava/lang/String;)Ljava/lang/Process; can be vulnerable to Command Injection
The highlighted API is used to execute a system command. If unfiltered input is passed to this API, it can lead to arbitrary command execution.
Vulnerable Code:
import java.lang.Runtime;
Runtime r = Runtime.getRuntime();
r.exec("/bin/sh -c some_tool" + input);
References
OWASP: Command Injection
OWASP: Top 10 2013-A1-Injection
CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Rank: Scary (5), confidence: High
Pattern: COMMAND_INJECTION
Type: SECCI, Category: SECURITY (Security)
XML output:
<BugInstance type="COMMAND_INJECTION" priority="1" rank="5" abbrev="SECCI" category="SECURITY" first="1">
This usage of java/lang/Runtime.exec([Ljava/lang/String;)Ljava...be vulnerable to Command Injection [Scary(5), High confidence]
Writable
Smart Insert
25 : 26

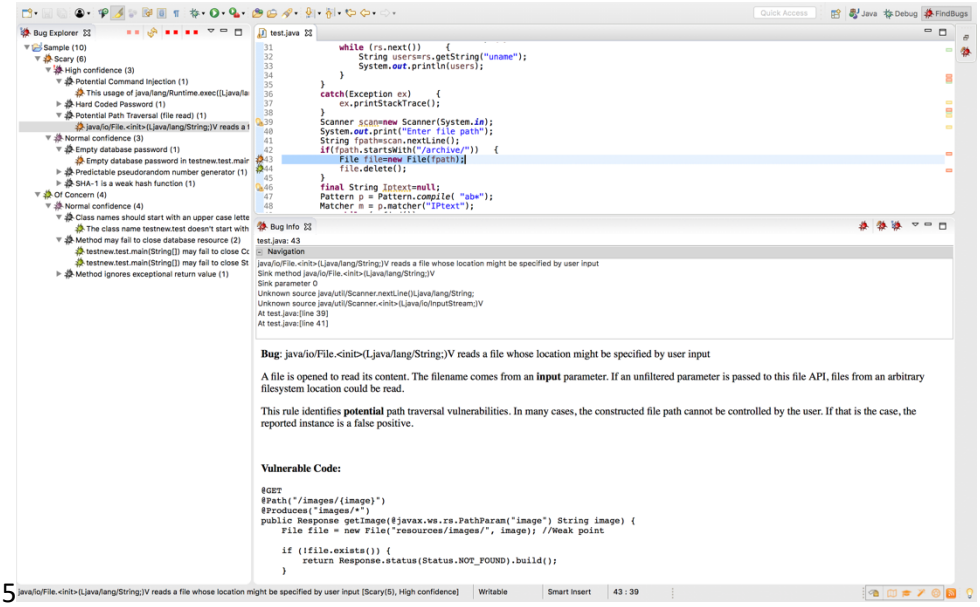
o. Bug 3


Bug Explorer
Sample (10)
Scary (6)
High confidence (3)
Potential Command Injection (1)
This usage of java/lang/Runtime.exec([Ljava/la
Hard Coded Password (1)
Potential Path Traversal (file read) (1)
java/io/File.<init>([Ljava/lang/String;)V reads a f
Normal confidence (3)
Empty database password (1)
Empty database password in testnew.test.mair
Predictable pseudorandom number generator (1)
SHA-1 is a weak hash function (1)
Of Concern (4)
Normal confidence (4)
Class names should start with an upper case lette
The class name testnew.test doesn't start with
Method may fail to close database resource (2)
Method ignores exceptional return value (1)
test.java
public static void main(String[] args) throws IOException {
Connection con=null;
if(args.length <1) {
System.out.println("No arguments given");
System.exit(1);
}
//Returns the Runtime object
Runtime rt = Runtime.getRuntime();
String[] cmd = new String[3];
cmd[0] = "cmd.exe" ;
cmd[1] = "/C";
cmd[2] = "dir " + args[0];
rt.exec(cmd);
try {
con = DriverManager.getConnection("", "APP", "");
Statement stmt=con.createStatement();
String qry="select * from bank_accounts where ac_name='"+ username+"'";
ResultSet rs =stmt.executeQuery(qry);
Bug Info
test.java: 27
Navigation
Empty database password in testnew.test.main(String[])
Bug: Empty database password in testnew.test.main(String[])
This code creates a database connect using a blank or empty password. This indicates that the database is not protected by a password.
Rank: Scary (7), confidence: Normal
Pattern: DMI_EMPTY_DB_PASSWORD
Type: Dm, Category: SECURITY (Security)
XML output:
<BugInstance type="DMI_EMPTY_DB_PASSWORD" priority="2" rank="7" abbrev="Dm" category="SECURITY" first="1">
<Class classname="testnew.test">
<SourceLine classname="testnew.test" sourcefile="test.java" sourcepath="testnew/test.java"/>
</Class>
<Method classname="testnew.test" name="main" signature="([Ljava/lang/String;)V" isStatic="true">
<SourceLine classname="testnew.test" start="14" end="81" startBytecode="0" endBytecode="204" sourcefile="test.java" sourcepath="t
</Method>
<SourceLine classname="testnew.test" start="27" end="27" startBytecode="72" endBytecode="72" sourcefile="test.java" sourcepath="tes
<SourceLine classname="testnew.test" start="27" end="27" startBytecode="72" endBytecode="72" sourcefile="test.java" sourcepath="tes
</BugInstance>
testnew.test.main(String[]) may fail to close Connection [Of Concern(16), Normal confidence]
Writable
Smart Insert
27 : 62

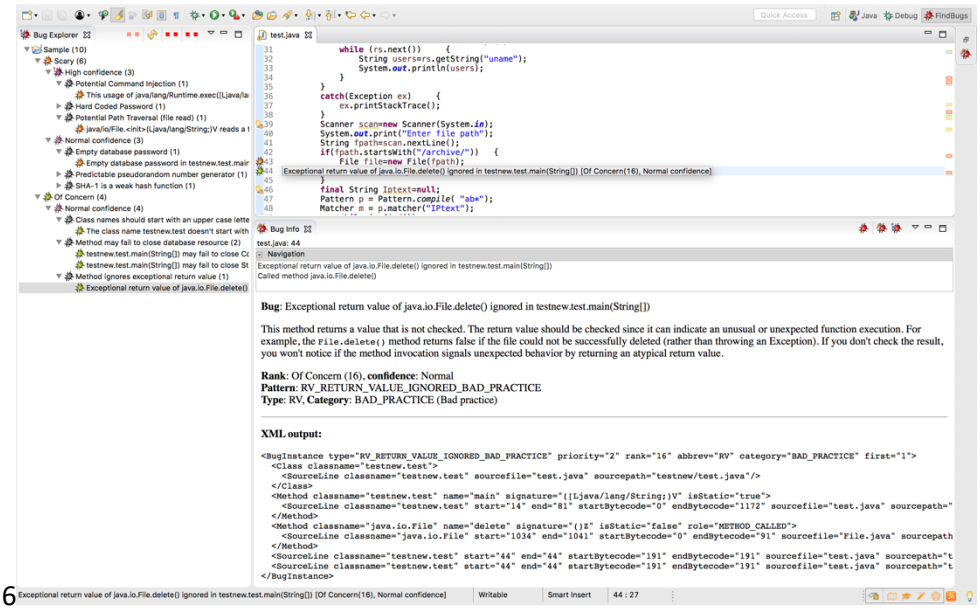
p. Bug 4


Bug Explorer
Sample (10)
Scary (6)
High confidence (3)
Potential Command Injection (1)
This usage of java/lang/Runtime.exec([Ljava/la
Hard Coded Password (1)
Potential Path Traversal (file read) (1)
java/io/File.<init>([Ljava/lang/String;)V reads a f
Normal confidence (3)
Empty database password (1)
Empty database password in testnew.test.mair
Predictable pseudorandom number generator (1)
SHA-1 is a weak hash function (1)
Of Concern (4)
Normal confidence (4)
Class names should start with an upper case lette
The class name testnew.test doesn't start with
Method may fail to close database resource (2)
testnew.test.main(String[]) may fail to close Cc
testnew.test.main(String[]) may fail to close St
Method ignores exceptional return value (1)
test.java
public static void main(String[] args) throws IOException {
Connection con=null;
if(args.length <1) {
System.out.println("No arguments given");
System.exit(1);
}
//Returns the Runtime object
Runtime rt = Runtime.getRuntime();
String[] cmd = new String[3];
cmd[0] = "cmd.exe" ;
cmd[1] = "/C";
cmd[2] = "dir " + args[0];
rt.exec(cmd);
try {
con = DriverManager.getConnection("", "APP", "");
Statement stmt=con.createStatement();
String qry="select * from bank_accounts where ac_name='"+ username+"'";
ResultSet rs =stmt.executeQuery(qry);
Bug Info
test.java: 28
Navigation
testnew.test.main(String[]) may fail to close Statement
Need to close java.sql.Statement
Bug: testnew.test.main(String[]) may fail to close Statement
The method creates a database resource (such as a database connection or row set), does not assign it to any fields, pass it to other methods, or return it, and does not appear to close the object on all paths out of the method. Failure to close database resources on all paths out of a method may result in poor performance, and could cause the application to have problems communicating with the database.
Rank: Of Concern (16), confidence: Normal
Pattern: ODR_OPEN_DATABASE_RESOURCE
Type: ODR, Category: BAD_PRACTICE (Bad practice)
XML output:
<BugInstance type="ODR_OPEN_DATABASE_RESOURCE" priority="2" rank="16" abbrev="ODR" category="BAD_PRACTICE" first="1">
<Class classname="testnew.test">
<SourceLine classname="testnew.test" sourcefile="test.java" sourcepath="testnew/test.java"/>
</Class>
<Method classname="testnew.test" name="main" signature="([Ljava/lang/String;)V" isStatic="true">
<SourceLine classname="testnew.test" start="14" end="81" startBytecode="0" endBytecode="204" sourcefile="test.java" sourcepath="t
</Method>
<Type descriptor="Ljava/sql/Statement;" role="TYPE_CLOSEIT">
<SourceLine classname="java.sql.Statement" start="1092" end="1368" sourcefile="Statement.java" sourcepath="java/sql/Statement.jav
</Type>
<SourceLine classname="testnew.test" start="28" end="28" startBytecode="77" endBytecode="77" sourcefile="test.java" sourcepath="tes
<SourceLine classname="testnew.test" start="28" end="28" startBytecode="77" endBytecode="77" sourcefile="test.java" sourcepath="tes
</BugInstance>
testnew.test.main(String[]) may fail to close Statement [Of Concern(16), Normal confidence]
Writable
Smart Insert
28 : 50

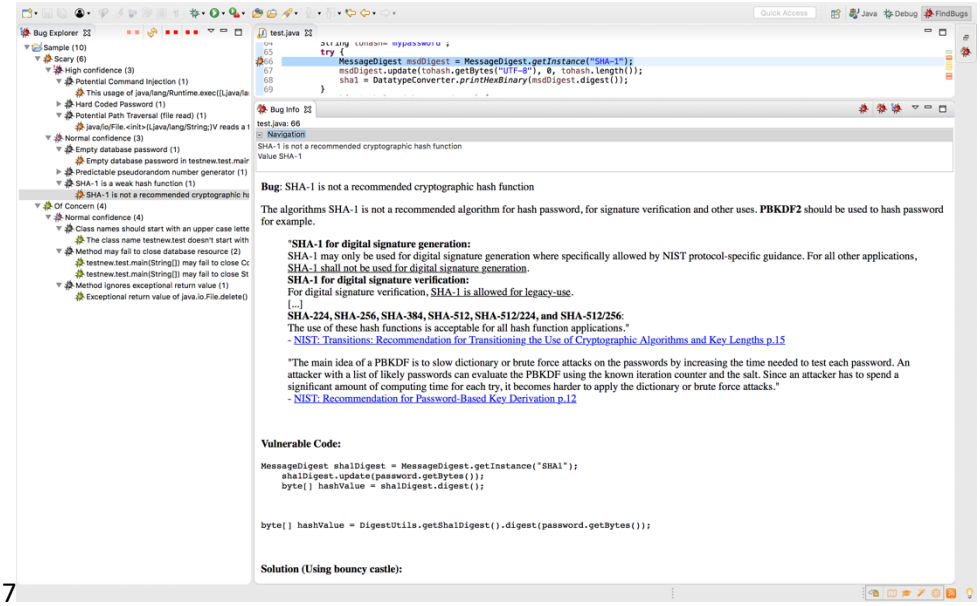
q. Bug 5


Bug Explorer
Sample (10)
Scary (6)
High confidence (3)
Potential Command Injection (1)
This usage of java/lang/Runtime.exec([Ljava/la
Hard Coded Password (1)
Potential Path Traversal (file read) (1)
java/io/File.<init>(Ljava/lang/String;)V reads a
Normal confidence (3)
Empty database password (1)
Empty database password in testnew.test.mair
Predictable pseudorandom number generator (1)
SHA-1 is a weak hash function (1)
Of Concern (4)
Normal confidence (4)
Class names should start with an upper case lette
The class name testnew.test doesn't start with
Method may fail to close database resource (2)
testnew.test.main(String[]) may fail to close Cc
testnew.test.main(String[]) may fail to close St
Method ignores exceptional return value (1)
test.java
while (rs.next())
String users=rs.getString("uname");
System.out.println(users);
}
}
catch(Exception ex)
ex.printStackTrace();
}
Scanner scan=new Scanner(System.in);
System.out.print("Enter file path");
String fpath=scan.nextLine();
if(fpath.startsWith("/archive/"))
File file=new File(fpath);
file.delete();
}
final String lotext=null;
Pattern p = Pattern.compile("ab*");
Matcher m = p.matcher("IPtext");
Bug Info
test.java: 43
Navigation
java/io/File.<init>(Ljava/lang/String;)V reads a file whose location might be specified by user input
Sink method java/io/File.<init>(Ljava/lang/String;)V
Sink parameter 0
Unknown source java/util/Scanner.nextLine()Ljava/lang/String;
Unknown source java/util/Scanner.<init>(Ljava/io/InputStream;)V
At test.java:[line 39]
At test.java:[line 41]
Bug: java/io/File.<init>(Ljava/lang/String;)V reads a file whose location might be specified by user input
A file is opened to read its content. The filename comes from an input parameter. If an unfiltered parameter is passed to this file API, files from an arbitrary filesystem location could be read.
This rule identifies potential path traversal vulnerabilities. In many cases, the constructed file path cannot be controlled by the user. If that is the case, the reported instance is a false positive.
Vulnerable Code:
@GET
@Path("/images/{image}")
@Produces("images/*")
public Response getImage(@javax.ws.rs.PathParam("image") String image) {
File file = new File("resources/images/", image); //Weak point
if (!file.exists()) {
return Response.status(Status.NOT_FOUND).build();
}
java/io/File.<init>(Ljava/lang/String;)V reads a file whose location might be specified by user input [Scary(5), High confidence]
Writable
Smart Insert
43 : 39

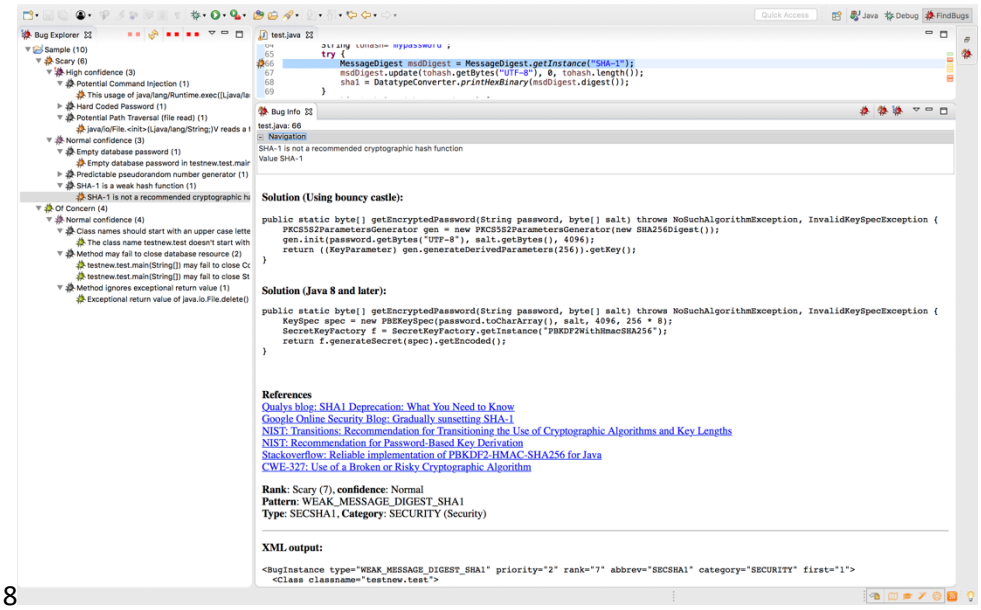
r. Bug 6

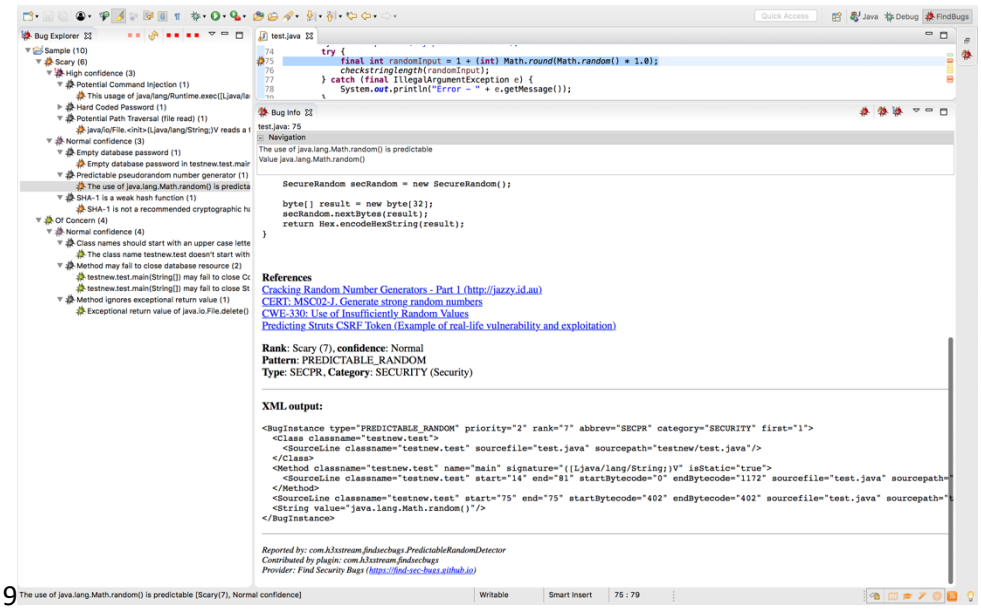

Bug Explorer
Sample (10)
Scary (6)
High confidence (3)
Potential Command Injection (1)
This usage of java/lang/Runtime.exec([Ljava/la
Hard Coded Password (1)
Potential Path Traversal (file read) (1)
java/io/File.<init>(Ljava/lang/String;)V reads a
Normal confidence (3)
Empty database password (1)
Empty database password in testnew.test.mair
Predictable pseudorandom number generator (1)
SHA-1 is a weak hash function (1)
Of Concern (4)
Normal confidence (4)
Class names should start with an upper case lette
The class name testnew.test doesn't start with
Method may fail to close database resource (2)
testnew.test.main(String[]) may fail to close Cc
testnew.test.main(String[]) may fail to close St
Method ignores exceptional return value (1)
Exceptional return value of java.io.File.delete()
test.java
while (rs.next())
String users=rs.getString("uname");
System.out.println(users);
}
}
catch(Exception ex)
ex.printStackTrace();
}
Scanner scan=new Scanner(System.in);
System.out.print("Enter file path");
String fpath=scan.nextLine();
if(fpath.startsWith("/archive/"))
File file=new File(fpath);
Exceptional return value of java.io.File.delete() ignored in testnew.test.main(String[]) [Of Concern(16), Normal confidence]
}
final String lotext=null;
Pattern p = Pattern.compile("ab*");
Matcher m = p.matcher("IPtext");
Bug Info
test.java: 44
Navigation
Exceptional return value of java.io.File.delete() ignored in testnew.test.main(String[])
Called method java.io.File.delete()
Bug: Exceptional return value of java.io.File.delete() ignored in testnew.test.main(String[])
This method returns a value that is not checked. The return value should be checked since it can indicate an unusual or unexpected function execution. For example, the File.delete() method returns false if the file could not be successfully deleted (rather than throwing an Exception). If you don't check the result, you won't notice if the method invocation signals unexpected behavior by returning an atypical return value.
Rank: Of Concern (16), confidence: Normal
Pattern: RV_RETURN_VALUE_IGNORED_BAD_PRACTICE
Type: RV, Category: BAD_PRACTICE (Bad practice)
XML output:
<BugInstance type="RV_RETURN_VALUE_IGNORED_BAD_PRACTICE" priority="2" rank="16" abbrev="RV" category="BAD_PRACTICE" first="1">
<Class classname="testnew.test">
<SourceLine classname="testnew.test" sourcefile="test.java" sourcepath="testnew/test.java"/>
</Class>
<Method classname="testnew.test" name="main" signature="([Ljava/lang/String;)V" isStatic="true">
<SourceLine classname="testnew.test" start="14" end="81" startBytecode="0" endBytecode="1172" sourcefile="test.java" sourcepath="
</Method>
<Method classname="java.io.File" name="delete" signature="()Z" isStatic="false" role="METHOD_CALLED">
<SourceLine classname="java.io.File" start="1034" end="1041" startBytecode="0" endBytecode="91" sourcefile="File.java" sourcepath
</Method>
<SourceLine classname="testnew.test" start="44" end="44" startBytecode="191" endBytecode="191" sourcefile="test.java" sourcepath="t
<SourceLine classname="testnew.test" start="44" end="44" startBytecode="191" endBytecode="191" sourcefile="test.java" sourcepath="t
</BugInstance>
Exceptional return value of java.io.File.delete() ignored in testnew.test.main(String[]) [Of Concern(16), Normal confidence]
Writable
Smart Insert
44 : 27

s. Bug 7


Bug Explorer
Sample (10)
Scary (6)
High confidence (3)
Potential Command Injection (1)
This usage of java/lang/Runtime.exec([Ljava/la
Hard Coded Password (1)
Potential Path Traversal (file read) (1)
java/io/File.<init>(Ljava/lang/String;)V reads a
Normal confidence (3)
Empty database password (1)
Empty database password in testnew.test.mair
Predictable pseudorandom number generator (1)
SHA-1 is a weak hash function (1)
SHA-1 is not a recommended cryptographic hi
Of Concern (4)
Normal confidence (4)
Class names should start with an upper case lette
The class name testnew.test doesn't start with
Method may fail to close database resource (2)
testnew.test.main(String[]) may fail to close Cc
testnew.test.main(String[]) may fail to close St
Method ignores exceptional return value (1)
Exceptional return value of java.io.File.delete()
test.java
try {
MessageDigest msdDigest = MessageDigest.getInstance("SHA-1");
msdDigest.update(tohash.getBytes("UTF-8"), 0, tohash.length());
sha1 = DatatypeConverter.printHexBinary(msdDigest.digest());
}
Bug Info
test.java: 66
Navigation
SHA-1 is not a recommended cryptographic hash function
Value SHA-1
Bug: SHA-1 is not a recommended cryptographic hash function
The algorithms SHA-1 is not a recommended algorithm for hash password, for signature verification and other uses. PBKDF2 should be used to hash password for example.
"SHA-1 for digital signature generation:
SHA-1 may only be used for digital signature generation where specifically allowed by NIST protocol-specific guidance. For all other applications, SHA-1 shall not be used for digital signature generation.
SHA-1 for digital signature verification:
For digital signature verification, SHA-1 is allowed for legacy-use.
[...]
SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256:
The use of these hash functions is acceptable for all hash function applications."
- NIST: Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths p.15
"The main idea of a PBKDF is to slow dictionary or brute force attacks on the passwords by increasing the time needed to test each password. An attacker with a list of likely passwords can evaluate the PBKDF using the known iteration counter and the salt. Since an attacker has to spend a significant amount of computing time for each try, it becomes harder to apply the dictionary or brute force attacks."
- NIST: Recommendation for Password-Based Key Derivation p.12
Vulnerable Code:
MessageDigest shaldigest = MessageDigest.getInstance("SHA1");
shaldigest.update(password.getBytes());
byte[] hashValue = shaldigest.digest();
byte[] hashValue = DigestUtils.getShaldigest().digest(password.getBytes());
Solution (Using bouncy castle):

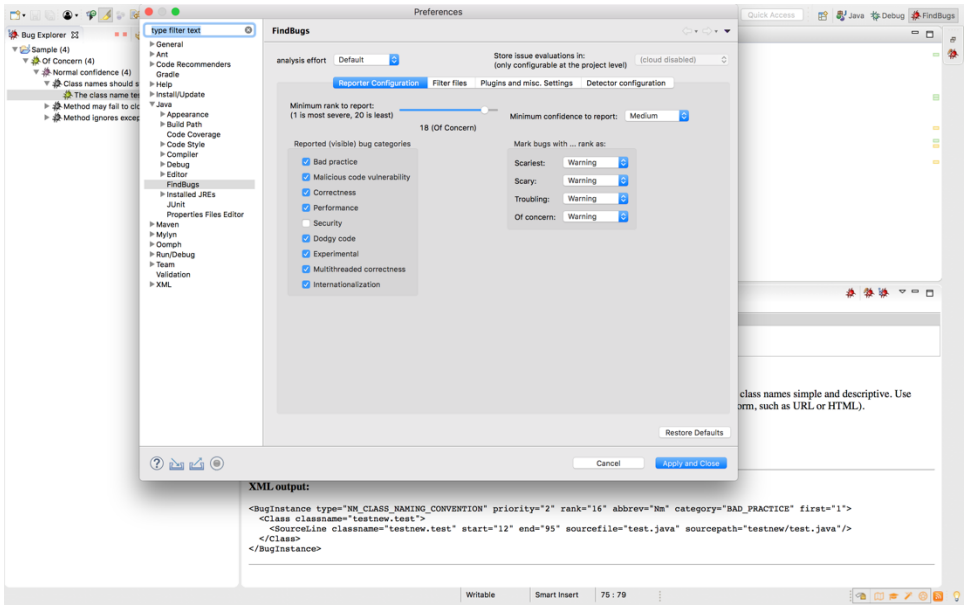
t. Bug 8


Bug Explorer
Sample (10)
Scary (6)
High confidence (3)
Potential Command Injection (1)
This usage of java/lang/Runtime.exec([Ljava/la
Hard Coded Password (1)
Potential Path Traversal (file read) (1)
java/io/File.<init>(Ljava/lang/String;)V reads a
Normal confidence (3)
Empty database password (1)
Empty database password in testnew.test.mair
Predictable pseudorandom number generator (1)
SHA-1 is a weak hash function (1)
SHA-1 is not a recommended cryptographic hi
Of Concern (4)
Normal confidence (4)
Class names should start with an upper case lette
The class name testnew.test doesn't start with
Method may fail to close database resource (2)
testnew.test.main(String[]) may fail to close Cc
testnew.test.main(String[]) may fail to close St
Method ignores exceptional return value (1)
Exceptional return value of java.io.File.delete()
test.java
try {
MessageDigest msdDigest = MessageDigest.getInstance("SHA-1");
msdDigest.update(tohash.getBytes("UTF-8"), 0, tohash.length());
sha1 = DatatypeConverter.printHexBinary(msdDigest.digest());
}
Bug Info
test.java: 66
Navigation
SHA-1 is not a recommended cryptographic hash function
Value SHA-1
Solution (Using bouncy castle):
public static byte[] getEncryptedPassword(String password, byte[] salt) throws NoSuchAlgorithmException, InvalidKeySpecException {
PKCS5S2ParametersGenerator gen = new PKCS5S2ParametersGenerator(new SHA256Digest());
gen.init(password.getBytes("UTF-8"), salt.getBytes(), 4096);
return ((KeyParameter) gen.generateDerivedParameters(256)).getKey();
}
Solution (Java 8 and later):
public static byte[] getEncryptedPassword(String password, byte[] salt) throws NoSuchAlgorithmException, InvalidKeySpecException {
KeySpec spec = new PBEKeySpec(password.toCharArray(), salt, 4096, 256 * 8);
SecretKeyFactory f = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
return f.generateSecret(spec).getEncoded();
}
References
Qualys blog: SHA1 Deprecation: What You Need to Know
Google Online Security Blog: Gradually sunsetting SHA-1
NIST: Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths
NIST: Recommendation for Password-Based Key Derivation
Stackoverflow: Reliable implementation of PBKDF2-HMAC-SHA256 for Java
CWE-327: Use of a Broken or Risky Cryptographic Algorithm
Rank: Scary (7), confidence: Normal
Pattern: WEAK_MESSAGE_DIGEST_SHA1
Type: SECSHA1, Category: SECURITY (Security)
XML output:
<BugInstance type="WEAK_MESSAGE_DIGEST_SHA1" priority="2" rank="7" abbrev="SECSHA1" category="SECURITY" first="1">
<Class classname="testnew.test">
Quick Access
Java
Debug
FindBugs

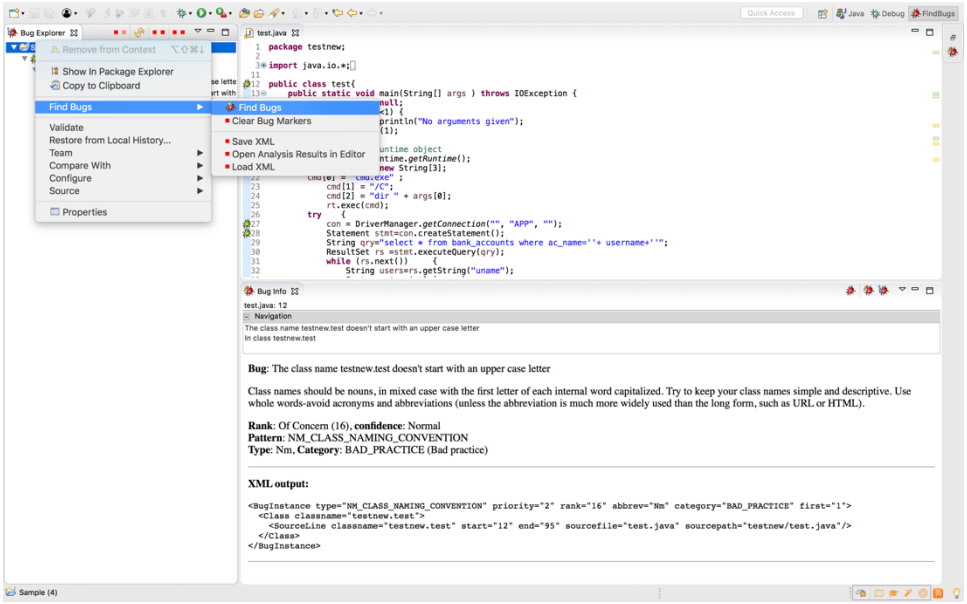


u. Bug 9

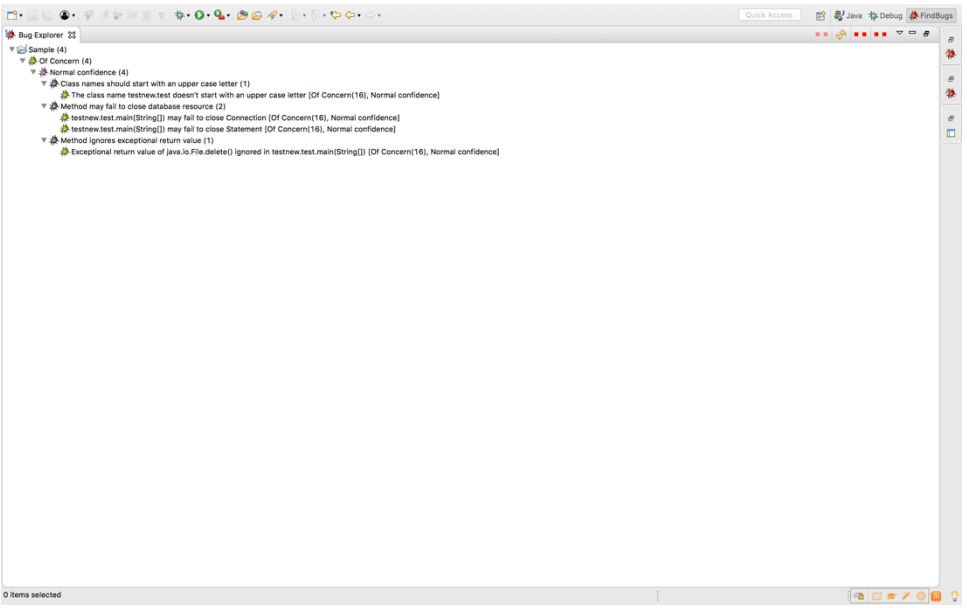
a. When the Security Mode is not enabled, ie only findbugs Able to find 4 bugs(Of Concern)



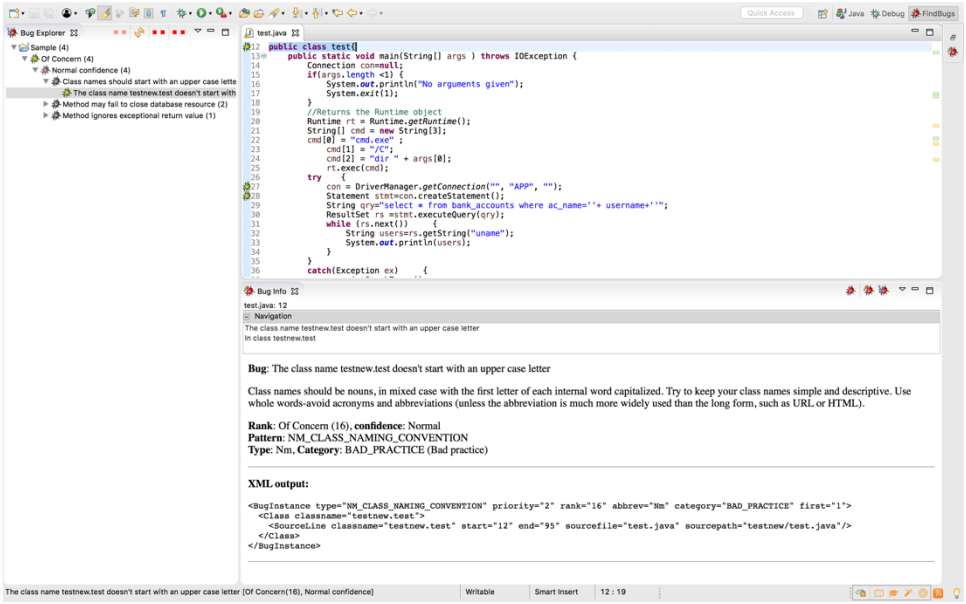
b.


Bug Explorer
Remove from Context
Show In Package Explorer
Copy to Clipboard
Find Bugs
Validate
Restore from Local History...
Team
Compare With
Configure
Source
Properties
Find Bugs
Clear Bug Markers
Save XML
Open Analysis Results in Editor
Load XML
test.java
Bug Info
test.java: 12
Navigation
The class name testnew.test doesn't start with an upper case letter
In class testnew.test
Bug: The class name testnew.test doesn't start with an upper case letter
Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).
Rank: Of Concern (16), confidence: Normal
Pattern: NM_CLASS_NAMING_CONVENTION
Type: Nm, Category: BAD_PRACTICE (Bad practice)
XML output:
<BugInstance type="NM_CLASS_NAMING_CONVENTION" priority="2" rank="16" abbrev="Nm" category="BAD_PRACTICE" first="1">
<Class classname="testnew.test">
<SourceLine classname="testnew.test" start="12" end="95" sourcefile="test.java" sourcepath="testnew/test.java"/>
</Class>
</BugInstance>
Sample (4)

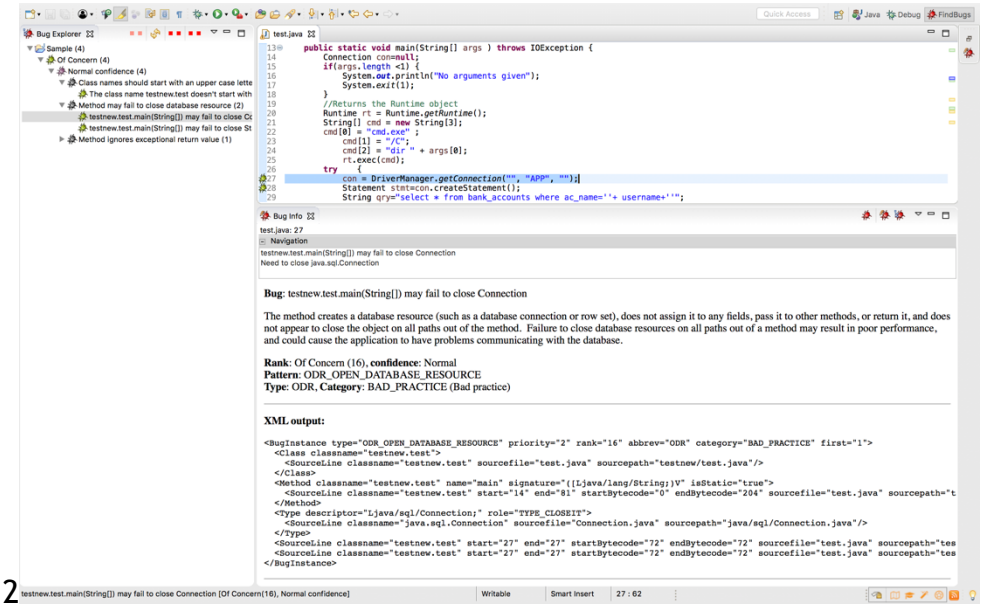
c.


Bug Explorer
Sample (4)
Of Concern (4)
Normal confidence (4)
Class names should start with an upper case letter (1)
The class name testnew.test doesn't start with an upper case letter [Of Concern(16), Normal confidence]
Method may fail to close database resource (2)
testnew.test.main(String[]) may fail to close Connection [Of Concern(16), Normal confidence]
testnew.test.main(String[]) may fail to close Statement [Of Concern(16), Normal confidence]
Method ignores exceptional return value (1)
Exceptional return value of java.io.File.delete() ignored in testnew.test.main(String[]) [Of Concern(16), Normal confidence]
0 items selected

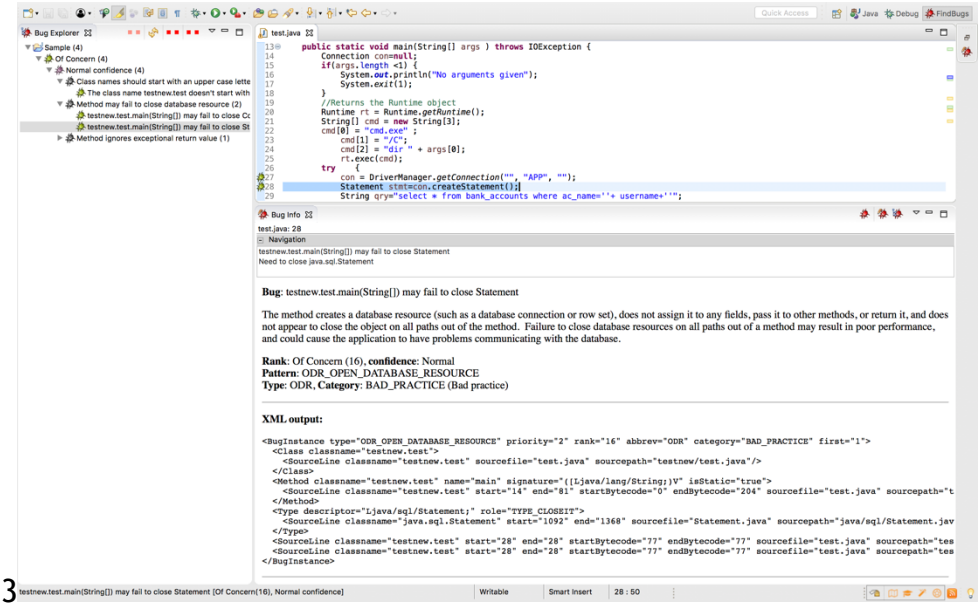
d. Bug 1


Bug Explorer
Sample (4)
Of Concern (4)
Normal confidence (4)
Class names should start with an upper case lette
The class name testnew.test doesn't start with
Method may fail to close database resource (2)
Method ignores exceptional return value (1)
test.java
public class test{
public static void main(String[] args) throws IOException {
Connection con=null;
if(args.length <1) {
System.out.println("No arguments given");
System.exit(1);
}
//Returns the Runtime object
Runtime rt = Runtime.getRuntime();
String[] cmd = new String[3];
cmd[0] = "cmd.exe" ;
cmd[1] = "/C";
cmd[2] = "dir " + args[0];
rt.exec(cmd);
try {
con = DriverManager.getConnection("", "APP", "");
Statement stmt=con.createStatement();
String qry="select * from bank_accounts where ac_name='"+ username+"'";
ResultSet rs =stmt.executeQuery(qry);
while (rs.next()) {
String users=rs.getString("uname");
System.out.println(users);
}
}
catch(Exception ex) {
Bug Info
test.java: 12
Navigation
The class name testnew.test doesn't start with an upper case letter
In class testnew.test
Bug: The class name testnew.test doesn't start with an upper case letter
Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).
Rank: Of Concern (16), confidence: Normal
Pattern: NM_CLASS_NAMING_CONVENTION
Type: Nm, Category: BAD_PRACTICE (Bad practice)
XML output:
<BugInstance type="NM_CLASS_NAMING_CONVENTION" priority="2" rank="16" abbrev="Nm" category="BAD_PRACTICE" first="1">
<Class classname="testnew.test">
<SourceLine classname="testnew.test" start="12" end="95" sourcefile="test.java" sourcepath="testnew/test.java"/>
</Class>
</BugInstance>
The class name testnew.test doesn't start with an upper case letter [Of Concern(16), Normal confidence]
Writable
Smart Insert
12 : 19

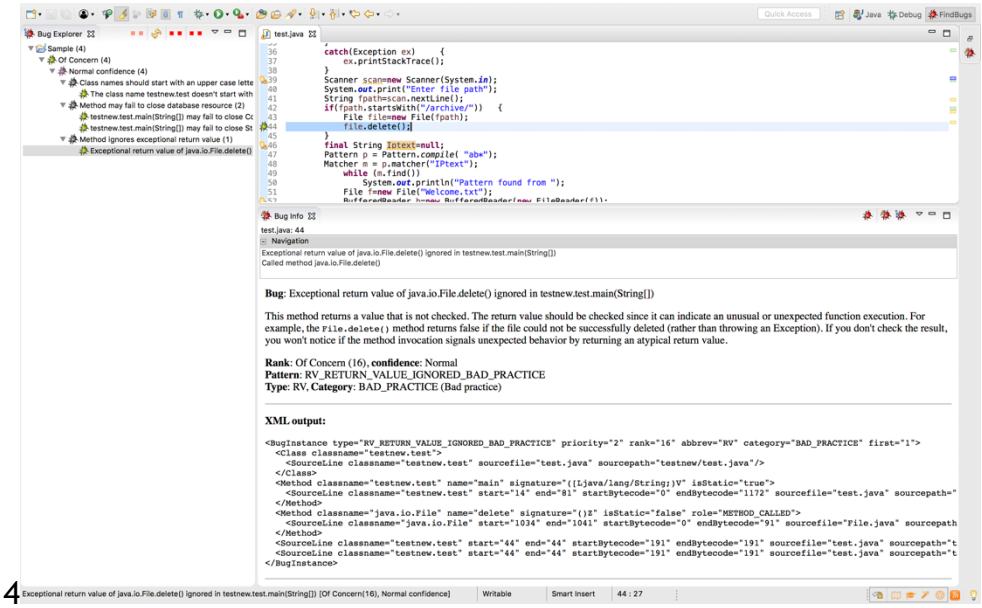
e. Bug 2


Bug Explorer
Sample (4)
Of Concern (4)
Normal confidence (4)
Class names should start with an upper case lette
The class name testnew.test doesn't start with
Method may fail to close database resource (2)
testnew.test.main(String[]) may fail to close Cc
testnew.test.main(String[]) may fail to close St
Method ignores exceptional return value (1)
test.java
public static void main(String[] args) throws IOException {
Connection con=null;
if(args.length <1) {
System.out.println("No arguments given");
System.exit(1);
}
//Returns the Runtime object
Runtime rt = Runtime.getRuntime();
String[] cmd = new String[3];
cmd[0] = "cmd.exe" ;
cmd[1] = "/C";
cmd[2] = "dir " + args[0];
rt.exec(cmd);
try {
con = DriverManager.getConnection("", "APP", "");
Statement stmt=con.createStatement();
String qry="select * from bank_accounts where ac_name='"+ username+"'";
Bug Info
test.java: 27
Navigation
testnew.test.main(String[]) may fail to close Connection
Need to close java.sql.Connection
Bug: testnew.test.main(String[]) may fail to close Connection
The method creates a database resource (such as a database connection or row set), does not assign it to any fields, pass it to other methods, or return it, and does not appear to close the object on all paths out of the method. Failure to close database resources on all paths out of a method may result in poor performance, and could cause the application to have problems communicating with the database.
Rank: Of Concern (16), confidence: Normal
Pattern: ODR_OPEN_DATABASE_RESOURCE
Type: ODR, Category: BAD_PRACTICE (Bad practice)
XML output:
<BugInstance type="ODR_OPEN_DATABASE_RESOURCE" priority="2" rank="16" abbrev="ODR" category="BAD_PRACTICE" first="1">
<Class classname="testnew.test">
<SourceLine classname="testnew.test" sourcefile="test.java" sourcepath="testnew/test.java"/>
</Class>
<Method classname="testnew.test" name="main" signature="([Ljava/lang/String;)V" isStatic="true">
<SourceLine classname="testnew.test" start="14" end="81" startBytecode="0" endBytecode="204" sourcefile="test.java" sourcepath="t
</Method>
<Type descriptor="Ljava/sql/Connection;" role="TYPE_CLOSEIT">
<SourceLine classname="java.sql.Connection" sourcefile="Connection.java" sourcepath="java/sql/Connection.java"/>
</Type>
<SourceLine classname="testnew.test" start="27" end="27" startBytecode="72" endBytecode="72" sourcefile="test.java" sourcepath="tes
<SourceLine classname="testnew.test" start="27" end="27" startBytecode="72" endBytecode="72" sourcefile="test.java" sourcepath="tes
</BugInstance>
testnew.test.main(String[]) may fail to close Connection [Of Concern(16), Normal confidence]
Writable
Smart Insert
27 : 62

f. Bug 3

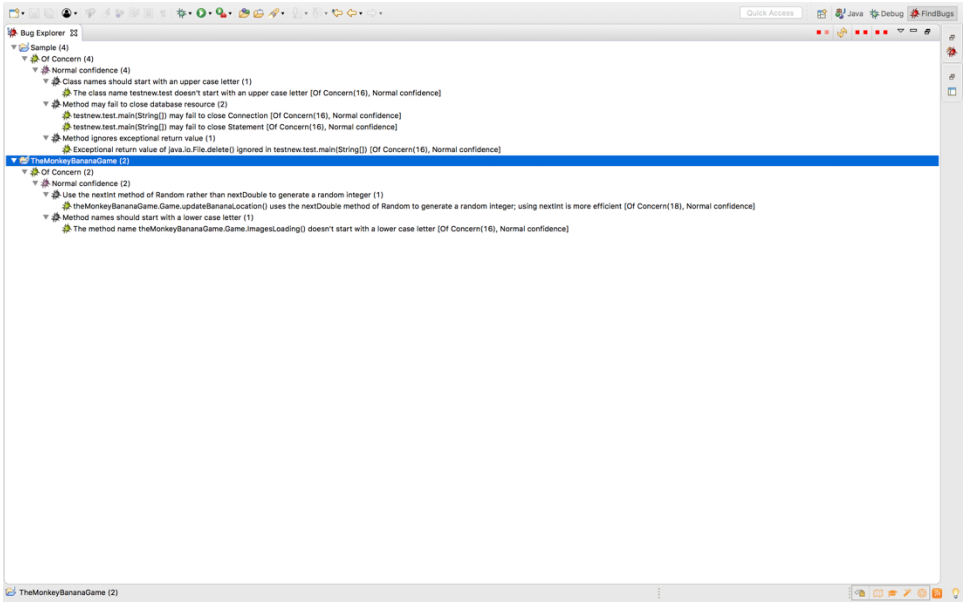


g. Bug 4

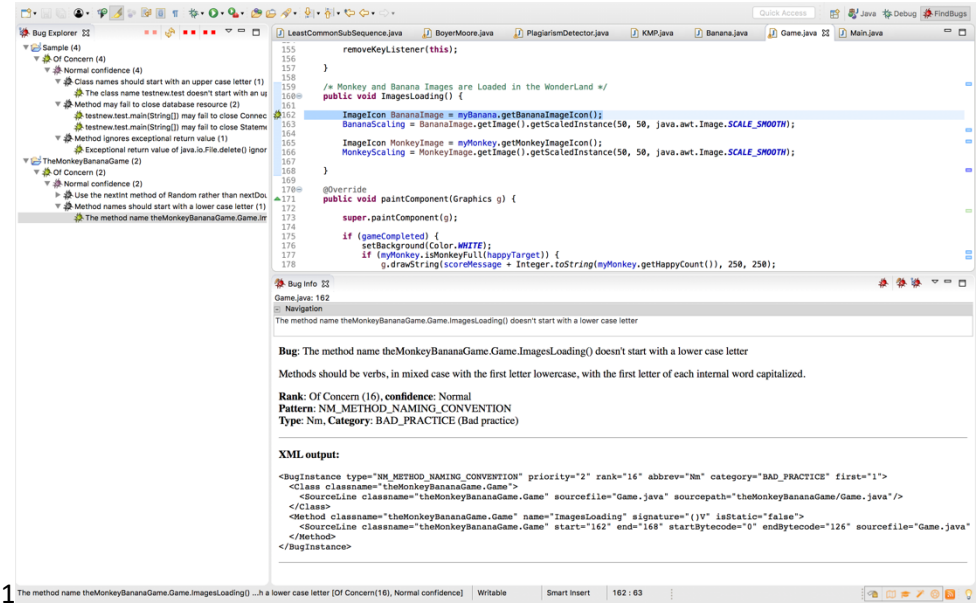


My code (TheMonkeyBananaGame):
Reference: https://github.com/svrswetha/Software-Design-Patterns
I have used findbugs & Security Bugs is to limit the scan to Securityonly bug detectors on My Code(TheMonkeyBananaGame). With Security mode disabled(Findbugs), it gave 2 bugs

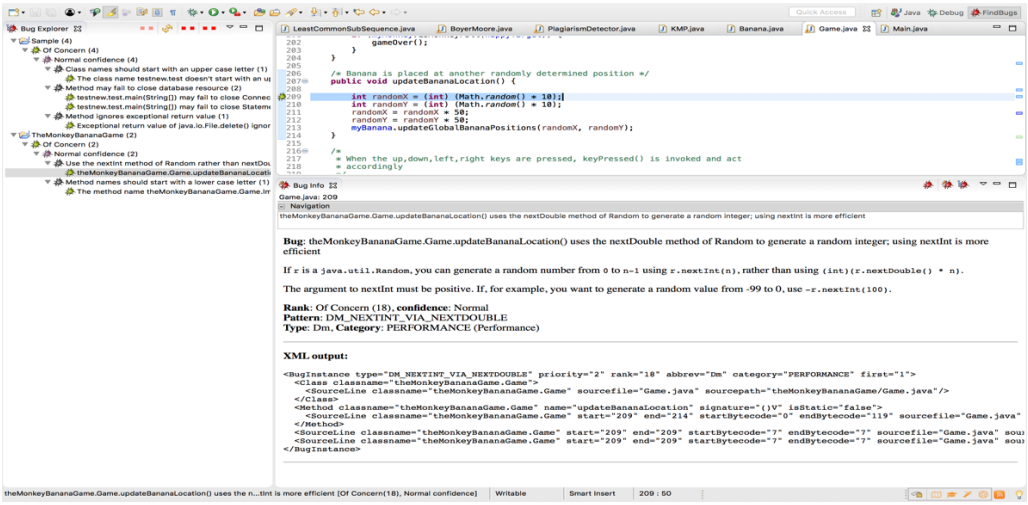
a.



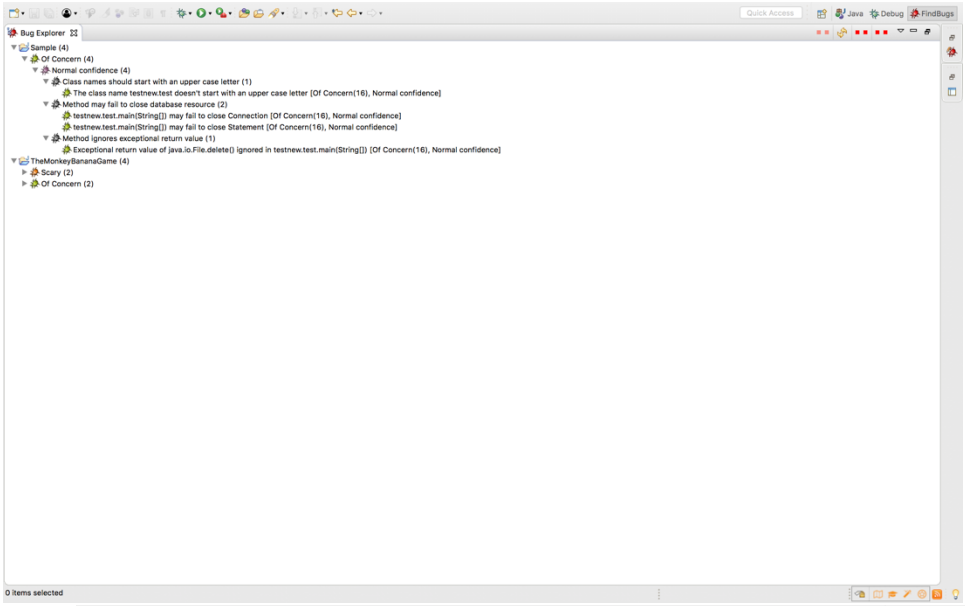
b. Bug 1



c. Bug 2

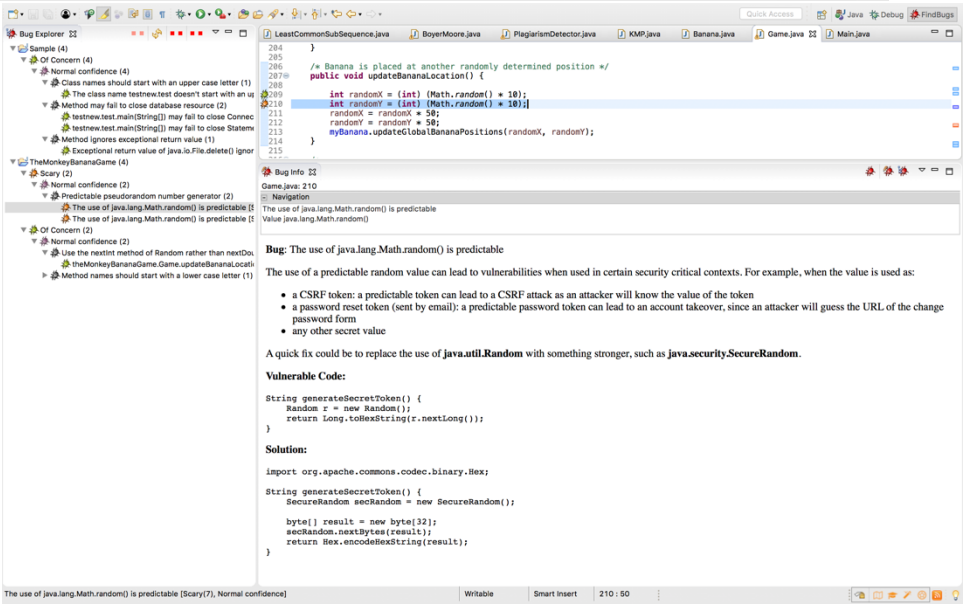


d. When the Security mode in enabled (FindSecurityBugs), It gave 4 bugs

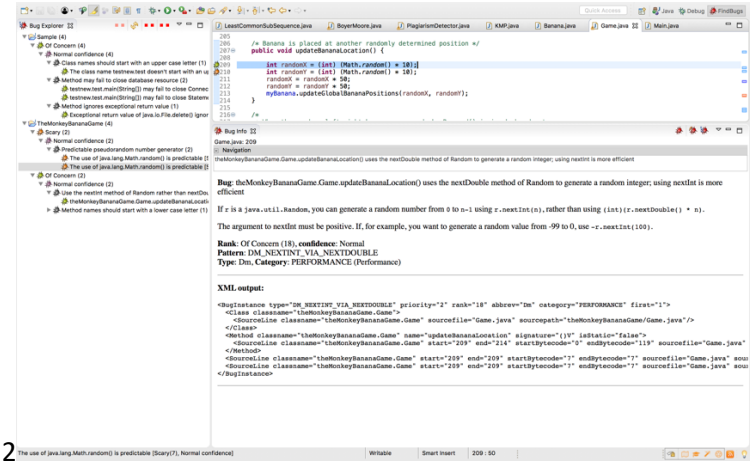


e.

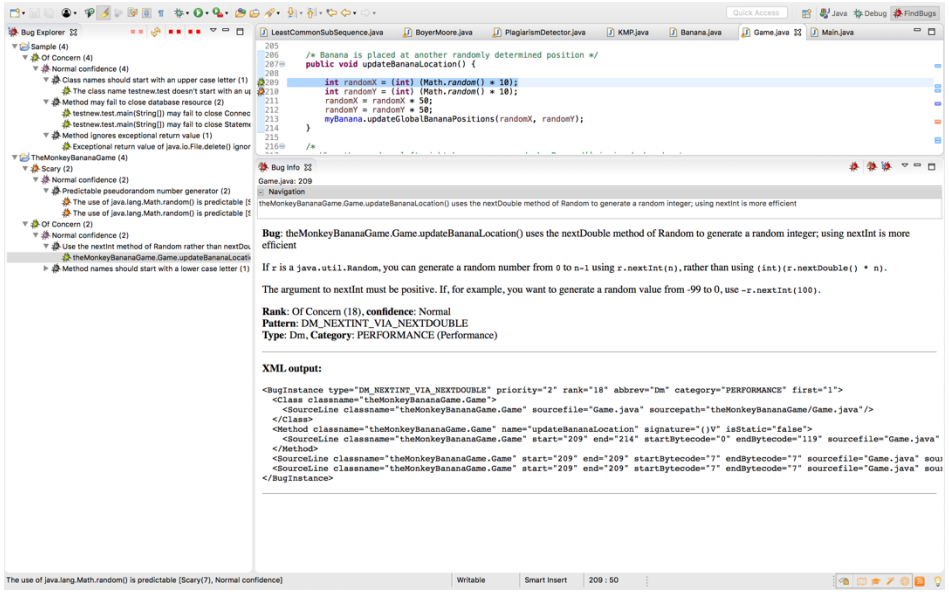
f. Bug 1



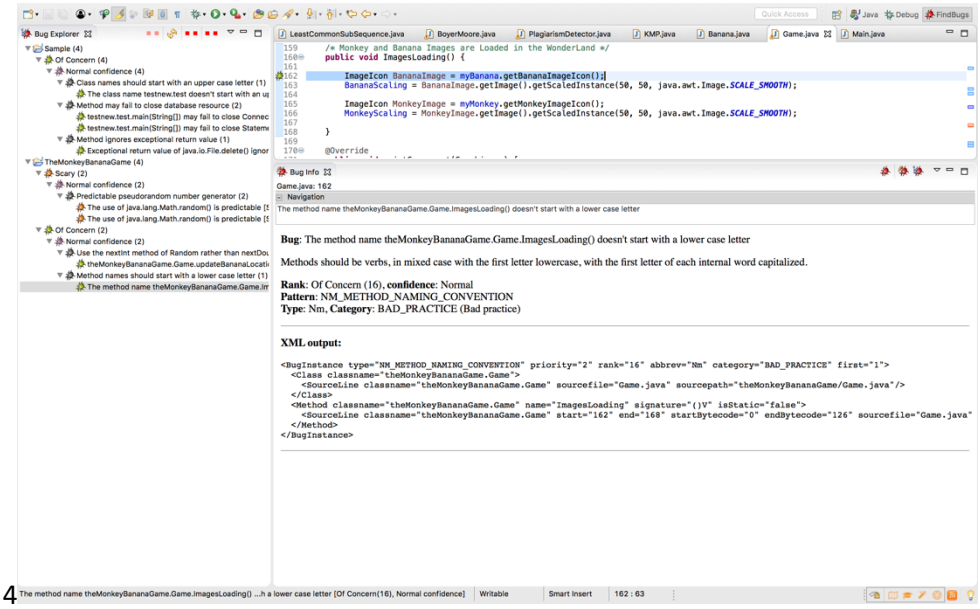
g. Bug 2



h. Bug 3



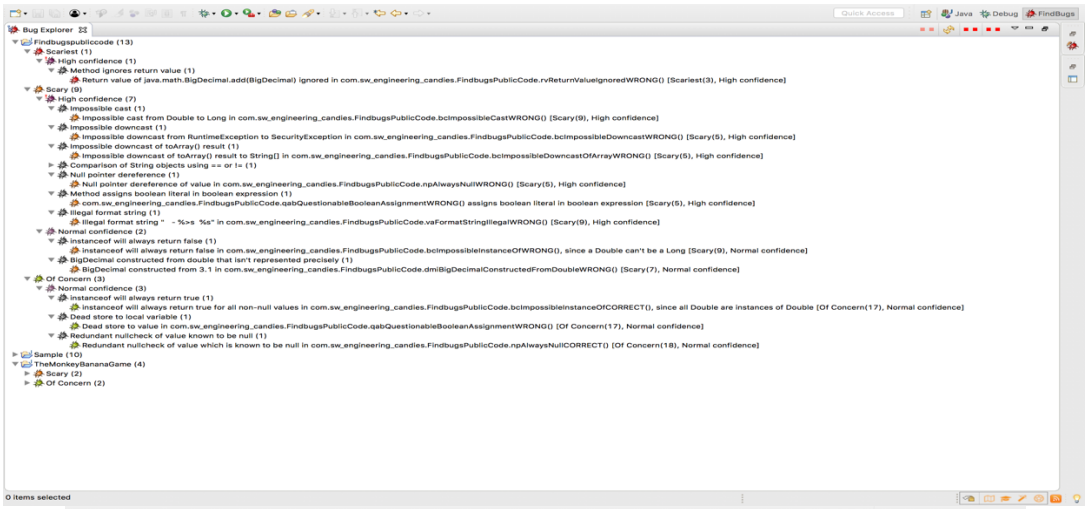
i. Bug 4

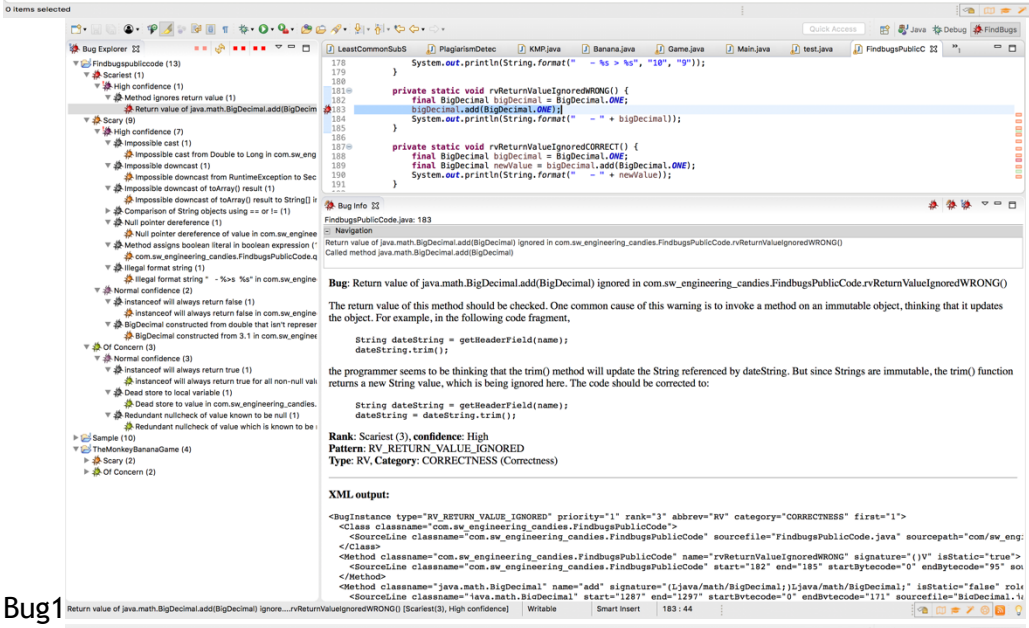


Public Code:

With the findbugs along with security category enabled or disabled, the below code is giving 13 bugs.

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





Bug1

Bug: Return value of java.math.BigDecimal.add(BigDecimal) ignored in com.sw_engineering_candies.FindbugsPublicCode.rvReturnValueIgnoredWRONG()

The return value of this method should be checked. One common cause of this warning is to invoke a method on an immutable object, thinking that it updates the object. For example, in the following code fragment,

```
String dateString = getHeaderField(name);
dateString.trim();
```

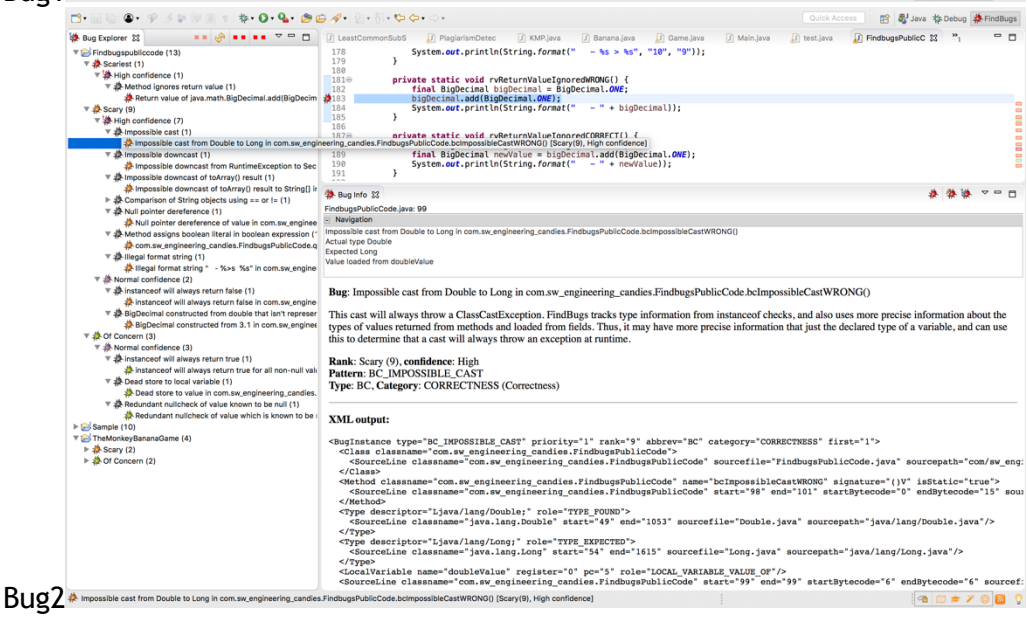
the programmer seems to be thinking that the trim() method will update the String referenced by dateString. But since Strings are immutable, the trim() function returns a new String value, which is being ignored here. The code should be corrected to:

```
String dateString = getHeaderField(name);
dateString = dateString.trim();
```

Rank: Scariest (3), **confidence**: High
Pattern: RV_RETURN_VALUE_IGNORED
Type: RV, **Category**: CORRECTNESS (Correctness)

XML output:

```
<BugInstance type="RV_RETURN_VALUE_IGNORED" priority="1" rank="3" abbrev="RV" 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="rvReturnValueIgnoredWRONG" signature="()V" isStatic="true">
    <SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="182" end="185" startBytecode="0" endBytecode="95" so
  </Method>
  <Method classname="java.math.BigDecimal" name="add" signature="(Ljava/math/BigDecimal;)Ljava/math/BigDecimal;" isStatic="false" role
    <SourceLine classname="java.math.BigDecimal" start="1287" end="1297" startBytecode="0" endBytecode="171" sourcefile="BigDecimal.
```



Bug2

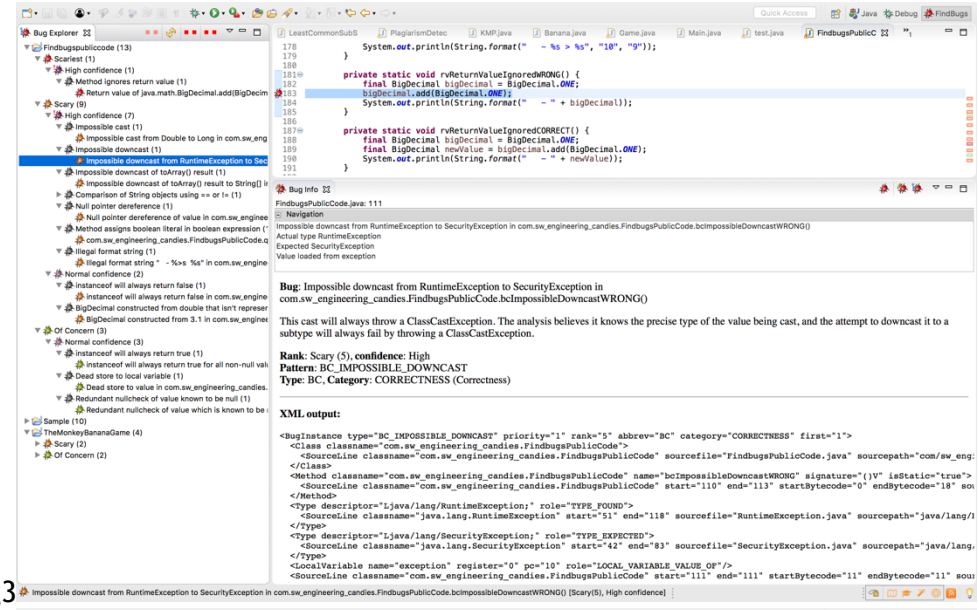
Bug: Impossible cast from Double to Long in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleCastWRONG()

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
```



Bug3

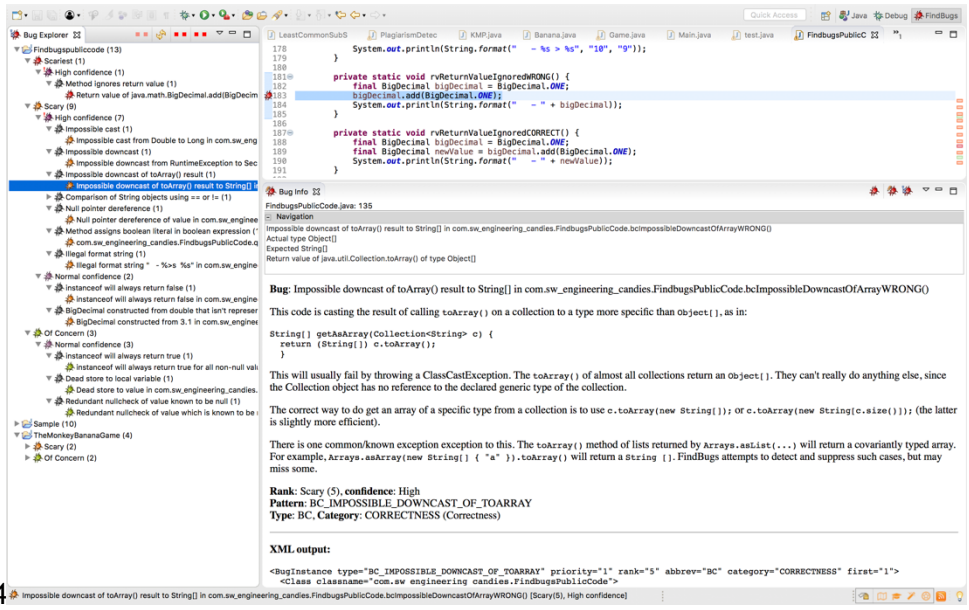
Bug: Impossible downcast from RuntimeException to SecurityException in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleDowncastWRONG()

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" so
  </Method>
  <Type descriptor="Ljava/lang/RuntimeException;" role="TYPE_FOUND">
    <SourceLine classname="java.lang.RuntimeException" start="51" end="118" sourcefile="RuntimeException.java" sourcepath="java/lang/
  </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
```



Bug4

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.

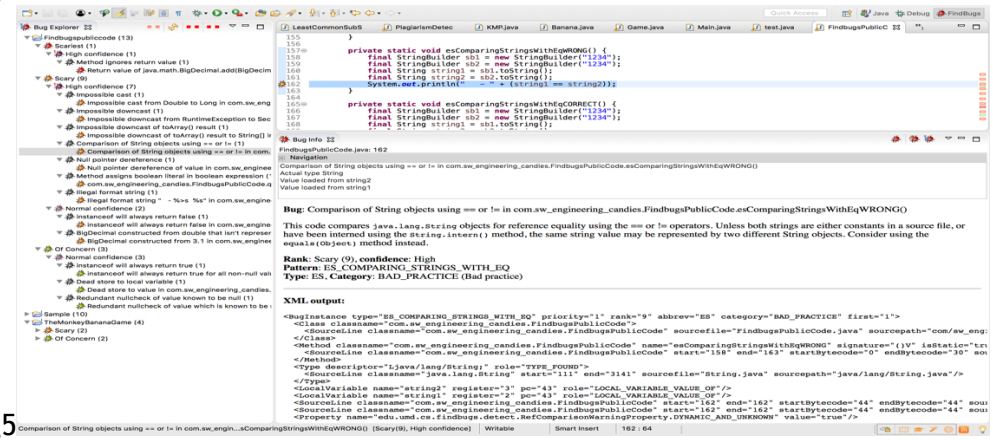
The correct way to do get an array of a specific type from a collection is to use `c.toArray(new String[])`; or `c.toArray(new String[c.size()])`; (the latter is slightly more efficient).

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">
```



Bug5

Bug: Comparison of String objects using == or != in com.sw_engineering_candies.FindbugsPublicCode.esComparingStringsWithEqWRONG()

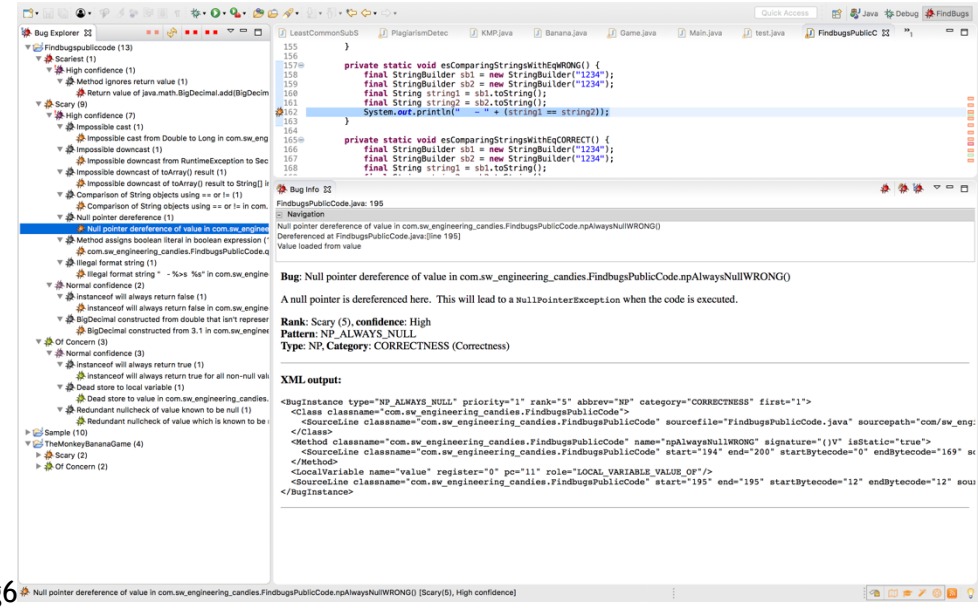
This code compares `java.lang.String` objects for reference equality using the == or != operators. Unless both strings are either constants in a source file, or have been interned using the `String.intern()` method, the same string value may be represented by two different String objects. Consider using 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="tr
    <SourceLine classname="com.sw_engineering_candies.FindbugsPublicCode" start="158" end="163" startBytecode="0" endBytecode="30" so
  </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"/>
```

Bug6



Bug: Null pointer dereference of value in com.sw_engineering_candies.FindbugsPublicCode.npAlwaysNullWRONG()

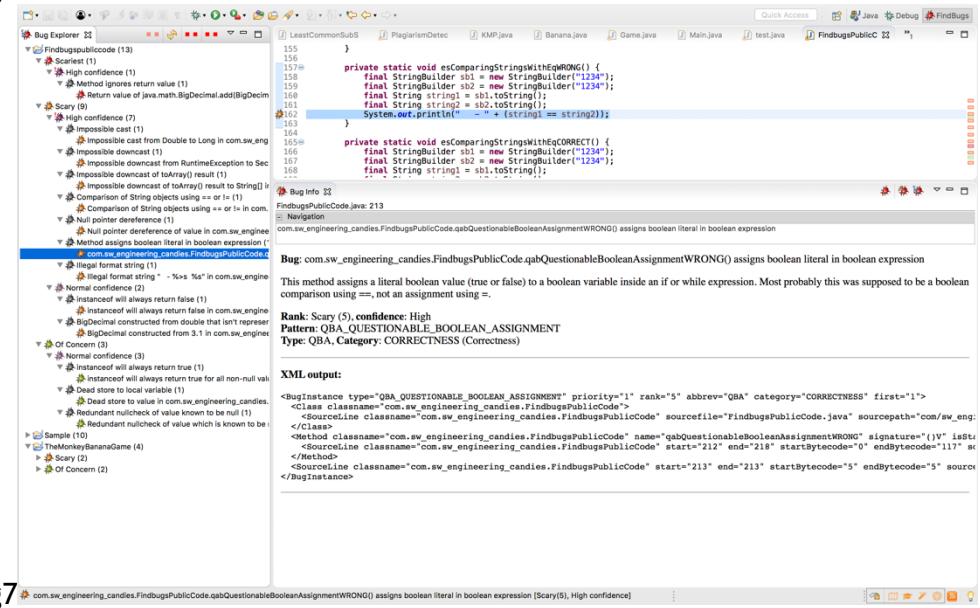
A null pointer is dereferenced here. This will lead to a `NullPointerException` when the code is executed.

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>
```

Bug7



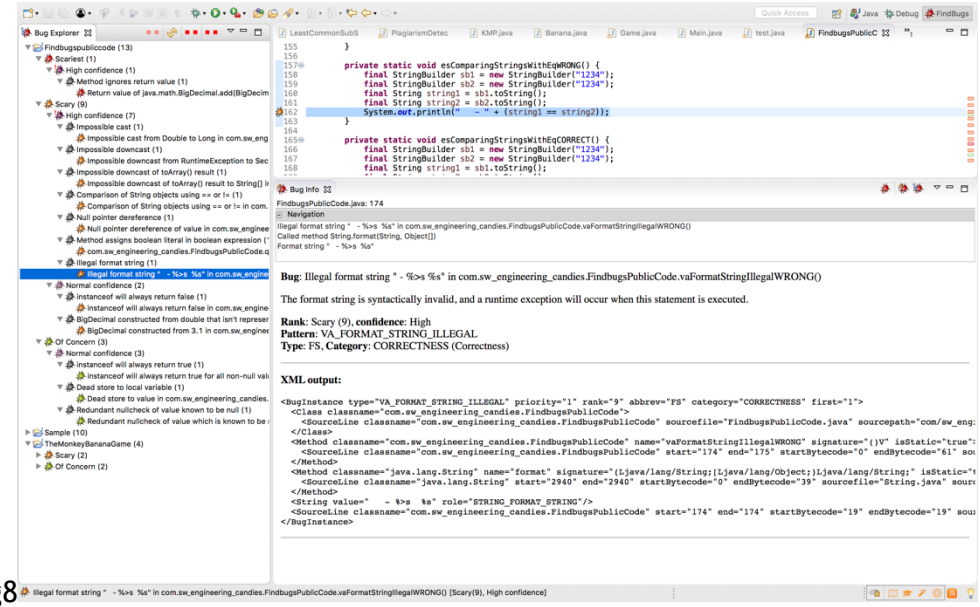
Bug: com.sw_engineering_candies.FindbugsPublicCode.qabQuestionableBooleanAssignmentWRONG() assigns boolean literal in boolean expression

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" isSta
    <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" source
</BugInstance>
```



Bug8

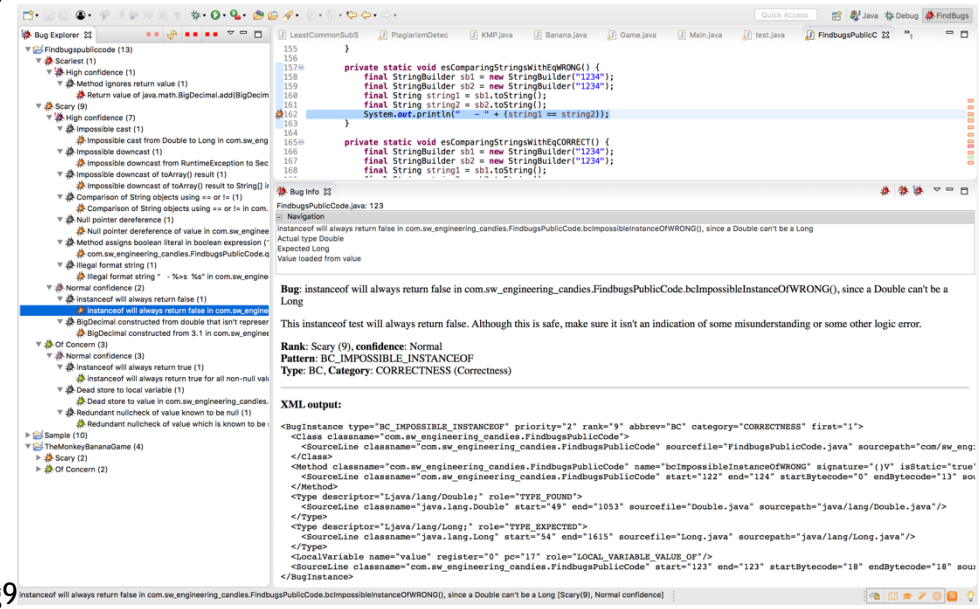
Bug: Illegal format string " - %>s %s" in com.sw_engineering_candies.FindbugsPublicCode.vaFormatStringIllegalWRONG()

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>
```



Bug9

Bug: instanceof will always return false in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleInstanceOfWRONG(), since a Double can't be a Long

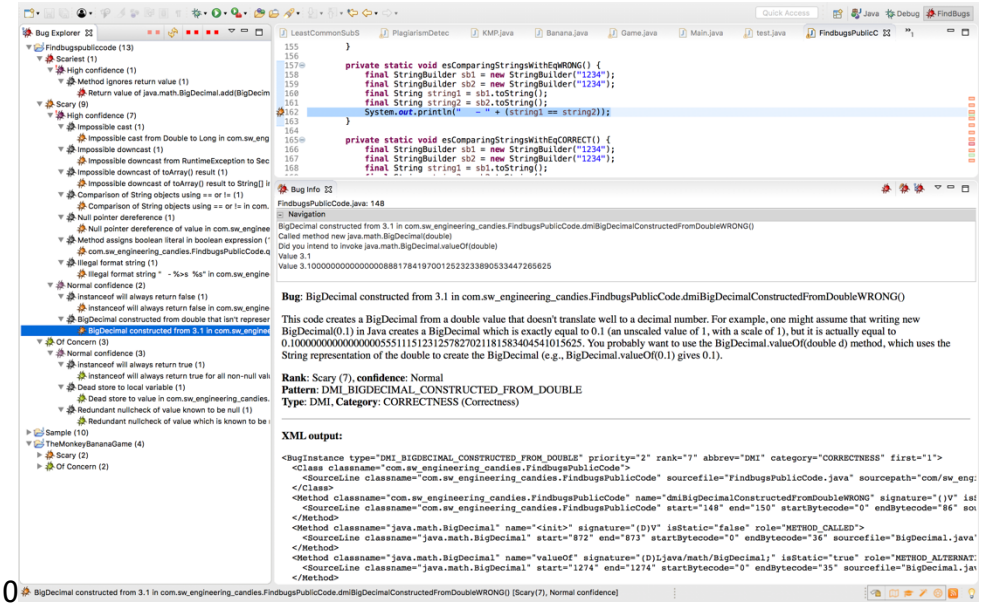
This instanceof test will always return false. Although this is safe, make sure it isn't an indication of some misunderstanding or some other logic error.

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>
```

Bug10



Bug: BigDecimal constructed from 3.1 in com.sw_engineering_candies.FindbugsPublicCode.dmiBigDecimalConstructedFromDoubleWRONG()

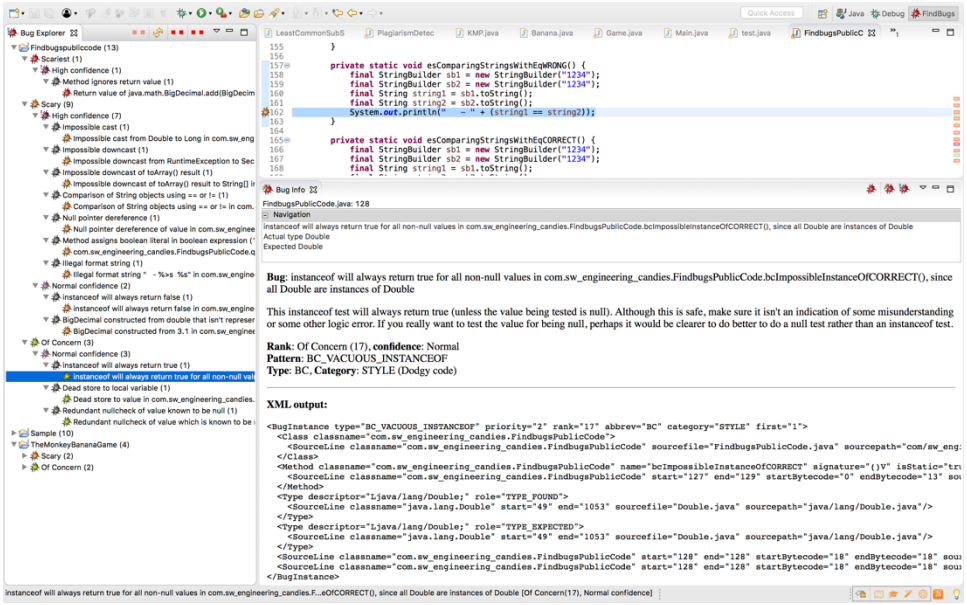
This code creates a BigDecimal from a double value that doesn't translate well to a decimal number. For example, one might assume that writing new BigDecimal(0.1) in Java creates a BigDecimal which is exactly equal to 0.1 (an unscaled value of 1, with a scale of 1), but it is actually equal to 0.1000000000000000055511151231257827021181583404541015625. 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_engi
  </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>
```

Bug11



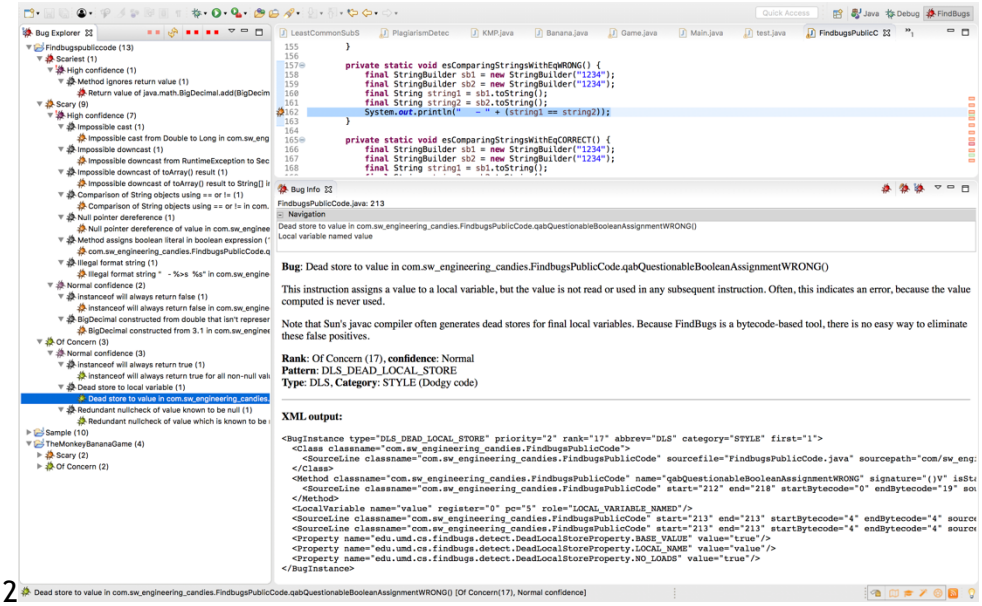
Bug: instanceof will always return true for all non-null values in com.sw_engineering_candies.FindbugsPublicCode.bcImpossibleInstanceOfCORRECT(), since all Double are instances of Double

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.

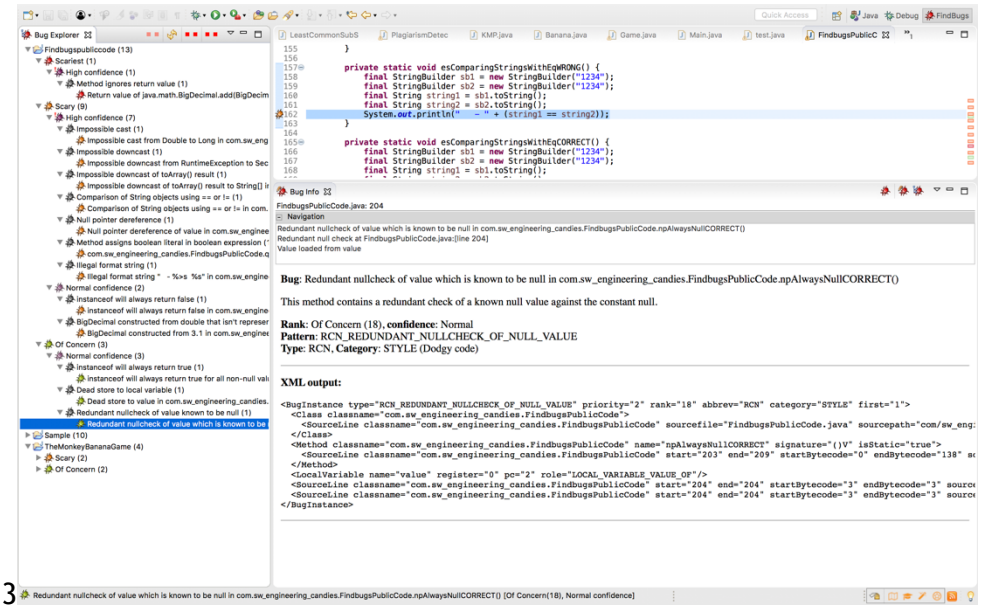
Rank: Of Concern (17), **confidence**: Normal
Pattern: BC_VACUOUS_INSTANCEOF
Type: BC, **Category**: STYLE (Dodgy code)

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_engi
  </Class>
  <Method classname="com.sw_engineering_candies.FindbugsPublicCode" name="bcImpossibleInstanceOfCORRECT" signature="()V" isStatic="tr
    <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>
```



Bug12



Bug13

1. What does it take as input?
It takes Java Byte code as an Input.
2. What “bugs” does it find? (And not find?)
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, ie when the security category is enabled it is FindSecurityBugs**]:** Security
3. How?
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.