Spring 2020

CSE-5382-001 – Secure Programming

Homework Assignment 10 – Understanding and Using Static Code Analysis Tools

Name	UTA ID
Goutami Padmanabhan	1001669338

Part 1 :

The purpose of this task is to analyze a web server with analysis tools like SpotBugs and SonarLint.

```
45      public void processRequest(Socket s) throws Exception {
46      /* used to read data from the client */
47      BufferedReader br =
48          new BufferedReader (
49                  new InputStreamReader (s.getInputStream()));
50
```

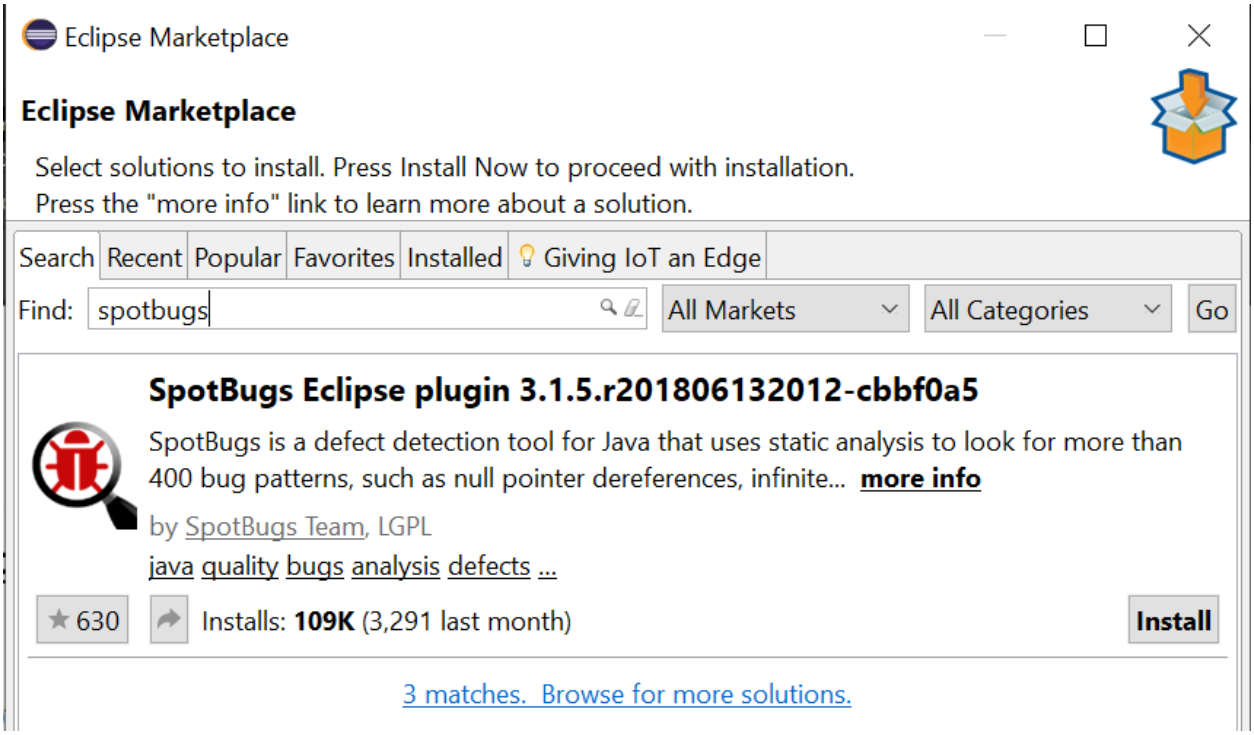
```
101     /* try to open file specified by pathname */
102     try {
103         fr = new FileReader (pathname);
104         c = fr.read();
105     }
106     catch (Exception e) {
107         /* if the file is not found,return the
108            appropriate HTTP response code  */
109         osw.write ("HTTP/1.0 404 Not Found\n\n");
110         return;
111     }
112
```

Manual Analysis: Initially, the code is reviewed to see if we can identify any problems. By just looking at the code for SimpleWebServer.java we can see that the BuffreReader object br creates an InputStream and used but not closed. Always close the streams. It is a good habit which helps you to avoid some odd behavior. Calling close() method also calls flush() so you don't have do this manually. FileReader object has been created, but it is also not closed. By not closing these two objects there may be performance issues, garbage value collection and may produce wrong undesired results as output.

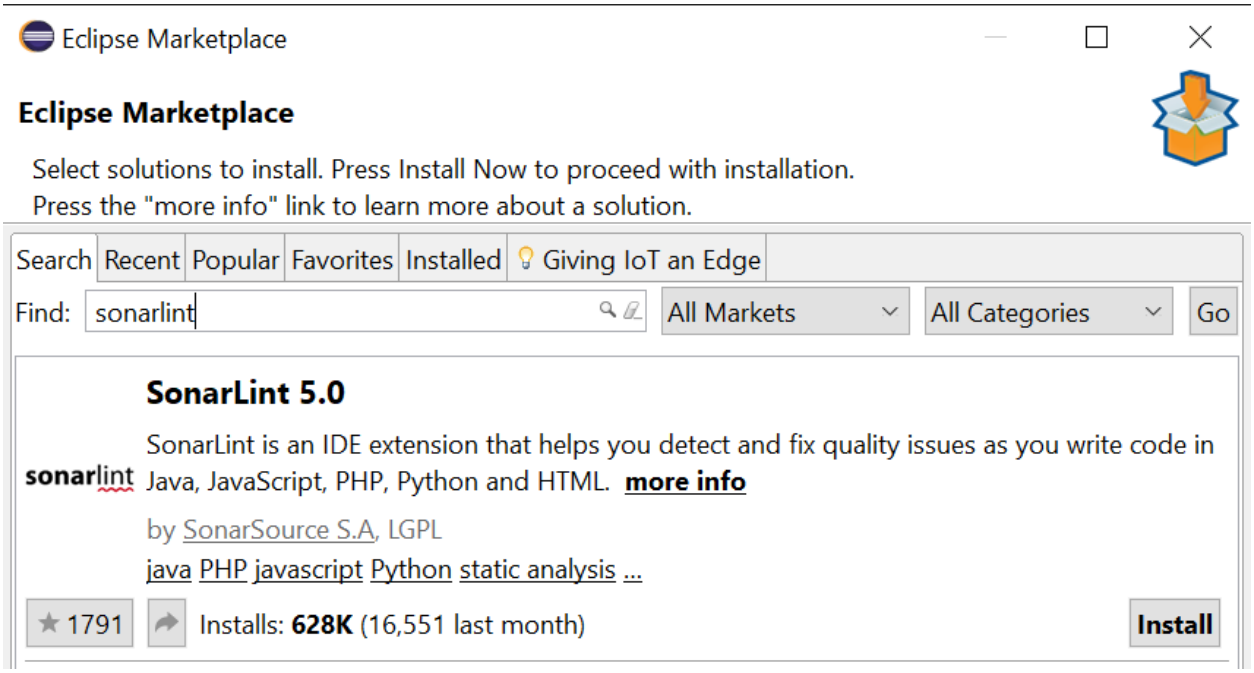
Tool Choices/Versions:

I have chosen two tools namely SpotBugs and SonarLint.

a) SpotBugs Eclipse plugin 3.1.5.r201806132012-cbbf0a5



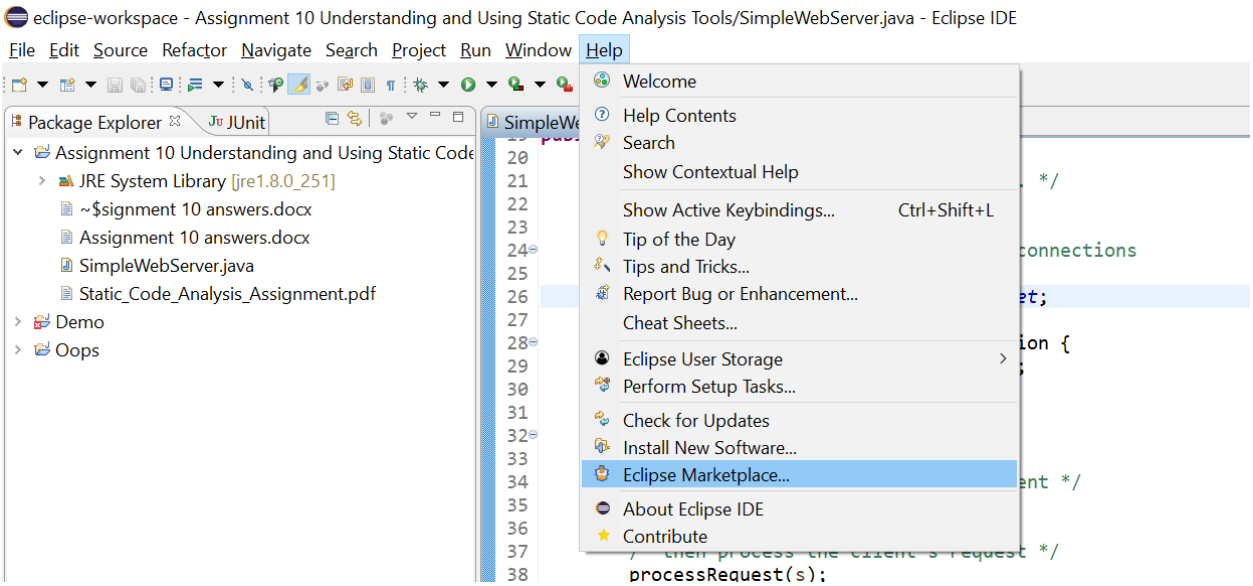
b) SonarLint 5.0

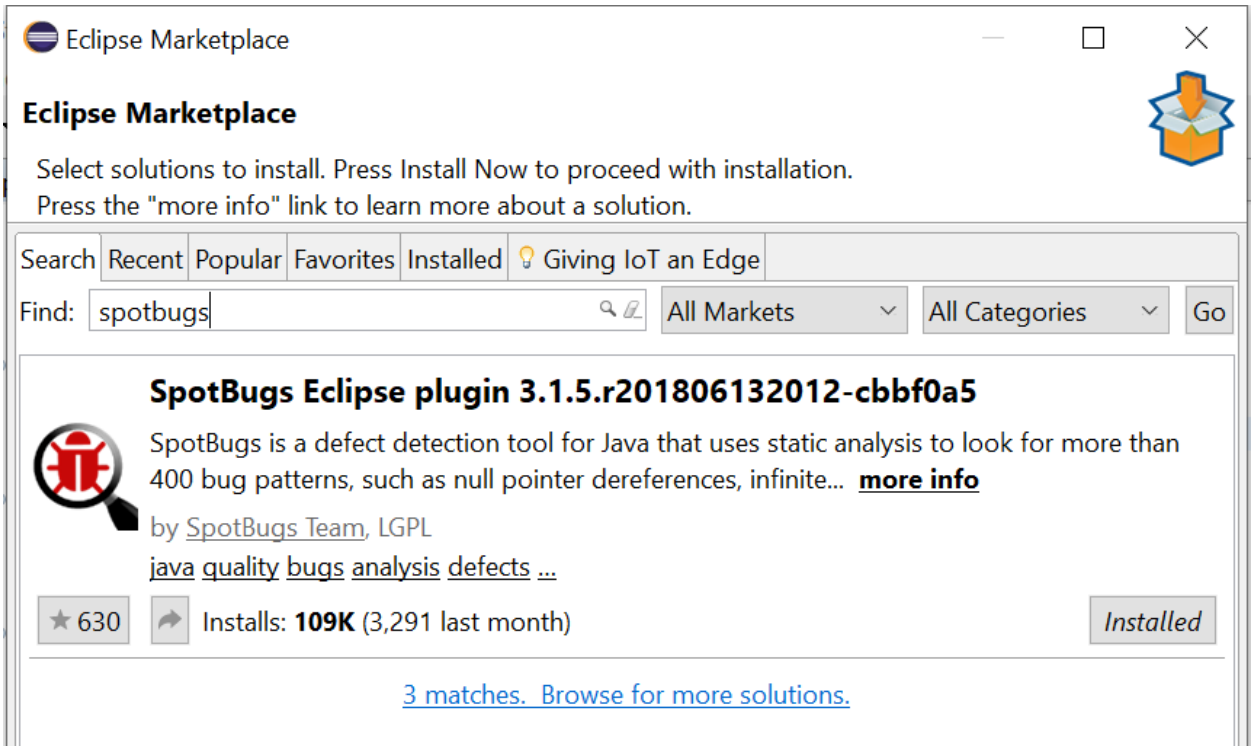


Tool Invocation Process:

a) SpotBugs: In Eclipse IDE, navigate to Help and find Elipse Marketplace. Then search for SpotBugs and install it.

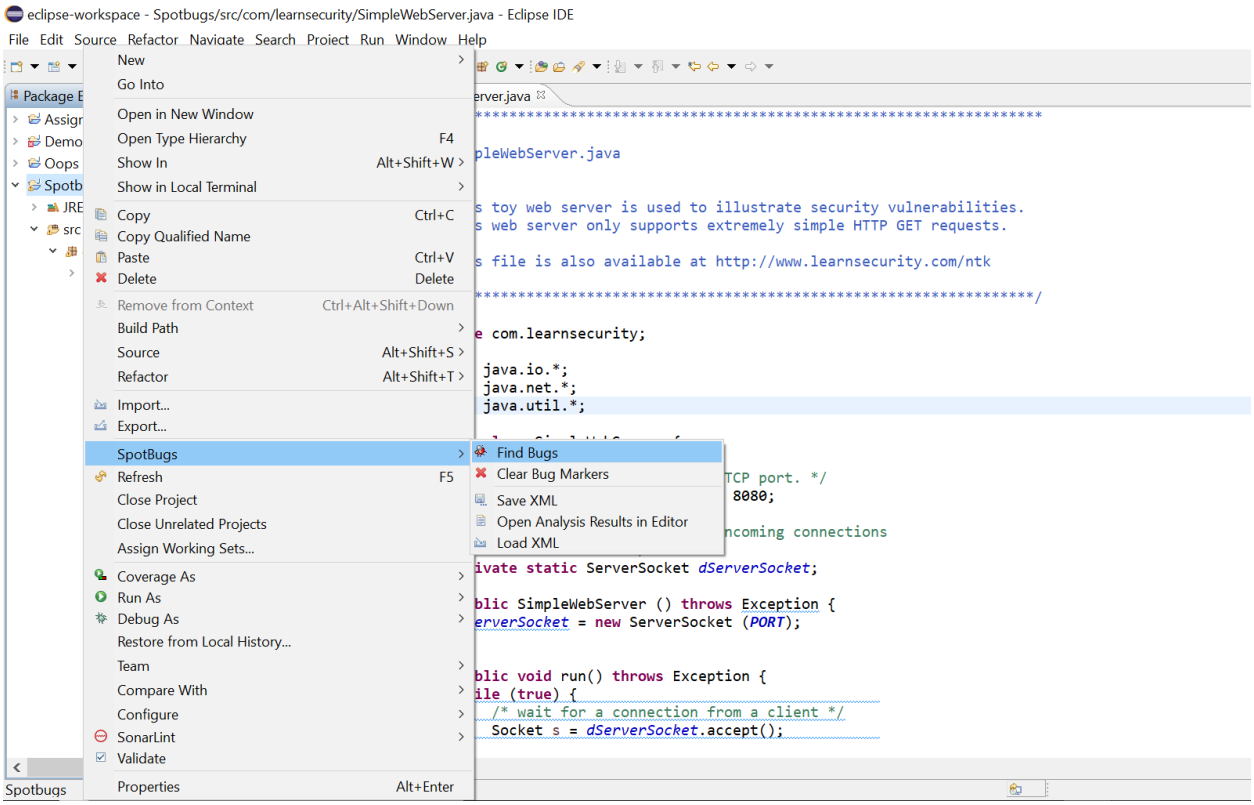
Help => Eclipse Marketplace => SpotBugs



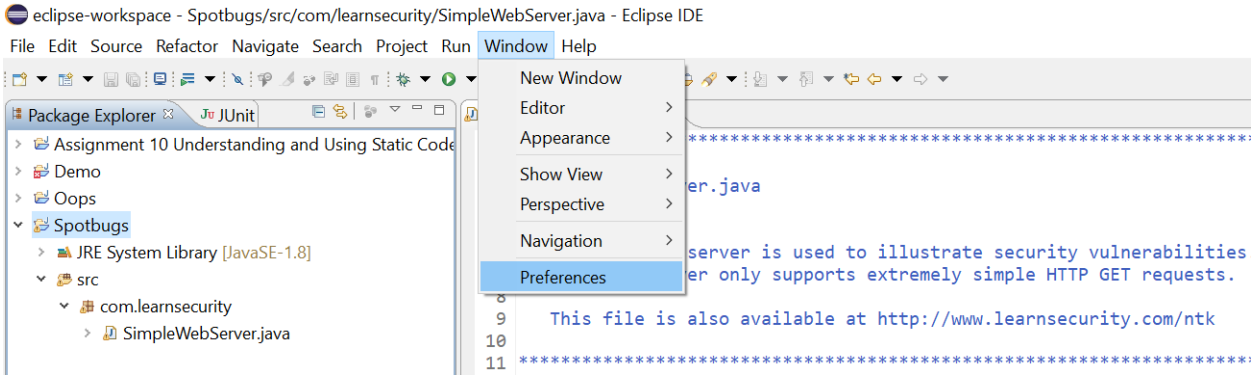


Screenshot shows that the SpotBugs plugin has been installed. We can confirm this by checking if the code has bugs.

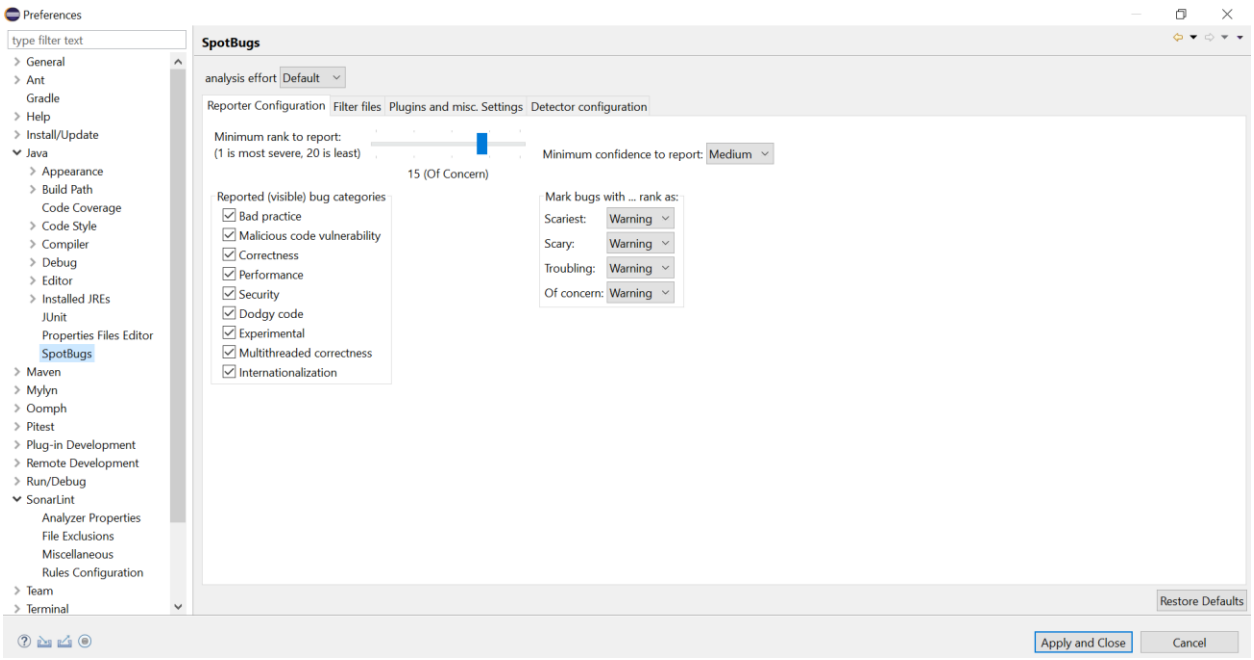
We take the SimpleWebServer.java file and right click on the project in which it is present. We can see SpotBugs option. If we click that we get Find Bugs. By clicking on Find Bugs we can identify all the bugs in the program. But before doing that, in eclipse IDE, we navigate to Window => Preferences.

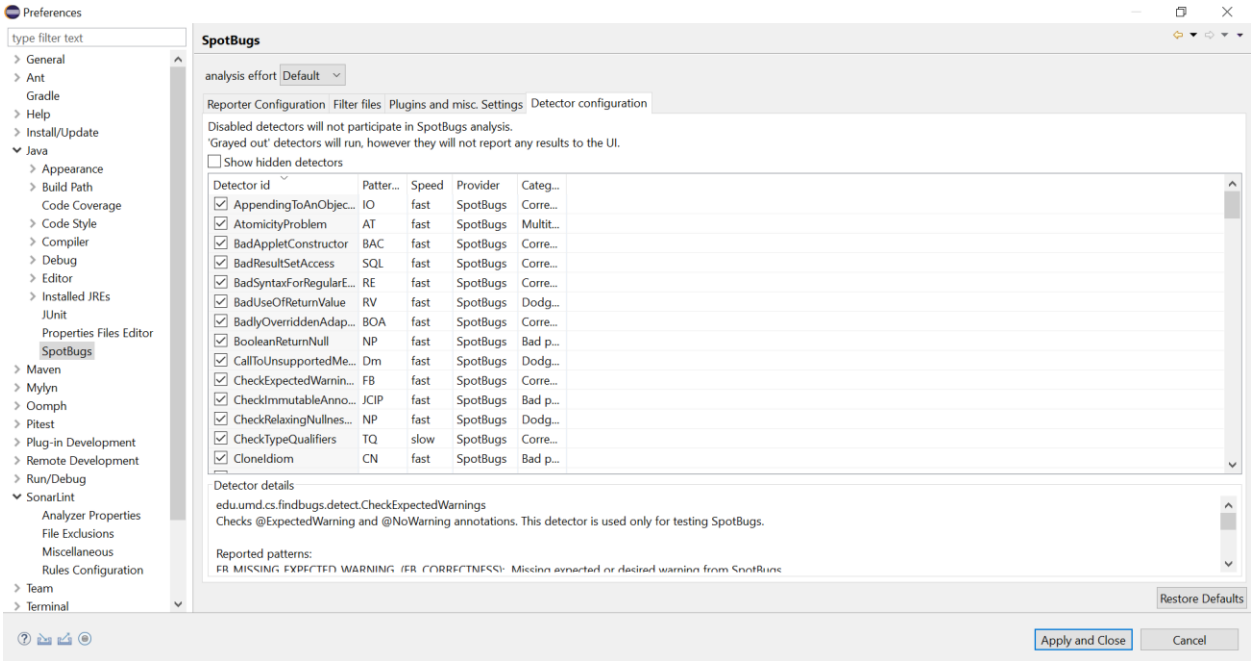


Information about tools settings and plugins used:

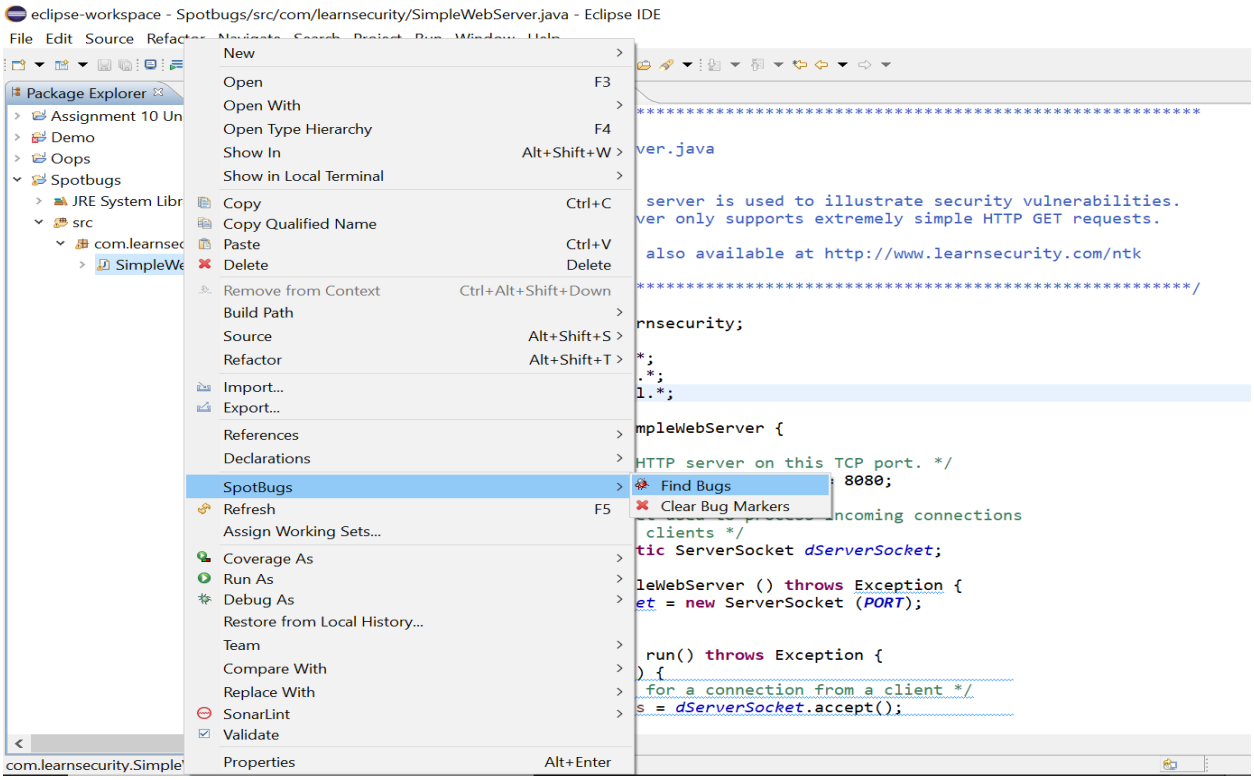


We navigate to Java=> SpotBugs in Preferences and make sure we mark all the check boxes for Reported Bug categories in Reported Configuration and Detector Configuration. We want to detect all kinds of bugs found in the program using SpotBugs.

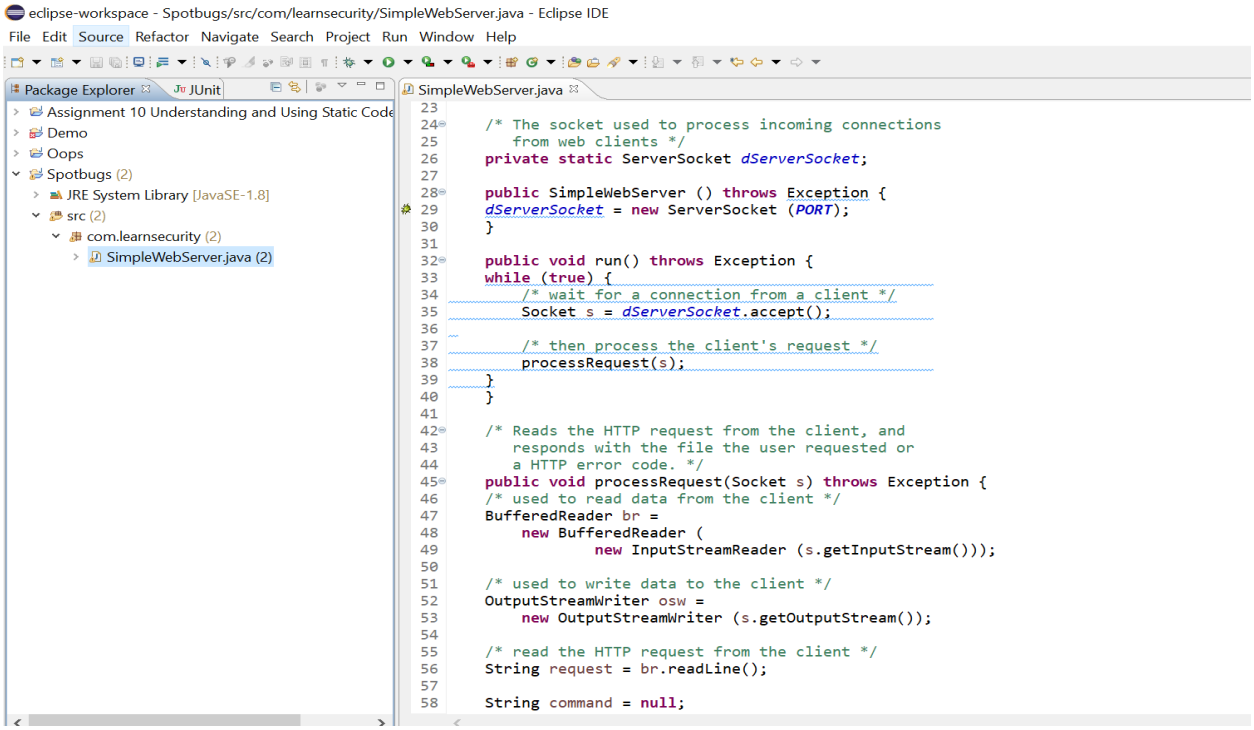




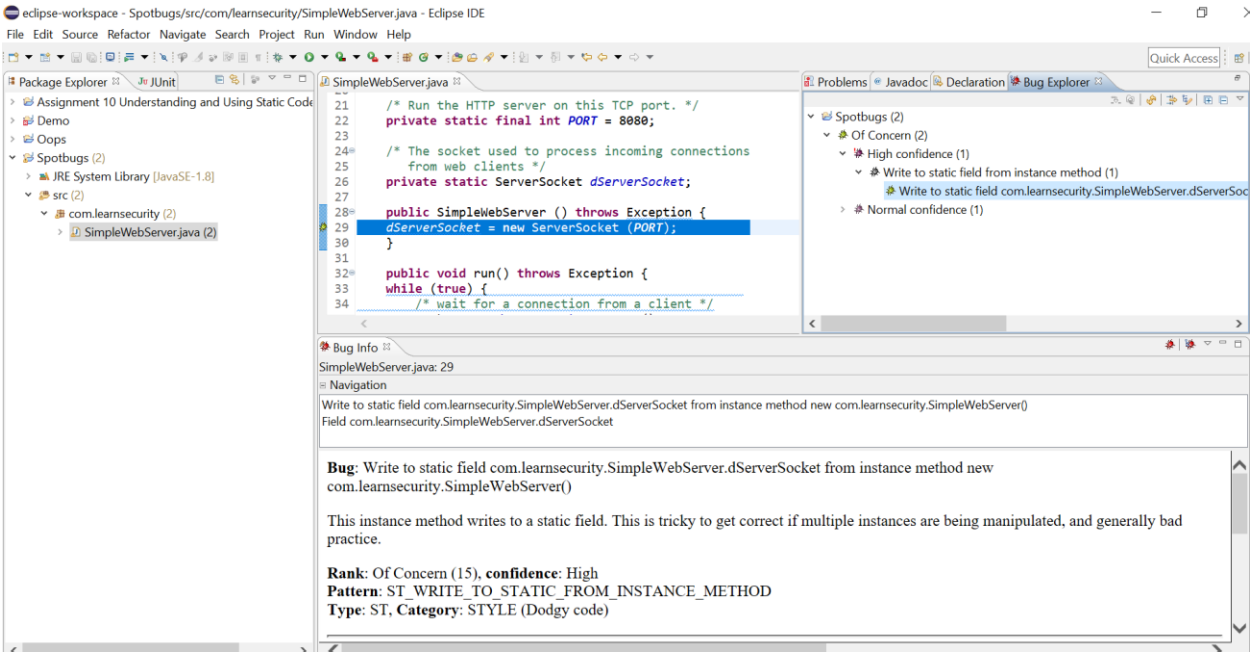
We find all the bugs now by right clicking on file SimpleWebServer.java going to SpotBugs => Find Bugs



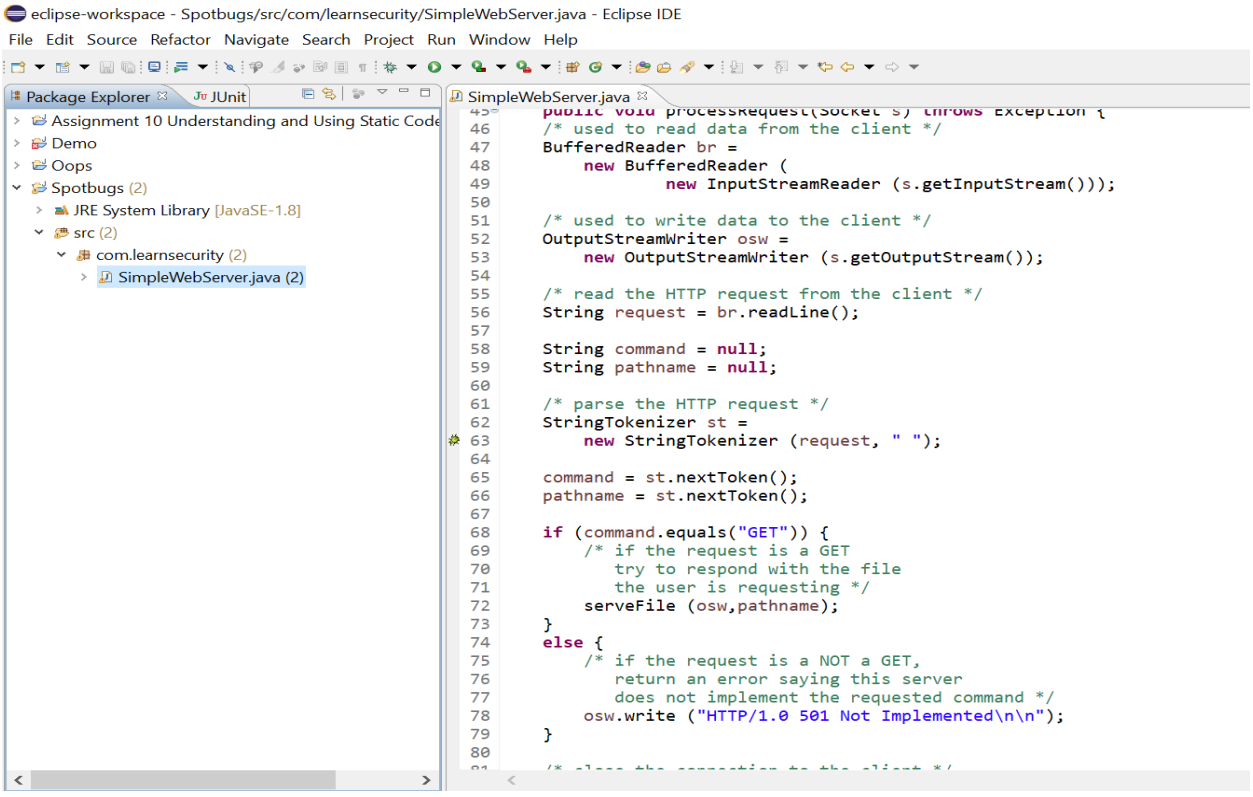
We find a bug in Line 29 as per the screenshot.



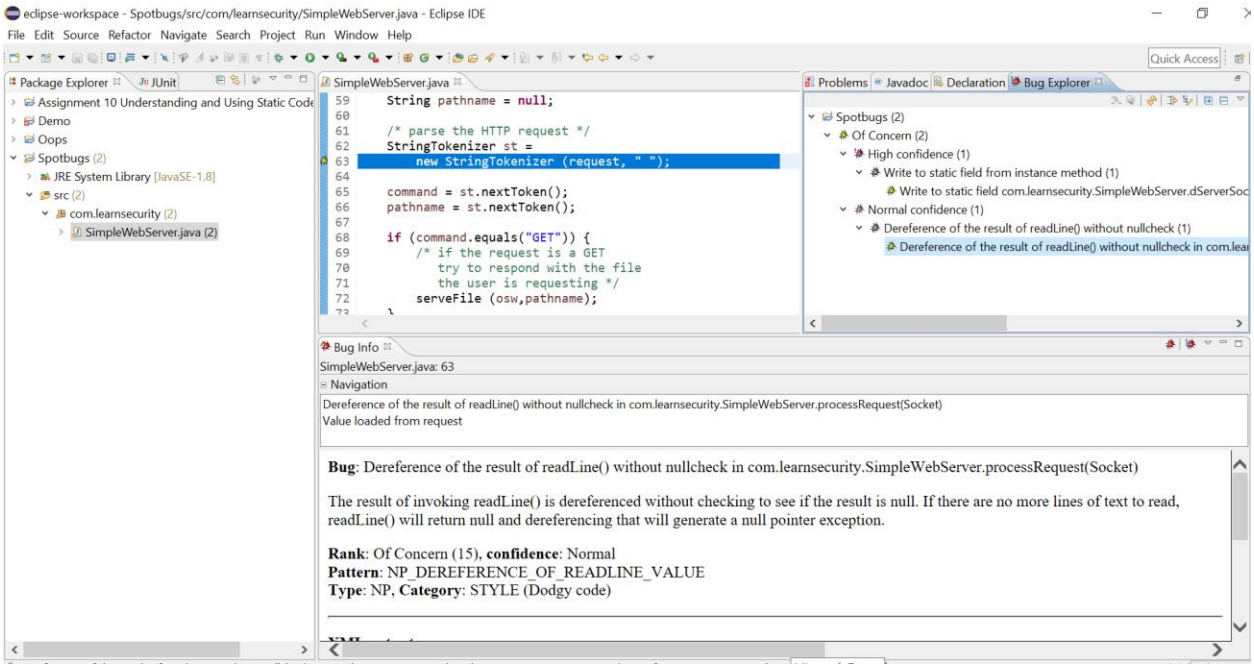
In line 26, *dServerSocket* has been created as a static variable. This is causing a bug in line 29, where we try to write instance method **new ServerSocket (*PORT*)**to a static variable. The instance method writes to a static field. This is tricky to get correct if multiple instances are being manipulated, and generally bad practice.



We find a bug in Line 63 as per the screenshot.



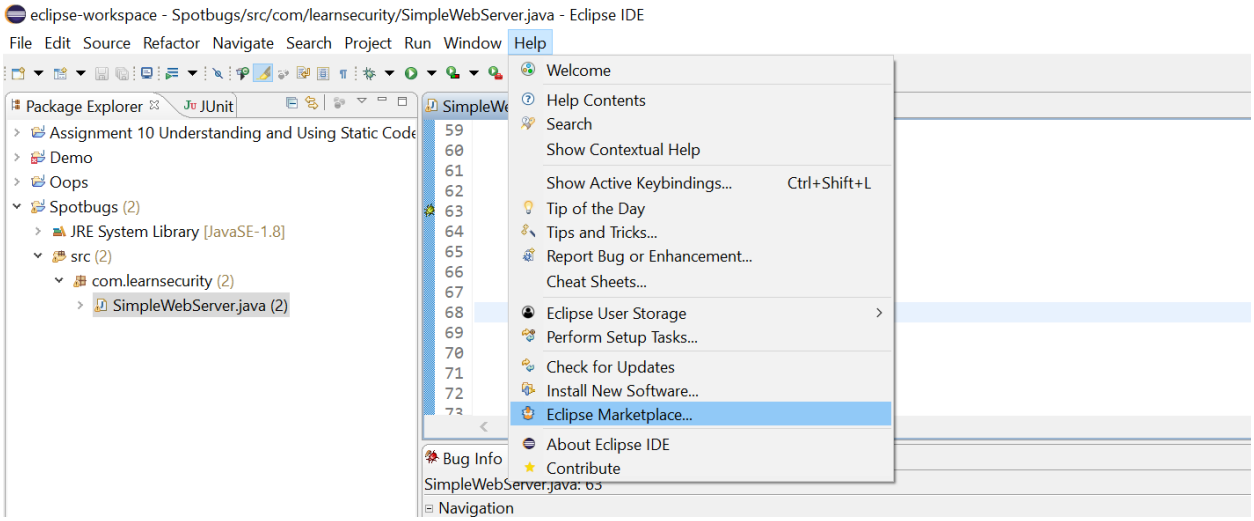
In line 63, there is a dereference of the output of readLine() without nullcheck. The result of invoking readLine() is dereferenced without checking to see if the result is null. If there are no more lines of text to read, readLine() will return null and dereferencing that will generate a null pointer exception.



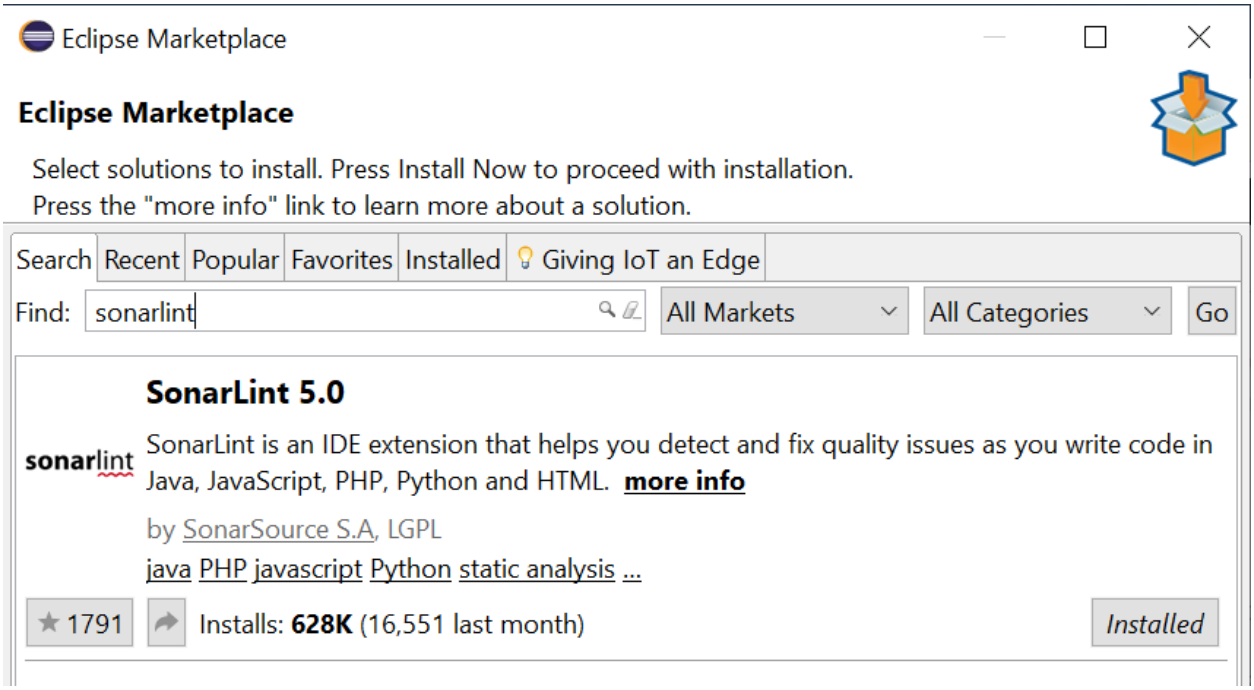
b) SonarLint

In Eclipse IDE, navigate to Help and find Elipse Marketplace. Then search for SonarLint and install it.

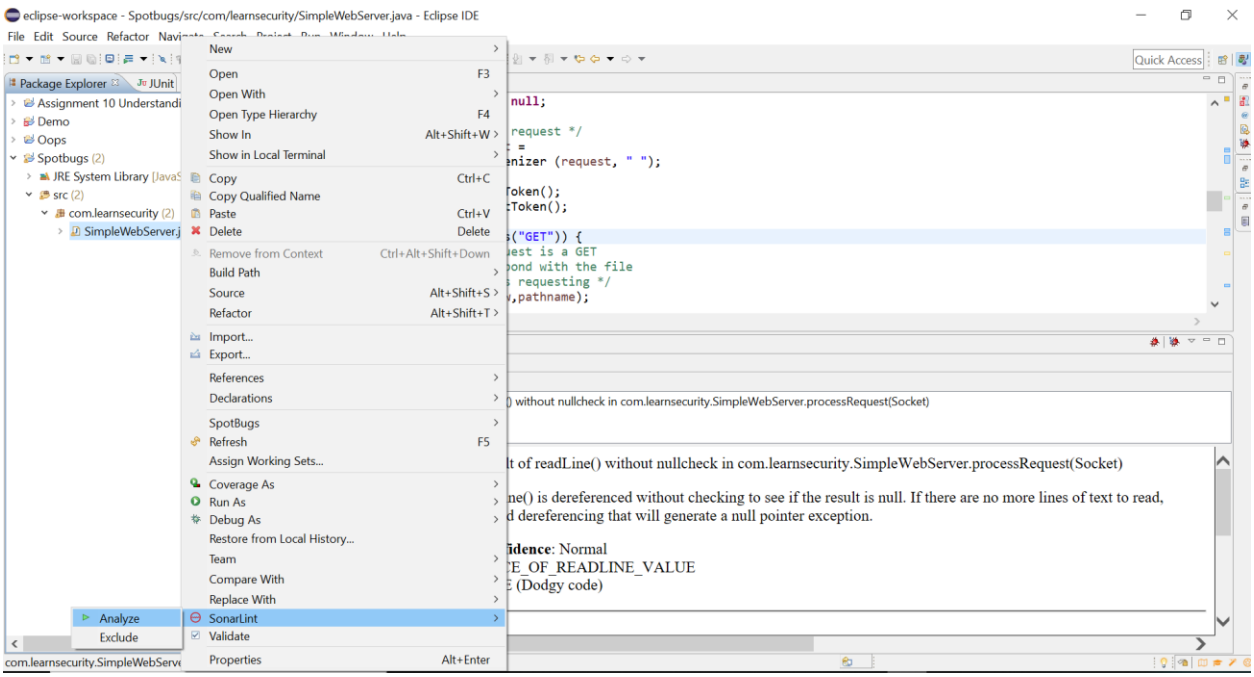
Help => Eclipse Marketplace => SonarLint



Screenshot shows that the SonarLint plugin has been installed. We can confirm this by checking if the code has bugs.



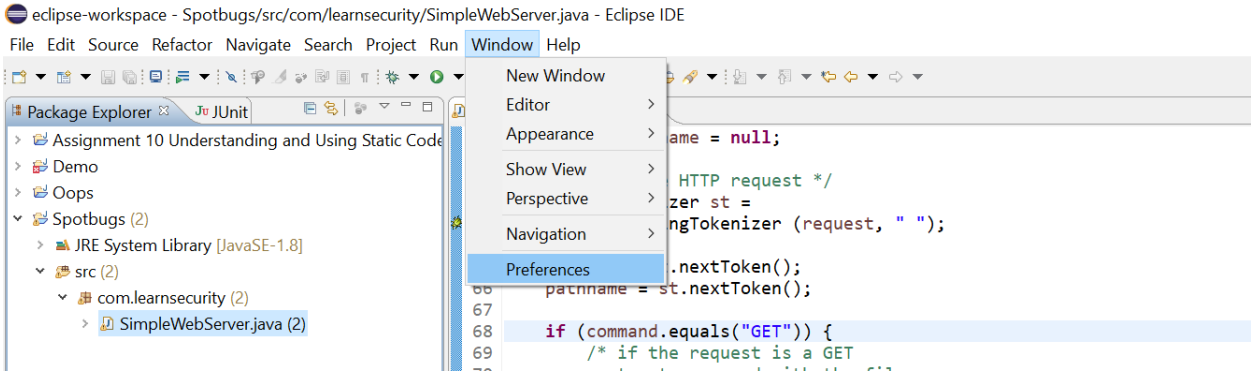
We take the SimpleWebServer.java file and right click on the project in which it is present. We can see SonarLint option. If we click that we get two options - Analyze, Exclude. By clicking on Analyze we can identify all the bugs in the program. But before doing that, in eclipse IDE, we navigate to Window => Preferences.

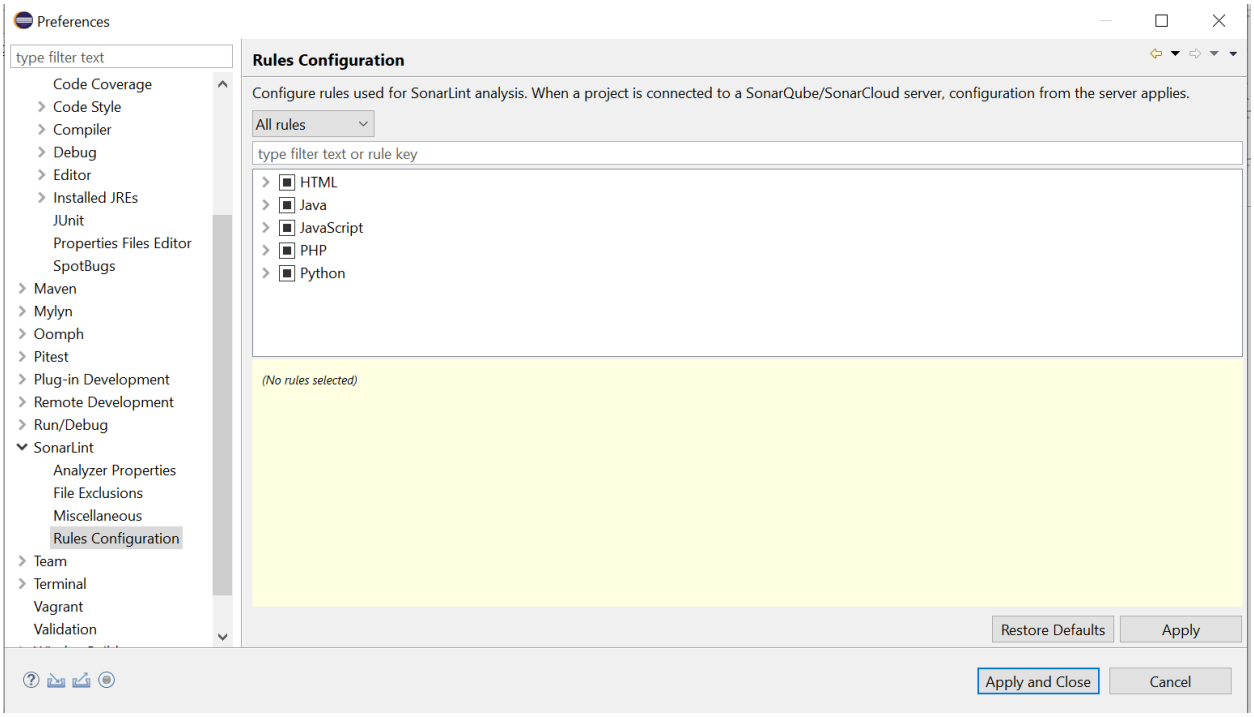


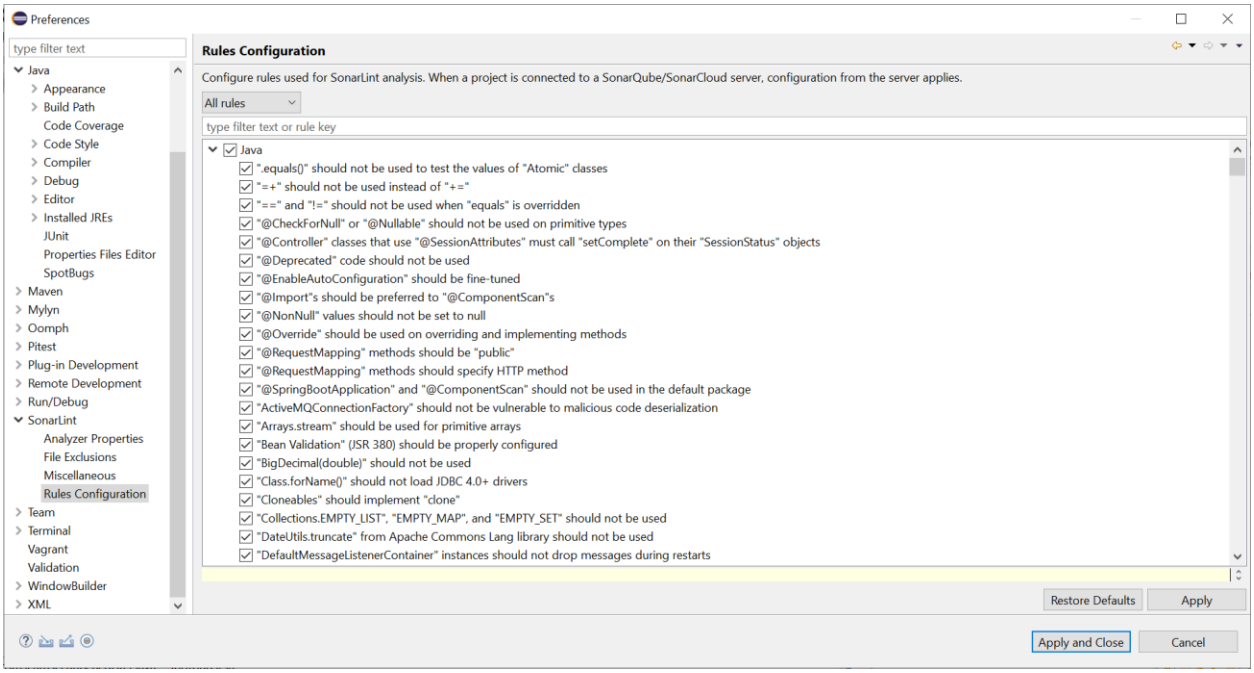
Information about tools settings and plugins used:

In eclipse IDE, we navigate to Window => Preferences.

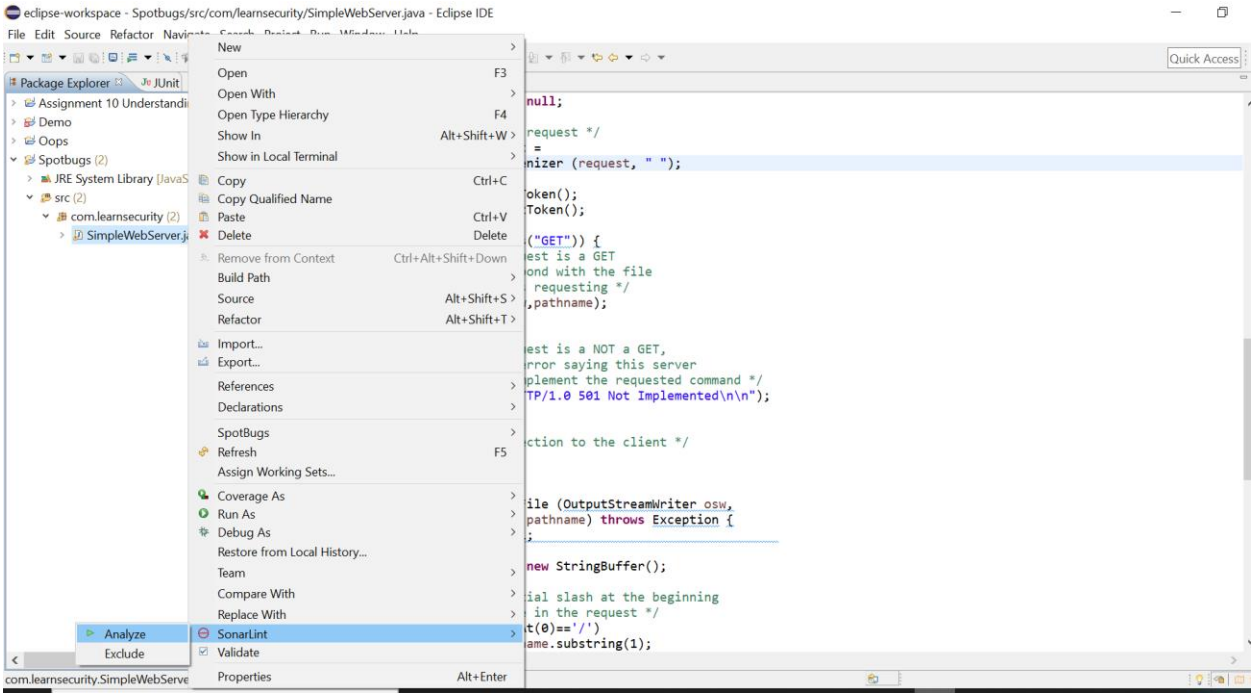
We navigate to Java=> SonarLint in Preferences and make sure we mark all the check boxes for Java in Rules Configuration. We want to detect all kinds of bugs found in the program using SonarLint.



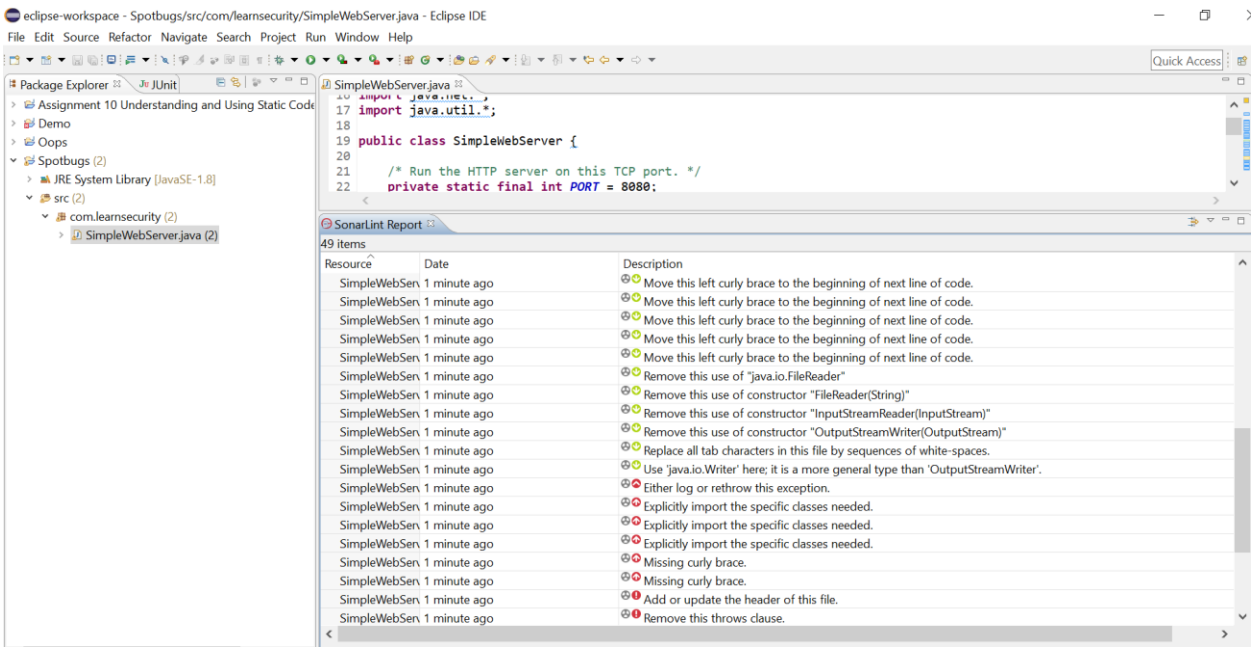




When all the rules are configured during SonarLint analysis, we will be able to detect all kinds of bugs in the entire program. We make this check so that we don't implement leaving bugs in the program.



We find all the bugs now by right clicking on file SimpleWebServer.java going to SonarLint => Analyze



Some new bugs and suggestions have been provided and found with SonarLint tool. Some of them listed below:

i) *Explicitly import the specific classes needed* – Since import java.util.* is a very common import and all libraries get imported, it is not a best practice and it is not secure. So specify the exact library always.

ii) *Missing curly brace* – After if (condition) we always need to give curly braces and specify the boundaries or the block of statements for the action to be taken when the condition is true. Otherwise, only pone line after this condition is considered as action.

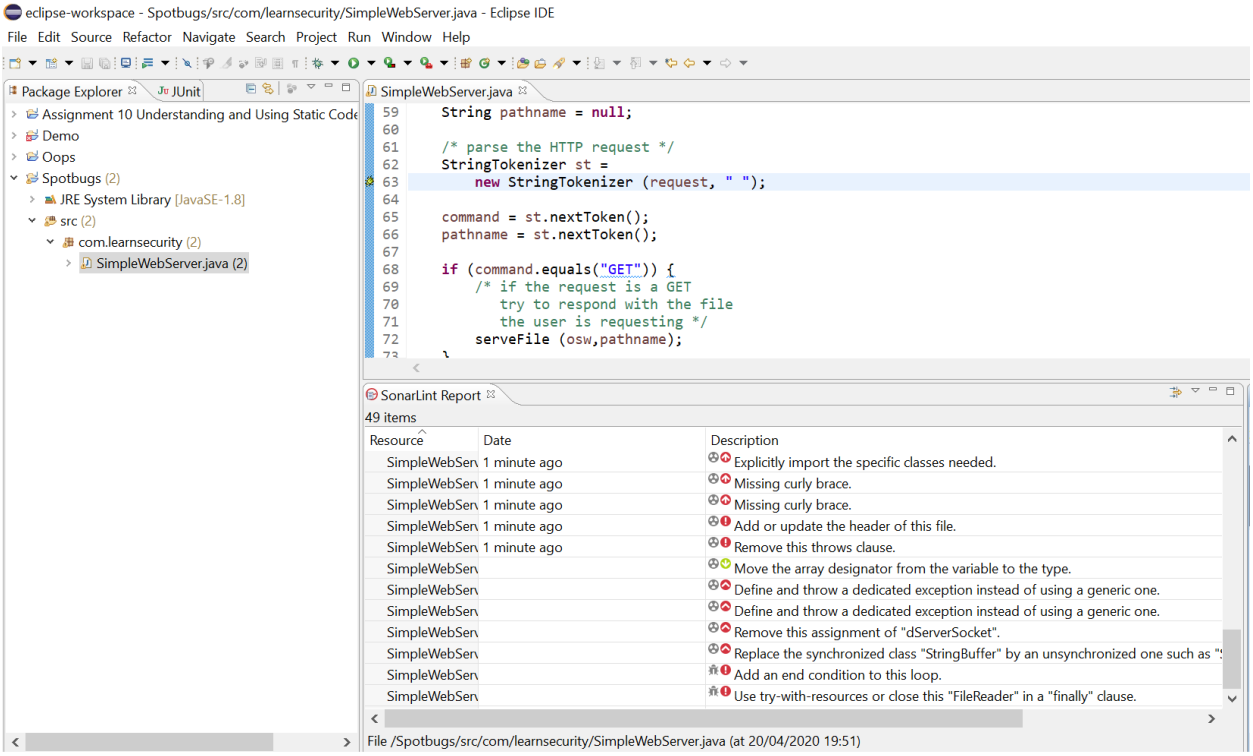
iii) *Define and throw a dedicated exception instead of using a generic one* – This bug is shown when the type of Exception to be thrown is not given in throw clause.

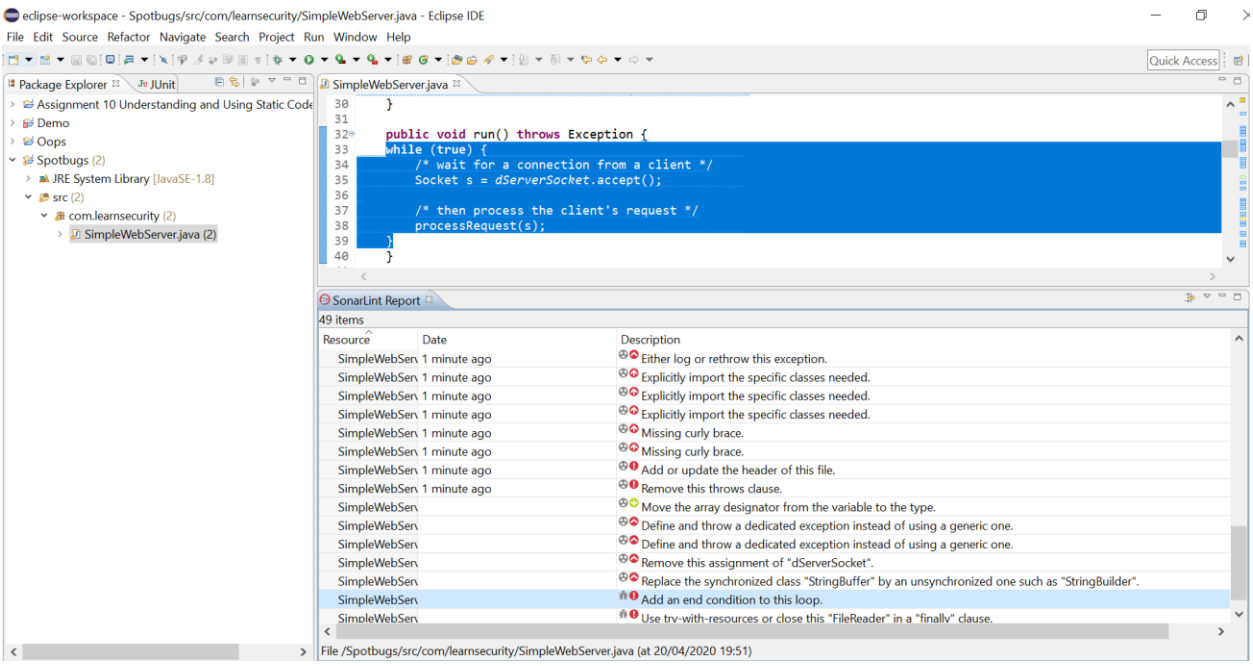
```
public void serveFile (OutputStreamWriter osw,
                       String pathname) throws Exception
```

The above code shows that we have not given what exception to be thrown like IOException or NullPointerException or which kind. We have just specified a generic Exception after throw clause.

iv) *Replace the synchronized class "StringBuffer" by an unsynchronized one such as "StingBuilder"* – In String Buffer, the methods are synchronized where necessary so that all the operations on any particular instance behave as if they occur in some serial order that is consistent with the order of the method calls made by each of the individual threads involved. In case of StringBuilder, this class is designed for use as a drop-in replacement for StringBuffer in places where the string buffer was being used by a single thread (as is generally the case). Where possible, it is recommended that this class be used in preference to StringBuffer as it will be faster under most implementations.

v) *Add an end-condition to this loop* – The while(true) loop in line 33 may go into infinite loop as there is no condition to end the loop


eclipse-workspace - Spotbugs/src/com/learnsecurity/SimpleWebServer.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
Assignment 10 Understanding and Using Static Code
Demo
Oops
Spotbugs (2)
JRE System Library [JavaSE-1.8]
src (2)
com.learnsecurity (2)
SimpleWebServer.java (2)
SimpleWebServer.java
59 String pathname = null;
60
61 /* parse the HTTP request */
62 StringTokenizer st =
63 new StringTokenizer (request, " ");
64
65 command = st.nextToken();
66 pathname = st.nextToken();
67
68 if (command.equals("GET")) {
69 /* if the request is a GET
70 try to respond with the file
71 the user is requesting */
72 serveFile (osw,pathname);
SonarLint Report
49 items
Resource
Date
Description
SimpleWebServ 1 minute ago Explicitly import the specific classes needed.
SimpleWebServ 1 minute ago Missing curly brace.
SimpleWebServ 1 minute ago Missing curly brace.
SimpleWebServ 1 minute ago Add or update the header of this file.
SimpleWebServ 1 minute ago Remove this throws clause.
SimpleWebServ Move the array designator from the variable to the type.
SimpleWebServ Define and throw a dedicated exception instead of using a generic one.
SimpleWebServ Define and throw a dedicated exception instead of using a generic one.
SimpleWebServ Remove this assignment of "dServerSocket".
SimpleWebServ Replace the synchronized class "StringBuffer" by an unsynchronized one such as "
SimpleWebServ Add an end condition to this loop.
SimpleWebServ Use try-with-resources or close this "FileReader" in a "finally" clause.
File /Spotbugs/src/com/learnsecurity/SimpleWebServer.java (at 20/04/2020 19:51)


eclipse-workspace - Spotbugs/src/com/learnsecurity/SimpleWebServer.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
JUnit
Assignment 10 Understanding and Using Static Code
Demo
Oops
Spotbugs (2)
JRE System Library [JavaSE-1.8]
src (2)
com.learnsecurity (2)
SimpleWebServer.java (2)
SimpleWebServer.java
30 }
31
32 public void run() throws Exception {
33 while (true) {
34 /* wait for a connection from a client */
35 Socket s = dServerSocket.accept();
36
37 /* then process the client's request */
38 processRequest(s);
39 }
40 }
SonarLint Report
49 items
Resource
Date
Description
SimpleWebServ 1 minute ago Either log or rethrow this exception.
SimpleWebServ 1 minute ago Explicitly import the specific classes needed.
SimpleWebServ 1 minute ago Explicitly import the specific classes needed.
SimpleWebServ 1 minute ago Explicitly import the specific classes needed.
SimpleWebServ 1 minute ago Missing curly brace.
SimpleWebServ 1 minute ago Missing curly brace.
SimpleWebServ 1 minute ago Add or update the header of this file.
SimpleWebServ 1 minute ago Remove this throws clause.
SimpleWebServ Move the array designator from the variable to the type.
SimpleWebServ Define and throw a dedicated exception instead of using a generic one.
SimpleWebServ Define and throw a dedicated exception instead of using a generic one.
SimpleWebServ Remove this assignment of "dServerSocket".
SimpleWebServ Replace the synchronized class "StringBuffer" by an unsynchronized one such as "StringBuilder".
SimpleWebServ Add an end condition to this loop.
SimpleWebServ Use try-with-resources or close this "FileReader" in a "finally" clause.
File /Spotbugs/src/com/learnsecurity/SimpleWebServer.java (at 20/04/2020 19:51)

Comparison/Contrast Tools:

Does the tool analyze source or binary as input?

SpotBugs, which is the continuation of FindBugs, analyzes the binary while others analyze source directly. It analyzes the Java bytecode to find bugs in the code.

SonarLint is an IDE extension that helps you detect and fix quality issues as you write code. Like a spell checker, SonarLint squiggles flaws so that they can be fixed before committing code. SonarLint analyzes the source. It makes the management of code easy. It takes care of maintainability and security.

Which category of tools is it?

SpotBugs detects possible bugs in Java programs. Potential errors are classified in four ranks: scariest, scary, troubling and of concern. This is a hint to the developer about their possible impact or severity

SpotBugs comes under the following category of tools:

i) Type checking
ii) Bug finding
iii) Security review

SonarLint is an IDE extension that helps you detect and fix quality issues as you write code. Like a spell checker, it squiggles flaws so that they can be fixed before committing code.

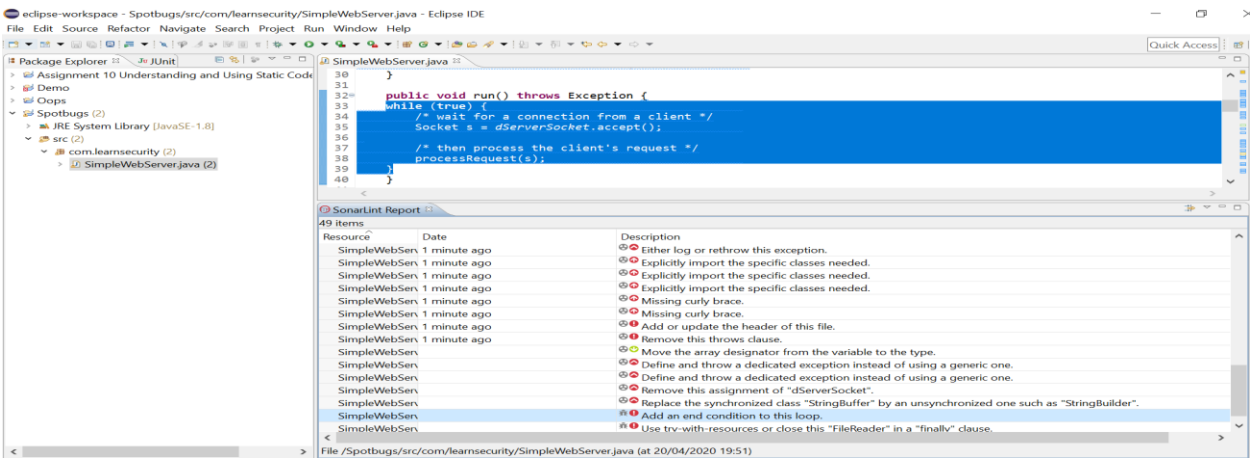
SonarLint comes under the following category of tools:

i) Style checking
ii) Bug finding
iii) Security review

Show an example (if one exists) of a finding that is reported by one tool and not others:

SoanrLint recognizes some new bugs in the program as explained earlier. One of the finding reported in SonarLint and not in SpotBugs is the exception below.

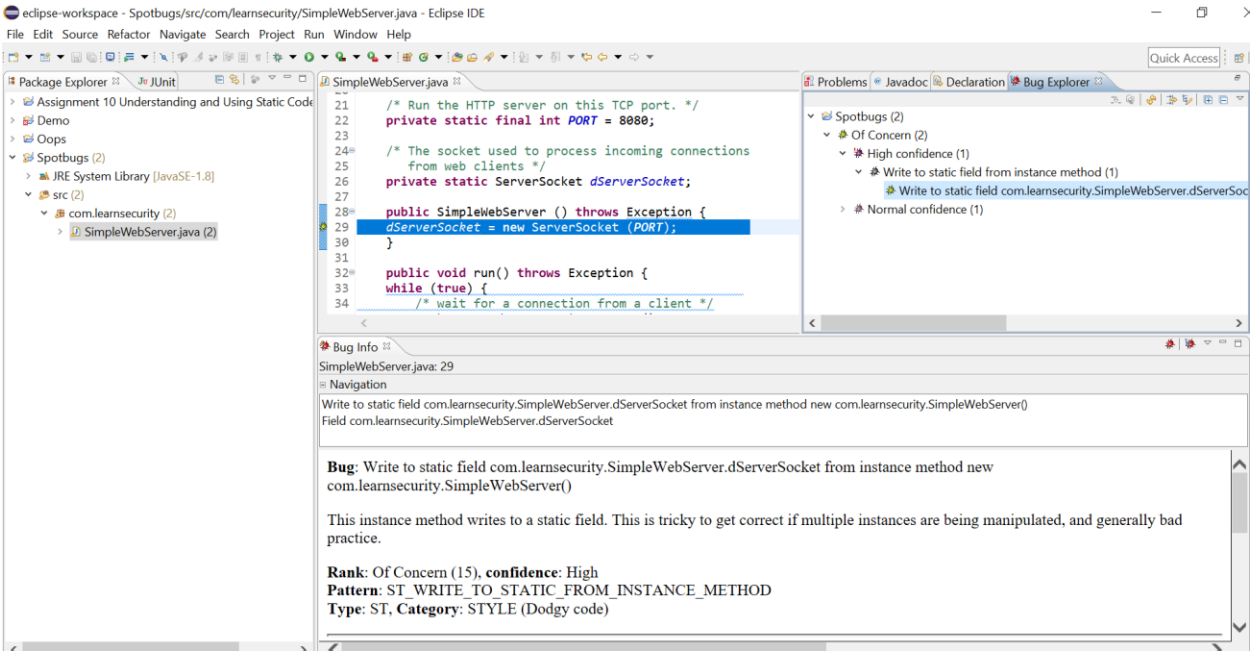
Add an end-condition to this loop – The while(true) loop in line 33 may go into infinite loop as there is no condition to end the loop.



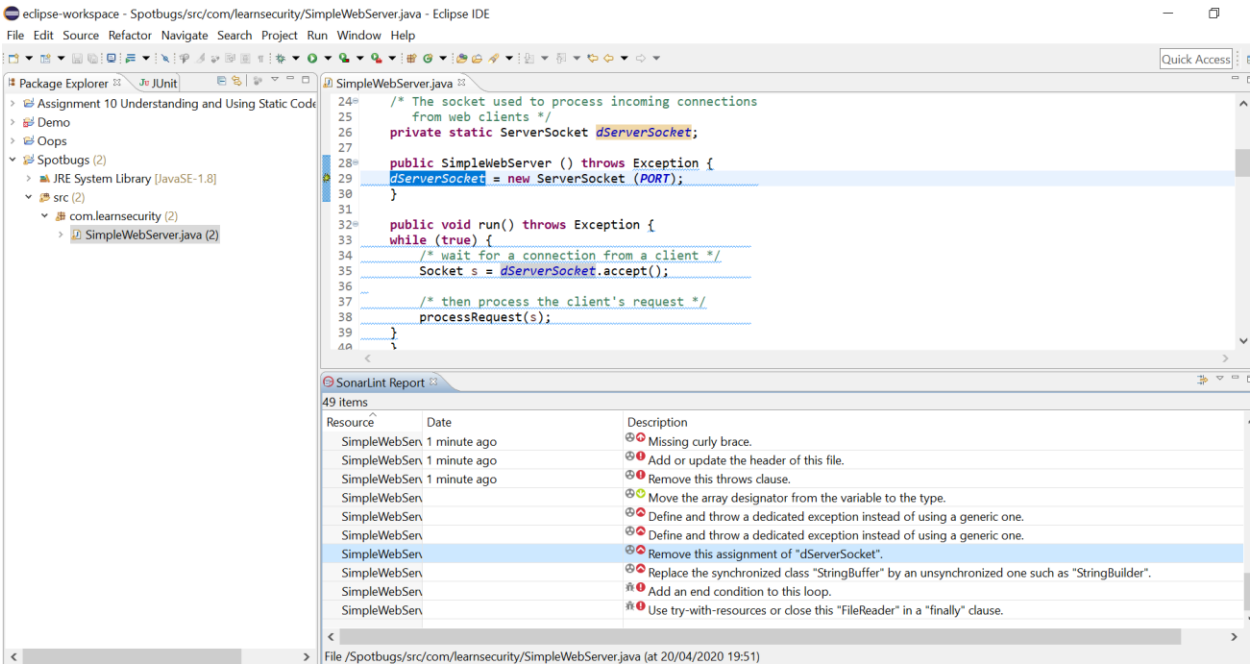
Show an example (if one exists) of a finding reported by multiple tools:

The following bug was found and reported by both SpotBugs and SonarLint.

SpotBugs – Write to static field from instance method. In line 26, `dServerSocket` has been created as a static variable. This is causing a bug in line 29, where we try to write instance method `new ServerSocket (PORT)` to a static variable. The instance method writes to a static field. This is tricky to get correct if multiple instances are being manipulated, and generally bad practice.

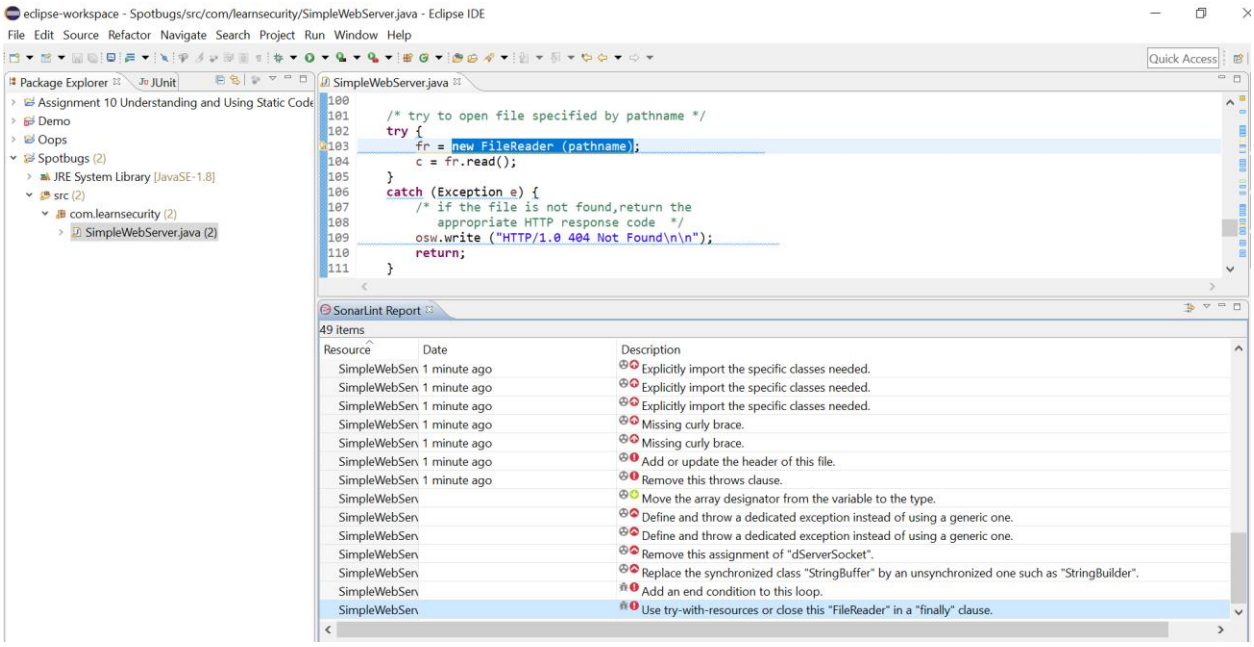


SonarLint - Remove this assignemnet of dServerSocket. This bug also points line 29 specified earlier.

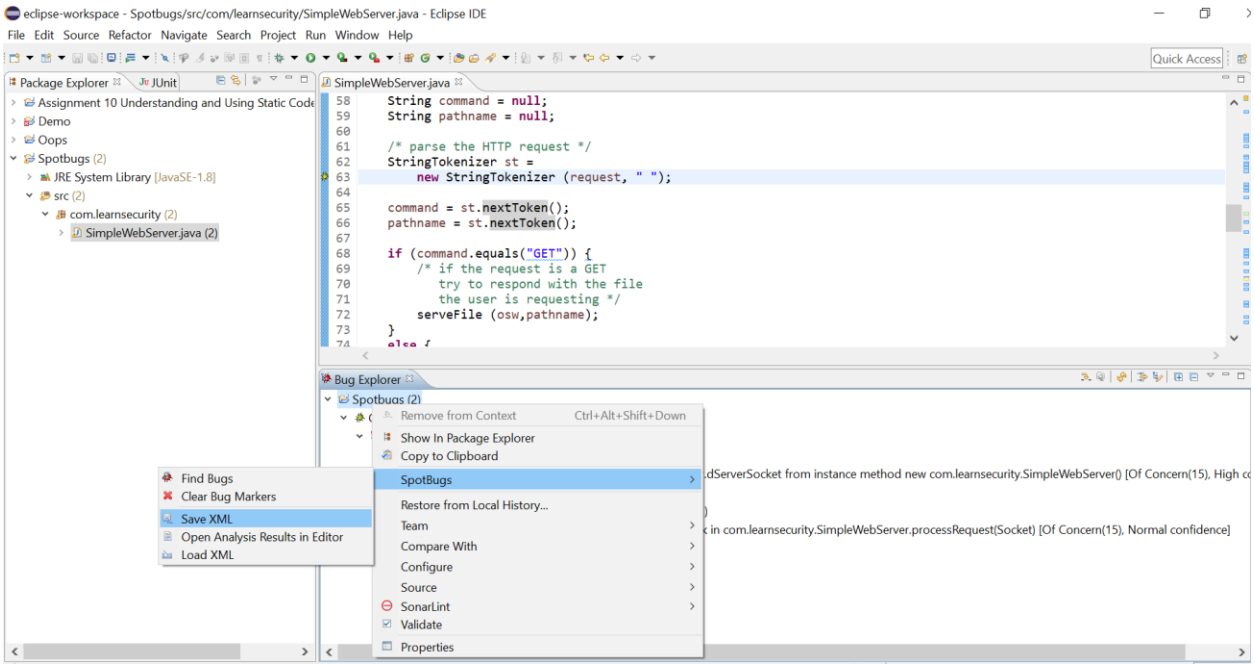


For the known flaw in the code used, document which tools reported it (true negative) and which tools did not (false positive):

We know that the known flaw from the Manual analysis done earlier is that the FileReader object has been created, but it is also not closed. Always closing the streams is a good habit which helps us to avoid some odd behavior. SonarLint identified this bug with the exception “Use try-with-resources or close this FileReader in a finally clause”. SpotBugs did not identify this bug.



We can extract the report of SpotBugs in an XML file as shown in the screenshot. This XML file is attached to the assignment submission.



XML file preview:

```
<?xml version="1.0" encoding="UTF-8"?>
<BugCollection version="4.0.1" sequence="1" timestamp="1587429245823" analysisTimestamp="1587429240634" release="">
  <Project projectName="Spotbugs">
    <Jar>C:\Users\gouta\eclipse-workspace\Spotbugs\bin\com\learnsecurity\SimpleWebServer.class</Jar>
    <AuxClasspathEntry>C:\Users\gouta\eclipse-workspace\Spotbugs\bin</AuxClasspathEntry>
    <AuxClasspathEntry>C:\Program Files\Java\jre1.8.0_251\lib\rt.jar</AuxClasspathEntry>
    <AuxClasspathEntry>C:\Program Files\Java\jre1.8.0_251\lib\jce.jar</AuxClasspathEntry>
    <SrcDir>C:\Users\gouta\eclipse-workspace\Spotbugs\src</SrcDir>
  </Project>
  <BugInstance type="DM_DEFAULT_ENCODING" priority="1" rank="19" abbrev="Dm" category="I18N" first="1">
    <Class classname="com.learnsecurity.SimpleWebServer">
      <SourceLine classname="com.learnsecurity.SimpleWebServer" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
    </Class>
    <Method classname="com.learnsecurity.SimpleWebServer" name="processRequest" signature="(Ljava/net/Socket;)V" isStatic="false">
      <SourceLine classname="com.learnsecurity.SimpleWebServer" start="48" end="83" startBytecode="0" endBytecode="324" sourcefile="SimpleWebServer.java" sourcepath
    </Method>
    <Method classname="java.io.InputStreamReader" name="&lt;init&gt;" signature="(Ljava/io/InputStream;)V" isStatic="false" role="METHOD_CALLED">
      <SourceLine classname="java.io.InputStreamReader"/>
    </Method>
    <SourceLine classname="com.learnsecurity.SimpleWebServer" start="49" end="49" startBytecode="12" endBytecode="12" sourcefile="SimpleWebServer.java" sourcepath="
    <SourceLine classname="com.learnsecurity.SimpleWebServer" start="49" end="49" startBytecode="12" endBytecode="12" sourcefile="SimpleWebServer.java" sourcepath="
  </BugInstance>
  <BugInstance type="DM_DEFAULT_ENCODING" priority="1" rank="19" abbrev="Dm" category="I18N" first="1">
    <Class classname="com.learnsecurity.SimpleWebServer">
      <SourceLine classname="com.learnsecurity.SimpleWebServer" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
    </Class>
    <Method classname="com.learnsecurity.SimpleWebServer" name="processRequest" signature="(Ljava/net/Socket;)V" isStatic="false">
      <SourceLine classname="com.learnsecurity.SimpleWebServer" start="48" end="83" startBytecode="0" endBytecode="324" sourcefile="SimpleWebServer.java" sourcepath
    </Method>
    <Method classname="java.io.OutputStreamWriter" name="&lt;init&gt;" signature="(Ljava/io/OutputStream;)V" isStatic="false" role="METHOD_CALLED">
      <SourceLine classname="java.io.OutputStreamWriter"/>
    </Method>
    <SourceLine classname="com.learnsecurity.SimpleWebServer" start="53" end="53" startBytecode="27" endBytecode="27" sourcefile="SimpleWebServer.java" sourcepath="
    <SourceLine classname="com.learnsecurity.SimpleWebServer" start="53" end="53" startBytecode="27" endBytecode="27" sourcefile="SimpleWebServer.java" sourcepath="
  </BugInstance>
  <BugInstance type="DM_DEFAULT_ENCODING" priority="1" rank="19" abbrev="Dm" category="I18N" first="1">
    <Class classname="com.learnsecurity.SimpleWebServer">
      <SourceLine classname="com.learnsecurity.SimpleWebServer" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
```

spotbugReport.xml

Part 2:

The code provided is some code written in the past. It is written in Java.

Code Provided:

```
package com.learnsecurity;

import java.util.ArrayList;
/**
 * Class to Register an individual
 */
public class Register {
	/** The check input value. */
	// This should be "Individual" or "Business" or "Admin"
	private String typeOfUser= null;
	// Name of the user
	private String name = null;

    /**
     * The Constructor.
     */
    public Register() {
        this.typeOfUser = null;
        this.name = null;
    }
    public void run() {
       individualInput();
       IndividualValidation();
       ArrayList<String> list = new ArrayList<>();
         list.add(this.typeOfUser);
         list.add(this.name);
         System.out.println("Imformation of the name and type of user: "+list.get(2));
    }
    /**
     * Check the input given in the individual field is for Individual user.
     */
    private void individualInput() {
        if (this.typeOfUser == "Individual") {
            System.out.println("This is valid individual user");
        } else if (this.typeOfUser == "Business") {
            System.out.println("This is valid Business user");
        } else return;
    }

    /**
     * Validate the Individual and give information where to find the details.
     */
    private static void IndividualValidation() {
       boolean dropDown = false;
        if (dropDown = true) {
            System.out.println("You can check the complete details in the Information
page");
        } else {
            System.exit(0);
```

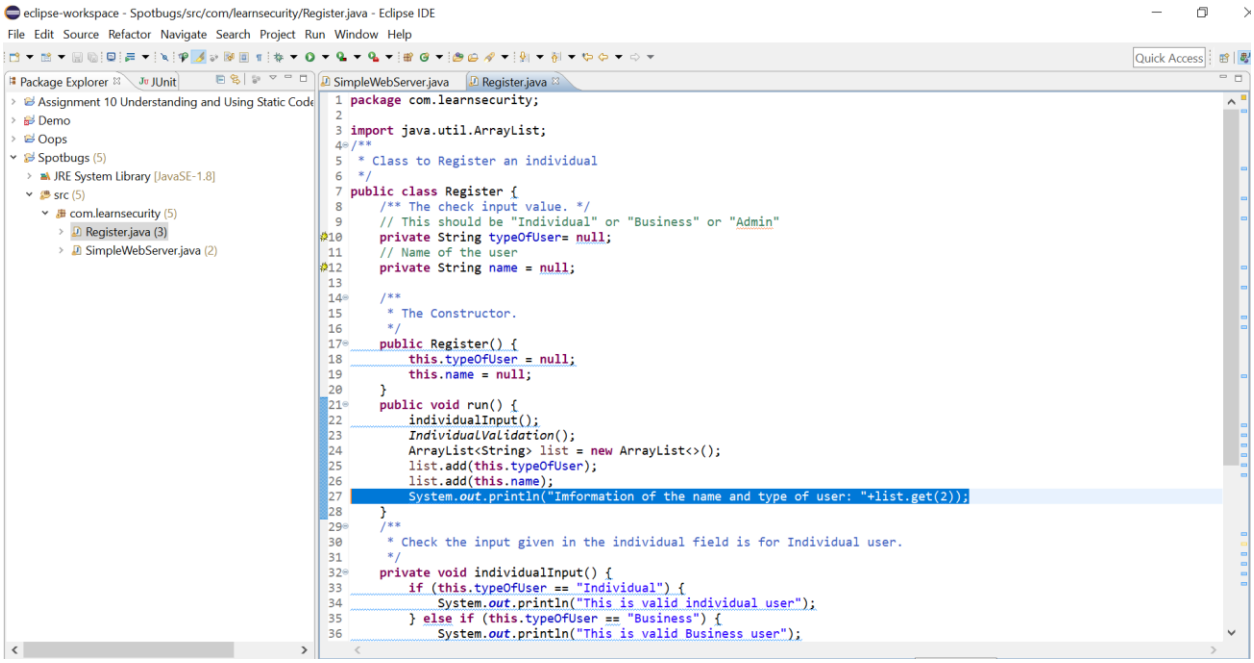
```
        }
    }
}
```

Manual Analysis:

By just looking at the code, we can identify that

i) The string variables typeOfUser and name are initialized to null in lines 10 and 12 as well as in the constructor. So, these two fields remain as null throughout the program.

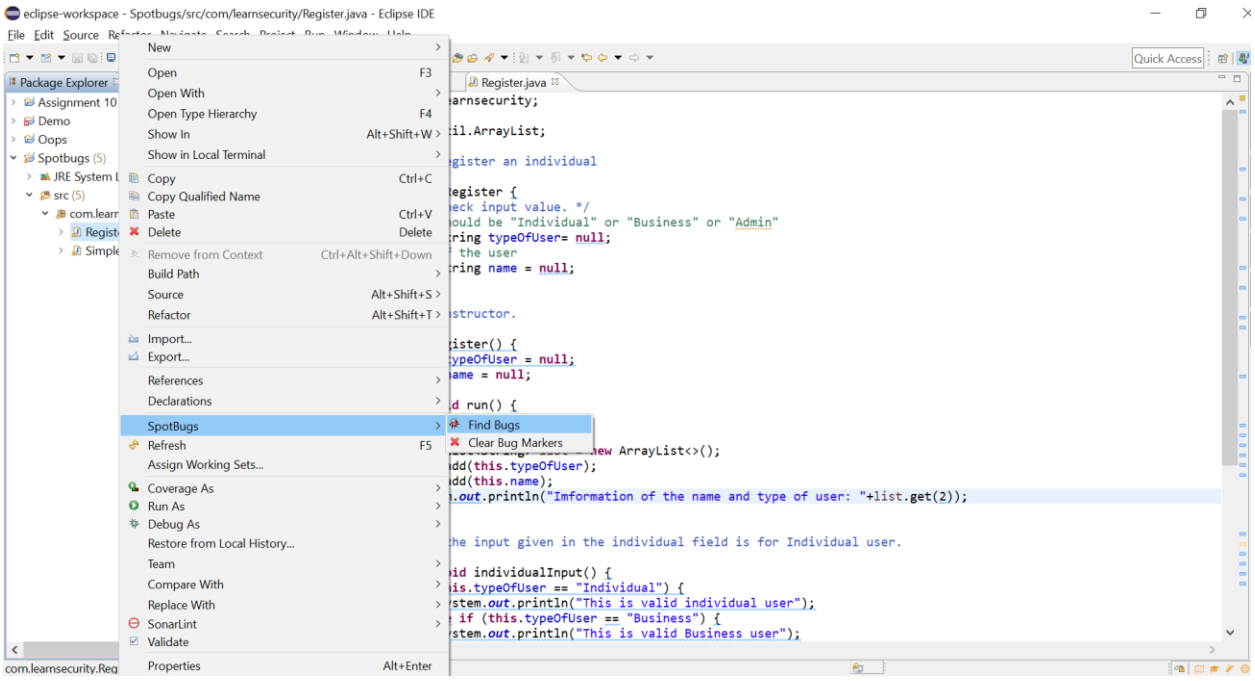
ii) There is one more statement line 27, where the ArrayList list tries to retrieve and element at index 2 which does not exist. The size of list is 2 but it is trying to retrieve the third element. This may cause ArrayOutOfBounds exception.

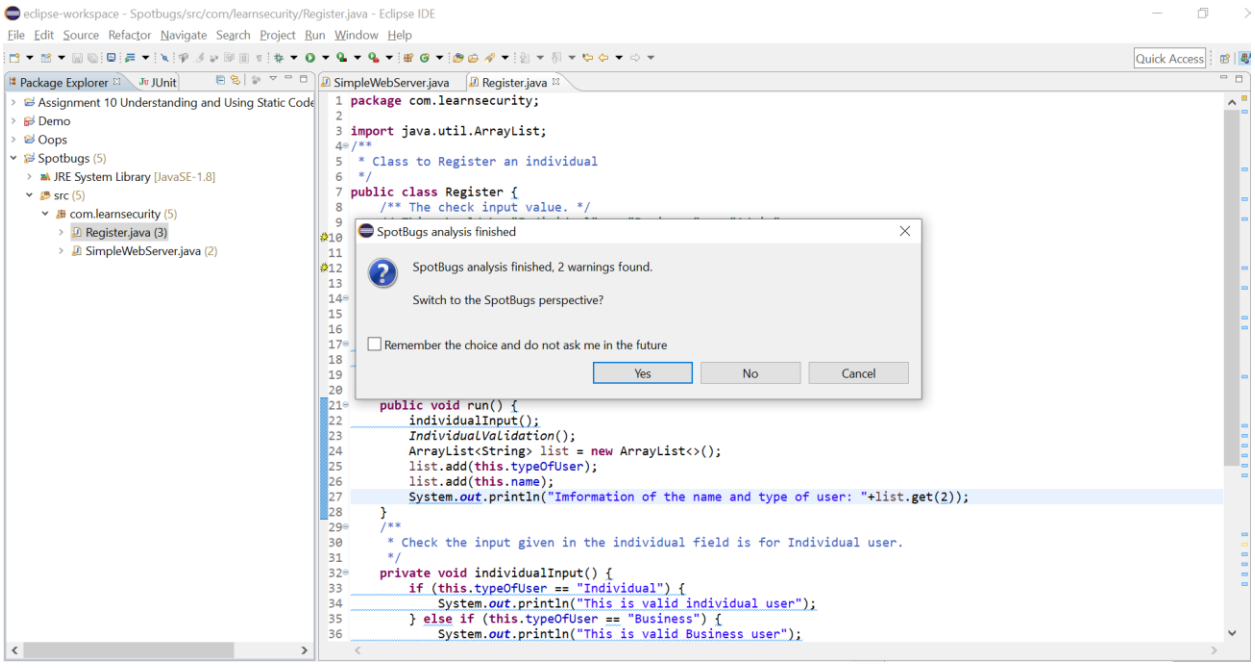


Results:

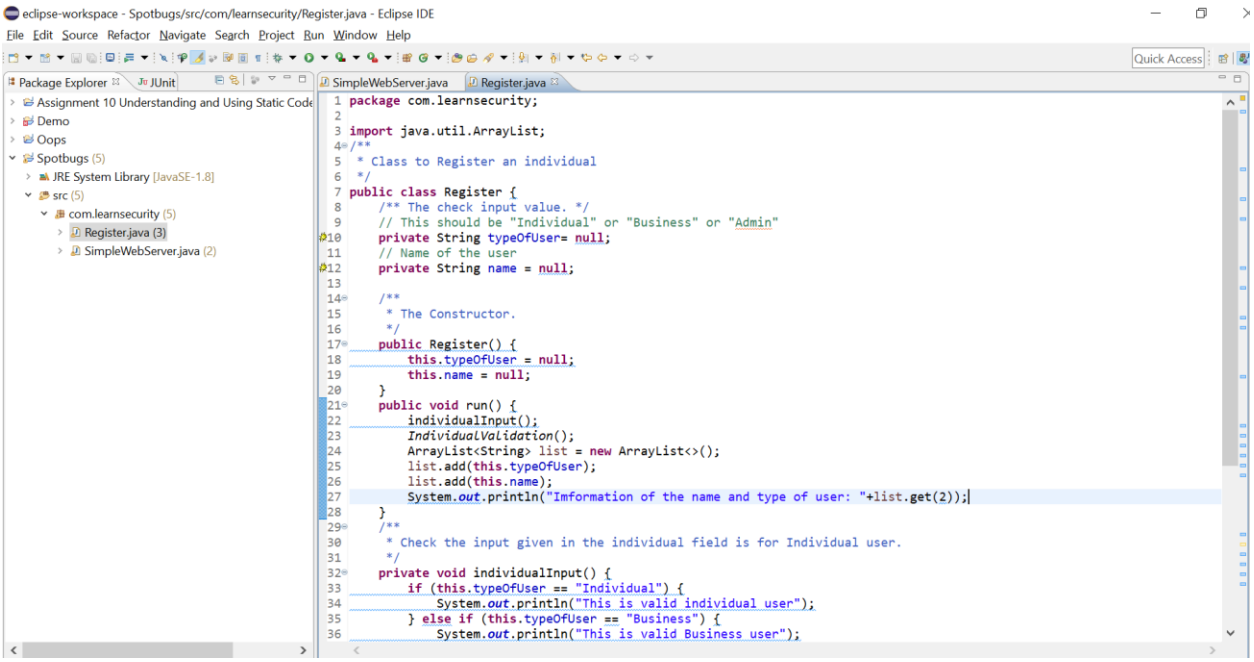
SpotBugs:

We retrieve the bugs in the program by using the SpotBugs tool first.

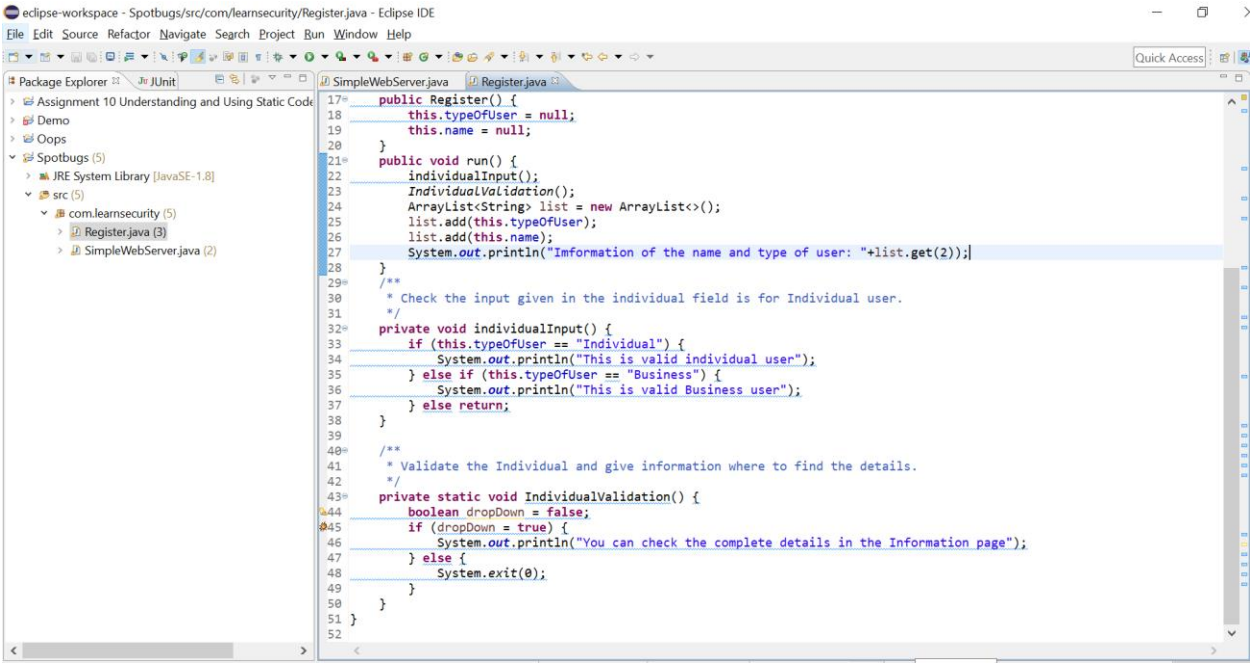




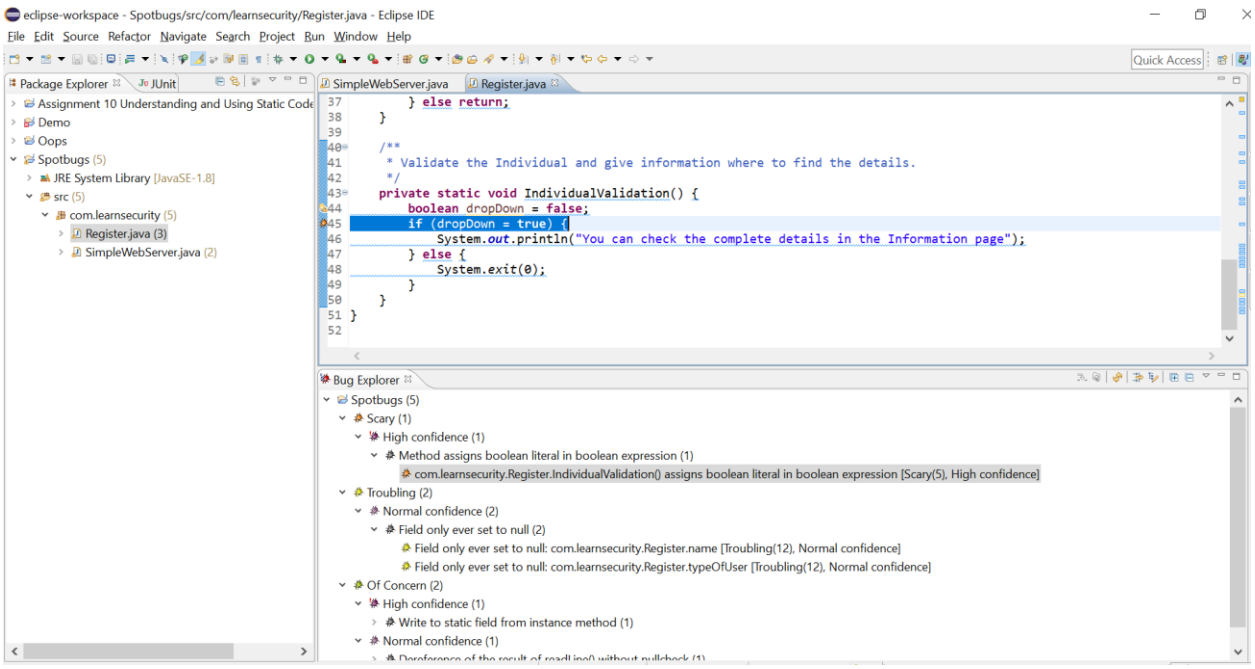
We find 3 bugs found in the program as shown in the results below. The bug symbol near the line shows where the bug is present in the code.



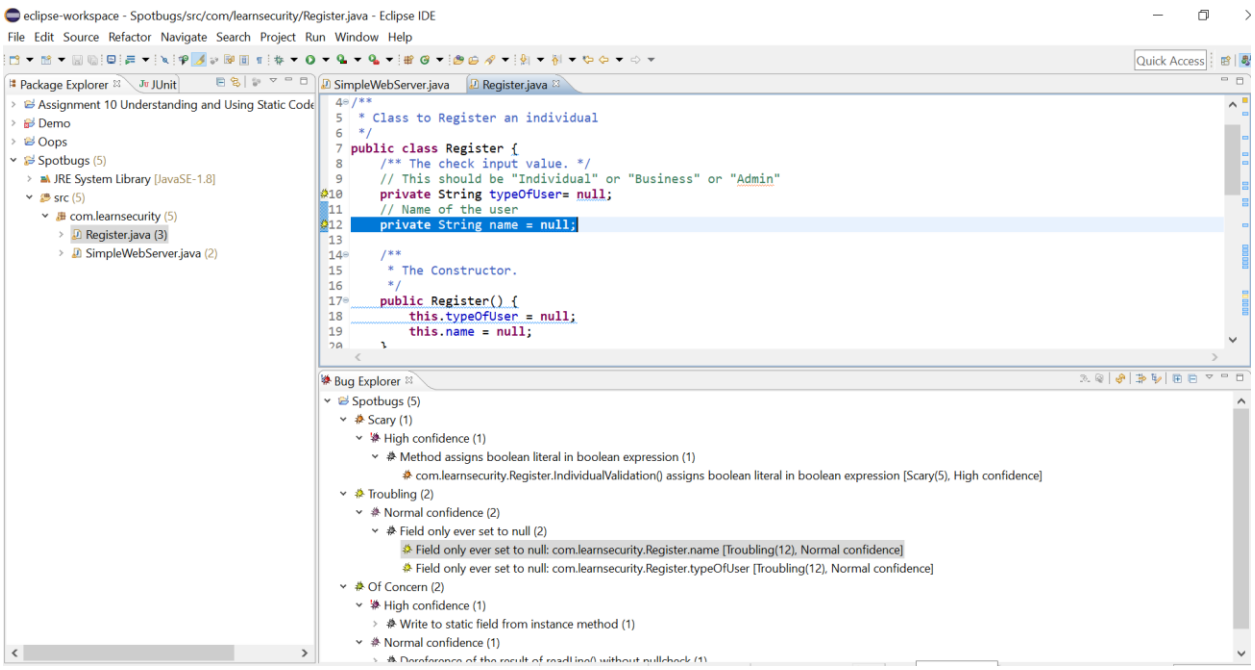
As mentioned in manual analysis, SpotBugs detect that the string variables typeOfUser and name are initialized to null in lines 10 and 12 as well as in the constructor.

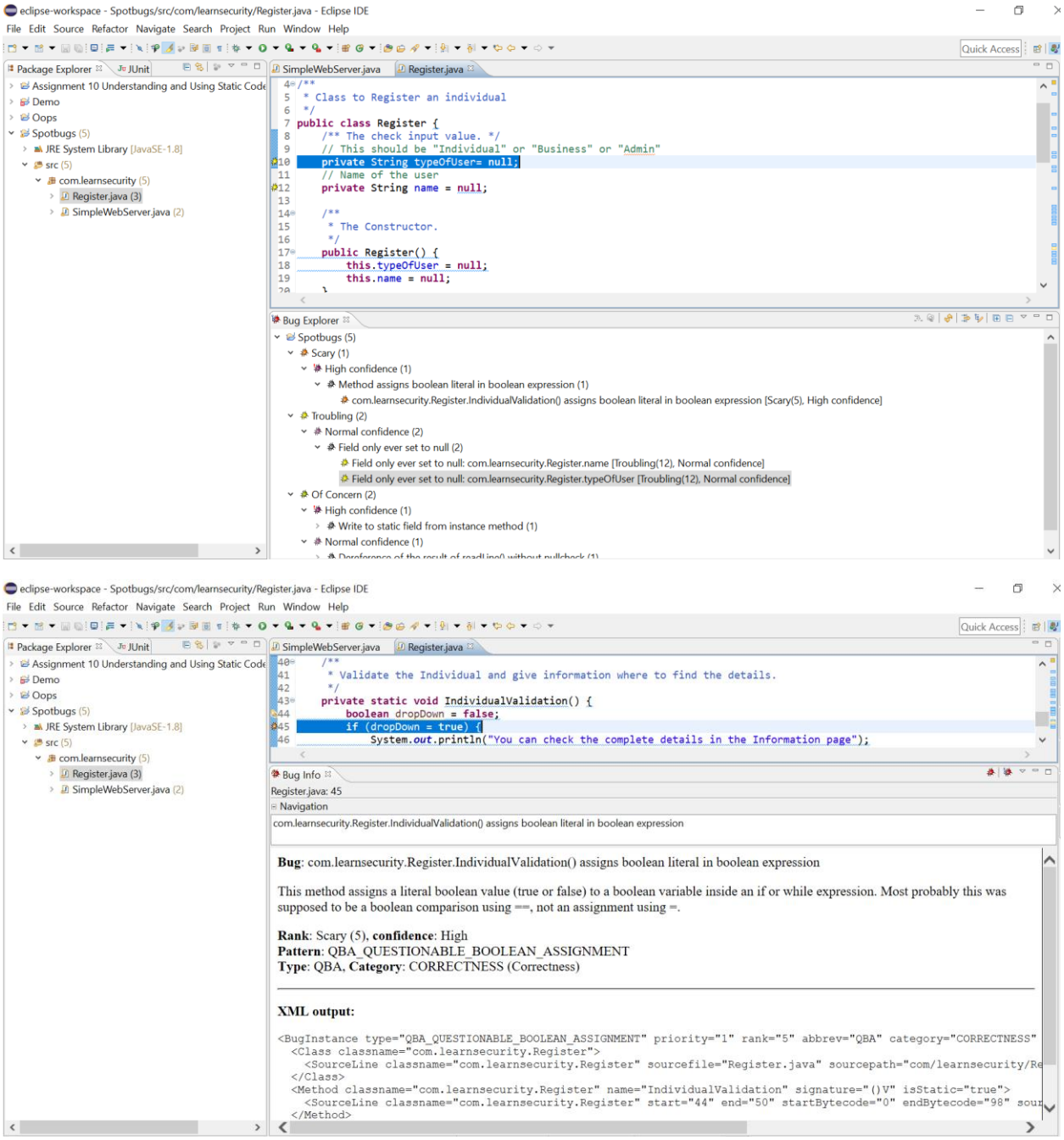


There is one more bug detected as Scary bug which was not noticed in manual analysis. Line 45 shows that dropDown Boolean variable is being assigned as true in the if condition instead of comparing the value of dropDown to true. This is a very dangerous bug. Whenever the if statement is used, the condition value should be comparing two values that should give true or false. If condition should not have any kind of variable assignment.

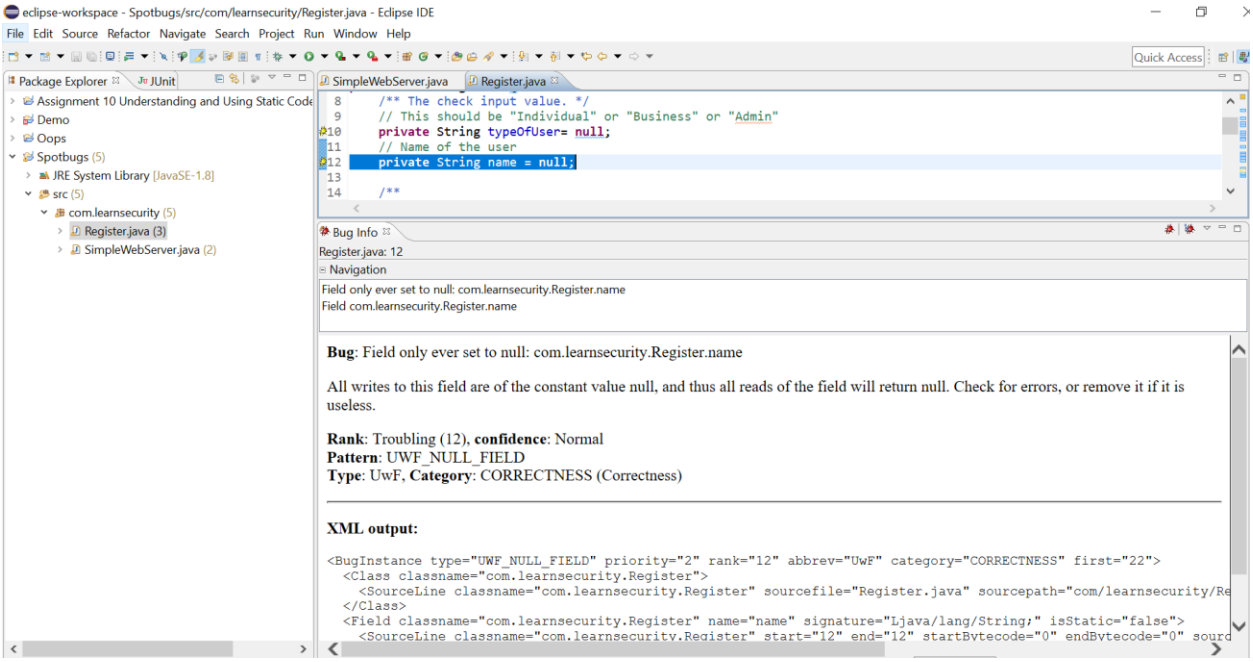


Also, the `boolean dropDown = false;` statement should not be placed inside the function `IndividualValidation()` because each time this function is called, dropDown variable is assigned as false and it never goes through the condition `if (dropDown = true)` It always goes to the else statement.

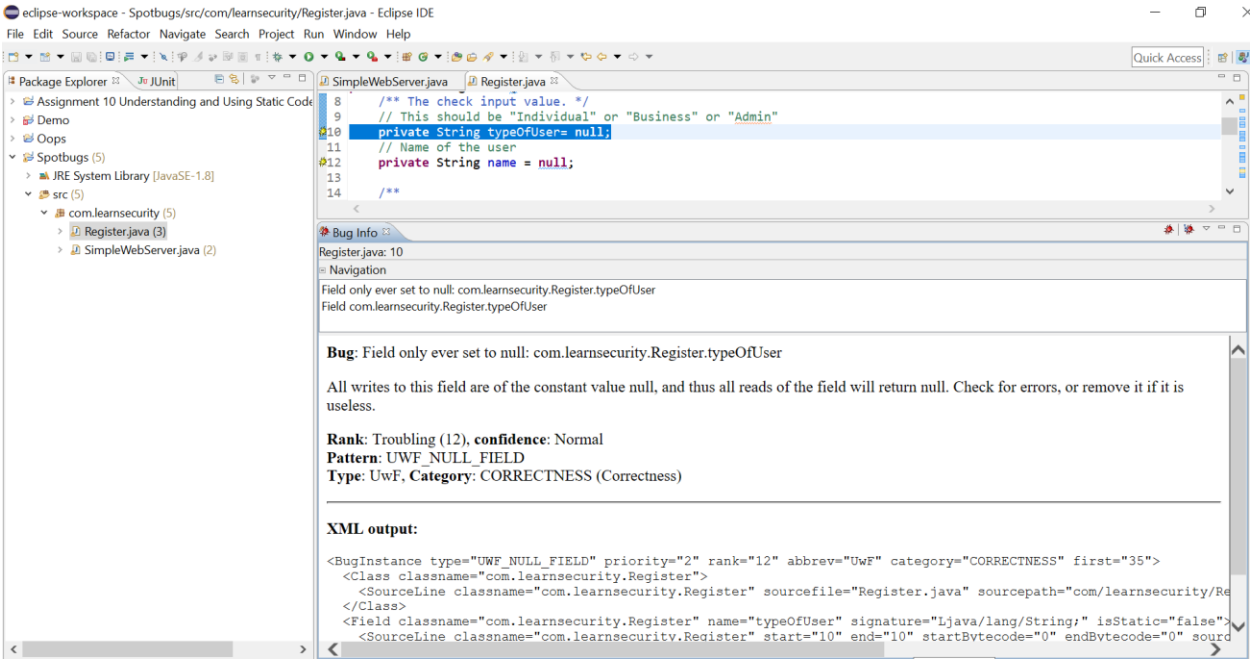




SpotBugs detects the bug that InputValidation() assigns Boolean literal in Boolean expression. This method assigns a literal boolean value (true or false) to a boolean variable insidean if or while expression. Most probably this was supposed to be a boolean comparison using==, not an assignment using =.

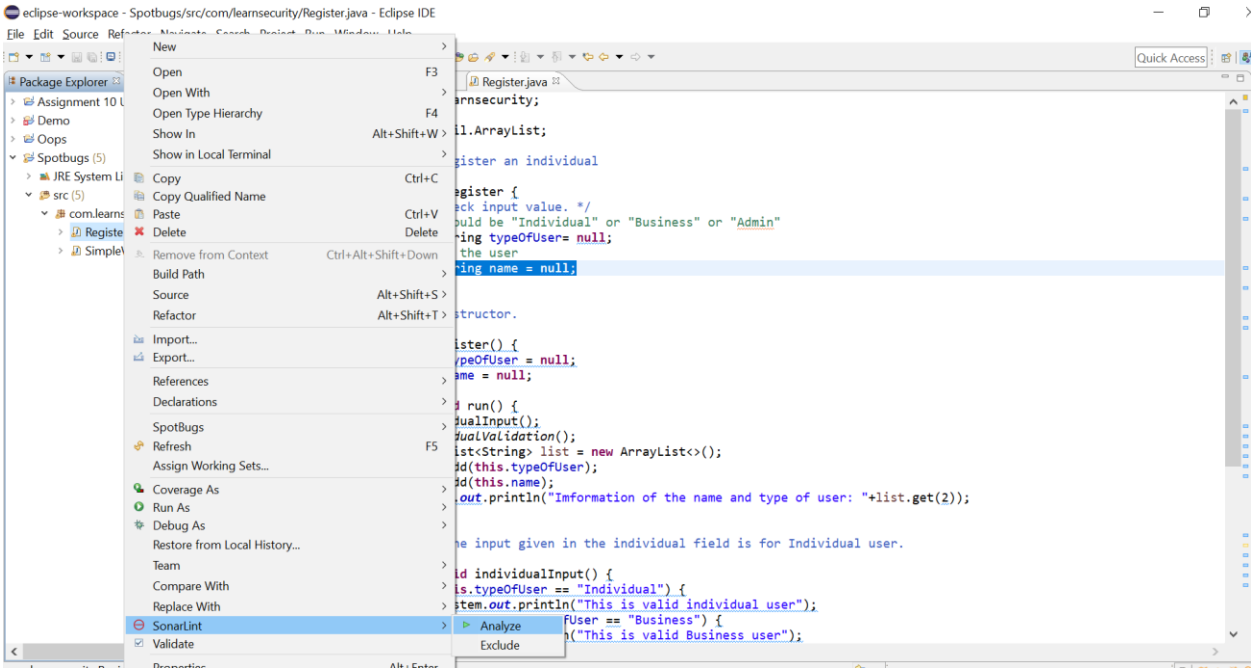


SpotBugs detects – Field only ever set to null. All writes to this field are of the constant value null, and thus all reads of the field will return null. Check for errors or remove it if it is useless.

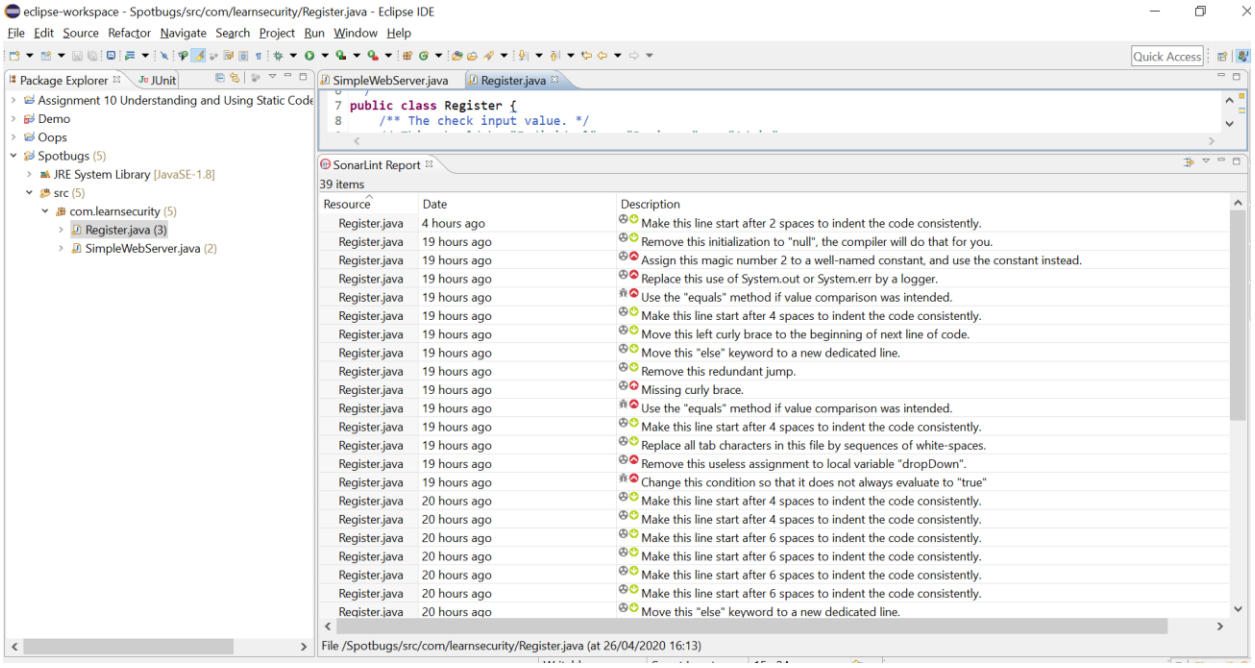


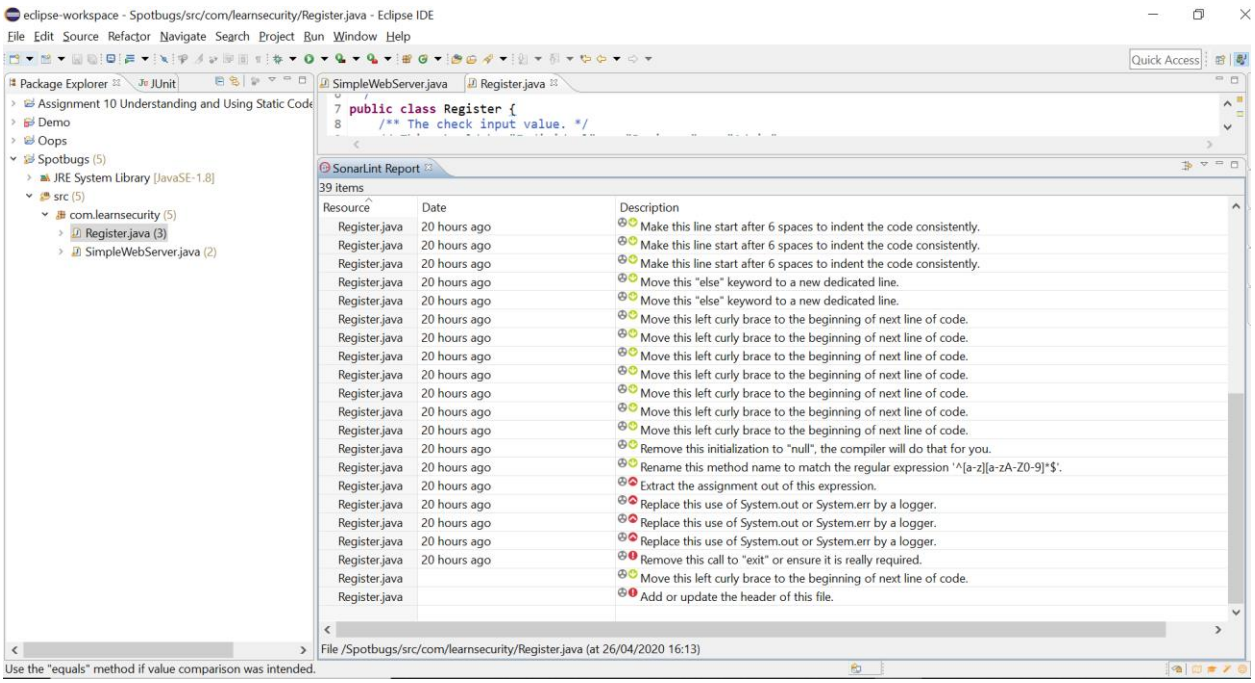
b) SonarLint:

We retrieve the bugs in the program by using the SonarLint tool next.



We retrieve new bugs in SonarLint as shown in the screenshot.

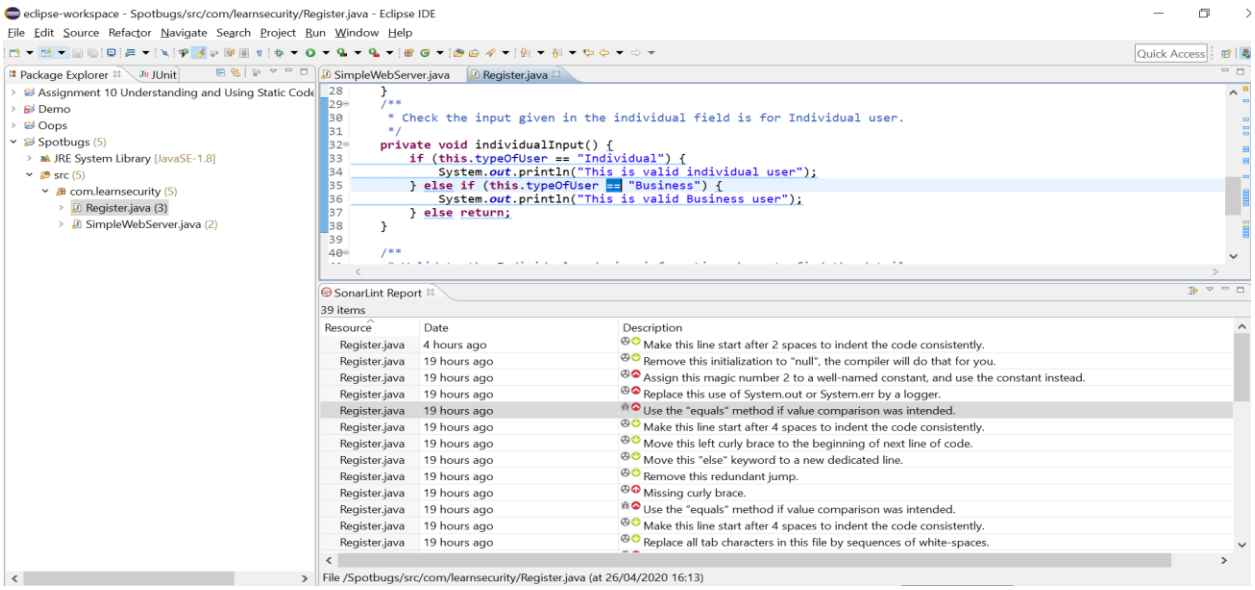




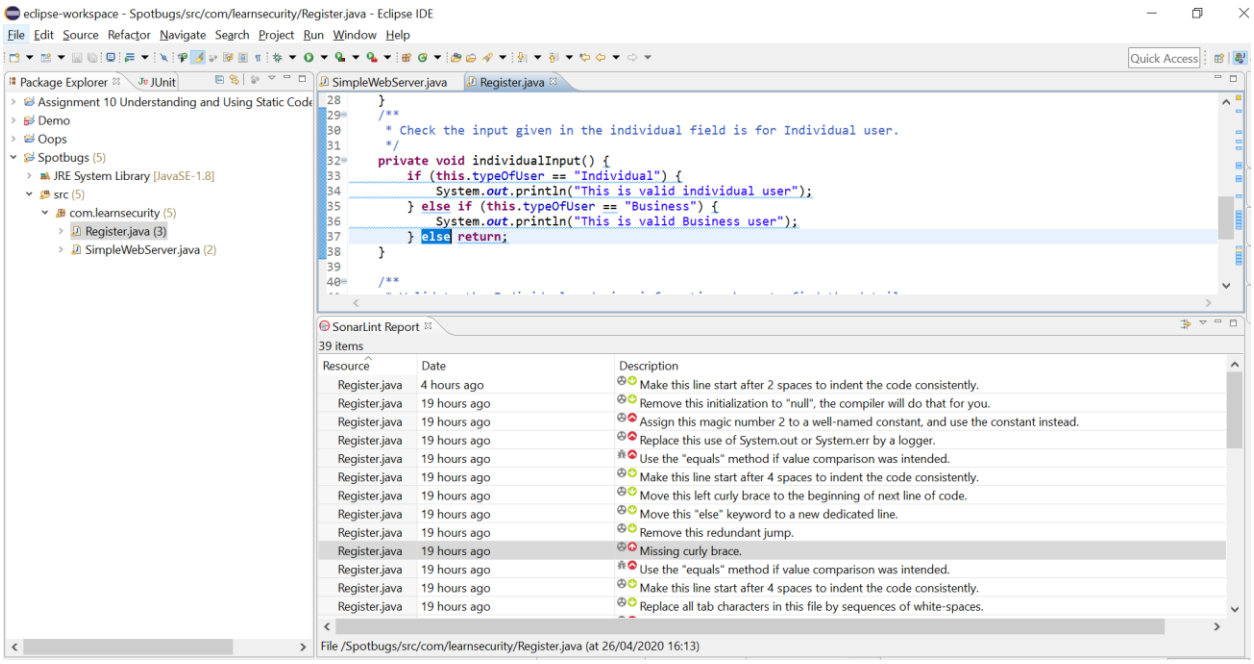
Fixes:

Some of the new bugs found in SonarLint and their fixes are as follows.

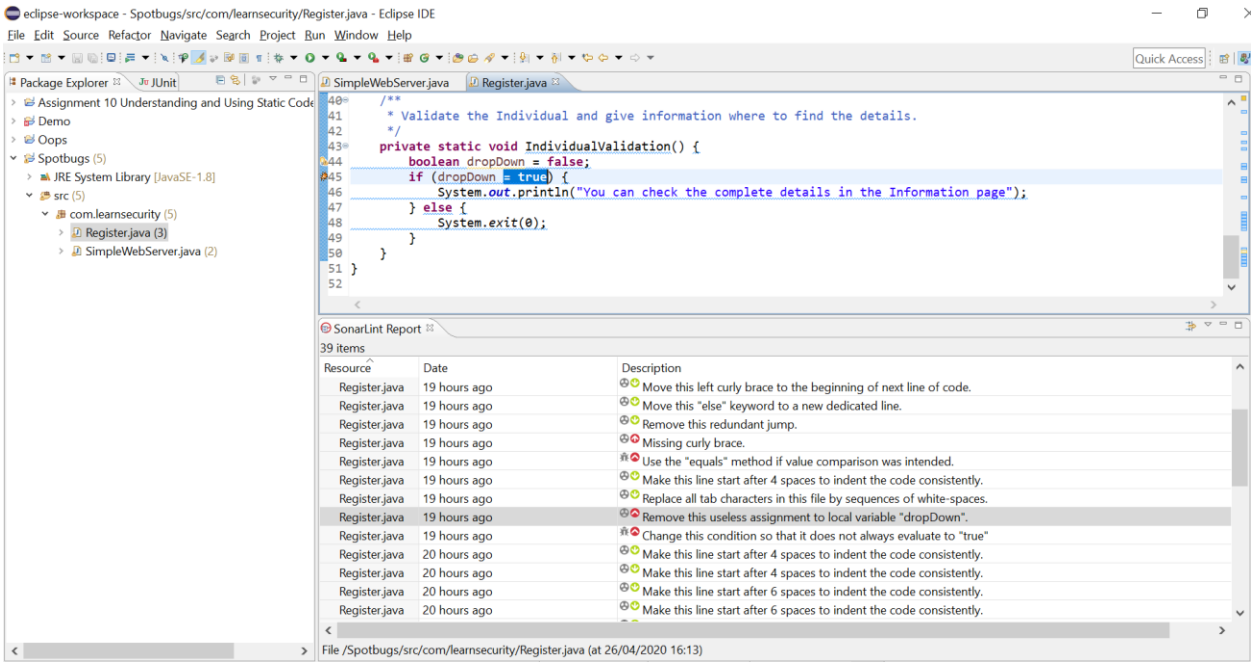
i) Use the equals method if value comparison was intended – This bug suggests that we use string.equals() method instead of using == while comparing two strings. This line `if (this.typeOfUser == "Business")` should be replaced by this line `if (this.typeOfUser.equals("Business"))`. If we use == for string comparison it may compare the address of the two strings instead of value. Similarly the line `if (this.typeOfUser == "Individual")` should be replaced by this line `if (this.typeOfUser.equals("Individual"))`.



ii) Missing curly braces - After if (condition) or else condition we always need to give curly braces and specify the boundaries or the block of statements for the action to be taken when the condition is true. Otherwise, only pone line after this condition is considered as action. We have to give `else { return}`



iii) Remove this useless assignment to local variable “dropDown” – We should not assign values to variables in the if condition. Condition should be compared and evaluate to true or false. Change this statement to `if (dropDown == true)`.



Also, place the `boolean dropDown = false;` statement before the individaulValidation() method so that dropDown does not always be false and it will never enter the if condition. It will always go to else for any logic or data provided to the program.

iv) Remove this call to "exit" or ensure it is really required – System.exit(0) function is not recommended to use in the program because it terminates the program abruptly. So we use return statement. Replace System.exit(0) with return

SonarLint has not detected that we always assign null to the string variables typeOfUser and name as found by SpotBugs. Field only ever set to null. All writes to this field are of the constant value null, and thus all reads of the field will return null. We have to assign other than null values to both these variables or just remove null assignment as the compiler will do it by itself. These two string variables may be initialized with some other values in the constructor.

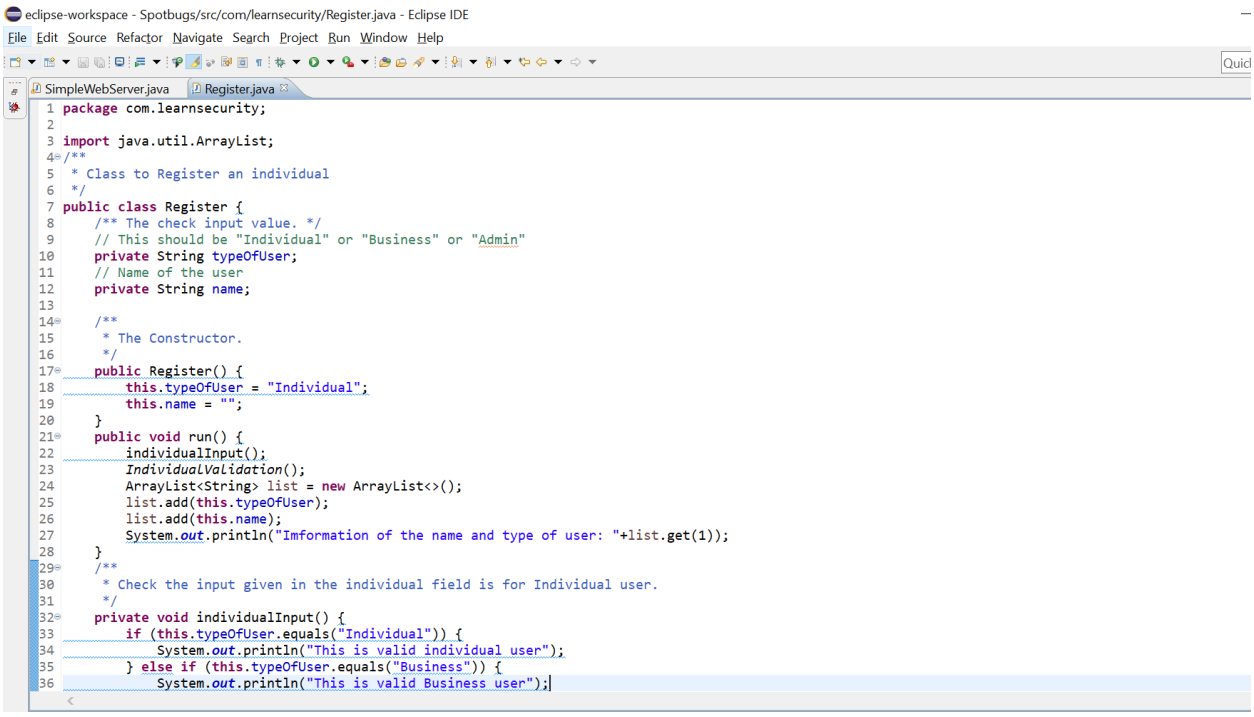
The bug detected in Manual analysis that line 27, where the ArrayList list tries to retrieve and element at index 2 which does not exist was not found both in SpotBugs and in SonarLint. The size of list is 2 but it is trying to retrieve the third element. This may cause ArrayOutOfBounds exception. This run time error was not detected by both the tools. We have to change the logic and check what we want as output.

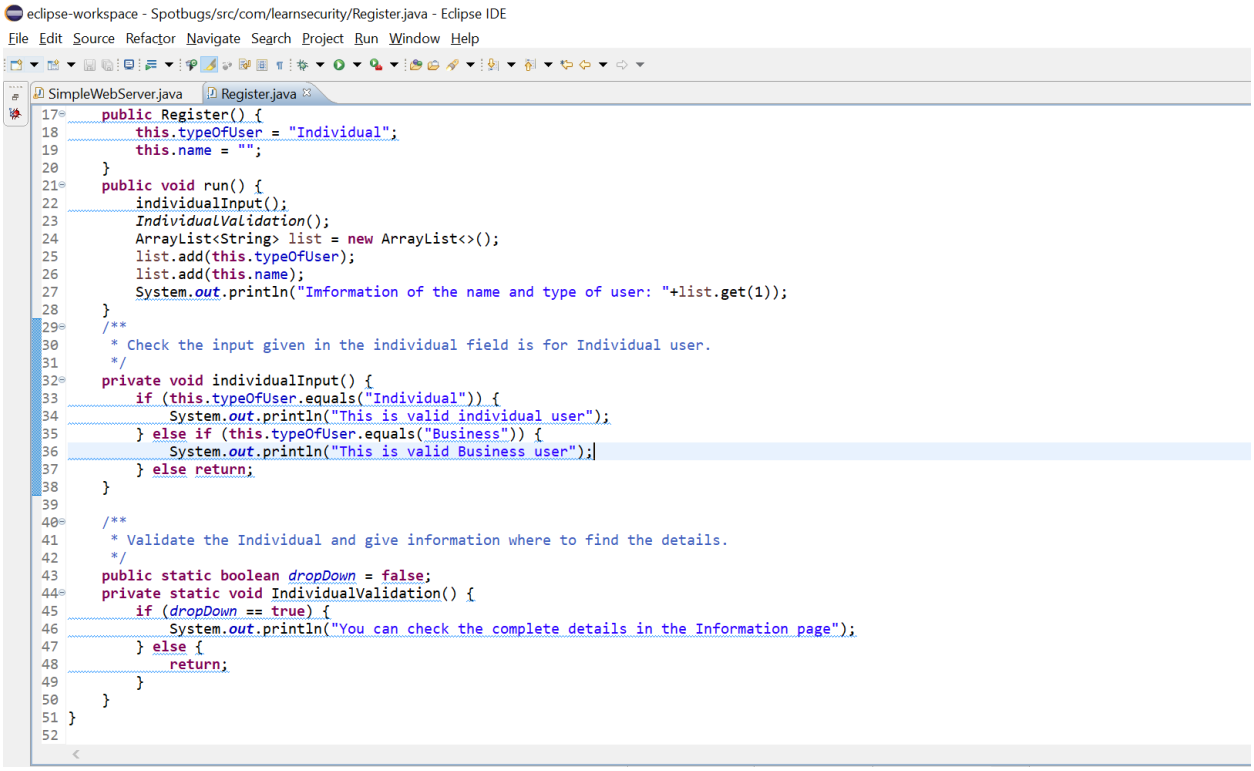
Fixed Code:

```
package com.learnsecurity;

import java.util.ArrayList;
/**
 * Class to Register an individual
 */
public class Register {
	/** The check input value. */
	// This should be "Individual" or "Business" or "Admin"
	private String typeOfUser;
	// Name of the user
	private String name;

    /**
     * The Constructor.
     */
    public Register() {
        this.typeOfUser = "Individual";
        this.name = "";
    }
    public void run() {
       individualInput();
       IndividualValidation();
       ArrayList<String> list = new ArrayList<>();
         list.add(this.typeOfUser);
         list.add(this.name);
         System.out.println("Imformation of the name and type of user: "+list.get(1));
    }
    /**
     * Check the input given in the individual field is for Individual user.
     */
    private void individualInput() {
```

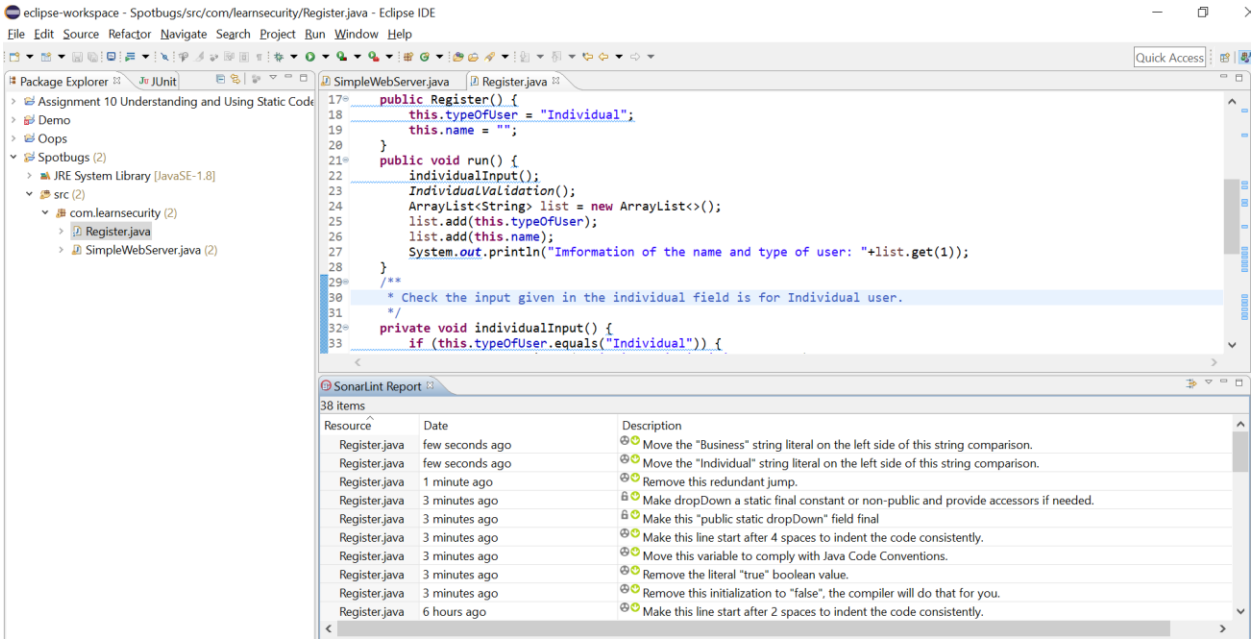
```
        if (this.typeOfUser.equals("Individual")) {
            System.out.println("This is valid individual user");
        } else if (this.typeOfUser.equals("Business")) {
            System.out.println("This is valid Business user");
        } else {
             return;
             }
    }

    /**
     * Validate the Individual and give information where to find the details.
     */
    public static boolean dropDown = false;
    private static void IndividualValidation() {
        if (dropDown == true) {
            System.out.println("You can check the complete details in the Information
page");
        } else {
            return;
        }
    }
}
```

Spotbugs result for Fixed code:

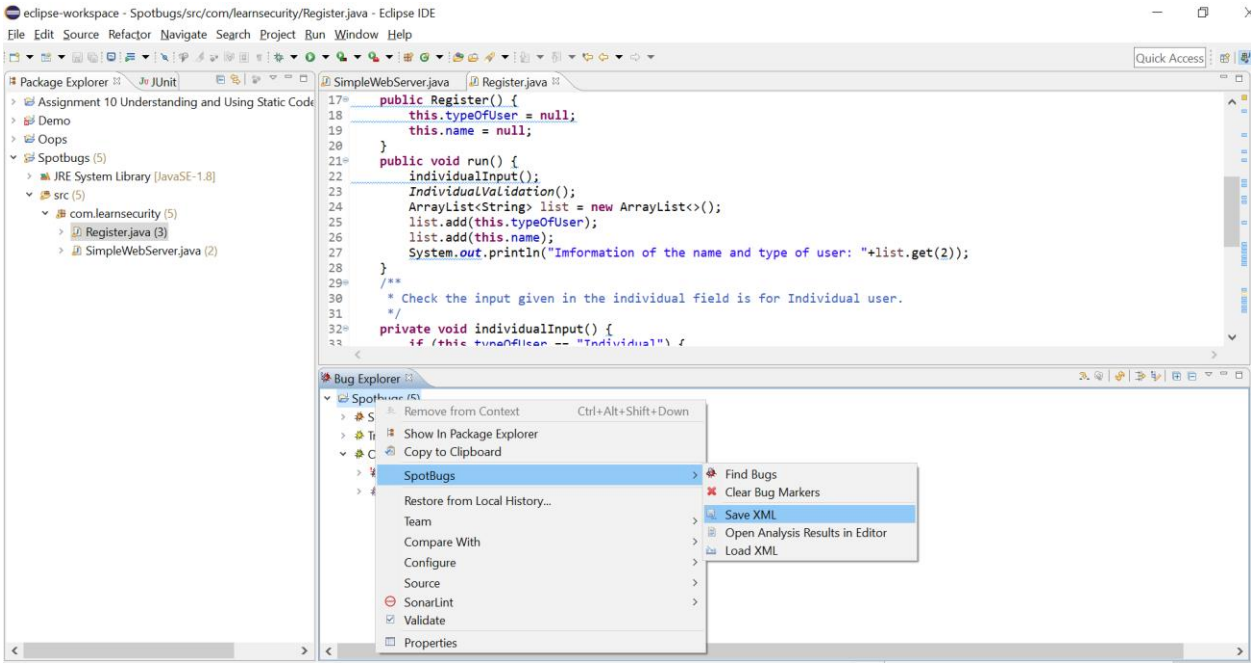




SonarLint Result for fixed code:



XML Report:



XML Preview:

```
<?xml version="1.0" encoding="UTF-8"?>

<BugCollection version="4.0.1" sequence="40" timestamp="1587937796534" analysisTimestamp="1587937794970" release="">
  <Project projectName="Spotbugs">
    <Jar>C:\Users\gouta\eclipse-workspace\Spotbugs\bin\com\learnsecurity\Register.class</Jar>
    <AuxClasspathEntry>C:\Users\gouta\eclipse-workspace\Spotbugs\bin</AuxClasspathEntry>
    <AuxClasspathEntry>C:\Program Files\Java\jre1.8.0_251\lib\rt.jar</AuxClasspathEntry>
    <AuxClasspathEntry>C:\Program Files\Java\jre1.8.0_251\lib\jce.jar</AuxClasspathEntry>
    <SrcDir>C:\Users\gouta\eclipse-workspace\Spotbugs\src</SrcDir>
  </Project>
  <BugInstance type="SBSC_USE_STRINGBUFFER_CONCATENATION" priority="2" rank="18" abbrev="SBSC" category="PERFORMANCE" first="14">
    <Class classname="com.learnsecurity.Analysis">
      <SourceLine classname="com.learnsecurity.Analysis" sourcefile="Analysis.java" sourcepath="com/learnsecurity/Analysis.java"/>
    </Class>
    <Method classname="com.learnsecurity.Analysis" name="test" signature="()V" isStatic="false">
      <SourceLine classname="com.learnsecurity.Analysis" start="12" end="22" startBytecode="0" endBytecode="193" sourcefile="Analysis.java" sourcepath="com/learnsec
    </Method>
    <SourceLine classname="com.learnsecurity.Analysis" start="17" end="17" startBytecode="38" endBytecode="38" sourcefile="Analysis.java" sourcepath="com/learnsecur
  </BugInstance>
  <BugInstance type="UUF_UNUSED_FIELD" priority="2" rank="18" abbrev="UuF" category="PERFORMANCE" first="14">
    <Class classname="com.learnsecurity.Analysis">
      <SourceLine classname="com.learnsecurity.Analysis" sourcefile="Analysis.java" sourcepath="com/learnsecurity/Analysis.java"/>
    </Class>
    <Field classname="com.learnsecurity.Analysis" name="abc" signature="I" isStatic="false">
      <SourceLine classname="com.learnsecurity.Analysis" start="6" end="6" startBytecode="0" endBytecode="0" sourcefile="Analysis.java" sourcepath="com/learnsecurit
    </Field>
  </BugInstance>
  <BugInstance type="CD_CIRCULAR_DEPENDENCY" priority="2" rank="17" abbrev="CD" category="STYLE" first="20" last="24" removedByChange="true">
    <Class classname="com.learnsecurity.Client">
      <SourceLine classname="com.learnsecurity.Client" start="51" end="297" sourcefile="Client.java" sourcepath="com/learnsecurity/Client.java"/>
    </Class>
    <Class classname="com.learnsecurity.Client$2">
      <SourceLine classname="com.learnsecurity.Client$2" sourcefile="Client.java" sourcepath="com/learnsecurity/Client.java"/>
    </Class>
  </BugInstance>
  <BugInstance type="CD_CIRCULAR_DEPENDENCY" priority="2" rank="17" abbrev="CD" category="STYLE" first="20" last="24" removedByChange="true">
    <Class classname="com.learnsecurity.Client">
      <SourceLine classname="com.learnsecurity.Client" start="51" end="297" sourcefile="Client.java" sourcepath="com/learnsecurity/Client.java"/>
    </Class>
```

spotbugRegisterReport.xml

References:

https://stackoverflow.com/questions/9376183/closing-an-buffer-reader-is-compulsory?lq=1

https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html

https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuffer.html

https://stackshare.io/stackups/findbugs-vs-sonarlint

https://www.sonarlint.org/