**Assignment 10 :- Understanding and Using Static Code Analysis Tools**

**Name :- Aparna Krishna Bhat**

**ID :- 1001255079**

**Part 1**

The aim of this task is to use research tools like SpotBugs and SonarLint to examine a web server code.

A screenshot of a computer screen

Description automatically generated with medium confidence

A screenshot of a computer

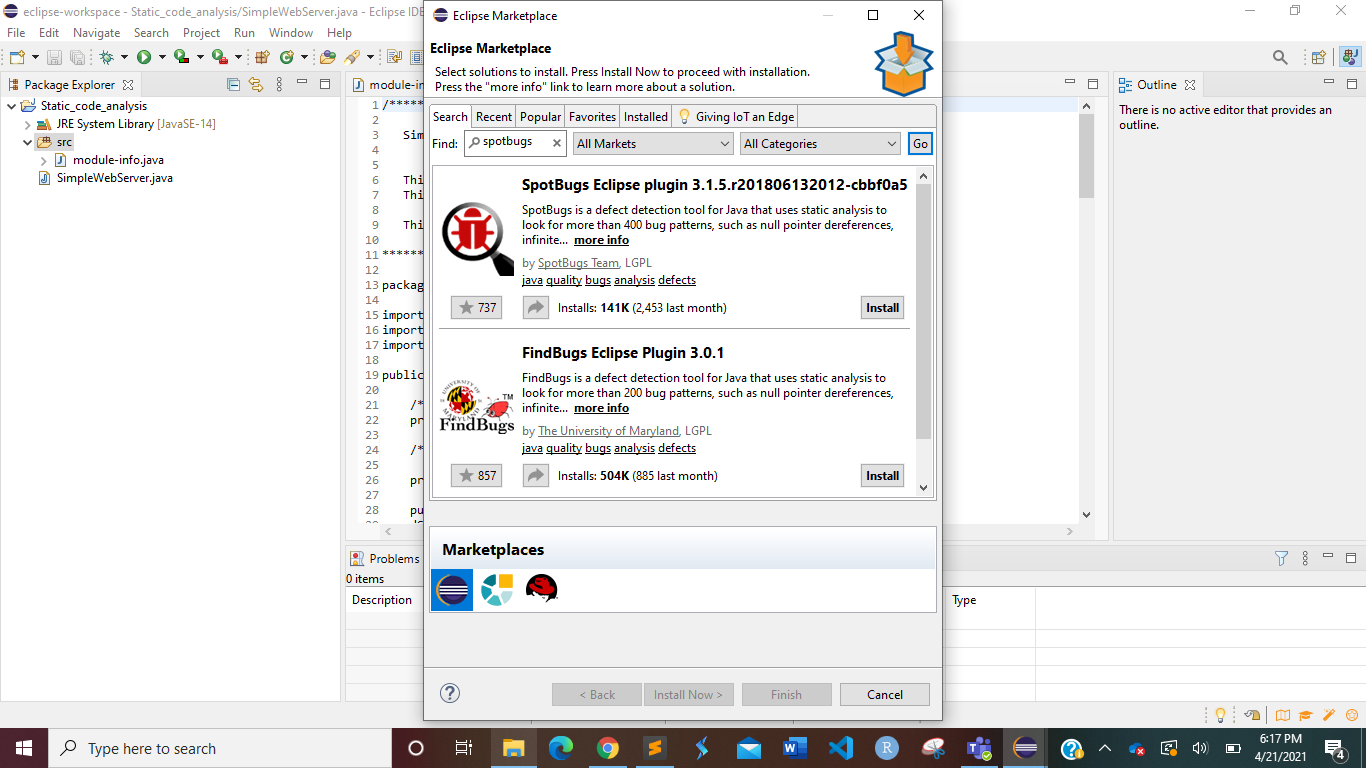
Description automatically generated with medium confidence

**Manual Analysis** :- The code is initially examined to see if any issues can be found. We can see from the code for SimpleWebServer.java that the BufferedReader object br generates an InputStream and uses it but does not close it. Close the streams at all times. While a FileReader object has been developed, it has not been closed. There may be performance problems, garbage value selection, and incorrect undesired output if these two artifacts are not closed.

**Tool Choices/Versions**

SpotBugs and SonarLint are the two resources I've selected.

1. **SpotBugs Eclipse plugin**



1. **SonarLint Eclipse plugin 5.8.1**

Graphical user interface, text, application

Description automatically generated

**Tool Invocation Process**

1. **SpotBugs**

Go to Help in the Eclipse IDE and look for Elipse Marketplace. Then look for SpotBugs and download it.

**Home -> Help -> Eclipse Marketplace -> Search Spotbugs -> Install**

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Right-click on the project in which the SimpleWebServer.java file is located. SpotBugs is an option that we can see. We get Find Bugs if we press that. We will discover all of the bugs in the software by clicking on Find Bugs.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

We go to Window -> Preferences in the Eclipse IDE. Then, in Preferences, go to Java -> SpotBugs and make sure that all of the check boxes for Reported Bug categories in Reported Configuration and Detector Configuration are checked. Using SpotBugs, we want to find all sorts of bugs in the software.

Graphical user interface, application

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer screen

Description automatically generated with medium confidence

Next, by right-clicking on SimpleWebServer.java and selecting SpotBugs -> Find Bugs, we can now find all the bugs.

Graphical user interface, text, application

Description automatically generated

dServerSocket has been declared as a static variable on line 26. This leads to a bug on line 29, where we attempt to write the instance method new ServerSocket (PORT) to a static variable. A static field is written by the instance process. If several instances are being exploited, this is difficult to get correct, and it's usually not a good a practice.

1. **SonarLint**

Navigate to Help in the Eclipse IDE and look for Eclipse Marketplace. Then search for SonarLint and install it.

**Help -> Eclipse MarketPlace -> Search Sonarlint -> Install**

Graphical user interface, text, application

Description automatically generated

Right-click on the project containing the SimpleWebServer.java package. SonarLint is a choice we can see. We have two choices when we press that: Analyze and Exclude. We will find all of the program's bugs by clicking on Analyze.

We go to Window -> Preferences in the Eclipse IDE. Then, in Preferences, go to Java -> SonarLint and double-check that all of the Java check boxes in Rules Configuration are checked. We want to use SonarLint to find all sorts of bugs in the software.

A screenshot of a computer

Description automatically generated

We'll be able to find all types of bugs in the entire software once all of the rules are installed during SonarLint review. This search is performed to ensure that the program does not contain any bugs. By right-clicking on SimpleWebServer.java and selecting SonarLint -> Analyze, we can now find all of the bugs.

Graphical user interface, application

Description automatically generated

With the SonarLint tool, several new bugs and suggestions have been given and discovered. Here are a few of them:

1)

Graphical user interface, text, application, email

Description automatically generated

1. Explicitly import the particular classes required – Because import java.util.\* is a very common import and all libraries are imported, it is not a best practice or a safe practice. As a result, always define the exact library.
2. Missing curly brace – We must always use curly braces after if (condition) to define the boundaries or block of statements for the action to be taken when the condition is valid. Otherwise, only pone line is called operation after this condition.
3. Replace the synchronized “StringBuffer” class with an unsynchronized “StingBuilder” class – The methods in String Buffer are synchronized where possible, so that all operations on any given instance behave as if they happened in a sequential order that matches the order of the method calls made by each of the threads involved.

**Comparison/Contrast Tools**

**Does the tool analyze source or binary as input?**

SpotBugs, which is a continuation of FindBugs, analyzes binary data while others focus on the source code. It examines Java bytecode for bugs. SonarLint is an IDE extension that assists you in detecting and correcting code quality issues as you write it. SonarLint squiggles errors like a spell checker, allowing them to be corrected before committing code. The source is analyzed by SonarLint. It simplifies the management of code. It ensures that the system is easy to manage and safe.

**Which category of tools is it?**

SpotBugs is a Java program that detects potential bugs. There are four levels of potential errors: ***scariest, scary, troubling, and of concern***. This is a warning to the developer about the potential effect or severity of their work.

SpotBugs belongs to the following tool categories:

1. Type checking
2. Bug detection
3. Security assessment

SonarLint is an IDE extension that assists you in detecting and resolving code quality issues as you write. It squiggles errors in code, just like a spell checker, so they can be corrected until the code is committed.

SonarLint is a tool that belongs to the following group:

1. Style checking
2. Bug finding
3. Security review

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

As previously mentioned, SonarLint detects some new bugs in the software. The exception below is one of the findings recorded in SonarLint but not in SpotBugs. Add an end-condition to this loop – Since there is no condition to end the while(true) loop in line 33, it can go on indefinitely.

Graphical user interface, text, application

Description automatically generated

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

Remove this dServerSocket assignment from SonarLint. This bug also refers to the line 29 that was previously mentioned.

Graphical user interface, text, application

Description automatically generated

Write to a static field from an instance method in SpotBugs. dServerSocket has been declared as a static variable on line 26. This leads to a bug on line 29, where we attempt to write the instance method new ServerSocket (PORT) to a static variable. A static field is written by the instance process. If several instances are being exploited, this is difficult to get correct, and it's usually poor practice.

Graphical user interface, text, application

Description automatically generated

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

The known error, as discovered during the manual study, is that the FileReader object has been generated but not closed. Always closing the streams is a good coding practice to have because it prevents us from acting strangely. The exception "Use try-with-resources or close this FileReader in a finally clause" was used by SonarLint to find this bug. This error was not found by SpotBugs.

A picture containing text, screenshot, computer, computer

Description automatically generated

I have copy pasted the issues found using SonarLint in an excel file. The file is attached in the submission folder.

As seen in the screenshot, we can save the SpotBugs report as an XML file. This XML file is attached to the submission of the task.

Graphical user interface, application

Description automatically generated

**Results**

The results in the raw format, excel sheet, xlm format have been provided in the submission folder. The best screenshots have been included in the report raw\_result.pdf contains more images including the ones that have not been used.

**Part 2**

**Buggy Java Code**

The code provided is some code written in Java.

package com.aa.lookahead.one.common;

import java.util.ArrayList;

import java.util.List;

public class Sample {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

ArrayList<Integer> list = null;

for(int i = 0 ; i <12 ;i++)

{

list.add(i);

}

int[] strs = new int[10];

for (int i = 0 ; i < strs.length+1 ; i++)

{

strs[i] = i+1;

}

System.out.println(list.size());

System.out.println(strs[10]);

}

}

**Manual Analysis**

By just looking at the code, we can identify that

1. At Line 14 Array list has been initialized as ***null***. So, it will give null pointer exception error
2. At Line 21 ***i < strs.length+1*** will give array out of bound exception as strs is initialized as empty array.

**SpotBugs report**

Could not find any bugs in spotbugs

**SonarLint report**

Text, table

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

SonarLint showed the Null pointer exception at two locations at Line 12 and at Line 15. The initial manual analysis for identifying the null pointer exception was correct.

**Fixes**

In order to fix the above buggy java code

1. At line 12 instead of ArrayList<Integer> list =**null**

Initialize it as ArrayList<Integer> list = **new** ArrayList();

This would fix the null pointer exception error

1. At line 19 instead of **for** (**int** i = 0 ; i < 10 ; i++)

Change to **for** (**int** i = 0 ; i < 10 ; i++)

This would fix the the issue.

**Fixed Code**

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Sample {

/\*\*

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

ArrayList<Integer> list = **new** ArrayList();

**for**(**int** i = 0 ; i <12 ;i++)

{

list.add(i);

}

**int**[] strs = **new** **int**[10];

**for** (**int** i = 0 ; i < 10 ; i++)

{

strs[i] = i+1;

}

System.***out***.println(list.size());

System.***out***.println(strs[9]);

}

}

**SonarLint report after Fixes**

Graphical user interface, text, application

Description automatically generated

Null pointer exceptions have been fixed.