

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.

```

45o 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

The screenshot shows the Eclipse Marketplace interface. At the top, there is a search bar with the text "spotbugs" and a "Go" button. Below the search bar, a card displays the "SpotBugs Eclipse plugin 3.1.5.r201806132012-cbbf0a5". The card includes a magnifying glass icon over a red bug, the plugin name, a brief description mentioning "400 bug patterns", the developer "SpotBugs Team, LGPL", and tags like "java quality bugs analysis defects ...". It also shows a rating of 630 stars and 109K installs. An "Install" button is located on the right side of the card. Below the card, a link says "3 matches. Browse for more solutions."

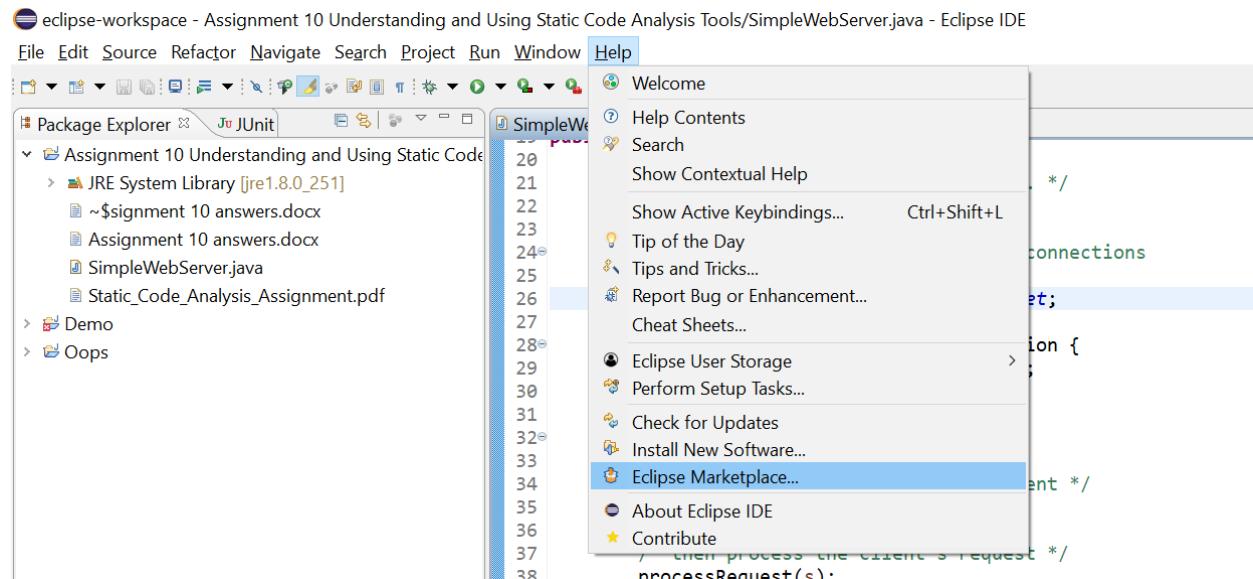
b) SonarLint 5.0

The screenshot shows the Eclipse Marketplace interface. At the top, there is a search bar with the text "sonarlint" and a "Go" button. Below the search bar, a card displays "SonarLint 5.0". The card includes a magnifying glass icon over a blue document, the plugin name, a brief description mentioning "Java, JavaScript, PHP, Python and HTML", the developer "SonarSource S.A, LGPL", and tags like "java PHP javascript Python static analysis ...". It also shows a rating of 1791 stars and 628K installs. An "Install" button is located on the right side of the card.

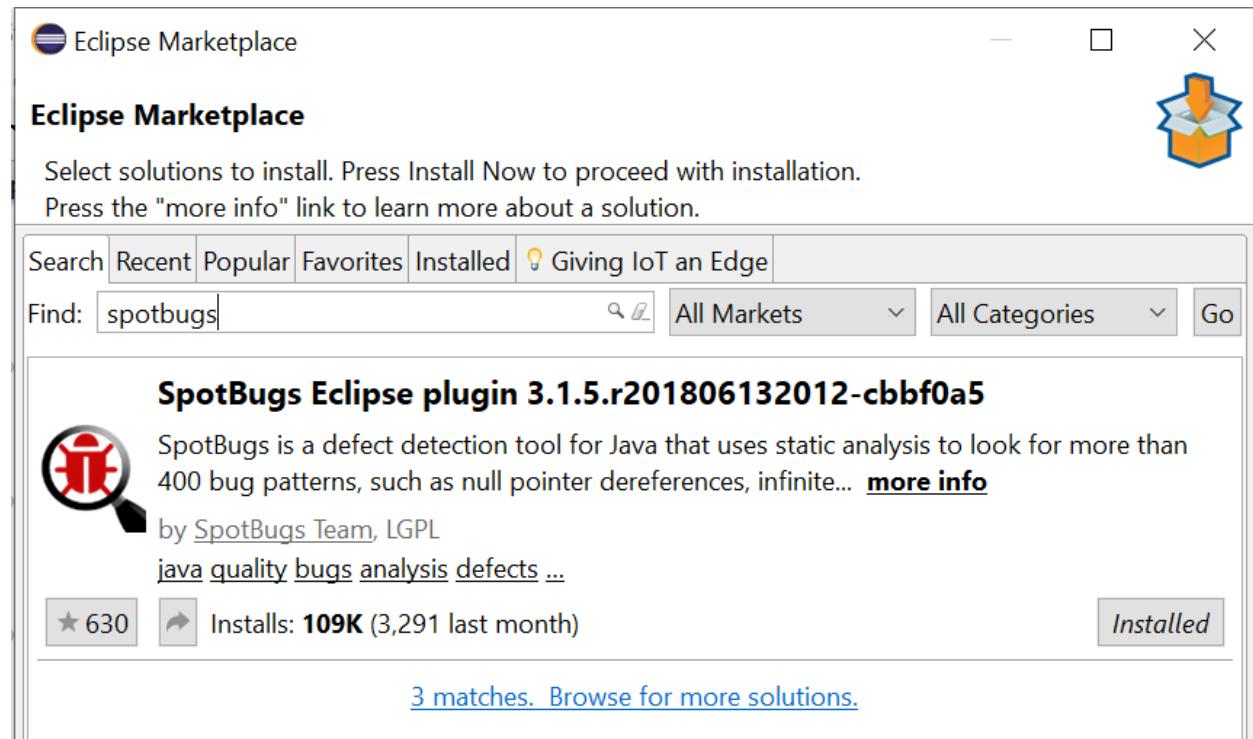
Tool Invocation Process:

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

Help => Eclipse Marketplace => SpotBugs



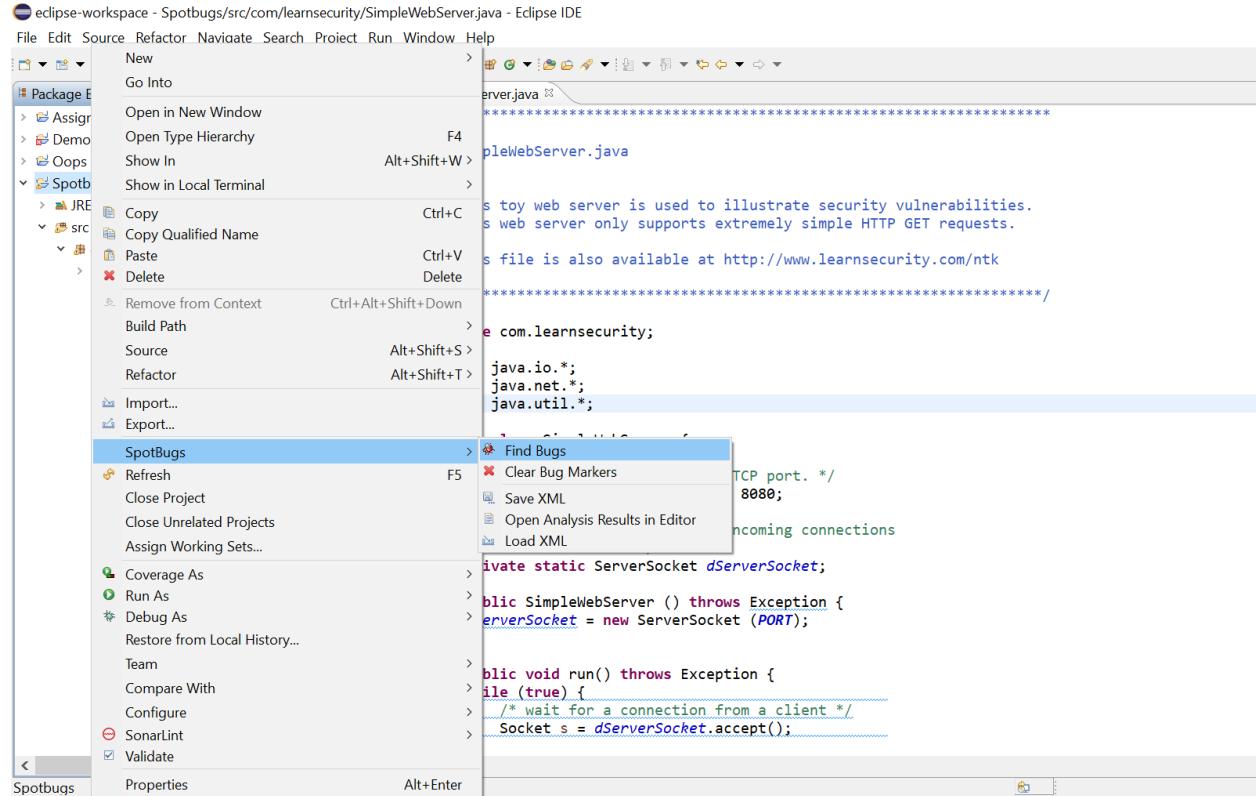
The screenshot shows the Eclipse IDE interface. The menu bar at the top has 'Help' selected. A dropdown menu is open under 'Help' with various options like 'Welcome', 'Help Contents', 'Search', and 'Eclipse Marketplace...'. The 'Eclipse Marketplace...' option is highlighted with a blue selection bar. Below this, the Java code editor shows a snippet of Java code related to a SimpleWebServer.



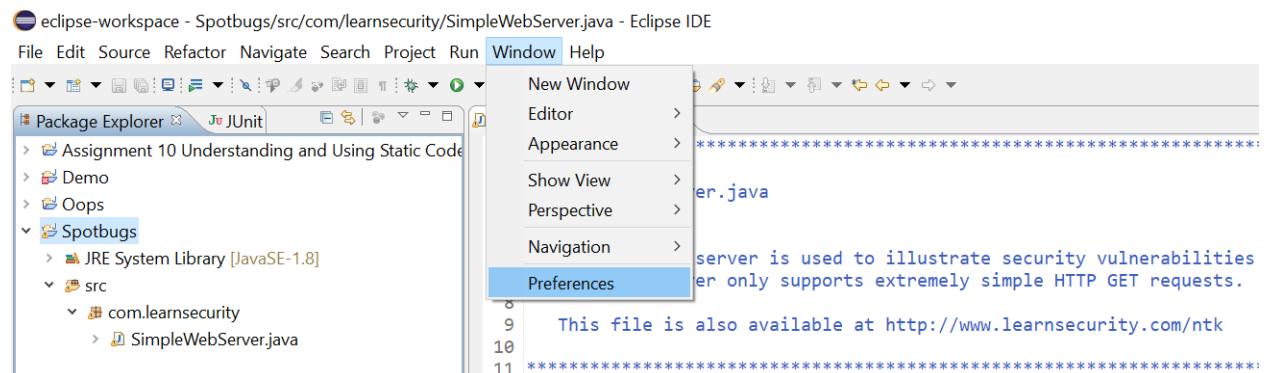
The screenshot shows the 'Eclipse Marketplace' window. At the top, it says 'Select solutions to install. Press Install Now to proceed with installation.' Below that is a search bar with 'spotbugs' typed into it. The main area displays a card for the 'SpotBugs Eclipse plugin 3.1.5.r201806132012-cbbf0a5'. The card includes a magnifying glass icon with a red bug, the plugin name, a description mentioning over 400 bug patterns, the developer 'SpotBugs Team, LGPL', and tags like 'java', 'quality', 'bugs', 'analysis', and 'defects'. It also shows a star rating of 630, 109K installs (3,291 last month), and an 'Installed' button.

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

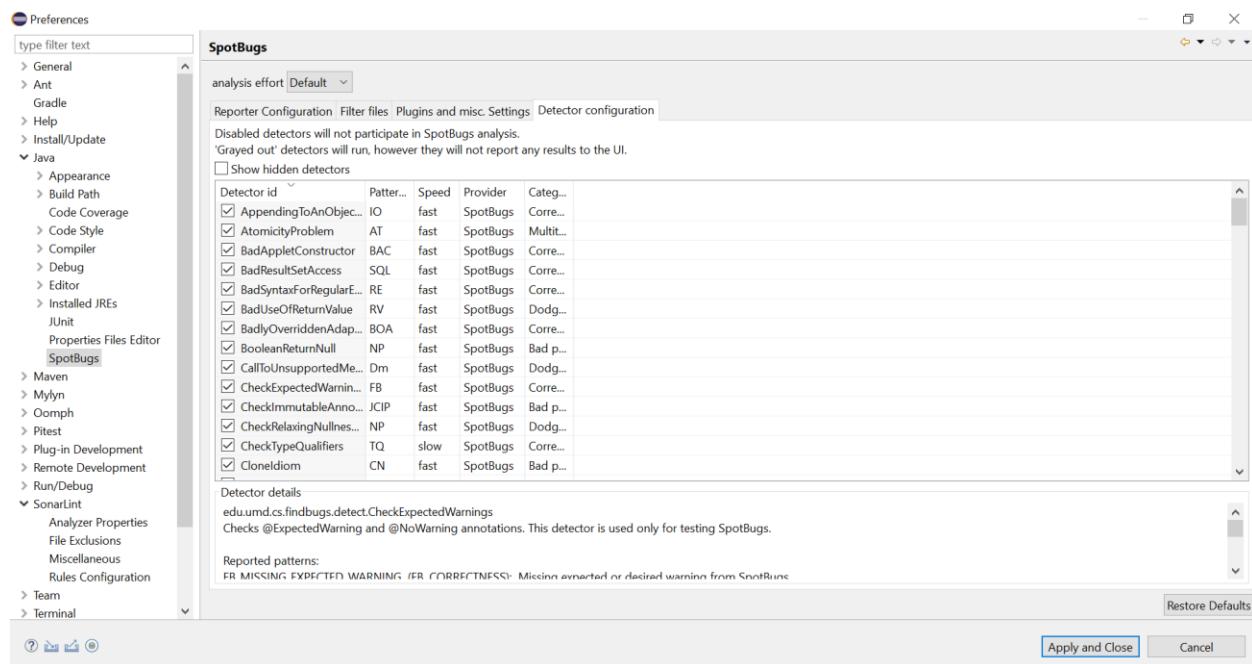
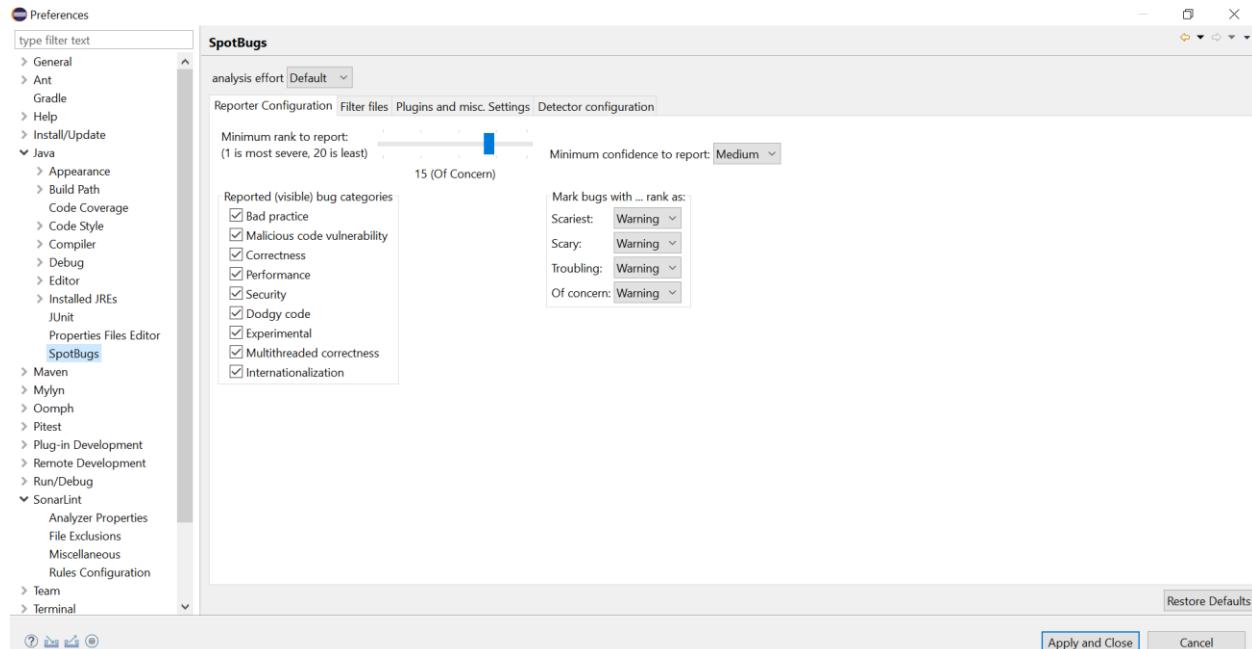
We take the SimpleWebServer.java file and right click on the project in which it is present. We can see SpotBugs option. If we click that we get Find Bugs. By clicking on Find Bugs we can identify all the bugs in the program. But before doing that, in eclipse IDE, we navigate to Window => Preferences.



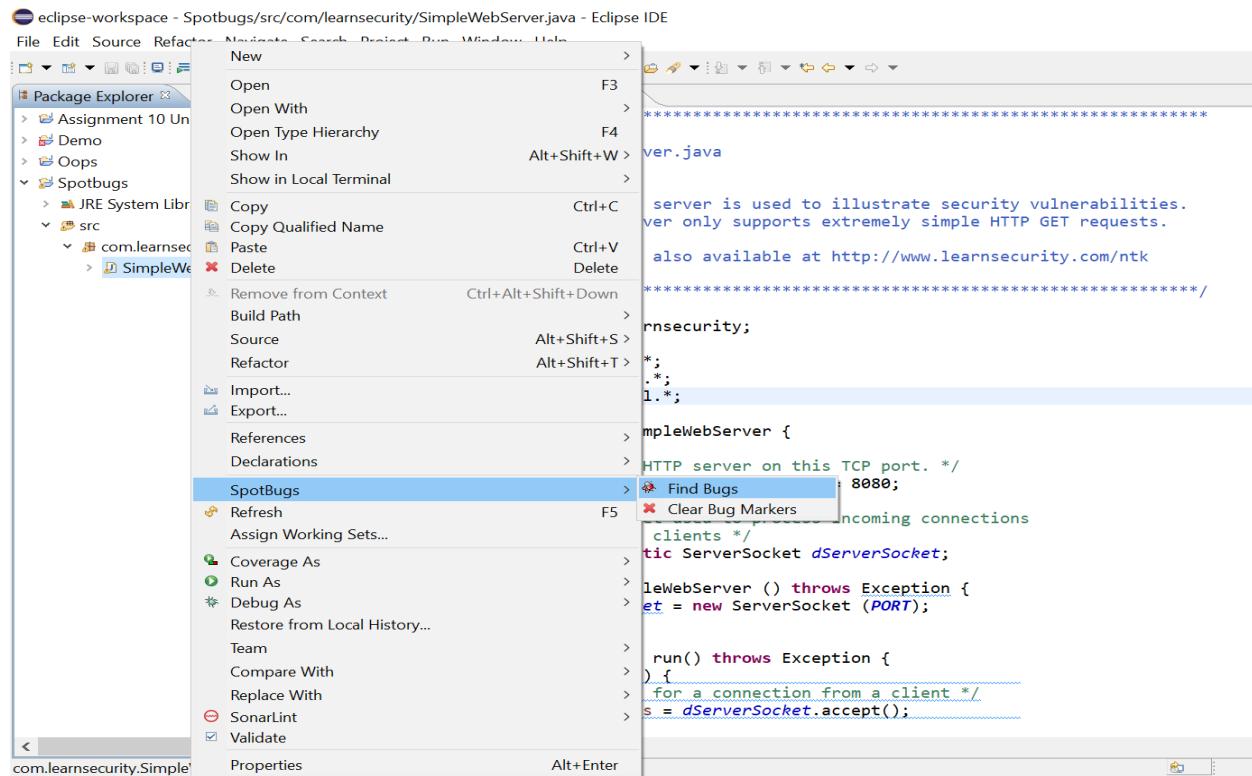
Information about tools settings and plugins used:



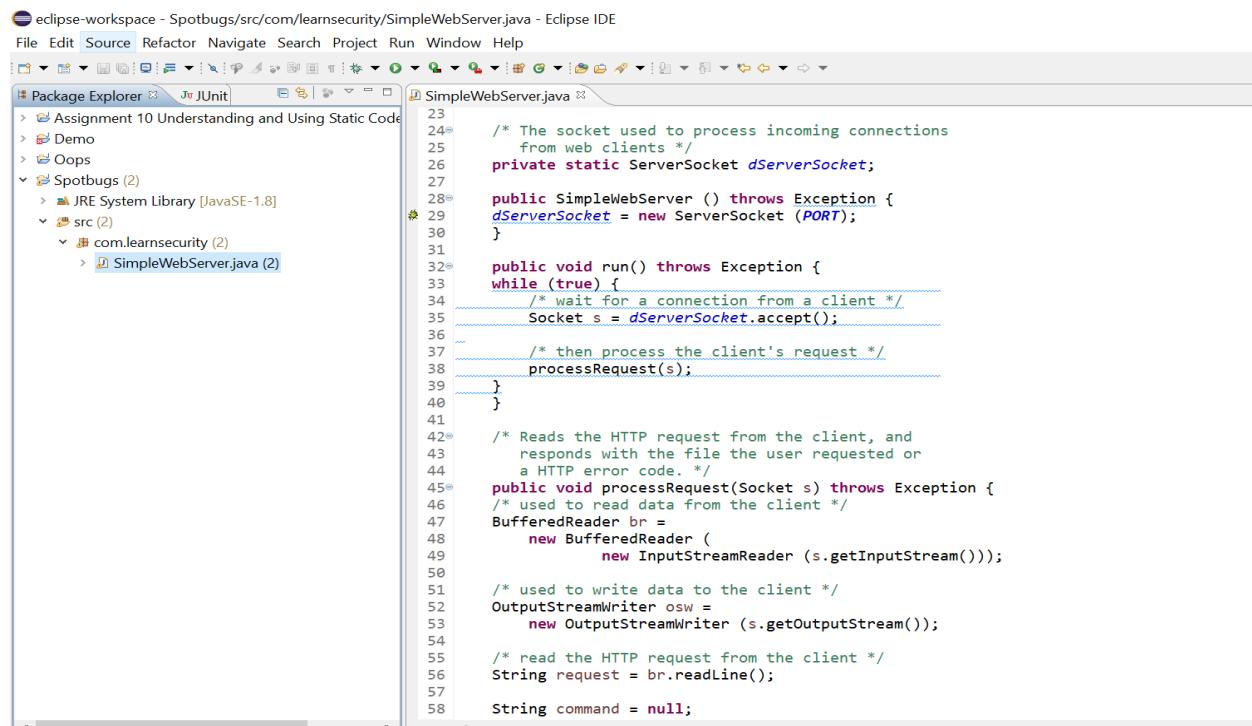
We navigate to Java=> SpotBugs in Preferences and make sure we mark all the check boxes for Reported Bug categories in Reported Configuration and Detector Configuration. We want to detect all kinds of bugs found in the program using SpotBugs.



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.

The screenshot shows the Eclipse IDE interface with the following details:

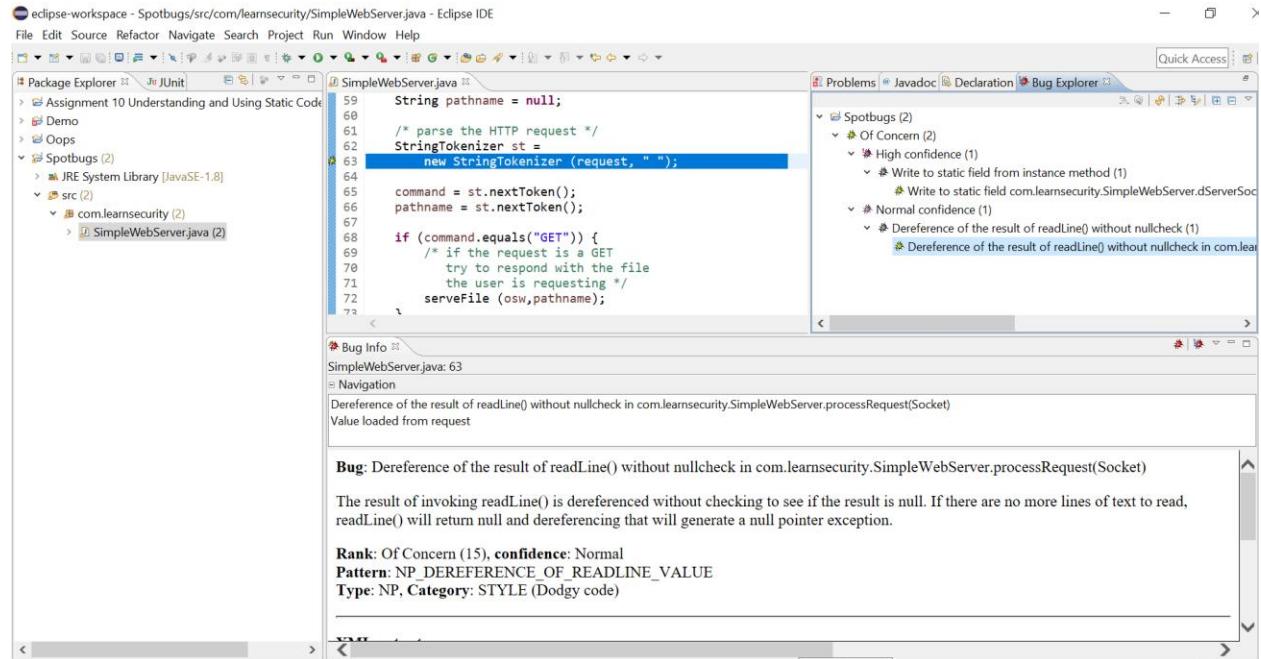
- File Explorer:** Shows a project structure with packages like Assignment 10 Understanding and Using Static Code, Demo, Oops, and Spotbugs.
- Code Editor:** Displays the file `SimpleWebServer.java` containing Java code. Line 29 is highlighted with a blue selection bar.
- Bug Explorer:** Shows a list of bugs under the "Spotbugs" category. One bug is listed: "Write to static field com.learnsecurity.SimpleWebServer.dServerSocket from instance method new com.learnsecurity.SimpleWebServer()".
- Bug Info View:** Provides detailed information about the bug, including the rank (Of Concern), confidence (High), pattern (ST_WRITE_TO_STATIC_FROM_INSTANCE_METHOD), and type (ST, Category: STYLE (Dodgy code)).

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

The screenshot shows the Eclipse IDE interface with the following details:

- File Explorer:** Shows a project structure with packages like Assignment 10 Understanding and Using Static Code, Demo, Oops, and Spotbugs.
- Code Editor:** Displays the file `SimpleWebServer.java` containing Java code. Line 63 is highlighted with a red selection bar.

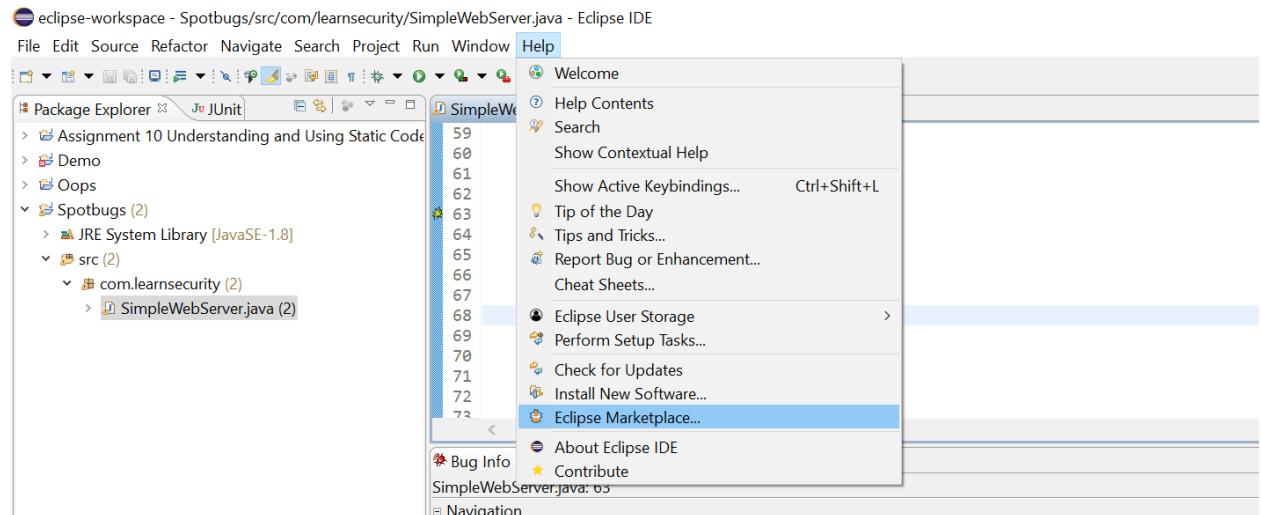
In line 63, there is a dereference of the output of `readLine()` without nullcheck. The result of invoking `readLine()` is dereferenced without checking to see if the result is null. If there are no more lines of text to read, `readLine()` will return null and dereferencing that will generate a null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



b) SonarLint

In Eclipse IDE, navigate to Help and find Eclipse 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.

The screenshot shows the Eclipse Marketplace interface. At the top, there is a search bar with the text "sonarlint" and a "Go" button. Below the search bar, the title "SonarLint 5.0" is displayed. A brief description follows: "SonarLint is an IDE extension that helps you detect and fix quality issues as you write code in Java, JavaScript, PHP, Python and HTML." It is developed by "SonarSource S.A, LGPL" and provides "java PHP javascript Python static analysis ...". The extension has 1791 installs and was last updated 16,551 times in the last month. A large "Installed" button is visible on the right side of the card.

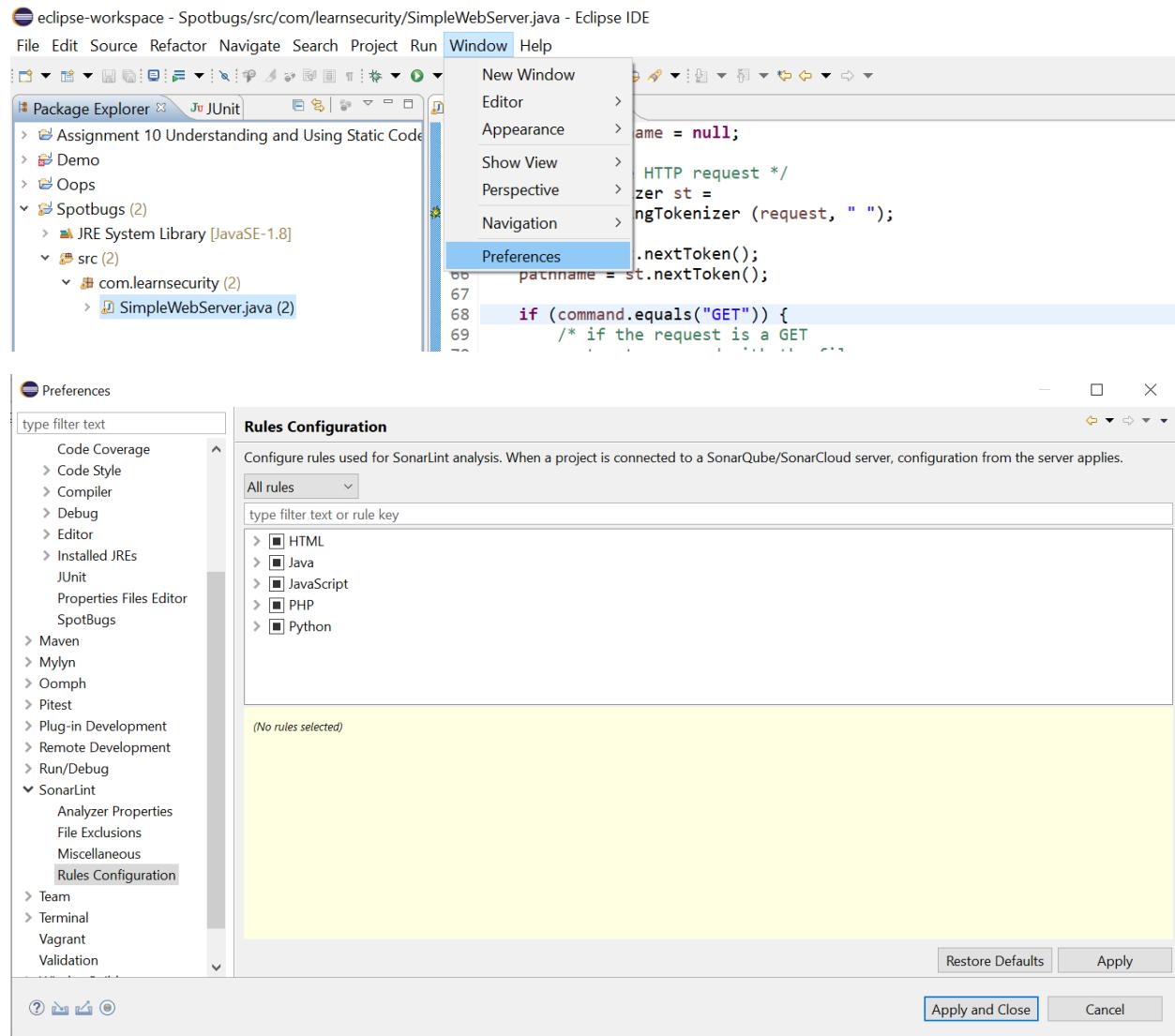
We take the SimpleWebServer.java file and right click on the project in which it is present. We can see SonarLint option. If we click that we get two options - Analyze, Exclude. By clicking on Analyze we can identify all the bugs in the program. But before doing that, in eclipse IDE, we navigate to Window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

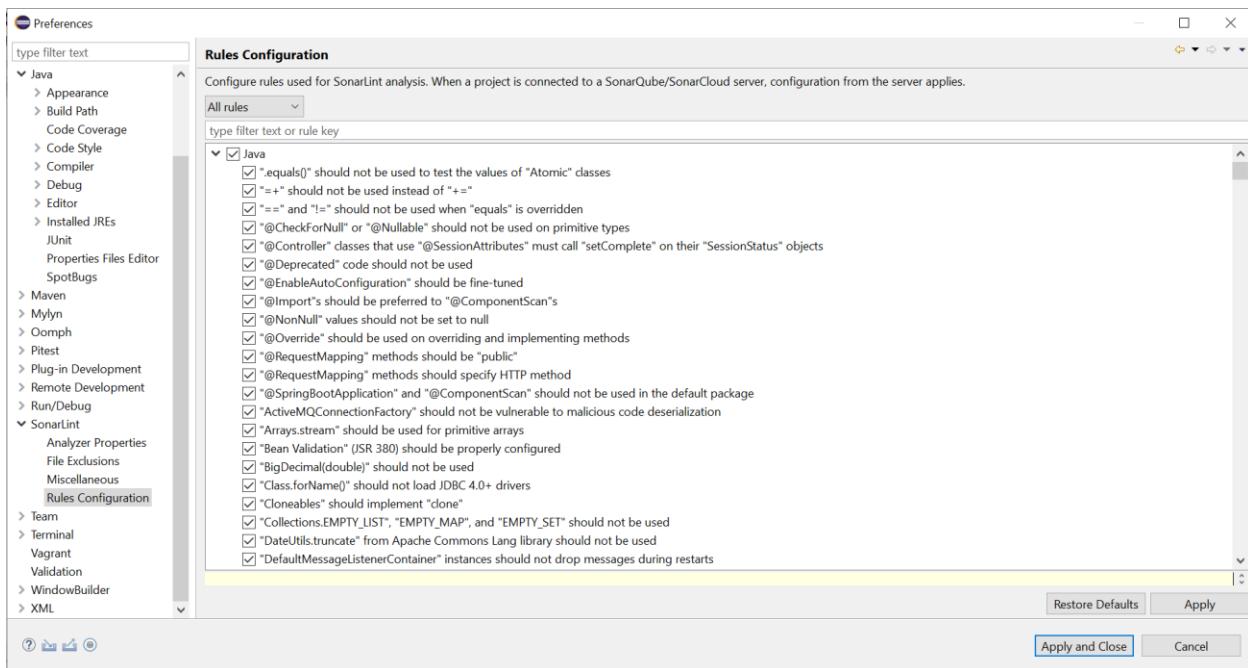
The screenshot shows the Eclipse IDE interface with the "SimpleWebServer" project selected in the Package Explorer. A context menu is open over a line of code in the editor. The "New" submenu is expanded, showing various options like Open, Open With, and SonarLint. The "SonarLint" option is highlighted with a blue selection bar. The status bar at the bottom shows "com.learnsecurity.SimpleWebServer".

Information about tools settings and plugins used:

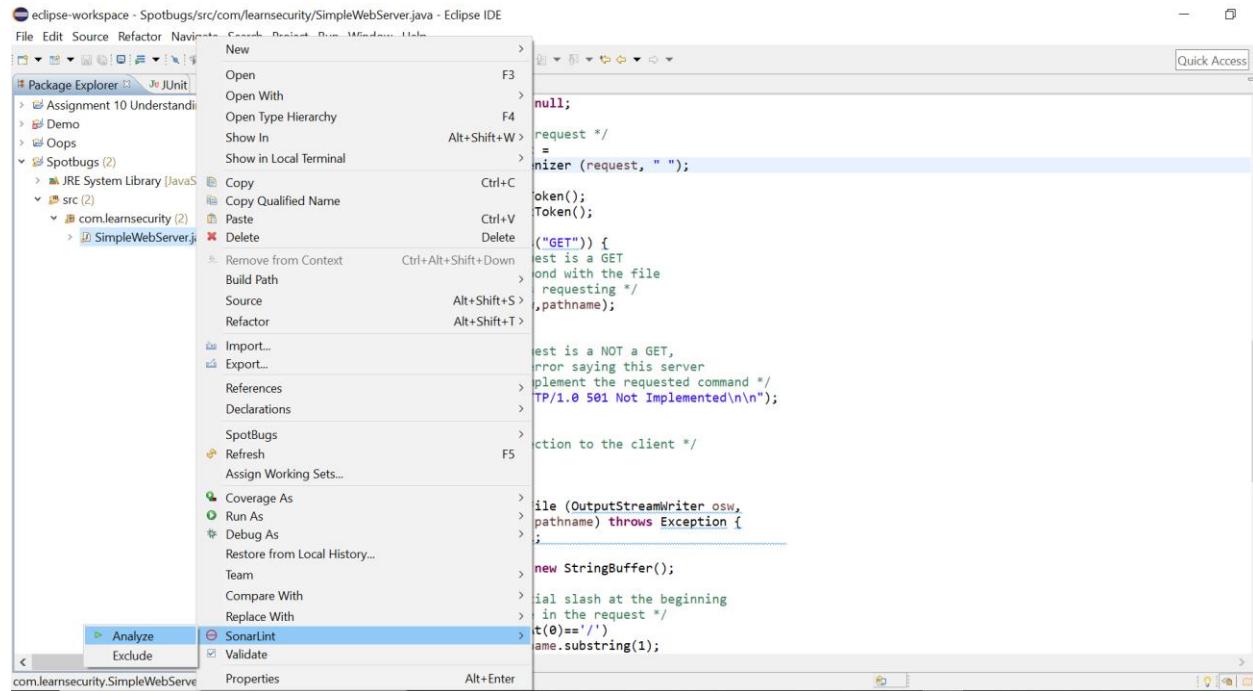
In eclipse IDE, we navigate to Window => Preferences.

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.

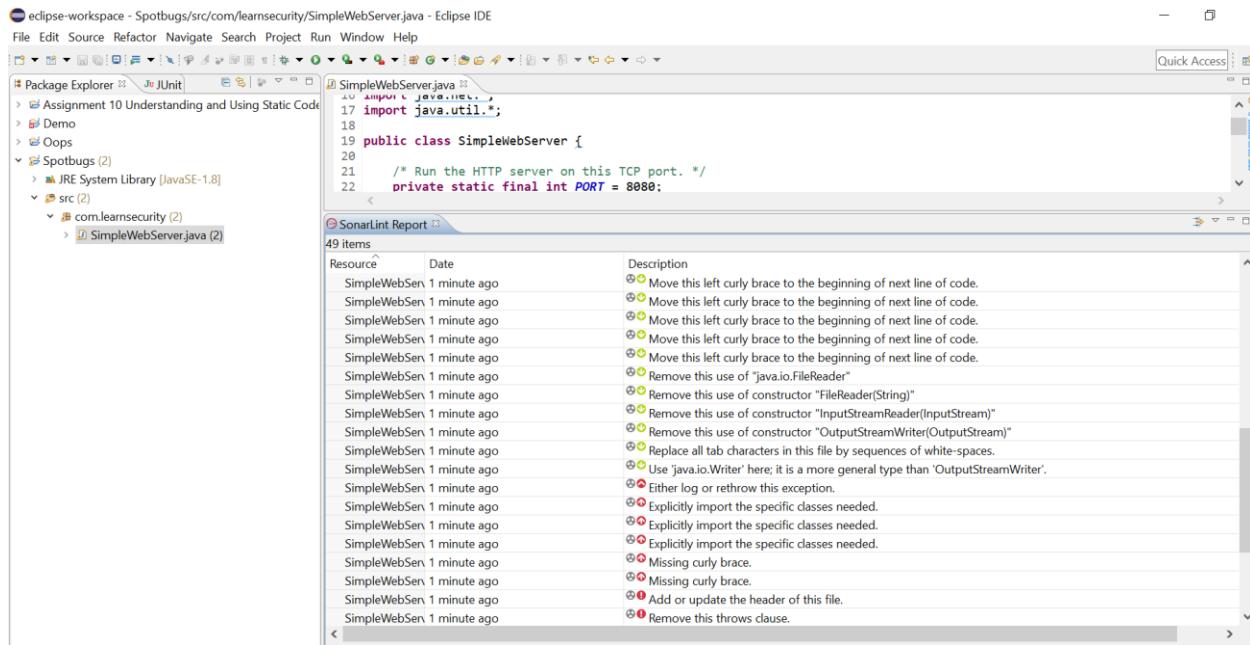




When all the rules are configured during SonarLint analysis, we will be able to detect all kinds of bugs in the entire program. We make this check so that we don't implement leaving bugs in the program.



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 one 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 "StringBuilder"* – 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

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);
73     }

```

SonarLint Report

49 items

| Resource | Date | Description |
|-----------------|--------------|---|
| SimpleWebSer... | 1 minute ago | Explicitly import the specific classes needed. |
| SimpleWebSer... | 1 minute ago | Missing curly brace. |
| SimpleWebSer... | 1 minute ago | Missing curly brace. |
| SimpleWebSer... | 1 minute ago | Add or update the header of this file. |
| SimpleWebSer... | 1 minute ago | Remove this throws clause. |
| SimpleWebSer... | | Move the array designator from the variable to the type. |
| SimpleWebSer... | | Define and throw a dedicated exception instead of using a generic one. |
| SimpleWebSer... | | Define and throw a dedicated exception instead of using a generic one. |
| SimpleWebSer... | | Replace this assignment of "dServerSocket". |
| SimpleWebSer... | | Replace the synchronized class "StringBuffer" by an unsynchronized one such as "StringBuilder". |
| SimpleWebSer... | | Add an end condition to this loop. |
| SimpleWebSer... | | 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

Package Explorer JUnit

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 |
|-----------------|--------------|---|
| SimpleWebSer... | 1 minute ago | Either log or rethrow this exception. |
| SimpleWebSer... | 1 minute ago | Explicitly import the specific classes needed. |
| SimpleWebSer... | 1 minute ago | Explicitly import the specific classes needed. |
| SimpleWebSer... | 1 minute ago | Explicitly import the specific classes needed. |
| SimpleWebSer... | 1 minute ago | Missing curly brace. |
| SimpleWebSer... | 1 minute ago | Missing curly brace. |
| SimpleWebSer... | 1 minute ago | Add or update the header of this file. |
| SimpleWebSer... | 1 minute ago | Remove this throws clause. |
| SimpleWebSer... | | Move the array designator from the variable to the type. |
| SimpleWebSer... | | Define and throw a dedicated exception instead of using a generic one. |
| SimpleWebSer... | | Define and throw a dedicated exception instead of using a generic one. |
| SimpleWebSer... | | Replace this assignment of "dServerSocket". |
| SimpleWebSer... | | Replace the synchronized class "StringBuffer" by an unsynchronized one such as "StringBuilder". |
| SimpleWebSer... | | Add an end condition to this loop. |
| SimpleWebSer... | | Use trv-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:

SonarLint 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.

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** eclipse-workspace - Spotbugs/src/com/learnsecurity/SimpleWebServer.java - Eclipse IDE
- Left Sidebar:** Package Explorer shows a project structure with packages like Assignment 10 Understanding and Using Static Code, Demo, and Spotbugs.
- Middle Area:** A code editor window displays the `SimpleWebServer.java` file. The code contains a `public void run() throws Exception { ... }` block. Line 33 contains the code `while (true) {`.
- Bottom Area:** The SonarLint Report view shows a table of findings. The first few rows of the table are as follows:

| Resource | Date | Description |
|-----------------|--------------|---|
| SimpleWebSer... | 1 minute ago | Either log or rethrow this exception. |
| SimpleWebSer... | 1 minute ago | Explicitly import the specific classes needed. |
| SimpleWebSer... | 1 minute ago | Explicitly import the specific classes needed. |
| SimpleWebSer... | 1 minute ago | Explicitly import the specific classes needed. |
| SimpleWebSer... | 1 minute ago | Missing curly brace. |
| SimpleWebSer... | 1 minute ago | Missing curly brace. |
| SimpleWebSer... | 1 minute ago | Add an end condition to the header of this file. |
| SimpleWebSer... | 1 minute ago | Remove the generic class. |
| SimpleWebSer... | | Move the array deserializer from the variable to the type. |
| SimpleWebSer... | | Define and throw a dedicated exception instead of using a generic one. |
| SimpleWebSer... | | Define and throw a dedicated exception instead of using a generic one. |
| SimpleWebSer... | | Remove this assignment of "dServerSocket". |
| SimpleWebSer... | | Replace the synchronized class "StringBuffer" by an unsynchronized one such as "StringBuilder". |
| SimpleWebSer... | | Add an end condition to this loop. |
| SimpleWebSer... | | Use trv-with-resources or close this "FileReader" in a "finally" clause. |

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.

The screenshot shows the Eclipse IDE interface with the following details:

- File Explorer:** Shows the project structure with `SimpleWebServer.java` selected.
- Code Editor:** Displays the Java code for `SimpleWebServer`. Line 29 is highlighted with a red underline: `dServerSocket = new ServerSocket(PORT);`.
- Bug Explorer:** Shows a single finding under the `Spotbugs` category.
 - Rank:** Of Concern (15)
 - Confidence:** High
 - Pattern:** ST_WRITE_TO_STATIC_FROM_INSTANCE_METHOD
 - Type:** ST, Category: STYLE (Dodgy code)
- Bug Info:** Provides a detailed description of the bug: "Write to static field com.learnsecurity.SimpleWebServer.dServerSocket from instance method new com.learnsecurity.SimpleWebServer()".
- Description:** States: "This 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 assignment of `dServerSocket`. This bug also points line 29 specified earlier.

The screenshot shows the Eclipse IDE interface with the following details:

- File Explorer:** Shows the project structure with `SimpleWebServer.java` selected.
- Code Editor:** Displays the Java code for `SimpleWebServer`. Line 29 is highlighted with a red underline: `dServerSocket = new ServerSocket(PORT);`.
- SonarLint Report:** Shows a list of 49 items with the following table:

| Resource | Date | Description |
|--------------|--------------|---|
| SimpleWebSer | 1 minute ago | Missing curly brace. |
| SimpleWebSer | 1 minute ago | Add or update the header of this file. |
| SimpleWebSer | 1 minute ago | Remove this throws clause. |
| SimpleWebSer | | Move the array designator from the variable to the type. |
| SimpleWebSer | | Define and throw a dedicated exception instead of using a generic one. |
| SimpleWebSer | | Define and throw a dedicated exception instead of using a generic one. |
| SimpleWebSer | | Remove this assignment of "dServerSocket". |
| SimpleWebSer | | Replace the synchronized class "StringBuffer" by an unsynchronized one such as "StringBuilder". |
| SimpleWebSer | | Add an end condition to this loop. |
| SimpleWebSer | | Use try-with-resources or close this "FileReader" in a "finally" clause. |

- File Status:** Shows the file path: `/Spotbugs/src/com/learnsecurity/SimpleWebServer.java` at 20/04/2020 19:51.

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.

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** eclipse-workspace - Spotbugs/src/com/learnsecurity/SimpleWebServer.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Left Sidebar:** Package Explorer (SimpleWebServer.java), JUnit, Assignment 10 Understanding and Using Static Code, Demo, Oops, Spotbugs (2).
- Center View:** SimpleWebServer.java code editor showing the following snippet:


```
100  /* try to open file specified by pathname */
101  try {
102      fr = new FileReader (pathname);
103      c = fr.read();
104  }
105  catch (Exception e) {
106      /* if the file is not found,return the
107       appropriate HTTP response code */
108      osw.write ("HTTP/1.0 404 Not Found\n\n");
109      return;
110  }
111 }
```
- Bottom View:** SonarLint Report (49 items) showing various code inspection findings for SimpleWebServer.java.

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.

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** eclipse-workspace - Spotbugs/src/com/learnsecurity/SimpleWebServer.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Left Sidebar:** Package Explorer (SimpleWebServer.java), JUnit, Assignment 10 Understanding and Using Static Code, Demo, Oops, Spotbugs (2).
- Center View:** SimpleWebServer.java code editor showing the following snippet:


```
58  String command = null;
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);
73  }
74 }
```
- Bottom View:** Bug Explorer (Spotbugs (2)) showing a context menu for a bug entry. The "Save XML" option is highlighted.

XML file preview:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Project NewVersion="1.0" ChangeLog="true">
3   <BugCollection version="4.0.1" sequence="1" timestamp="1587429245823" analysisTimestamp="1587429240634" release="">
4     <Project projectName="Spotbugs">
5       <Jar>C:\Users\gouta\eclipse-workspace\Spotbugs\bin\com\learnsecurity\SimpleWebServer.class</Jar>
6       <AuxClasspathEntry>C:\Program Files\Java\jre1.8.0_251\lib\rt.jar</AuxClasspathEntry>
7       <AuxClasspathEntry>C:\Program Files\Java\jre1.8.0_251\lib\jce.jar</AuxClasspathEntry>
8       <SrcDir>C:\Users\gouta\eclipse-workspace\Spotbugs\src</SrcDir>
9     </Project>
10    <BugInstance type="DM_DEFAULT_ENCODING" priority="1" rank="19" abbrev="Dm" category="I18N" first="1">
11      <Class classname="com.learnsecurity.SimpleWebServer">
12        <SourceLine classname="com.learnsecurity.SimpleWebServer" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
13      </Class>
14      <Method classname="com.learnsecurity.SimpleWebServer" name="processRequest" signature="(Ljava/net/Socket;)V" isStatic="false">
15        <SourceLine classname="com.learnsecurity.SimpleWebServer" start="48" end="83" startBytecode="0" endBytecode="324" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
16      </Method>
17      <Method classname="java.io.InputStreamReader" name="<init;>" signature="(Ljava/io/InputStream;)V" isStatic="false" role="METHOD_CALLED">
18        <SourceLine classname="java.io.InputStreamReader"/>
19      </Method>
20      <SourceLine classname="com.learnsecurity.SimpleWebServer" start="49" end="49" startBytecode="12" endBytecode="12" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
21      <SourceLine classname="com.learnsecurity.SimpleWebServer" start="49" end="49" startBytecode="12" endBytecode="12" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
22    </BugInstance>
23    <BugInstance type="DM_DEFAULT_ENCODING" priority="1" rank="19" abbrev="Dm" category="I18N" first="1">
24      <Class classname="com.learnsecurity.SimpleWebServer">
25        <SourceLine classname="com.learnsecurity.SimpleWebServer" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
26      </Class>
27      <Method classname="com.learnsecurity.SimpleWebServer" name="processRequest" signature="(Ljava/net/Socket;)V" isStatic="false">
28        <SourceLine classname="com.learnsecurity.SimpleWebServer" start="48" end="83" startBytecode="0" endBytecode="324" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
29      </Method>
30      <Method classname="java.io.OutputStreamWriter" name="<init;>" signature="(Ljava/io/OutputStream;)V" isStatic="false" role="METHOD_CALLED">
31        <SourceLine classname="java.io.OutputStreamWriter"/>
32      </Method>
33      <SourceLine classname="com.learnsecurity.SimpleWebServer" start="53" end="53" startBytecode="27" endBytecode="27" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
34      <SourceLine classname="com.learnsecurity.SimpleWebServer" start="53" end="53" startBytecode="27" endBytecode="27" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
35    </BugInstance>
36    <BugInstance type="DM_DEFAULT_ENCODING" priority="1" rank="19" abbrev="Dm" category="I18N" first="1">
37      <Class classname="com.learnsecurity.SimpleWebServer">
38        <SourceLine classname="com.learnsecurity.SimpleWebServer" sourcefile="SimpleWebServer.java" sourcepath="com/learnsecurity/SimpleWebServer.java"/>
39    </BugInstance>
```



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("Information 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 an 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 ArrayOutOfBoundsException exception.

```

eclipse-workspace - Spotbugs/src/com/learnsecurity/Register.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit Register.java
1 package com.learnsecurity;
2
3 import java.util.ArrayList;
4 /**
5 * Class to Register an individual
6 */
7 public class Register {
8     /** The check input value. */
9     // This should be "Individual" or "Business" or "Admin"
10    private String typeOfUser= null;
11    // Name of the user
12    private String name = null;
13
14    /**
15     * The Constructor.
16     */
17    public Register(){
18        this.typeOfUser = null;
19        this.name = null;
20    }
21    public void run() {
22        individualInput();
23        IndividualValidation();
24        ArrayList<String> list = new ArrayList<>();
25        list.add(this.typeOfUser);
26        list.add(this.name);
27        System.out.println("Information of the name and type of user: " + list.get(2));
28    }
29    /**
30     * Check the input given in the individual field is for Individual user.
31     */
32    private void individualInput() {
33        if (this.typeOfUser == "Individual") {
34            System.out.println("This is valid individual user");
35        } else if (this.typeOfUser == "Business") {
36            System.out.println("This is valid Business user");

```

Results:

SpotBugs:

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

The screenshot shows two instances of the Eclipse IDE interface. In the top instance, a context menu is open over a Java file named Register.java. The 'Find Bugs' option under the 'SpotBugs' section is highlighted. The code editor shows a snippet of Java code related to user registration. In the bottom instance, a modal dialog box titled 'SpotBugs analysis finished' is displayed, stating '2 warnings found'. The code editor shows the same Java code with several lines underlined in blue, indicating detected bugs. The 'Yes' button is highlighted in the dialog's button bar.

```
1 package com.learnsecurity;
2
3 import java.util.ArrayList;
4 /**
5 * Class to Register an individual
6 */
7 public class Register {
8     /** The check input value. */
9
10    public void run() {
11        individualInput();
12        IndividualValidation();
13        ArrayList<String> list = new ArrayList<>();
14        list.add(this.typeOfUser);
15        list.add(this.name);
16        System.out.println("Information of the name and type of user: "+list.get(2));
17    }
18    /**
19     * Check the input given in the individual field is for Individual user.
20     */
21    private void individualInput() {
22        if (this.typeOfUser == "Individual") {
23            System.out.println("This is valid individual user");
24        } else if (this.typeOfUser == "Business") {
25            System.out.println("This is valid Business user");
26        }
27    }
28 }
```

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.

```

1 package com.learnsecurity;
2
3 import java.util.ArrayList;
4
5 /**
6  * Class to Register an individual
7 */
8 public class Register {
9     /** The check input value. */
10    private String typeOfUser= null;
11    // Name of the user
12    private String name = null;
13
14    /**
15     * The Constructor.
16     */
17    public Register() {
18        this.typeOfUser = null;
19        this.name = null;
20    }
21    public void run() {
22        individualInput();
23        IndividualValidation();
24        ArrayList<String> list = new ArrayList<>();
25        list.add(this.typeOfUser);
26        list.add(this.name);
27        System.out.println("Information of the name and type of user: "+list.get(2));
28    }
29    /**
30     * Check the input given in the individual field is for Individual user.
31     */
32    private void individualInput() {
33        if (this.typeOfUser == "Individual") {
34            System.out.println("This is valid individual user");
35        } else if (this.typeOfUser == "Business") {
36            System.out.println("This is valid Business user");
37        } else return;
38    }
39
40    /**
41     * Validate the Individual and give information where to find the details.
42     */
43    private static void IndividualValidation() {
44        boolean dropdown = false;
45        if (dropdown == true) {
46            System.out.println("You can check the complete details in the Information page");
47        } else {
48            System.exit(0);
49        }
50    }
51 }
52

```

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.

```

17    public Register() {
18        this.typeOfUser = null;
19        this.name = null;
20    }
21    public void run() {
22        individualInput();
23        IndividualValidation();
24        ArrayList<String> list = new ArrayList<>();
25        list.add(this.typeOfUser);
26        list.add(this.name);
27        System.out.println("Information of the name and type of user: "+list.get(2));
28    }
29    /**
30     * Check the input given in the individual field is for Individual user.
31     */
32    private void individualInput() {
33        if (this.typeOfUser == "Individual") {
34            System.out.println("This is valid individual user");
35        } else if (this.typeOfUser == "Business") {
36            System.out.println("This is valid Business user");
37        } else return;
38    }
39
40    /**
41     * Validate the Individual and give information where to find the details.
42     */
43    private static void IndividualValidation() {
44        boolean dropdown = false;
45        if (dropdown == true) {
46            System.out.println("You can check the complete details in the Information page");
47        } else {
48            System.exit(0);
49        }
50    }
51 }
52

```

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.

eclipse-workspace - Spotbugs/src/com/learnsecurity/Register.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

```

37         } else return;
38     }
39
40     /**
41      * Validate the Individual and give information where to find the details.
42     */
43     private static void IndividualValidation() {
44         boolean dropDown = false;
45         if (dropDown == true) {
46             System.out.println("You can check the complete details in the Information page");
47         } else {
48             System.exit(0);
49         }
50     }
51 }

```

Bug Explorer

- Spotbugs (5)
 - Scary (1)
 - Method assigns boolean literal in boolean expression (1)
 - com.learnsecurity.Register.IndividualValidation() assigns boolean literal in boolean expression [Scary(5), High confidence]
 - Troubling (2)
 - Normal confidence (2)
 - Field only ever set to null (2)
 - Field only ever set to null: com.learnsecurity.Register.name [Troubling(12), Normal confidence]
 - Field only ever set to null: com.learnsecurity.Register.typeOfUser [Troubling(12), Normal confidence]
 - Of Concern (2)
 - High confidence (1)
 - Write to static field from instance method (1)
 - Normal confidence (1)
 - Difference of the result of read/write without null-check (1)

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.

eclipse-workspace - Spotbugs/src/com/learnsecurity/Register.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

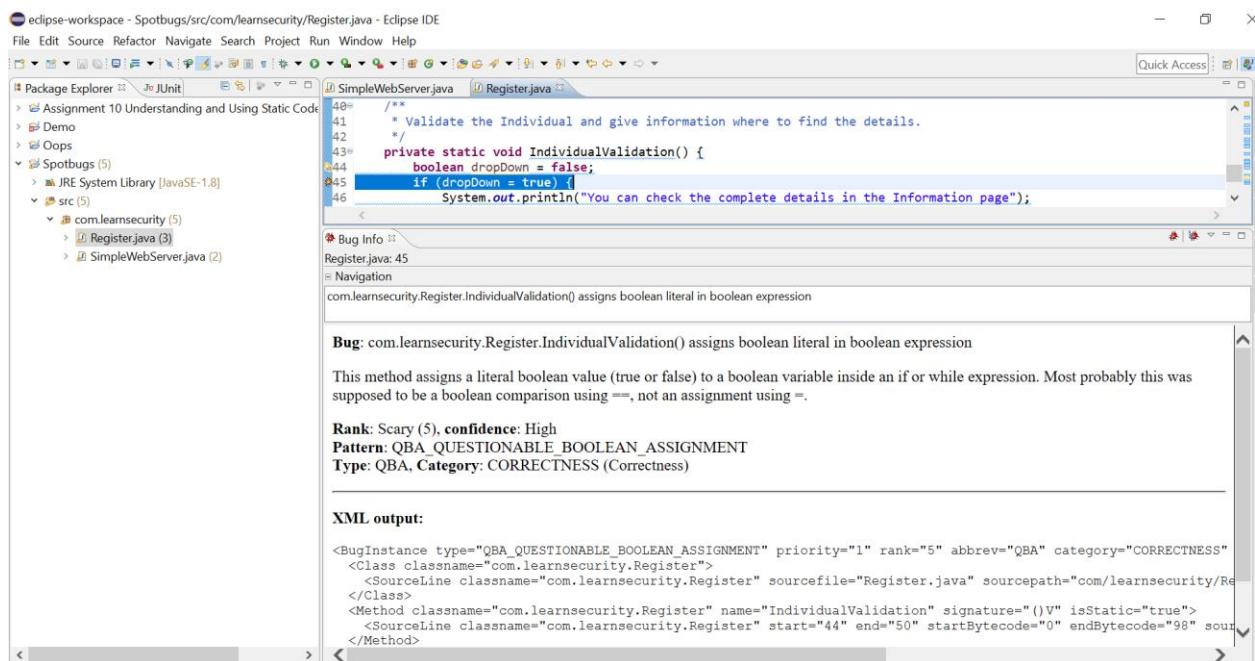
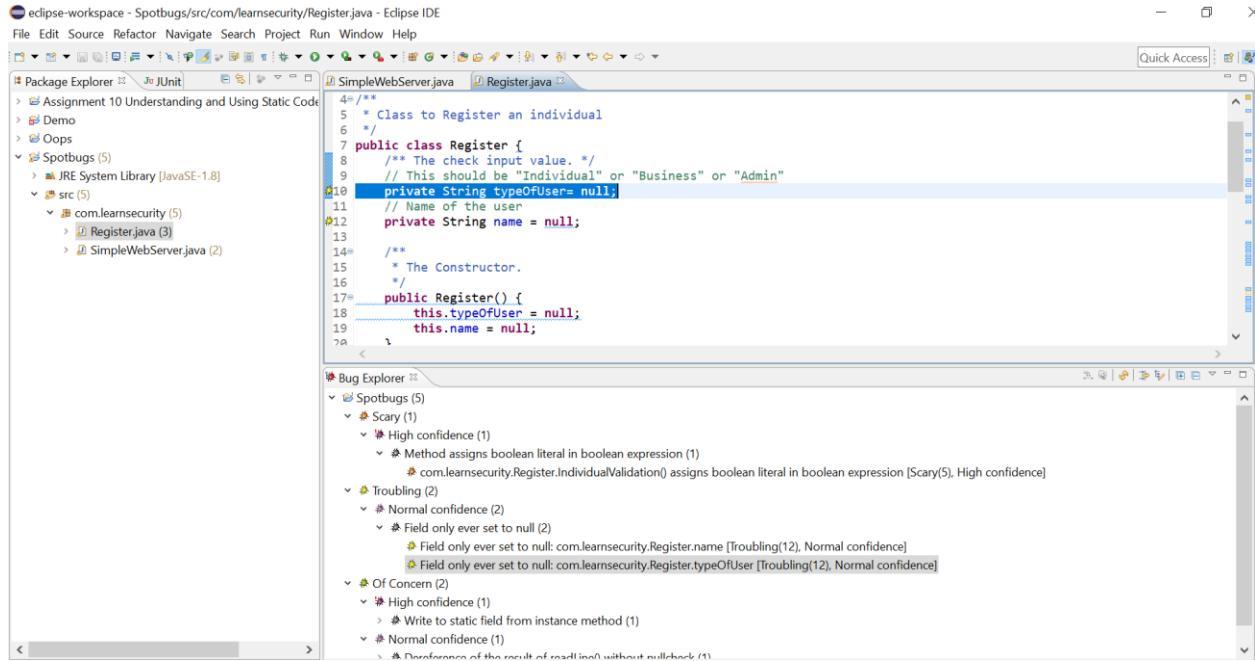
```

4 /**
5  * Class to Register an individual
6 */
7 public class Register {
8     /** The check input value. */
9     // This should be "Individual" or "Business" or "Admin"
10    private String typeOfUser= null;
11    // Name of the user
12    private String name = null;
13
14    /**
15     * The Constructor.
16     */
17    public Register(){
18        this.typeOfUser = null;
19        this.name = null;
20    }

```

Bug Explorer

- Spotbugs (5)
 - Scary (1)
 - High confidence (1)
 - Method assigns boolean literal in boolean expression (1)
 - com.learnsecurity.Register.IndividualValidation() assigns boolean literal in boolean expression [Scary(5), High confidence]
 - Troubling (2)
 - Normal confidence (2)
 - Field only ever set to null (2)
 - Field only ever set to null: com.learnsecurity.Register.name [Troubling(12), Normal confidence]
 - Field only ever set to null: com.learnsecurity.Register.typeOfUser [Troubling(12), Normal confidence]
 - Of Concern (2)
 - High confidence (1)
 - Write to static field from instance method (1)
 - Normal confidence (1)
 - Difference of the result of read/write without null-check (1)



SpotBugs detects the bug that `IndividualValidation()` 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 `=`.

eclipse-workspace - Spotbugs/src/com/learnsecurity/Register.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer JUnit

SimpleWebServer.java Register.java

```

8  /** The check input value. */
9  // This should be "Individual" or "Business" or "Admin"
10 private String typeOfUser= null;
11 // Name of the user
12 private String name = null;
13
14 /**

```

Bug Info

Register.java: 12

Navigation

Field only ever set to null: com.learnsecurity.Register.name
Field com.learnsecurity.Register.name

Bug: Field only ever set to null: com.learnsecurity.Register.name

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.

Rank: Troubling (12), **confidence:** Normal
Pattern: UWF_NULL_FIELD
Type: UwF, **Category:** CORRECTNESS (Correctness)

XML output:

```

<BugInstance type="UWF_NULL_FIELD" priority="2" rank="12" abbrev="UwF" category="CORRECTNESS" first="22">
    <Class classname="com.learnsecurity.Register">
        <SourceLine classname="com.learnsecurity.Register" sourcefile="Register.java" sourcepath="com/learnsecurity/Re
    </Class>
    <Field classname="com.learnsecurity.Register" name="name" signature="Ljava/lang/String;" isStatic="false">
        <SourceLine classname="com.learnsecurity.Register" start="12" end="12" startBytecode="0" endBytecode="0" sourd

```

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.

eclipse-workspace - Spotbugs/src/com/learnsecurity/Register.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer JUnit

SimpleWebServer.java Register.java

```

8  /** The check input value. */
9  // This should be "Individual" or "Business" or "Admin"
10 private String typeOfUser= null;
11 // Name of the user
12 private String name = null;
13
14 /**

```

Bug Info

Register.java: 10

Navigation

Field only ever set to null: com.learnsecurity.Register.typeOfUser
Field com.learnsecurity.Register.typeOfUser

Bug: Field only ever set to null: com.learnsecurity.Register.typeOfUser

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.

Rank: Troubling (12), **confidence:** Normal
Pattern: UWF_NULL_FIELD
Type: UwF, **Category:** CORRECTNESS (Correctness)

XML output:

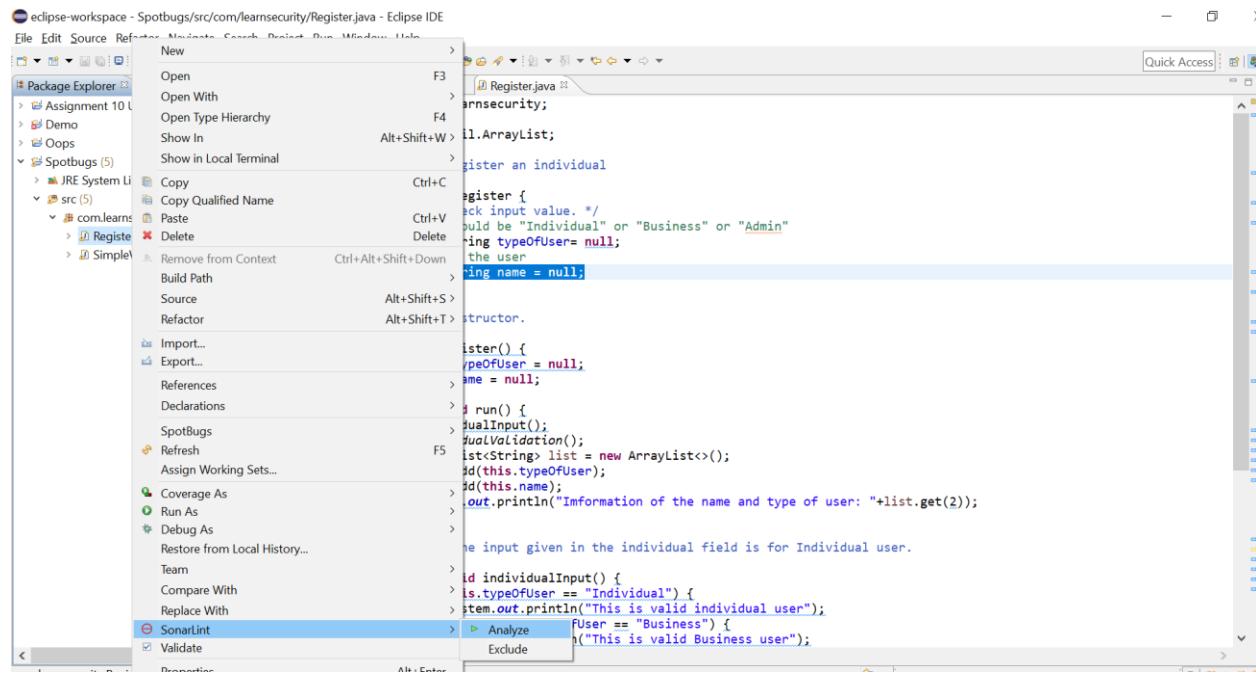
```

<BugInstance type="UWF_NULL_FIELD" priority="2" rank="12" abbrev="UwF" category="CORRECTNESS" first="35">
    <Class classname="com.learnsecurity.Register">
        <SourceLine classname="com.learnsecurity.Register" sourcefile="Register.java" sourcepath="com/learnsecurity/Re
    </Class>
    <Field classname="com.learnsecurity.Register" name="typeOfUser" signature="Ljava/lang/String;" isStatic="false">
        <SourceLine classname="com.learnsecurity.Register" start="10" end="10" startBytecode="0" endBytecode="0" sourd

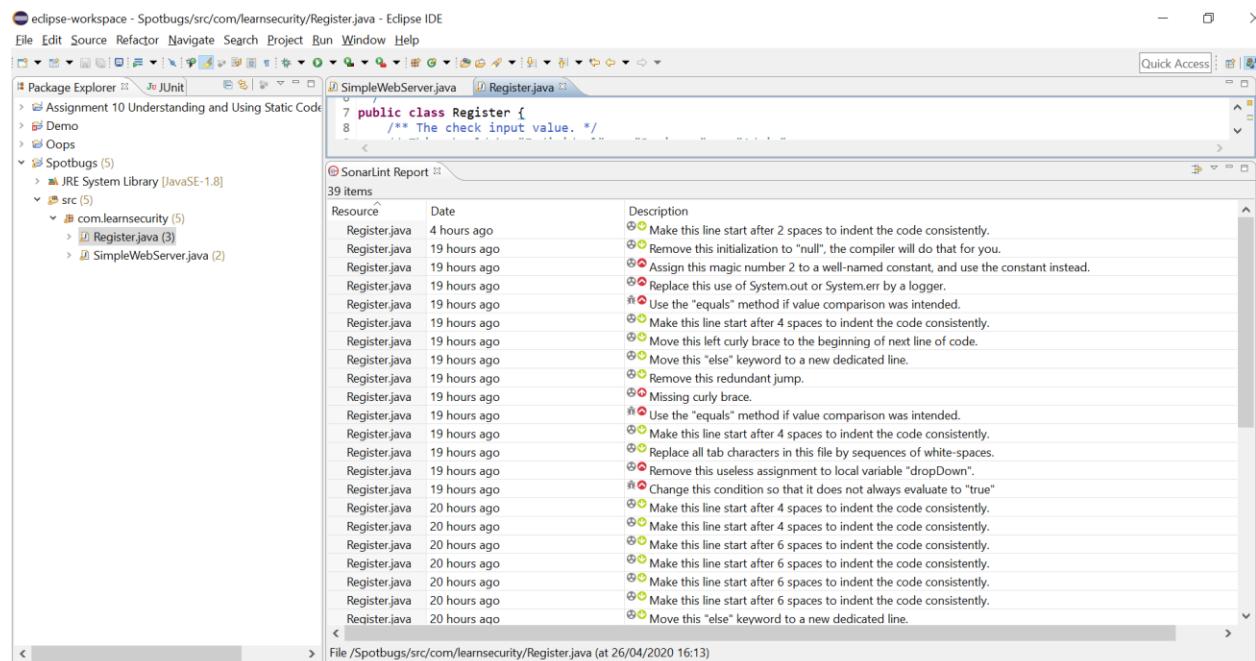
```

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.



The screenshot shows the Eclipse IDE interface with the SonarLint Report open. The report lists 39 items, mostly related to the `Register.java` file, with suggestions such as moving lines, renaming methods, and fixing System.out/err usage.

```

public class Register {
    /* The check input value. */
}

SonarLint Report
39 items
Resource Date Description
Register.java 20 hours ago Make this line start after 6 spaces to indent the code consistently.
Register.java 20 hours ago Make this line start after 6 spaces to indent the code consistently.
Register.java 20 hours ago Make this line start after 6 spaces to indent the code consistently.
Register.java 20 hours ago Move this "else" keyword to a new dedicated line.
Register.java 20 hours ago Move this "else" keyword to a new dedicated line.
Register.java 20 hours ago Move this left curly brace to the beginning of next line of code.
Register.java 20 hours ago Move this left curly brace to the beginning of next line of code.
Register.java 20 hours ago Move this left curly brace to the beginning of next line of code.
Register.java 20 hours ago Move this left curly brace to the beginning of next line of code.
Register.java 20 hours ago Move this left curly brace to the beginning of next line of code.
Register.java 20 hours ago Move this left curly brace to the beginning of next line of code.
Register.java 20 hours ago Remove this initialization to "null", the compiler will do that for you.
Register.java 20 hours ago Rename this method name to match the regular expression '^a-zA-Z0-9*$'.
Register.java 20 hours ago Extract the assignment out of this expression.
Register.java 20 hours ago Replace this use of System.out or System.err by a logger.
Register.java 20 hours ago Replace this use of System.out or System.err by a logger.
Register.java 20 hours ago Replace this use of System.out or System.err by a logger.
Register.java 20 hours ago Remove this call to "exit" or ensure it is really required.
Register.java 20 hours ago Move this left curly brace to the beginning of next line of code.
Register.java 20 hours ago Add or update the header of this file.

```

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"))`.

The screenshot shows the Eclipse IDE interface with the SonarLint Report open. A specific item in the report highlights the use of the `==` operator for string comparison, suggesting to use `String.equals()` instead.

```

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;
}

```

SonarLint Report

| Resource | Date | Description |
|---------------|--------------|--|
| Register.java | 4 hours ago | Make this line start after 2 spaces to indent the code consistently. |
| Register.java | 19 hours ago | Remove this initialization to "null", the compiler will do that for you. |
| Register.java | 19 hours ago | Assign this magic number 2 to a well-named constant, and use the constant instead. |
| Register.java | 19 hours ago | Replace this use of System.out or System.err by a logger. |
| Register.java | 19 hours ago | Use the "equals" method if value comparison was intended. |
| Register.java | 19 hours ago | Make this line start after 4 spaces to indent the code consistently. |
| Register.java | 19 hours ago | Move this left curly brace to the beginning of next line of code. |
| Register.java | 19 hours ago | Move this "else" keyword to a new dedicated line. |
| Register.java | 19 hours ago | Remove this redundant jump. |
| Register.java | 19 hours ago | Missing curly brace. |
| Register.java | 19 hours ago | Use the "equals" method if value comparison was intended. |
| Register.java | 19 hours ago | Make this line start after 4 spaces to indent the code consistently. |
| Register.java | 19 hours ago | Replace all tab characters in this file by sequences of white-spaces. |

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 one line after this condition is considered as action. We have to give `else { return}`

The screenshot shows the Eclipse IDE interface with the 'SonarLint Report' view open. The report lists 39 items, with the last item being 'Register.java 19 hours ago' which has a 'Missing curly brace' violation. The code editor shows a Java file with several print statements and an if-else block. The SonarLint report details the specific issue at line 37.

```

28    }
29    /**
30     * Check the input given in the individual field is for Individual user.
31     */
32    private void individualInput() {
33        if (this.typeOfUser == "Individual") {
34            System.out.println("This is valid individual user");
35        } else if (this.typeOfUser == "Business") {
36            System.out.println("This is valid Business user");
37        } else return;
38    }
39    /**
40

```

| Resource | Date | Description |
|---------------|--------------|--|
| Register.java | 4 hours ago | Make this line start after 2 spaces to indent the code consistently. |
| Register.java | 19 hours ago | Remove this initialization to "null", the compiler will do that for you. |
| Register.java | 19 hours ago | Assign this magic number 2 to a well-named constant, and use the constant instead. |
| Register.java | 19 hours ago | Replace this use of System.out or System.err by a logger. |
| Register.java | 19 hours ago | Use the "equals" method if value comparison was intended. |
| Register.java | 19 hours ago | Make this line start after 4 spaces to indent the code consistently. |
| Register.java | 19 hours ago | Move this left curly brace to the beginning of next line of code. |
| Register.java | 19 hours ago | Move this "else" keyword to a new dedicated line. |
| Register.java | 19 hours ago | Remove this redundant jump. |
| Register.java | 19 hours ago | Missing curly brace. |
| Register.java | 19 hours ago | Use the "equals" method if value comparison was intended. |
| Register.java | 19 hours ago | Make this line start after 4 spaces to indent the code consistently. |
| Register.java | 19 hours ago | Replace all tab characters in this file by sequences of white-spaces. |

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)`.

The screenshot shows the Eclipse IDE interface with the 'SonarLint Report' view open. The report lists 39 items, with the last item being 'Register.java 19 hours ago' which has a 'Useless assignment' violation. The code editor shows a Java file with a validation method that prints a message if dropDown is true. The SonarLint report details the specific issue at line 45.

```

40    /**
41     * Validate the Individual and give information where to find the details.
42     */
43    private static void IndividualValidation() {
44        boolean dropDown = false;
45        if (dropDown = true) {
46            System.out.println("You can check the complete details in the Information page");
47        } else {
48            System.exit(0);
49        }
50    }
51
52

```

| Resource | Date | Description |
|---------------|--------------|---|
| Register.java | 19 hours ago | Move this left curly brace to the beginning of next line of code. |
| Register.java | 19 hours ago | Move this "else" keyword to a new dedicated line. |
| Register.java | 19 hours ago | Remove this redundant jump. |
| Register.java | 19 hours ago | Missing curly brace. |
| Register.java | 19 hours ago | Use the "equals" method if value comparison was intended. |
| Register.java | 19 hours ago | Make this line start after 4 spaces to indent the code consistently. |
| Register.java | 19 hours ago | Replace all tab characters in this file by sequences of white-spaces. |
| Register.java | 19 hours ago | Remove this useless assignment to local variable "dropDown". |
| Register.java | 19 hours ago | Change this condition so that it does not always evaluate to "true". |
| Register.java | 20 hours ago | Make this line start after 4 spaces to indent the code consistently. |
| Register.java | 20 hours ago | Make this line start after 4 spaces to indent the code consistently. |
| Register.java | 20 hours ago | Make this line start after 6 spaces to indent the code consistently. |
| Register.java | 20 hours ago | Make this line start after 6 spaces to indent the code consistently. |

Also, place the `boolean dropDown = false;` statement before the `individualValidation()` 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 at line 27, where the `ArrayList` list tries to retrieve an 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 `ArrayOutOfBoundsException` 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("Information 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:

```

eclipse-workspace - Spotbugs/src/com/learnsecurity/Register.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
SimpleWebServer.java Register.java
1 package com.learnsecurity;
2
3 import java.util.ArrayList;
4 /*
5  * Class to Register an individual
6  */
7 public class Register {
8     /** The check input value. */
9     // This should be "Individual" or "Business" or "Admin"
10    private String typeOfUser;
11    // Name of the user
12    private String name;
13
14    /**
15     * The Constructor.
16     */
17    public Register() {
18        this.typeOfUser = "Individual";
19        this.name = "";
20    }
21    public void run() {
22        individualInput();
23        IndividualValidation();
24        ArrayList<String> list = new ArrayList<>();
25        list.add(this.typeOfUser);
26        list.add(this.name);
27        System.out.println("Information of the name and type of user: "+list.get(1));
28    }
29    /**
30     * Check the input given in the individual field is for Individual user.
31     */
32    private void individualInput() {
33        if (this.typeOfUser.equals("Individual")) {
34            System.out.println("This is valid individual user");
35        } else if (this.typeOfUser.equals("Business")) {
36            System.out.println("This is valid Business user");
37        }
38    }
}

```

eclipse-workspace - Spotbugs/src/com/learnsecurity/Register.java - Eclipse IDE

```

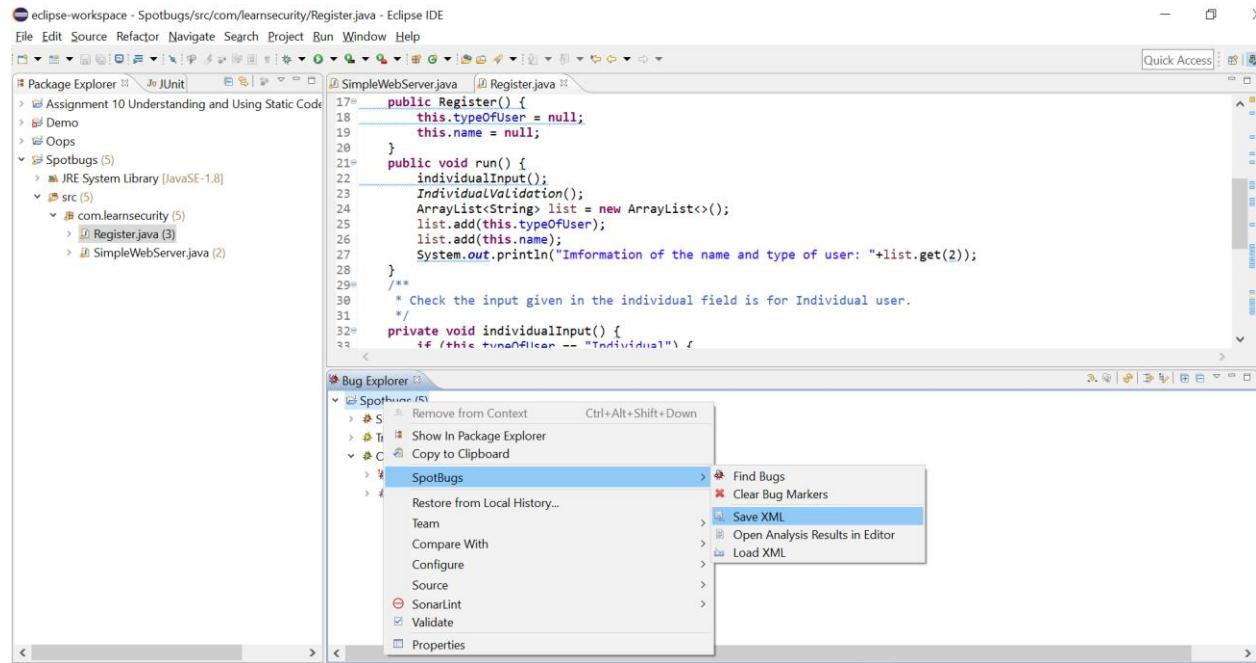
17= public Register() {
18     this.typeOfUser = "Individual";
19     this.name = "";
20 }
21= public void run() {
22     individualInput();
23     IndividualValidation();
24     ArrayList<String> list = new ArrayList<>();
25     list.add(this.typeOfUser);
26     list.add(this.name);
27     System.out.println("Information of the name and type of user: "+list.get(1));
28 }
29=
30 /**
31 * Check the input given in the individual field is for Individual user.
32 */
32= private void individualInput() {
33     if (this.typeOfUser.equals("Individual")){
34         System.out.println("This is valid individual user");
35     } else if (this.typeOfUser.equals("Business")) {
36         System.out.println("This is valid Business user");
37     } else return;
38 }
39
40 /**
41 * Validate the Individual and give information where to find the details.
42 */
43 public static boolean dropDown = false;
44= private static void IndividualValidation() {
45     if (dropDown == true) {
46         System.out.println("You can check the complete details in the Information page");
47     } else {
48         return;
49     }
50 }
51 }
52 
```

SonarLint Result for fixed code:

eclipse-workspace - Spotbugs/src/com/learnsecurity/Register.java - Eclipse IDE

| Resource | Date | Description |
|---------------|-----------------|--|
| Register.java | few seconds ago | Move the "Business" string literal on the left side of this string comparison. |
| Register.java | few seconds ago | Move the "Individual" string literal on the left side of this string comparison. |
| Register.java | 1 minute ago | Remove this redundant jump. |
| Register.java | 3 minutes ago | Make dropDown a static final constant or non-public and provide accessors if needed. |
| Register.java | 3 minutes ago | Make this "public static dropDown" field final. |
| Register.java | 3 minutes ago | Make this line start after 4 spaces to indent the code consistently. |
| Register.java | 3 minutes ago | Move this variable to comply with Java Code Conventions. |
| Register.java | 3 minutes ago | Remove the literal "true" boolean value. |
| Register.java | 3 minutes ago | Remove this initialization to "false", the compiler will do that for you. |
| Register.java | 6 hours ago | Make this line start after 2 spaces to indent the code consistently. |

XML Report:



XML Preview:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <BugCollection version="4.0.1" sequence="40" timestamp="1587937796534" analysisTimestamp="1587937794970" release="">
3   <Project projectName="Spotbugs">
4     <Jar>C:\Users\gouta\eclipse-workspace\Spotbugs\bin\com\learnsecurity\Register.class</Jar>
5     <AuxclasspathEntry>C:\Users\gouta\eclipse-workspace\Spotbugs\bin</AuxclasspathEntry>
6     <AuxclasspathEntry>C:\Program Files\Java\jre1.8.0_251\lib\rt.jar</AuxclasspathEntry>
7     <AuxclasspathEntry>C:\Program Files\Java\jre1.8.0_251\lib\jce.jar</AuxclasspathEntry>
8     <SrcDir>C:\Users\gouta\eclipse-workspace\Spotbugs\src</SrcDir>
9   </Project>
10  <BugInstance type="SBSC_USE_STRINGBUFFER_CONCATENATION" priority="2" rank="18" abbrev="SBSC" category="PERFORMANCE" first="14">
11    <Class classname="com.learnsecurity.Analysis">
12      <SourceLine classname="com.learnsecurity.Analysis" sourcefile="Analysis.java" sourcepath="com/learnsecurity/Analysis.java"/>
13    </Class>
14    <Method classname="com.learnsecurity.Analysis" name="test" signature="()V" isStatic="false">
15      <SourceLine classname="com.learnsecurity.Analysis" start="12" end="22" startBytecode="0" endBytecode="193" sourcefile="Analysis.java" sourcepath="com/learnsecu
16      <SourceLine classname="com.learnsecurity.Analysis" start="17" end="17" startBytecode="38" endBytecode="38" sourcefile="Analysis.java" sourcepath="com/learnsecu
17    </Method>
18    <SourceLine classname="com.learnsecurity.Analysis" start="17" end="17" startBytecode="38" endBytecode="38" sourcefile="Analysis.java" sourcepath="com/learnsecu
19  </BugInstance>
20  <BugInstance type="UUF_UNUSED_FIELD" priority="2" rank="18" abbrev="UuF" category="PERFORMANCE" first="14">
21    <Class classname="com.learnsecurity.Analysis">
22      <SourceLine classname="com.learnsecurity.Analysis" sourcefile="Analysis.java" sourcepath="com/learnsecurity/Analysis.java"/>
23    </Class>
24    <Field classname="com.learnsecurity.Analysis" name="abc" signature="I" isStatic="false">
25      <SourceLine classname="com.learnsecurity.Analysis" start="6" end="6" startBytecode="0" endBytecode="0" sourcefile="Analysis.java" sourcepath="com/learnsecu
26    </Field>
27  </BugInstance>
28  <BugInstance type="CD_CIRCULAR_DEPENDENCY" priority="2" rank="17" abbrev="CD" category="STYLE" first="20" last="24" removedByChange="true">
29    <Class classname="com.learnsecurity.Client">
30      <SourceLine classname="com.learnsecurity.Client" start="51" end="297" sourcefile="Client.java" sourcepath="com/learnsecurity/Client.java"/>
31    </Class>
32    <Class classname="com.learnsecurity.Client$2">
33      <SourceLine classname="com.learnsecurity.Client$2" sourcefile="Client.java" sourcepath="com/learnsecurity/Client.java"/>
34    </Class>
35  </BugInstance>
36  <BugInstance type="CD_CIRCULAR_DEPENDENCY" priority="2" rank="17" abbrev="CD" category="STYLE" first="20" last="24" removedByChange="true">
37    <Class classname="com.learnsecurity.Client">
38      <SourceLine classname="com.learnsecurity.Client" start="51" end="297" sourcefile="Client.java" sourcepath="com/learnsecurity/Client.java"/>
39    </Class>

```

spotbugRegisterRepo
rt.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/>