ASSIGNMENT – 10

Name: **Sudharsan Srinivasan**

UTA ID: **1001755919**

- The aim of this assignment is to analyse the give code and use our own code to manual analyse and find bugs or errors. Also, we make use of inbuilt plugins provided by Eclipse (we use **SpotBugs** and **Sonarlint**) for this assignment to analyse errors.
- Given code is a **SimpleWebServer.java** as shown below:

SimpleWebServer.java

```
43          responds with the file the user requested or
44          a HTTP error code. */
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
51      /* used to write data to the client */
52      OutputStreamWriter osw =
53          new OutputStreamWriter (s.getOutputStream());
54
55      /* read the HTTP request from the client */
56      String request = br.readLine();
57
58      String command = null;
59      String pathname = null;
```

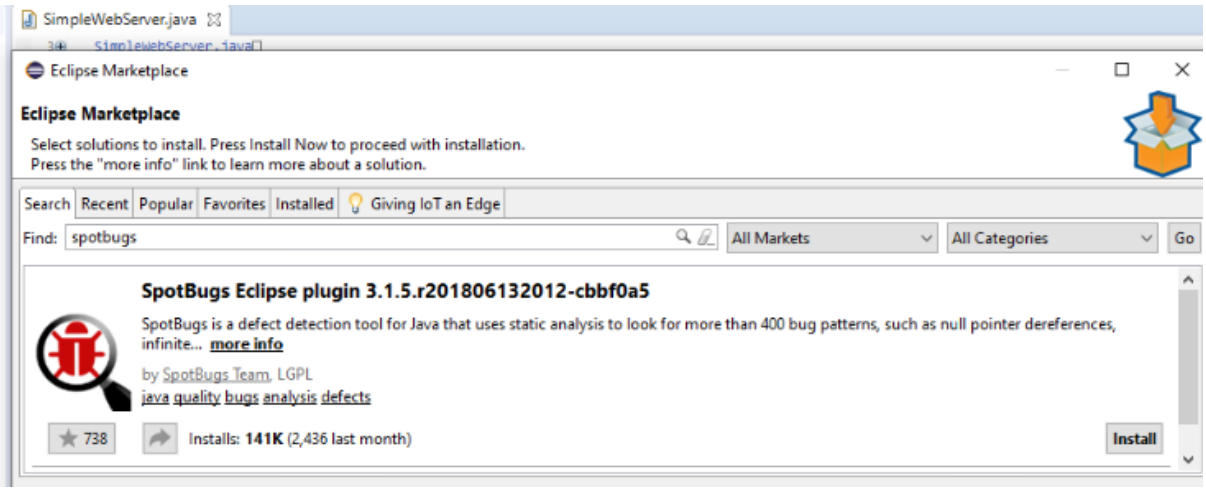
Manual Analysis:

On manually analysing the code without any plugins, we are able to locate few problems with the code. As seen above in Line 48, **BufferReader** object has been created for **InputStream** and **OutputStreamWriter** object for **OutputStream** and used. But, nowhere in the code they have been closed. It is important to note that, we have to call **close()** to make sure they are flushed out. Leaving them unclosed could result in wrong outputs or performance issues.

Tool Choices & Versions:

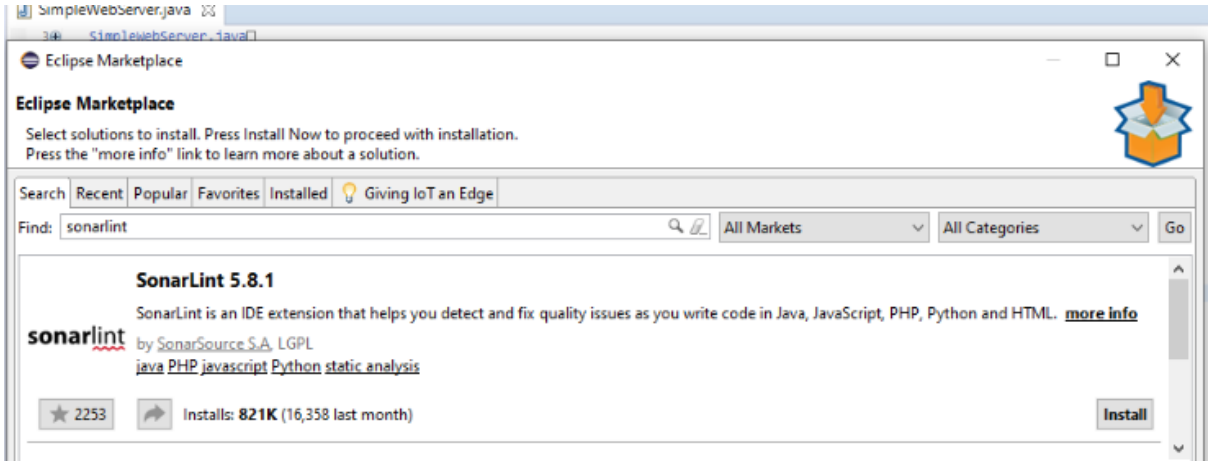
The two plugins/tools that will be used for this task are **SpotBugs** and **Sonarlint.**

a. SpotBugs:



The version used is **3.1.5.r201806132012-cbbf0a5**

b. **Sonarlint:**

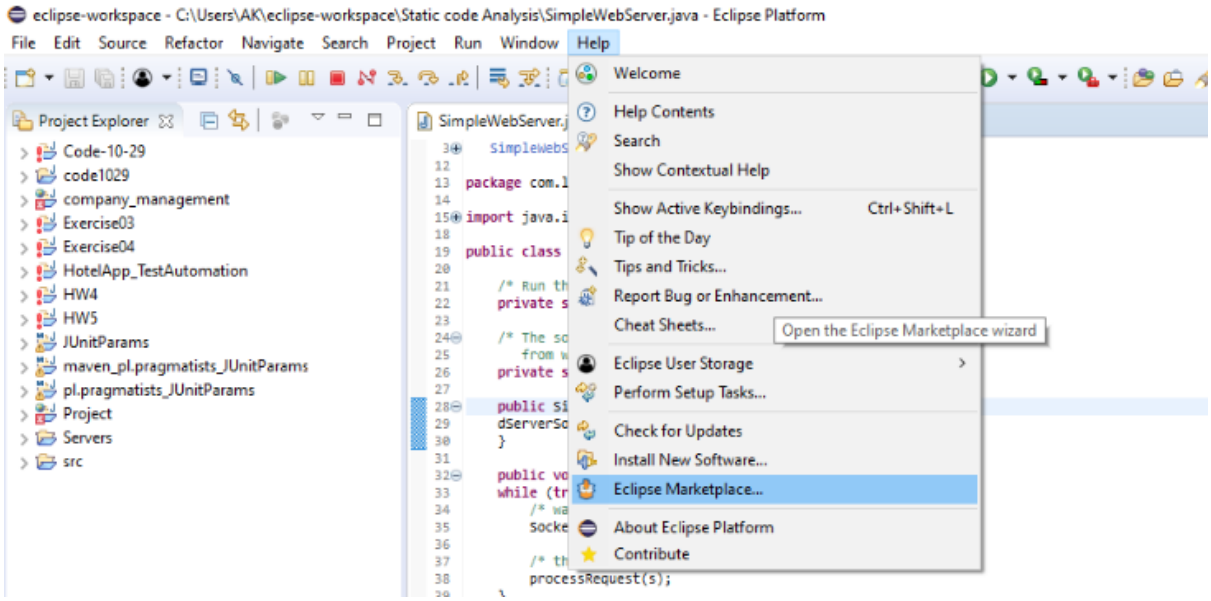


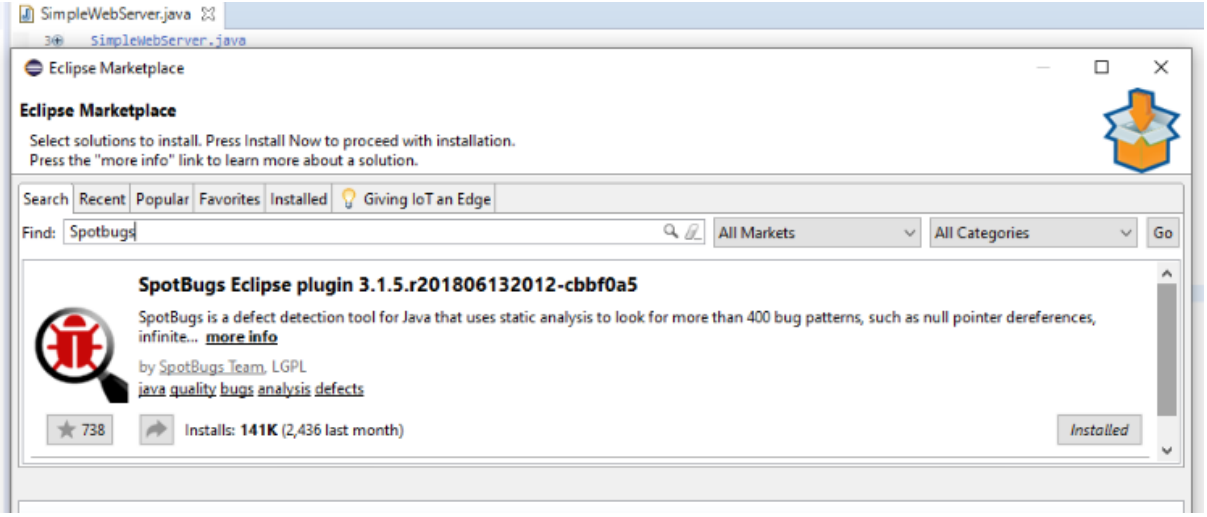
The version used here is **5.8.1**

Tool Invocation Process:

a) **SpotBugs:**

- To install this tool, setup this tool, we navigate to **Help -> Eclipse Marketplace** and search for SpotBugs and install it as shown below.

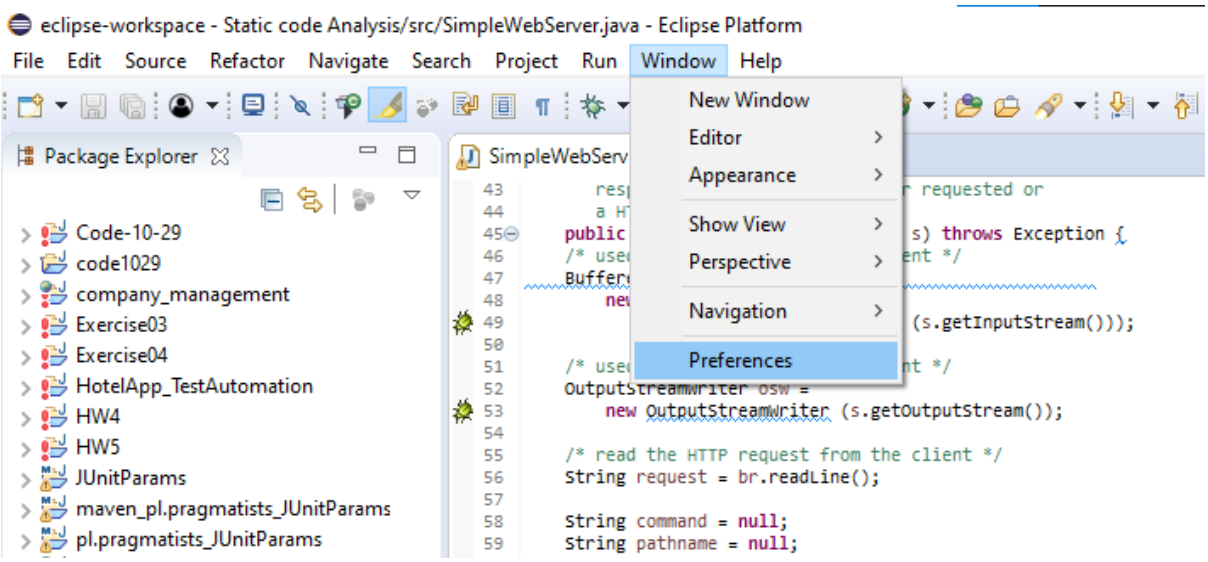




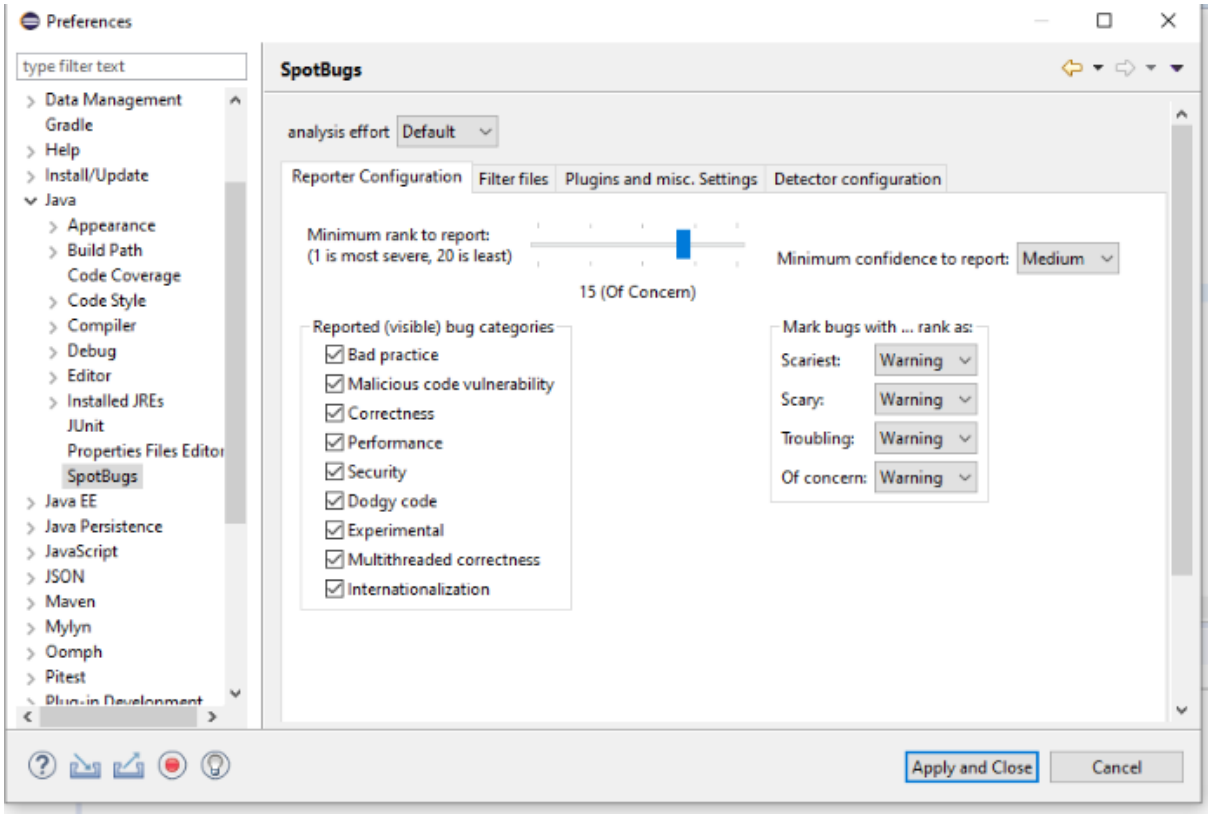
- The above screenshot shows that the SpotBugs plugin has been installed successfully.

SpotBugs settings and Plugins:

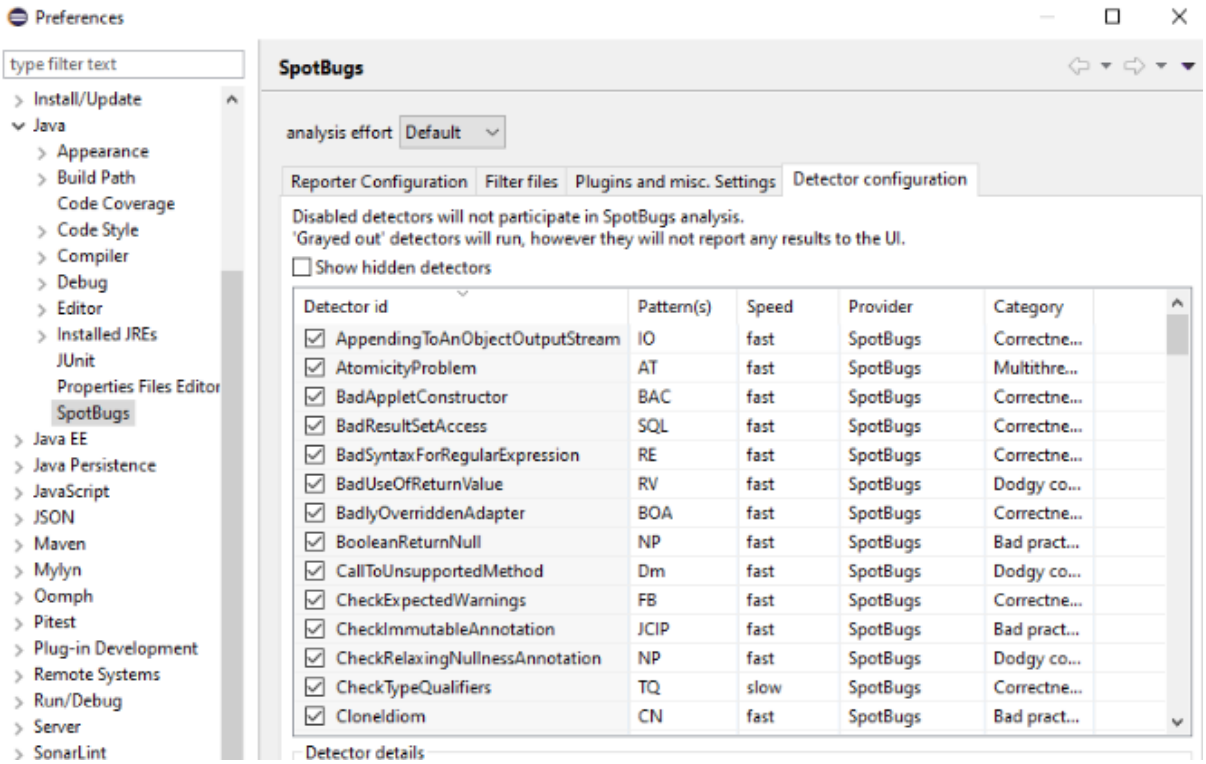
- Now the settings of this plugin will have to be configured to make sure it is able to identify all possible bugs from the code.
- To do so, head to **Windows -> Preferences** and navigate to **Java -> SpotBugs** tab as seen below.



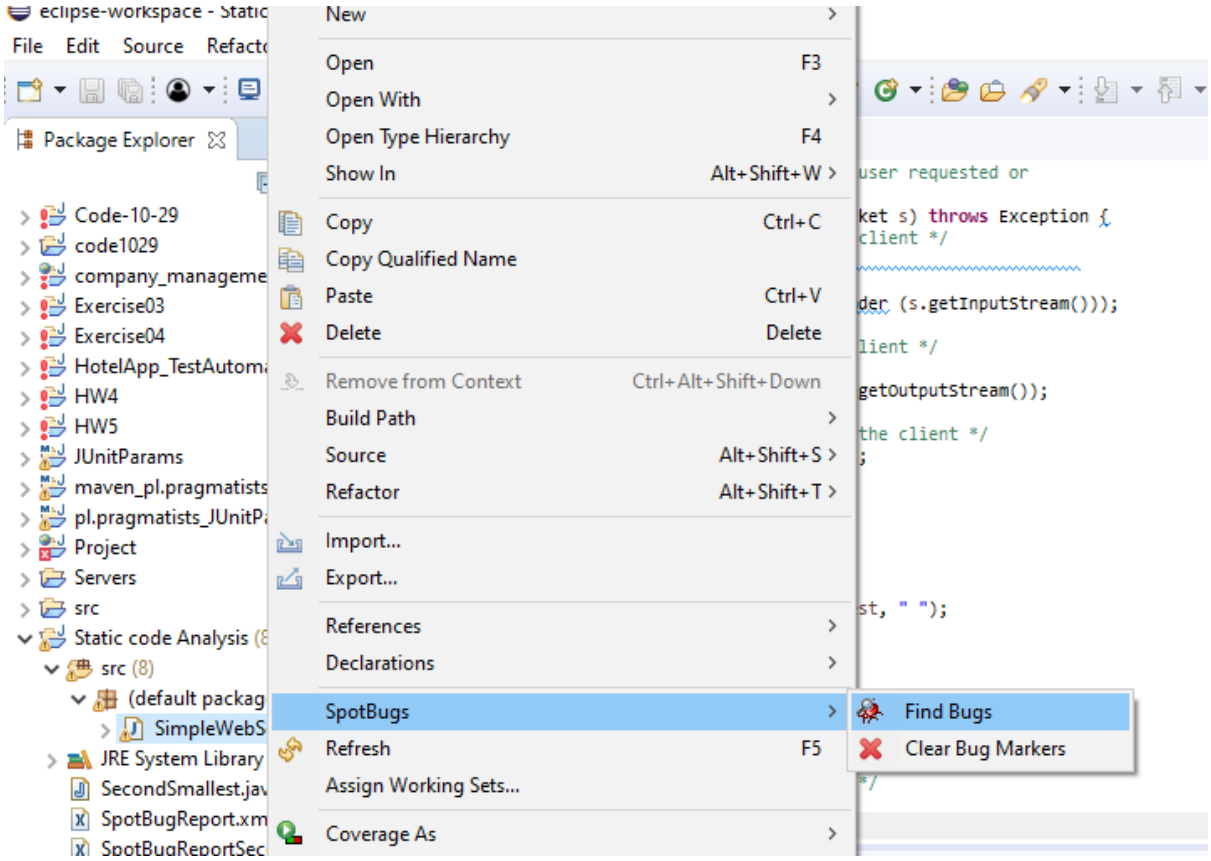
- In the Preference section of SpotBugs, we have to change the **Report Configuration** tab settings where we check all the bug categories for the plugin to be able to identify all of them.



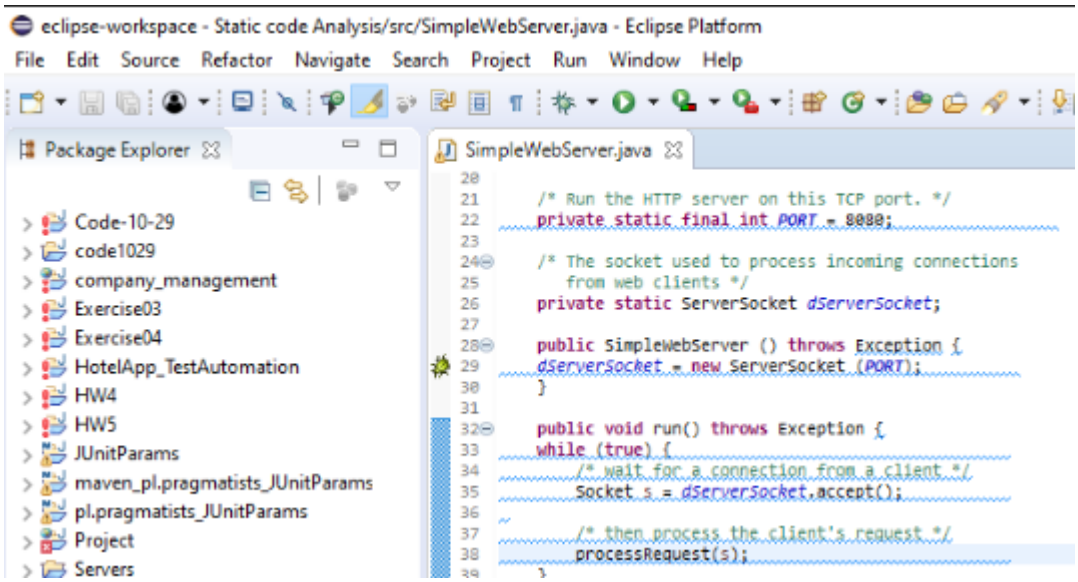
- Then, head over to **Detector Configuration** tab and check all the detectors and save them. These updated settings will help the plugin to identify all possible bugs in the code effectively.



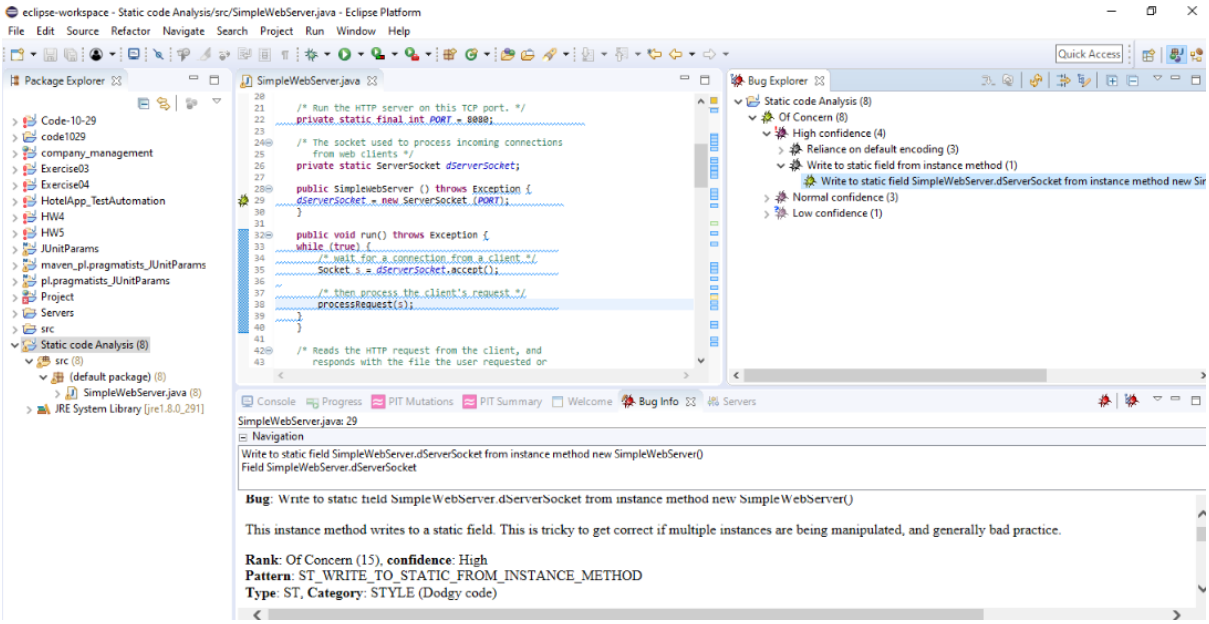
- After all the necessary settings are updated and saved, we use this plugin to find bugs in the code. We do this by right clicking on the file **SimpleWebServer.java** and do **SpotBugs -> Find Bugs**



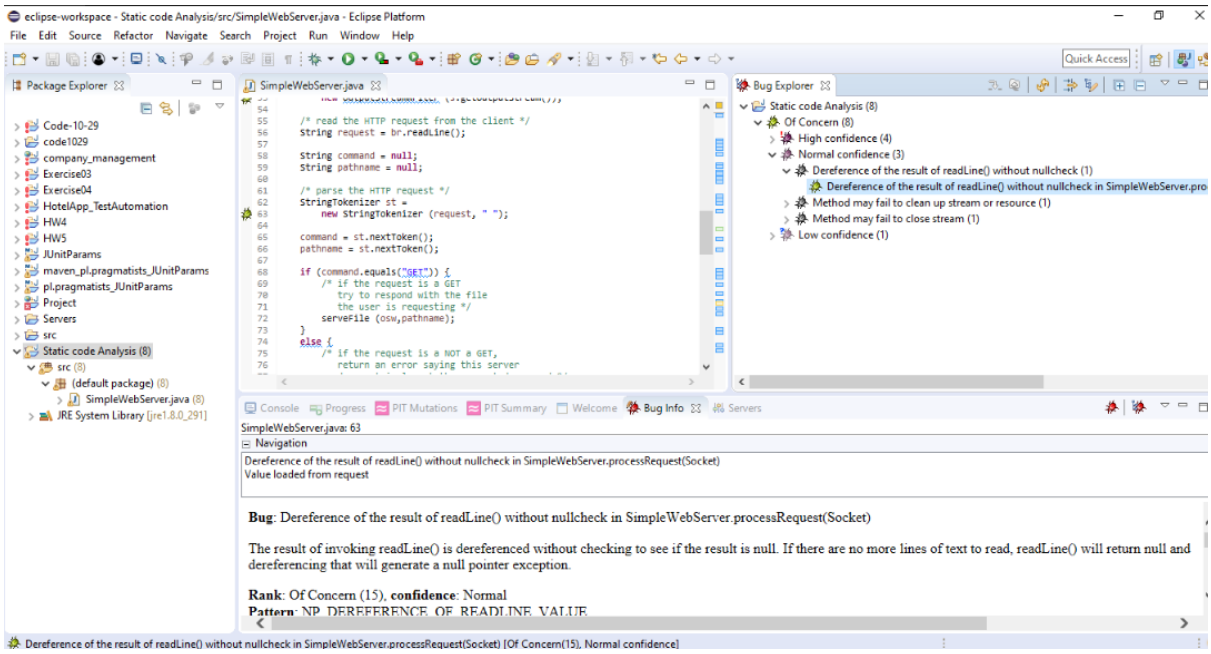
- The plugin runs and detects a bug in **Line 29**



- We can find out more info about the bug in the **Bug Explorer/Bug Info** tab as below.

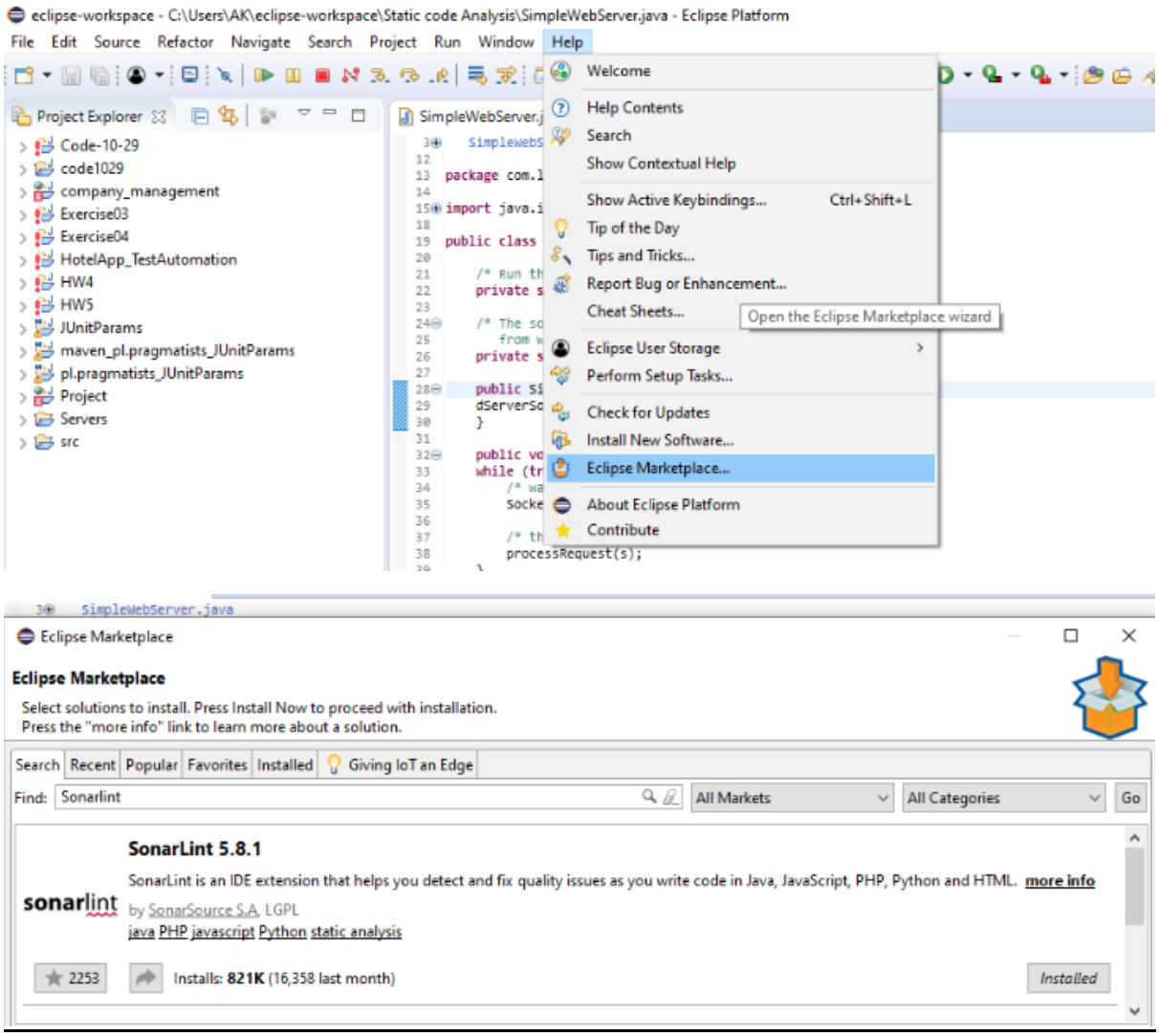


- The bug is present in the code, because **dServerSocket** static variable is declared earlier in Line 26. But in Line 29, the method **ServerSocket(PORT)** is trying to write into a static variable which is not advisable and may lead to undesirable outputs.
- Also, there is another bug found in **Line 63**. This bug is because there is a **request** string variable which stores the output of **readLine()**. But NULLCHECK is not done on the variable resulting in the bug.



b) <u>Sonarlint:</u>

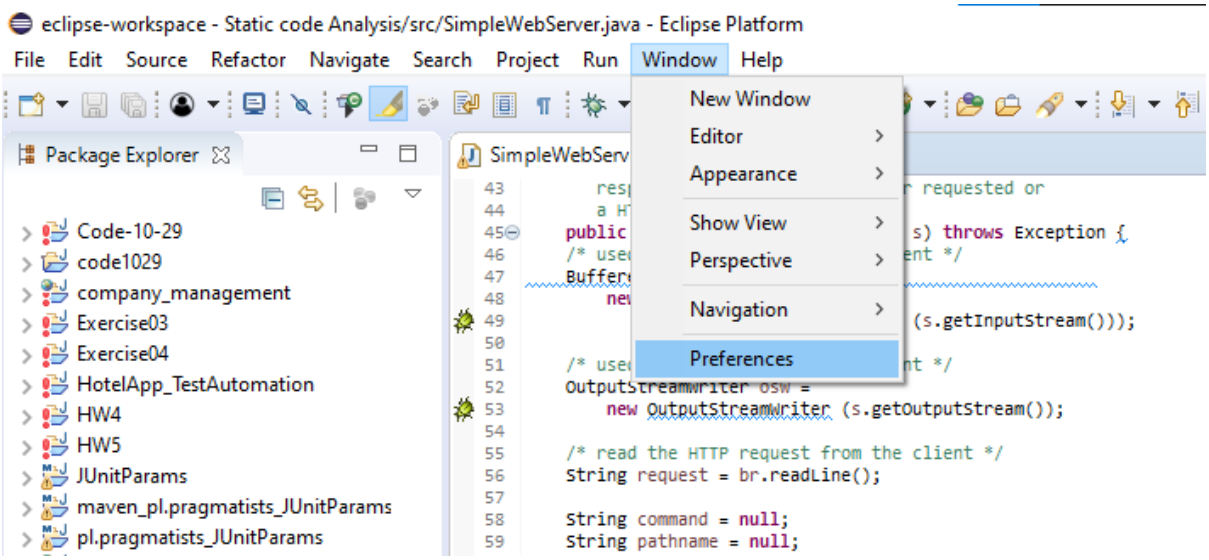
- Similar to SpotBugs plugin, we have to install Sonarlint, modify the preferences and settings to make it easier for it to detect all issues with the code. To install Sonarlint, head over to **Help -> Eclipse Marketplace** just like the previous plugin, search for Sonarlint and install it.



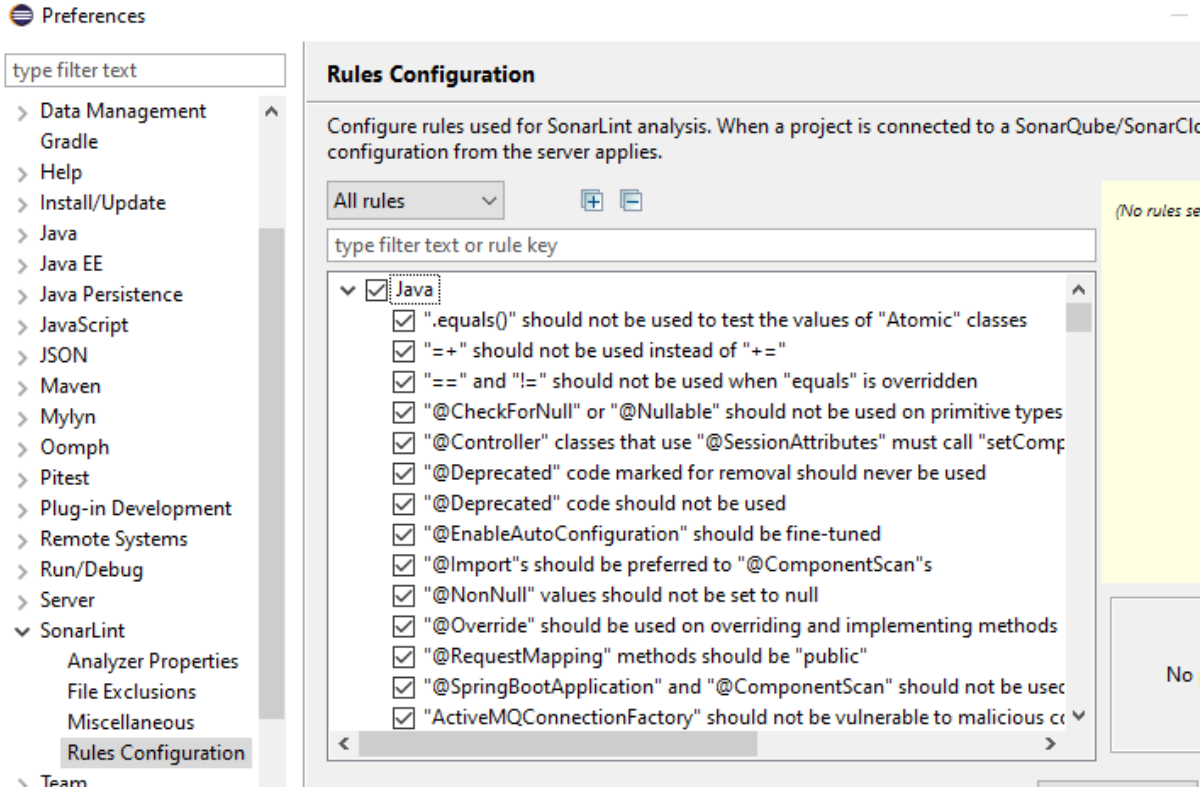
- The above image shows that the plugin has been successfully installed.

Sonarlint Settings and Plugins:

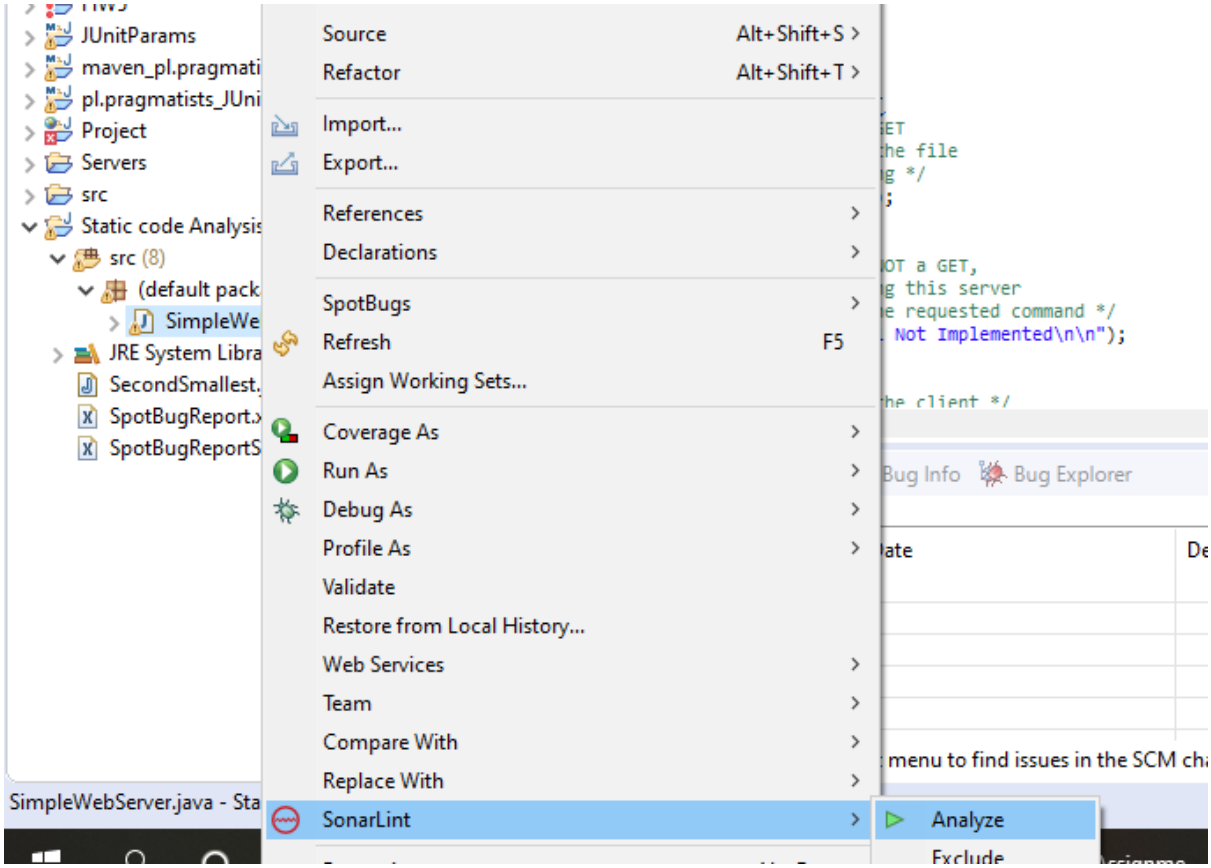
- We have to configure preferences and settings of Sonarlint plugin to identify errors in the code effectively.
- To do so, go to **Windows -> Preferences** and go to **Sonarlint** tab.



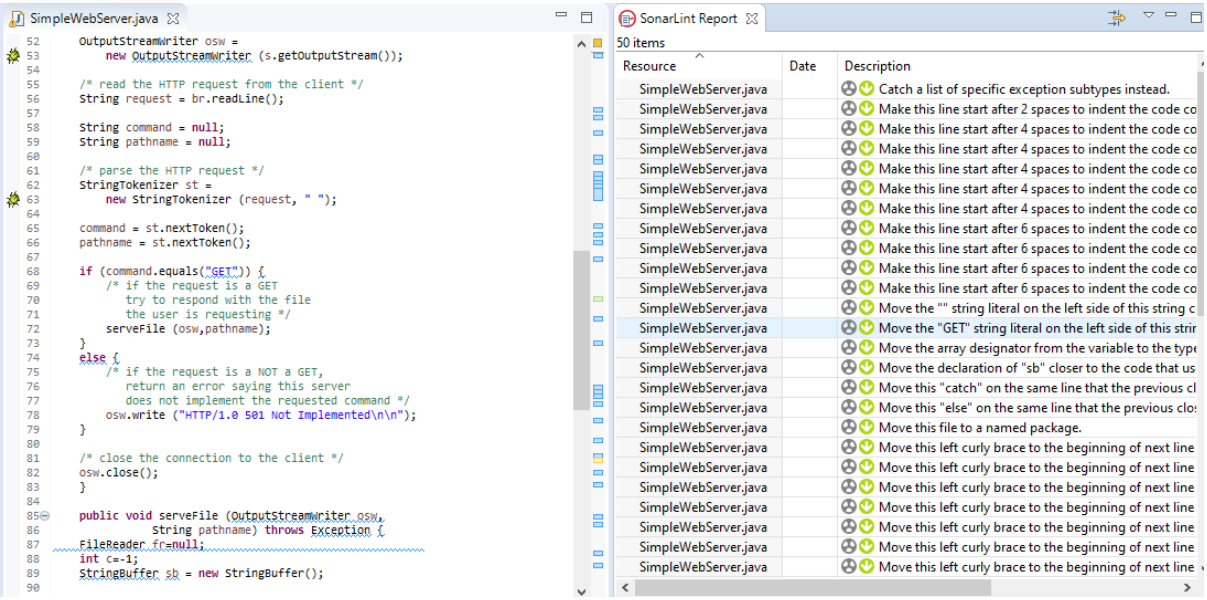
- To make sure, that the plugin detects all possible issues, we check all the boxes corresponding to rules of Java program.



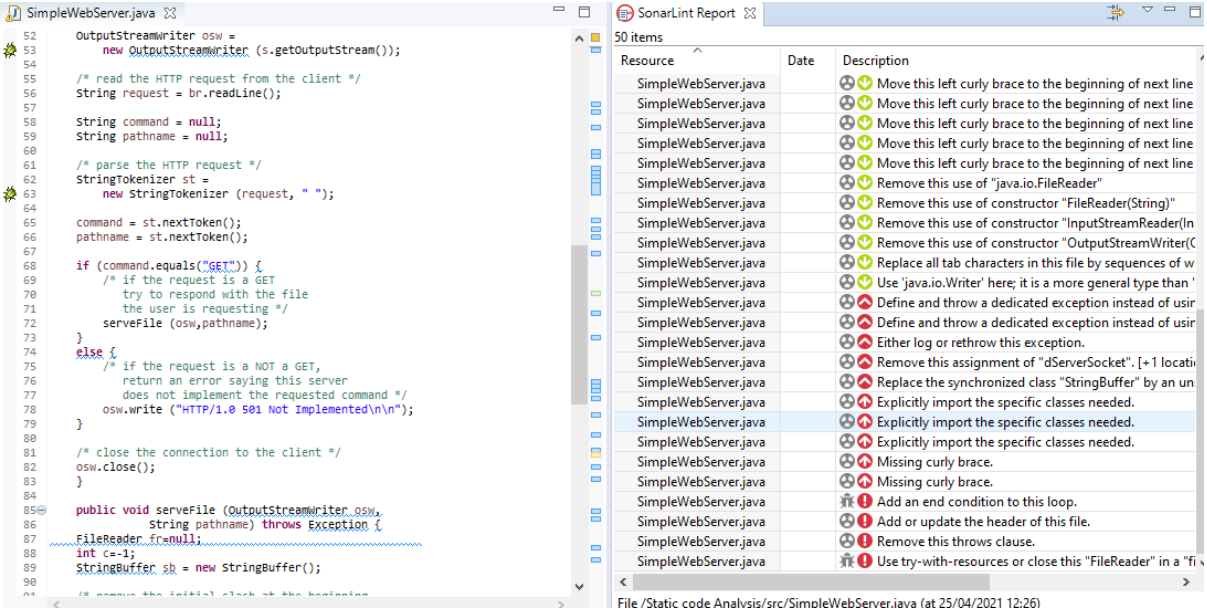
- After making the necessary updates to the settings of the plugin, we go ahead and run Sonarlint Report on the given Java program. To do so, right click on the file and select **SonarLint -> Analyse**



- A brief view of the SonarLint report is seen below. It can be seen from **SonarLint Report** tab. The green arrow indicates that they passed the check and red arrow indicates some of the potential issues.



- Some of the various bugs that were found as part of SonarLint are listed below.



- **<u>Missing curly brace:</u>**

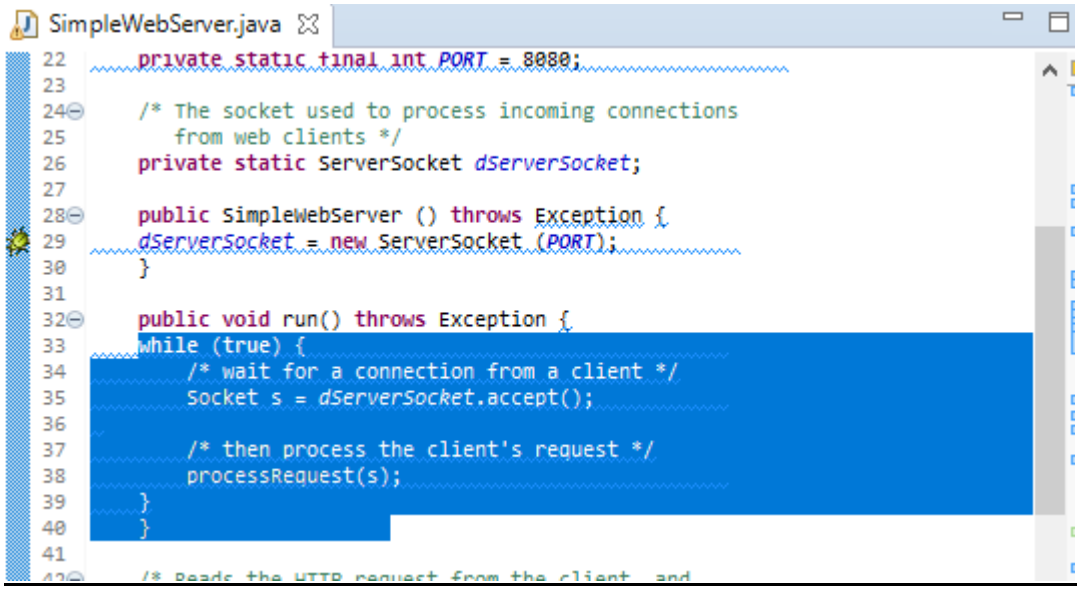
  This error is because we don't explicitly use {} for **if…else** loops in the program. This might lead to only one line of the loop getting executed and not the entire block, hence listed as an issue.

- **<u>Explicitly import the specific classes needed:</u>**

  The header file we have imported for the code is just **import java.util.*** assuming that this header file will inbuilt call and import all the necessary classes needed for the program. But a good practice would be separately adding all the necessary header files.

- **<u>Add an end condition to this loop:</u>**

  In Line 32 of the code, we have a run() function which uses a while loop to check for connection. But there is no end condition the loop. So potentially, this could end up being an infinite loop.



- **Define and throw a dedicated Exception:**

```
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      }
```

```
42      /* Reads the HTTP request from the client, and
43         responds with the file the user requested or
44         a HTTP error code. */
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
```

```
84
85      public void serveFile (OutputStreamWriter osw,
86                  String pathname) throws Exception {
87      FileReader fr=null;
88      int c=-1;
89      StringBuffer sb = new StringBuffer();
```

In different function of the program, where we use throw Exception, we are not explicitly mentioned the type of Exception we want the function to throw, ex: **IOException, NullPointerException** etc, hence the error.

Tool Comparison and Contrast:

Tool analysis of input as source or binary:

- SpotBugs plugins analyses the program as **Java Bytecode** to find potential issues with the code, because this plugin is an extension of FindBugs plugin.
- SonarLint plugin works more with respect to the view and presentability of the code, in that it checks for factors like code readability, security and syntactical flaws in the code etc.,

Tool Category:

- SpotBugs analyses the code and classifies the bugs into 4 categories namely **Scariest, Scary, Troubling, Of Concern** from the order of most vulnerable to least. It comes under the following category of tools which are

  Type Check

  Bug finding

  Security Review
- SonarLint works more with code readability and security and therefore come under the below category:

  Style Check

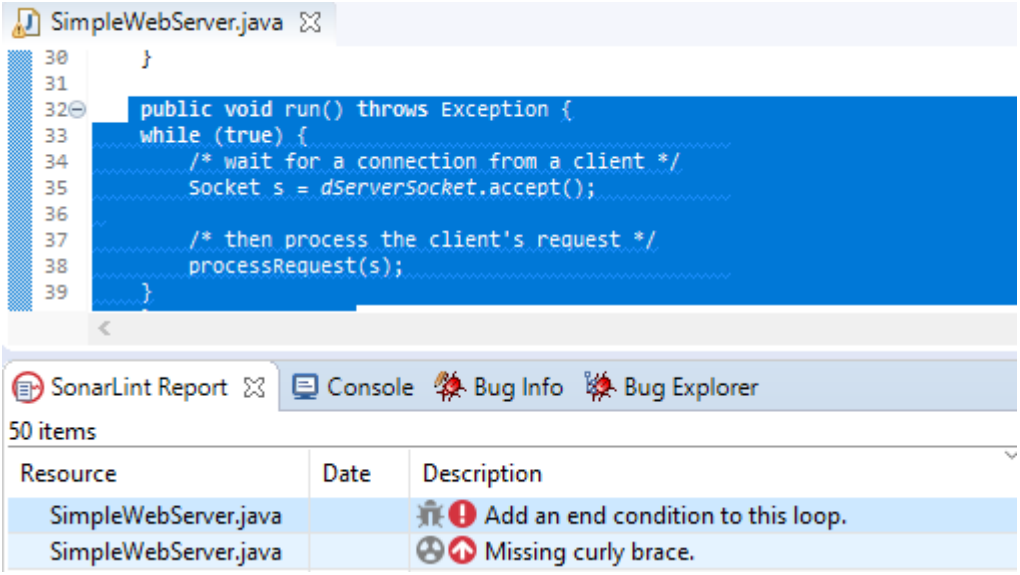
  Bug finding

  Security Review

Bug reported by only one tool:

Add an end condition to while loop:

- As seen earlier, this error occurs in run() function where while() loop does not have a corresponding end condition for the loop to terminate. This is reported by Sonarlint as seen below.

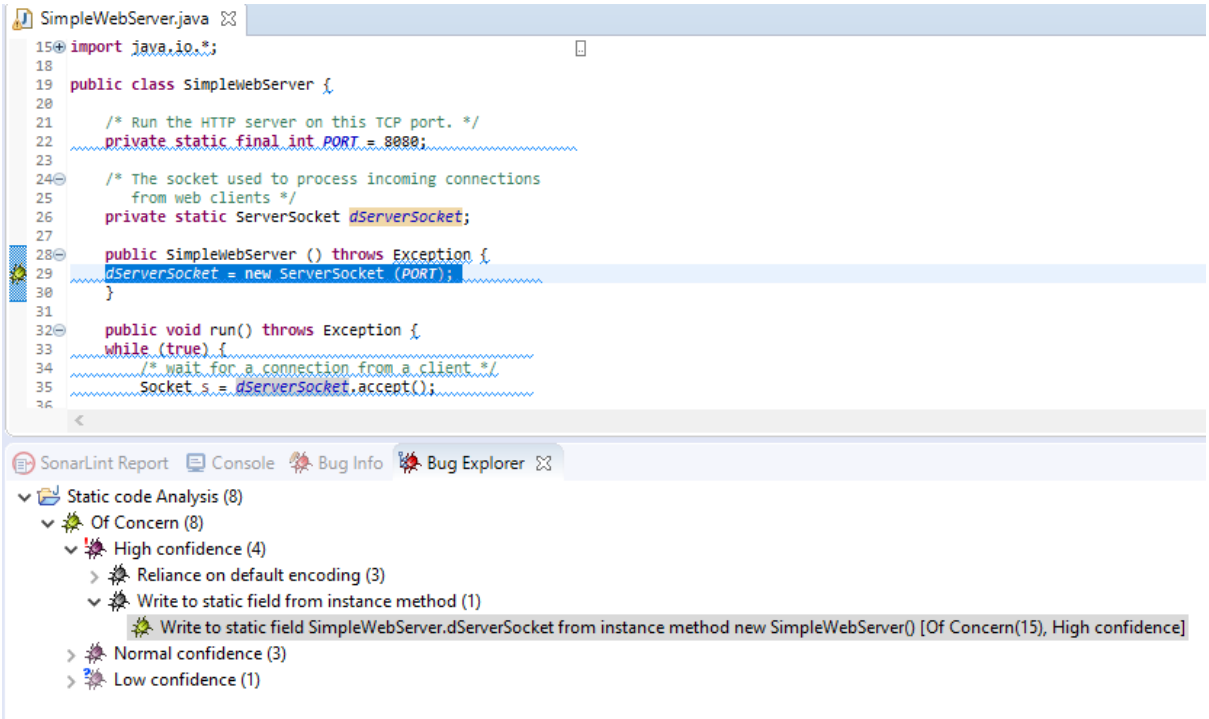


- But this same error is not reported as part of SpotBug analysis.

Bug reported by both tools:

- The bug we saw earlier while testing out SpotBugs plugin, corresponding to **dServerSocket** has been reported by both the plugins

SpotBugs:



SonarLint:

SimpleWebServer.java

```
18
19 public class SimpleWebServer {
20
21     /* Run the HTTP server on this TCP port. */
22     private static final int PORT = 8080;
23
24     /* The socket used to process incoming connections
25        from web clients */
26     private static ServerSocket dServerSocket;
27
28     public SimpleWebServer () throws Exception {
29     dServerSocket = new ServerSocket (PORT);
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     }
```

SonarLint Report | Console | Bug Info | Bug Explorer

50 items

Resource	Date	Description
SimpleWebServer.java		Remove this assignment of "dServerSocket". [+1 location]

- As seen from the above images of bug report from both SpotBugs and Sonarlint, the bug related to declaring a static variable **dSeverSocket** and later have a **ServerSocket(PORT)** write into the static variable is reported correctly by both the plugins.

Documenting known flaws in the code;

- Earlier in manual analysis, it was found that the stream object used were not appropriately closed. Sonarlint was able to identify this bug correctly.

SimpleWebServer.java

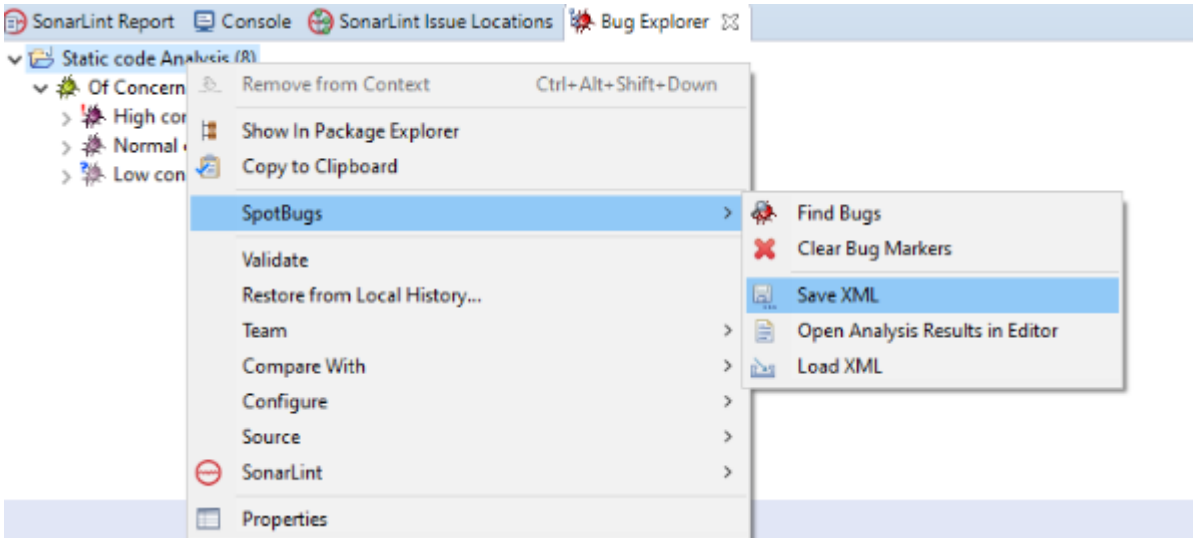
```
99         pathname="index.html";
100
101     /* try to open file specified by pathname */
102     try {
103         fr = new FileReader (pathname);
104         c = fr.read();
105     }
106     catch (Exception e) {
107         /* if the file is not found,return the
108            appropriate HTTP response code  */
109         osw.write ("HTTP/1.0 404 Not Found\n\n");
110         return;
111     }
112
113     /* if the requested file can be successfully opened
114        and read, then return an OK response code and
115        send the contents of the file */
```

SonarLint Report | Console | Bug Info | Bug Explorer

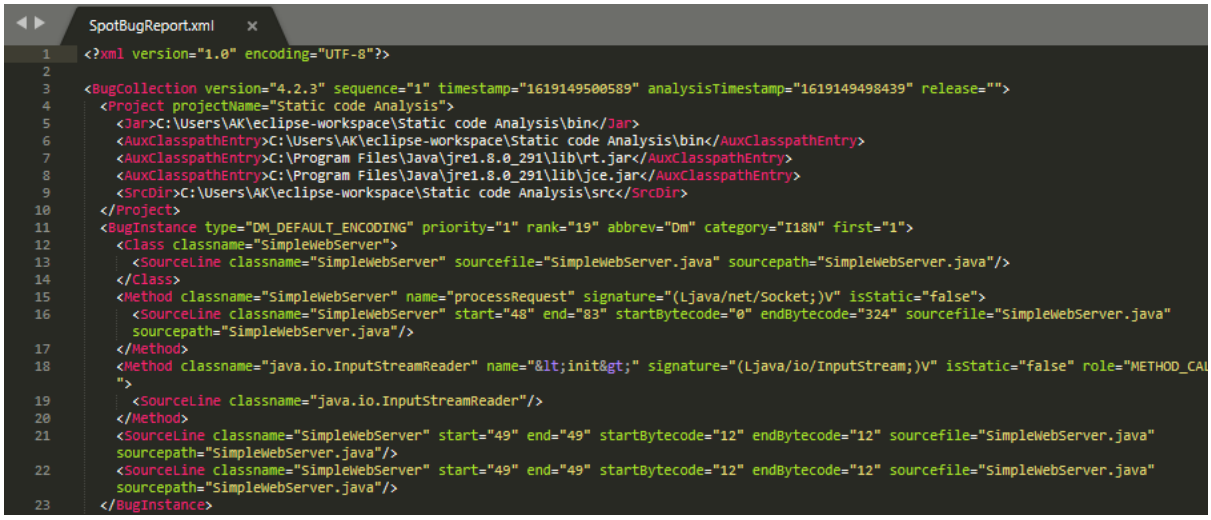
50 items

Resource	Date	Description
SimpleWebServer.java		Use try-with-resources or close this "FileReader" in a "finally" clause.

- But SpotBug plugin could not identify this issue we earlier found during manual analysis.
- Finally, the reports of SpotBugs can be extracted in the form of XML document by right clicking on the folder/file name under **Bug Explorer** tab and select **SpotBugs -> Save XML** option.



- A preview of the exported XML can be seen below.



SpotBugReport.xml

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

<BugCollection version="4.2.3" sequence="1" timestamp="1619149500589" analysisTimestamp="1619149498439" release="">
  <Project projectName="Static code Analysis">
    <Jar>C:\Users\AK\eclipse-workspace\Static code Analysis\bin</Jar>
    <AuxClasspathEntry>C:\Users\AK\eclipse-workspace\Static code Analysis\bin</AuxClasspathEntry>
    <AuxClasspathEntry>C:\Program Files\Java\jre1.8.0_291\lib\rt.jar</AuxClasspathEntry>
    <AuxClasspathEntry>C:\Program Files\Java\jre1.8.0_291\lib\jce.jar</AuxClasspathEntry>
    <SrcDir>C:\Users\AK\eclipse-workspace\Static code Analysis\src</SrcDir>
  </Project>
  <BugInstance type="DM_DEFAULT_ENCODING" priority="1" rank="19" abbrev="Dm" category="I18N" first="1">
    <Class classname="SimpleWebServer">
      <SourceLine classname="SimpleWebServer" sourcefile="SimpleWebServer.java" sourcepath="SimpleWebServer.java"/>
    </Class>
    <Method classname="SimpleWebServer" name="processRequest" signature="(Ljava/net/Socket;)V" isStatic="false">
      <SourceLine classname="SimpleWebServer" start="48" end="83" startBytecode="0" endBytecode="324" sourcefile="SimpleWebServer.java"
      sourcepath="SimpleWebServer.java"/>
    </Method>
    <Method classname="java.io.InputStreamReader" name="&lt;init&gt;" signature="(Ljava/io/InputStream;)V" isStatic="false" role="METHOD_CAL
    ">
      <SourceLine classname="java.io.InputStreamReader"/>
    </Method>
    <SourceLine classname="SimpleWebServer" start="49" end="49" startBytecode="12" endBytecode="12" sourcefile="SimpleWebServer.java"
    sourcepath="SimpleWebServer.java"/>
    <SourceLine classname="SimpleWebServer" start="49" end="49" startBytecode="12" endBytecode="12" sourcefile="SimpleWebServer.java"
    sourcepath="SimpleWebServer.java"/>
  </BugInstance>
```

Part 2 – Analyse Own Code:

- In this section, we use our own code and analyse the issues in the code using the above plugins and try to fix them. We use the already configured settings of both the plugins we made earlier for the previous task.
- The code used for the task can be seen in the below image.

SecondSmallest.java

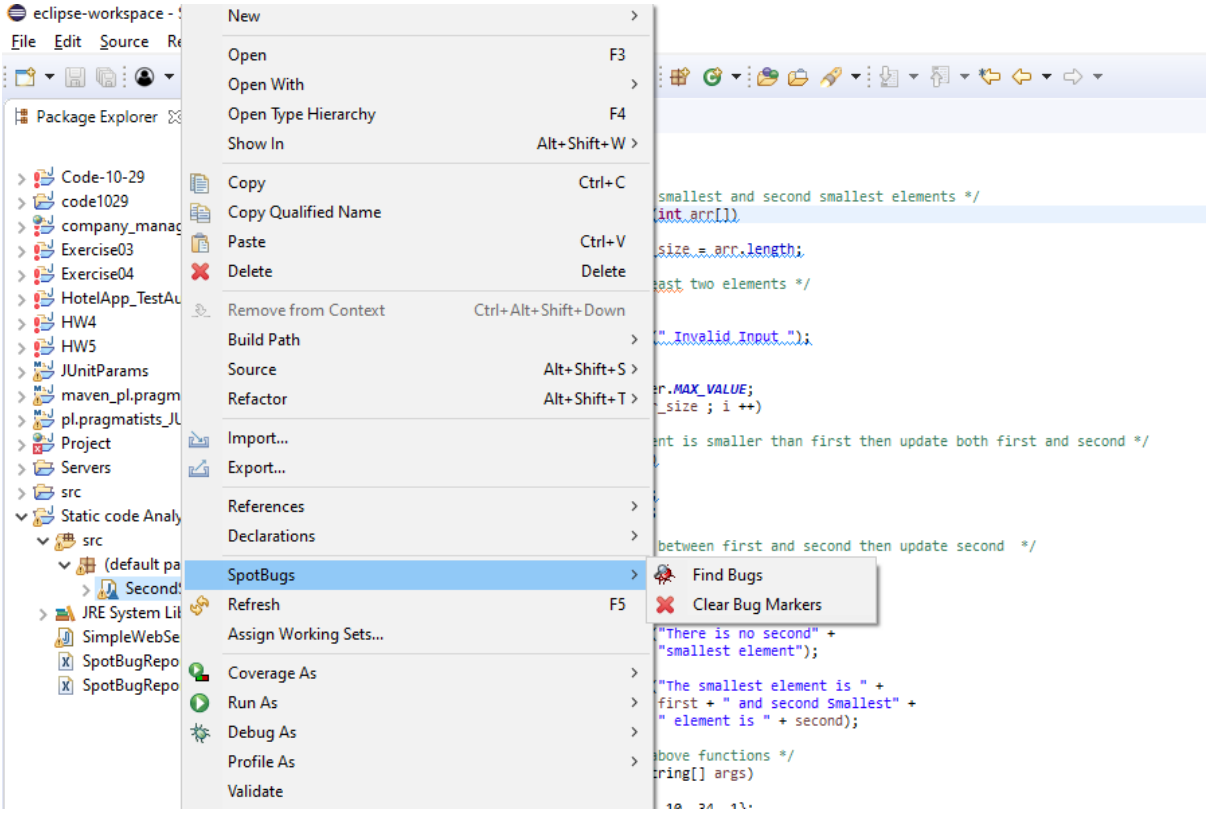
```java
import java.io.*;
 class SecondSmallest
{
    /* Function to print first smallest and second smallest elements */
    static void Print2Smallest(int arr[])
    {
        int first, second, arr_size = arr.length;

        /* There should be atleast two elements */
        if (arr_size < 2)
        {
            System.out.println(" Invalid Input ");
            return;
        }
        first = second = Integer.MAX_VALUE;
        for (int i = 0; i < arr_size ; i ++)
        {
            /* If current element is smaller than first then update both first and second */
            if (arr[i] < first)
            {
                second = first;
                first = arr[i];
            }
            /* If arr[i] is in between first and second then update second  */
            else if (arr[i] < second && arr[i] != first)
                second = arr[i];
        }
        if(second == Integer.MAX_VALUE)
            System.out.println("There is no second" +
                               "smallest element");
        else
            System.out.println("The smallest element is " +
                               first + " and second Smallest" +
                               " element is " + second);
    }
    /* Driver program to test above functions */
    public static void main (String[] args)
    {
        int arr[] = {12, 13, 1, 10, 34, 1};
        Print2Smallest(arr);
    }
}
```

Manual Analysis:

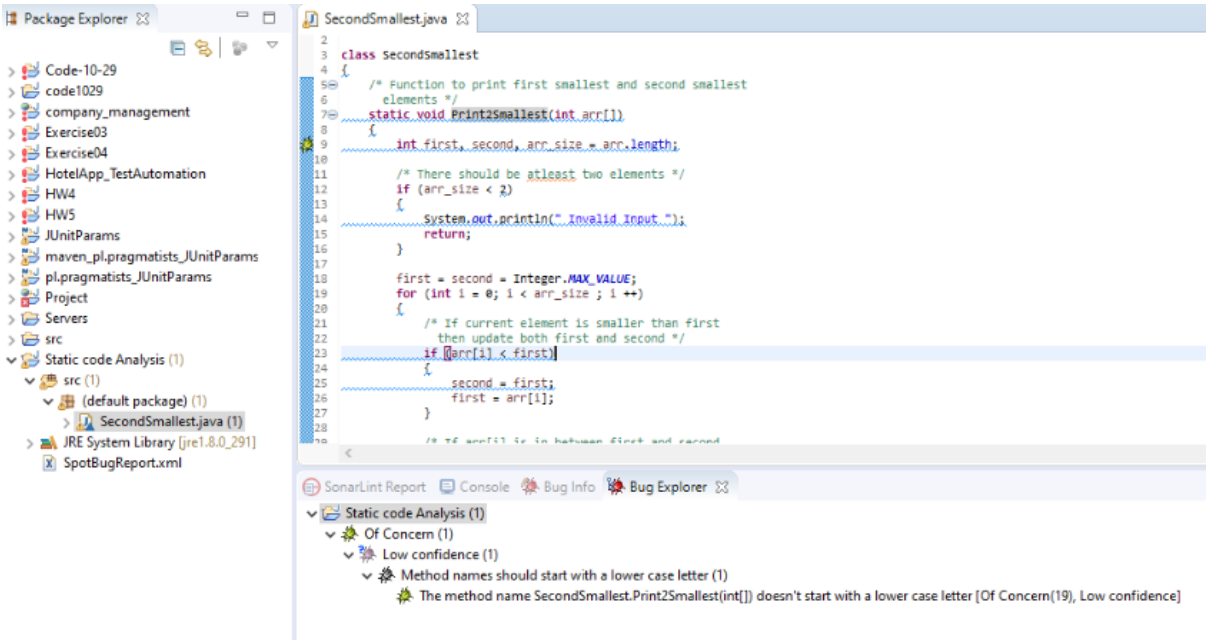
- On manually analysing the code, we can see that **if…else** condition loop used for print statement does not contain **curly braces**. In cases of multiple statements inside the loop, this could be a potential issue as we want the entire block of statements inside the loop to get executed and not just one line.

Analysis with SpotBugs:

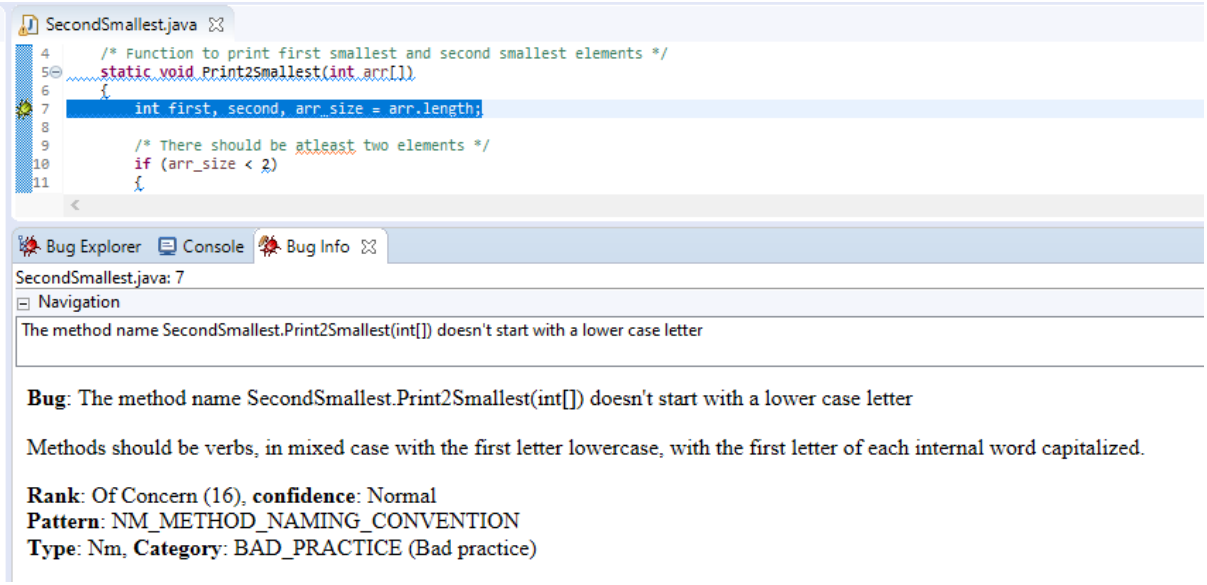
- We now try to find the bugs in the code using the plugin **SpotBugs** as we can see below.



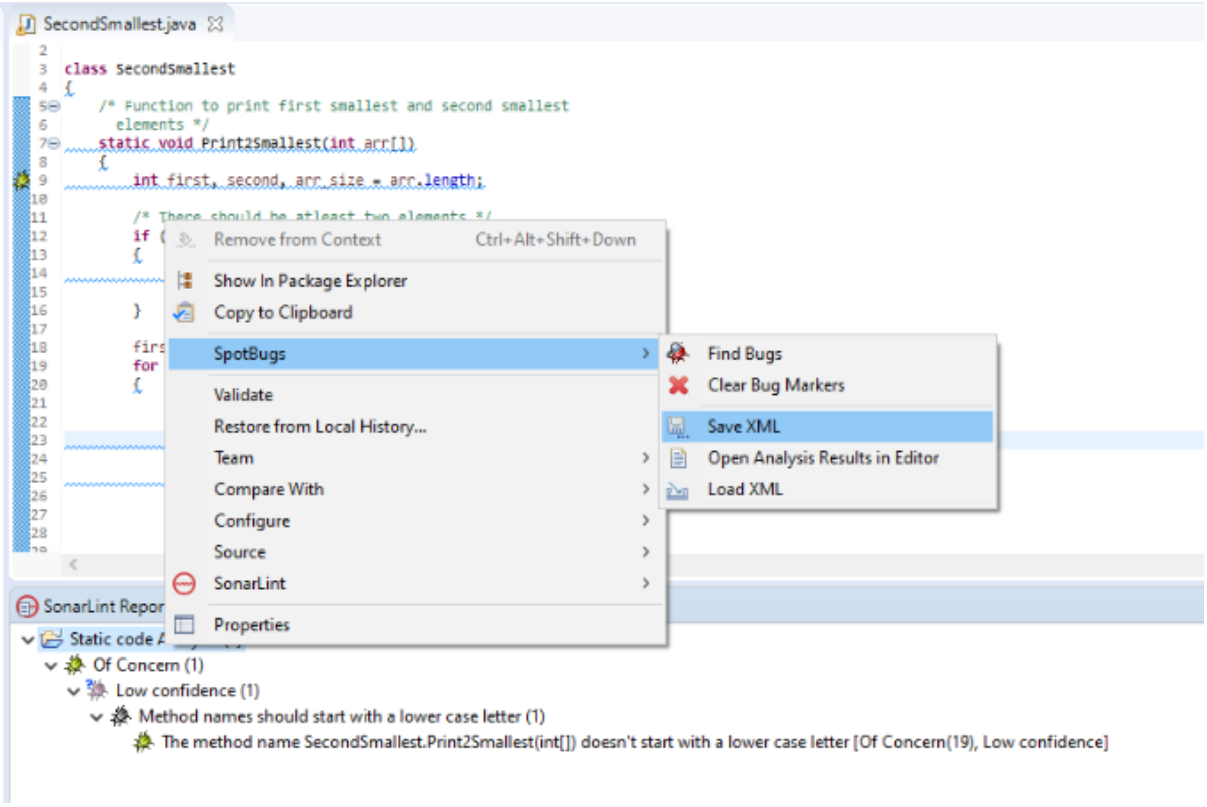
- The plugin was able to identify one issue with the code as seen below



- The error corresponds to the name of the function that is used in the code.



- As seen in the **Bug Info** section, we have used an uppercase letter for the first letter of the function name, which ideally is not a good coding practice and therefore, SpotBugs identifies this as a potential bug. Renaming the function to use Lowecase letter for the first letter of the function will fix the issue.
- We also extract the report of the bugs into an XML file as shown below:



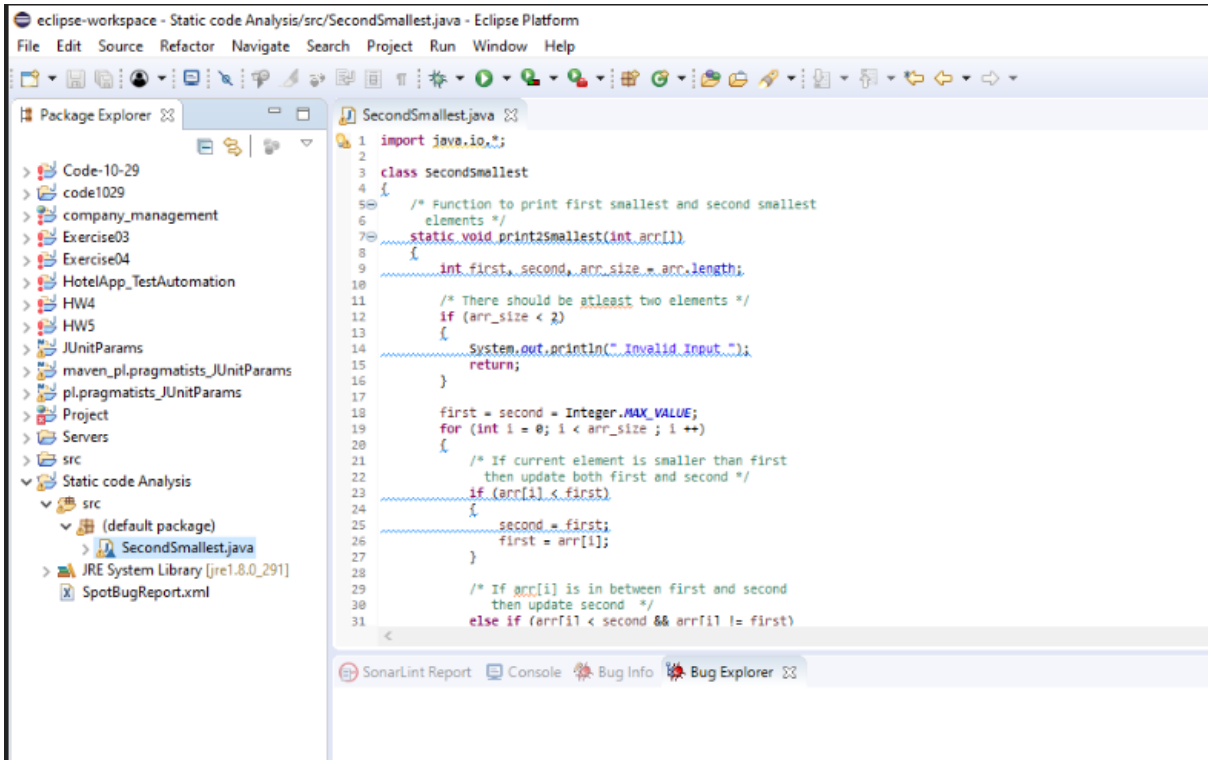
SpotBug run after error fix:

- The updated code with the error fixed regarding the function name is as follows;

SecondSmallest.java

```java
import java.io.*;
 class SecondSmallest
{
    /* Function to print first smallest and second smallest elements */
    static void print2Smallest(int arr[])
    {
        int first, second, arr_size = arr.length;

        /* There should be atleast two elements */
        if (arr_size < 2)
        {
            System.out.println(" Invalid Input ");
            return;
        }
        first = second = Integer.MAX_VALUE;
        for (int i = 0; i < arr_size ; i ++)
        {
            /* If current element is smaller than first then update both first and second */
            if (arr[i] < first)
            {
                second = first;
                first = arr[i];
            }
            /* If arr[i] is in between first and second then update second  */
            else if (arr[i] < second && arr[i] != first)
                second = arr[i];
        }
        if(second == Integer.MAX_VALUE)
            System.out.println("There is no second" +
                               "smallest element");
        else
            System.out.println("The smallest element is " +
                               first + " and second Smallest" +
                               " element is " + second);
    }
    /* Driver program to test above functions */
    public static void main (String[] args)
    {
        int arr[] = {12, 13, 1, 10, 34, 1};
        print2Smallest(arr);
    }
}
```

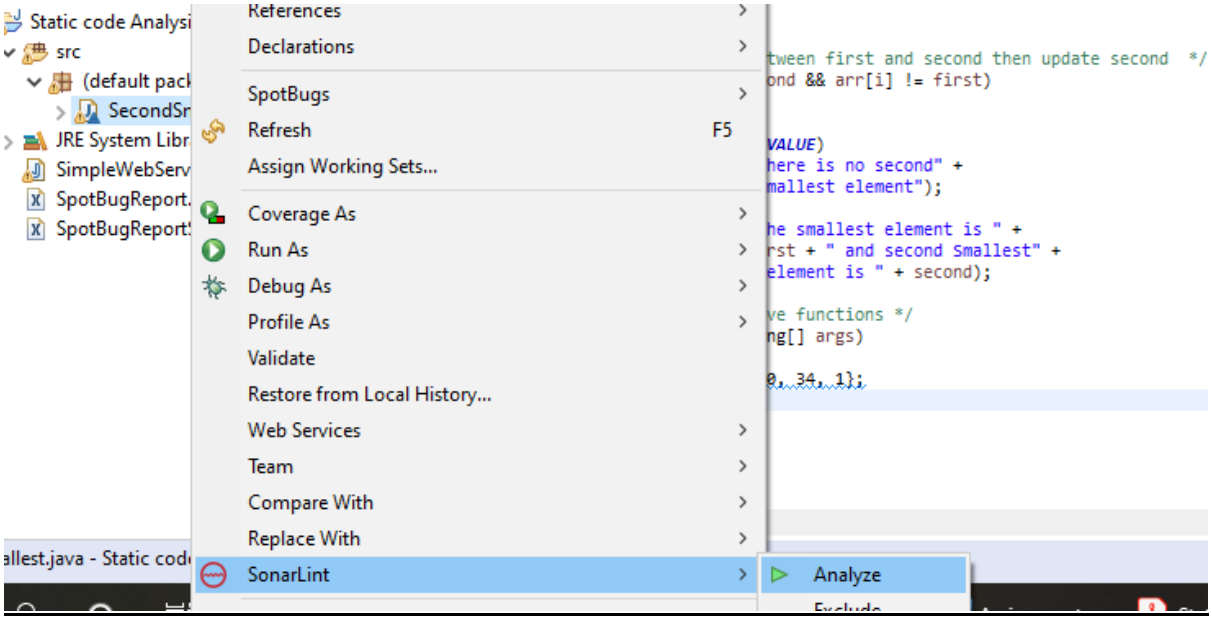
- Now, we run the SpotBugs plugin on this updated code and observe the results.



- As seen above, after the fix, SpotBugs didn't find any issues with the code, meaning our code is Bug free with respect to this plugin.

<u>Analysis with SonarLint:</u>

- We analyze the code using Sonarlint now to look for possible errors.



SecondSmallest.java

```java
import java.io.*;
 class SecondSmallest
{
    /* Function to print first smallest and second smallest elements */
    static void Print2Smallest(int arr[])
    {
        int first, second, arr_size = arr.length;

        /* There should be atleast two elements */
        if (arr_size < 2)
        {
            System.out.println(" Invalid Input ");
            return;
        }
        first = second = Integer.MAX_VALUE;
        for (int i = 0; i < arr_size ; i ++)
        {
            /* If current element is smaller than first then update both first and second */
            if (arr[i] < first)
            {
                second = first;
                first = arr[i];
            }
            /* If arr[i] is in between first and second then update second  */
            else if (arr[i] < second && arr[i] != first)
                second = arr[i];
        }
        if(second == Integer.MAX_VALUE)
            System.out.println("There is no second" +
```

SonarLint Report

38 items

Resource	Date	Description
SecondSmallest.java		"if ... else if" constructs should end with "else" clauses.
SecondSmallest.java		Explicitly import the specific classes needed.
SecondSmallest.java	18 mi...	Missing curly brace.
SecondSmallest.java	18 mi...	Missing curly brace.
SecondSmallest.java		Missing curly brace.
SecondSmallest.java		Add or update the header of this file.

- SonarLint returned some errors corresponding to the code. Some of the errors and their possible fixes.

Missing curly brace:

This corresponds to the error we found in Manual analysis where the if….else loop does not have braces enclosed. Adding them will fix this.

Explicitly import specific classes / Add or update header of the file:

Adding specific header corresponding to each class rather than 1 common header file will solve this issue.

- We make these fixes to the code to try and run the test again. The edited code image is below.

SecondSmallest.java

```java
import java.io.*;
 class SecondSmallest
{
    /* Function to print first smallest and second smallest elements */
    static void print2Smallest(int arr[])
    {
        int first, second, arr_size = arr.length;

        /* There should be atleast two elements */
        if (arr_size < 2)
        {
            System.out.println(" Invalid Input ");
            return;
        }
        first = second = Integer.MAX_VALUE;
        for (int i = 0; i < arr_size ; i ++)
        {
            /* If current element is smaller than first then update both first and second */
            if (arr[i] < first)
            {
                second = first;
                first = arr[i];
            }
            /* If arr[i] is in between first and second then update second  */
            else if (arr[i] < second && arr[i] != first)
                second = arr[i];
        }
        if(second == Integer.MAX_VALUE) {
            System.out.println("There is no second" +
                               "smallest element");}
        else {
            System.out.println("The smallest element is " +
                               first + " and second Smallest" +
                               " element is " + second);}
    }
    /* Driver program to test above functions */
    public static void main (String[] args)
    {
        int arr[] = {12, 13, 1, 10, 34, 1};
        print2Smallest(arr);
    }
}
```

- Now, we proceed to run the SonarLint test again and we see that the code has successfully passed all the criteria set by the Plugin.

SecondSmallest.java

```java
import java.io.*;

class SecondSmallest
{
    /* Function to print first smallest and second smallest
      elements */
    static void print2Smallest(int arr[])
    {
        int first, second, arr_size = arr.length;

        /* There should be atleast two elements */
        if (arr_size < 2)
        {
            System.out.println(" Invalid Input ");
            return;
        }

        first = second = Integer.MAX_VALUE;
        for (int i = 0; i < arr_size ; i ++)
        {
            /* If current element is smaller than first
              then update both first and second */
```

SonarLint Report | Console | Bug Info | Bug Explorer

41 items

Resource	Date	Description
SecondSmallest.java	few seconds ago	Make this line start after 6 spaces to indent the code consistently.
SecondSmallest.java	few seconds ago	Make this line start after 6 spaces to indent the code consistently.
SecondSmallest.java	few seconds ago	Move this "else" on the same line that the previous closing curly brace.
SecondSmallest.java	few seconds ago	Move this closing curly brace to the next line.
SecondSmallest.java	few seconds ago	Move this closing curly brace to the next line.
SecondSmallest.java	few seconds ago	Move this left curly brace to the beginning of next line of code.
SecondSmallest.java	few seconds ago	Move this left curly brace to the beginning of next line of code.
SecondSmallest.java	22 minutes ago	Declare "arr_size" on a separate line.
SecondSmallest.java	22 minutes ago	Declare "second" on a separate line.
SecondSmallest.java	22 minutes ago	Make this line start after 2 spaces to indent the code consistently.
SecondSmallest.java	22 minutes ago	Make this line start after 4 spaces to indent the code consistently.