

User Story for Phase 3:

=====

The documentation of the user story is also posted in: https://github.com/ShresthaSamita/CS5103_qkp232_Samita/issues/3

Title

Allow replacement of all occurrences of a given word to a given replacement word.

Description:

I want to extend the previous program and would like to add a third feature that replaces all occurrences of word. I would like the user to have a chance to choose one of the two options. Option 1 should take the user to have Word Stats- Phase 1 and Phase 2 while the second option should allow user to use the Phase 3 extension.

Acceptance Criteria:

- Program is the extension of the Phase I requirement.
- The replacement happens only when the given pattern word matches with a whole word.
- Program is Case- Sensitive.
- If the pattern does not match, no changes is made to the file and will notify user that the expected word is not found in the input file.

For Example: Suppose, Original content in input file is:

Harry ate an apple.

Enter the word you want to replace: a

Enter the word you want to replace it with: b

The word "a" not found in file.

Enter the word you want to replace: an

Enter the word you want to replace it with: na

Word replaced in file.

Modified content in file will now be:

Harry ate na apple.

Test Case based on User Story:

Functional Requirement

From Phase 1 and Phase 2:

- The program shall allow the user to pass the input from the command line for which they want to count the words and its frequency.
- This program is case sensitive; For example, "Test" is not same as "test".
- The program supports separators like space, tab, newline, and the combination of them.
- The input given should be in textual format.

Additional Requirements: Phase 3:

- The replacement happens only when the given pattern word matches with a whole word.
- If the pattern does not match, no changes is made to the file.
- Users should be prompted to two choices; Option 1 should take user to Phase 1 and Phase 2 whereas Option 2 should take user to Phase3.

Non-Functional requirement

- Error during usage, such as file not found, etc. are handled correctly.
- If a user enters other option than 1 or 2, they are notified.
- If the whole pattern which the user wants to replace is not found in the input file then user is notified.
- The program is developed using Maven development environment, so, the machine used to run this program must have the latest version of MAVEN and JDK installed.
- Works on both MACOS and Windows.

Explanation of Testcase from Test report table above:

1. For Single occurrence of the token

I created a file called "SingleOccurence.txt" containing only one instance of the word "world". This word was successfully identified by the matching pattern and subsequently replaced with the word "universe" after passing the arguments to the "replaceWord()" function within the "WordReplace" class. As a result, the test was successfully passed.

```

@Test
public void testSingleOccurence() throws IOException {
    String filename = "SingleOccurence.txt";
    String original = "And She said she is going to rule the world and make herself proud";
    Path testFile = Paths.get(filename);
    Files.writeString(testFile, original);

    WordReplace.replaceWord(filename, "world", "universe");
    String actualContent = Files.readString(testFile);
    String expectedContent = "And She said she is going to rule the universe and make herself proud";
    assertEquals(expectedContent, actualContent);
}

```

2. For Multiple occurrences of the pattern of token

I have created a text file named "MultipleOccurence.txt", which contains multiple occurrences of the word "strings" that match the token pattern. In this case, there are three instances of the word "strings" in the file. If the user wishes to replace these instances with different words, all three will be replaced. However, it is important to note that a similar token, "string", will not be replaced because the entire pattern must be matched, as per the user's requirements.

```

@Test
public void testMultipleOccurence() throws IOException {
    String filename = "MultipleOccurence.txt";
    String original = "This file has multiple strings in it. All the strings has to be "
        + "replaced by the new string to create a file with new strings .";
    Path testFile = Paths.get(filename);
    Files.writeString(testFile, original);

    WordReplace.replaceWord(filename, "strings", "words");
    String actualContent = Files.readString(testFile);
    String expectedContent = "This file has multiple words in it. All the words has to be "
        + "replaced by the new string to create a file with new words .";
    assertEquals(expectedContent, actualContent);
}

```

3. For Numbers

In this section, we will test how the program handles numbers. The specifications for numbers are the same as for words - all patterns must be fully matched. For instance, if we want to replace the number 20 with 21, we must ensure that the entire pattern is matched. In the example below, we can see that there are two instances of the number 20, as well as one instance of the number 2029 where "20" is included. However, the program will only replace the two instances of "20", and will not change "2029" because only part of the token is matched.

```
@Test
public void testForNumbers() throws IOException {
    String filename = "WithNumber.txt";

    String original = "I was 20 years old when I first moved 20 miles away from home 2029";
    Path testFile = Paths.get(filename);
    Files.writeString(testFile, original);

    WordReplace.replaceWord(filename, "20", "21");
    String actualContent = Files.readString(testFile);
    String expectedContent = "I was 21 years old when I first moved 21 miles away from home 2029";
    assertEquals(expectedContent, actualContent);
}
```

4. Input with no Match

The following test case is for when there is no match for the token. In such cases, the file remains unchanged. To make the program more user-friendly, we have added a notification for the user stating that no matches were found in the file.

```
@Test
public void testNoMatch() throws IOException {
    String filename = "NoMatch.txt";

    String original = "Hello, this is just a random file.";
    Path testFile = Paths.get(filename);
    Files.writeString(testFile, original);

    WordReplace.replaceWord(filename, "strings", "words");
    String actualContent = Files.readString(testFile);
    String expectedContent = "Hello, this is just a random file.";
    assertEquals(expectedContent, actualContent);
}
```

5. For the input with Uppercase and Lowercase

This test case aims to check the program's case sensitivity. For instance, in the given input, the user wants to replace "ham" with "chicken". We can observe that there are two instances of "ham" and one instance of "HaM". However, since the program is case sensitive, only the two instances of "ham" will be replaced, and the instance of "HaM" will remain unchanged.

```
@Test
public void testForUpperLower() throws IOException {
    String filename = "UpperLower.txt";

    String original = "I like ham burger and ham salad but he doesnot like HaM ";
    Path testFile = Paths.get(filename);
    Files.writeString(testFile, original);

    WordReplace.replaceWord(filename, "ham", "chicken");
    String actualContent = Files.readString(testFile);
    String expectedContent = "I like chicken burger and chicken salad but he doesnot like HaM ";
    assertEquals(expectedContent, actualContent);
}
```

Hence, all the testcases above are tested and are **PASSED**

Testing on Eclipse IDE:

The screenshot shows the Eclipse IDE's JUnit console. At the top, it says 'Console JUnit x' and 'Finished after 0.056 seconds'. Below this, a summary bar indicates 'Runs: 5/5', 'Errors: 0', and 'Failures: 0', with a green progress bar. The test results are listed under 'phase3.DriverTest [Runner: JUnit 4] (0.001 s)'. The tests and their durations are: 'testForNumbers (0.000 s)', 'testMultipleOccurence (0.000 s)', 'testSingleOccurence (0.000 s)', 'testForUpperLower (0.000 s)', and 'testNoMatch (0.000 s)'. A 'Failure Trace' button is visible on the right.

Testing on Command line:

```

downloaded From Central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-common-junit5/3.0.0/surefire-common-junit5-3.0.0.jar (12 KB at 240 KB/s)
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running phase1.WordCounterTest
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.089 s - in phase1.WordCounterTest
[INFO] Running phase2.DriverTest
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.019 s - in phase2.DriverTest
[INFO] Running phase3.DriverTest
Word replaced in file.
Word replaced in file.
Word replaced in file.
Word replaced in file.
The word "strings" not found in file.
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.049 s - in phase3.DriverTest
[INFO] Results:
[INFO] Tests run: 18, Failures: 0, Errors: 0, Skipped: 0
[INFO] BUILD SUCCESS
[INFO] Total time: 7.705 s
[INFO] Finished at: 2023-04-25T00:15:08-05:00
[INFO] -----
[INFO] --- jar:3.3.0:jar (default-jar) @ WordCounter ---
[INFO] Building jar: C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main\target\WordCounter-0.0.1-SNAPSHOT.jar
[INFO] BUILD SUCCESS
[INFO] Total time: 7.705 s
[INFO] Finished at: 2023-04-25T00:15:08-05:00
[INFO] -----

```

Sample Run:

1. SingleOccurence.txt

```

C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main>cat SingleOccurence.txt
And She said she is going to rule the world and make herself proud
C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main>java -cp target\WordCounter-0.0.1-SNAPSHOT.jar phase3.Driver
SingleOccurence.txt
=====
Welcome to CS5103
=====
Please select one:
1. Word Statistics
2. Word Replacement
-----
Enter your choice: 2
Enter the word you want to replace: world
Enter the word you want to replace it with: universe
Word replaced in file.

C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main>cat SingleOccurence.txt
And She said she is going to rule the universe and make herself proud
C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main>

```

2. MultipleOccurence.txt

```

C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main>cat MultipleOccurence.txt
This file has multiple strings in it. All the strings has to be replaced by the new string to create a file with new strings .
C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main>java -cp target\WordCounter-0.0.1-SNAPSHOT.jar phase3.Driver
MultipleOccurence.txt
=====
Welcome to CS5103
=====
Please select one:
1. Word Statistics
2. Word Replacement
-----

Enter your choice: 2
Enter the word you want to replace: strings
Enter the word you want to replace it with: words
Word replaced in file.

C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main>cat MultipleOccurence.txt
This file has multiple words in it. All the words has to be replaced by the new string to create a file with new words .
C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main>

```

3. NoMatch.txt

```

C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main>cat NoMatch.txt
ello, this is just a random file.
C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main>java -cp target\WordCounter-0.0.1-SNAPSHOT.jar phase3.Driver
NoMatch.txt
=====
elcome to CS5103
=====
lease select one:
. Word Statistics
. Word Replacement
-----

nter your choice: 2
nter the word you want to replace: y
nter the word you want to replace it with: xd
he word "y" not found in file.

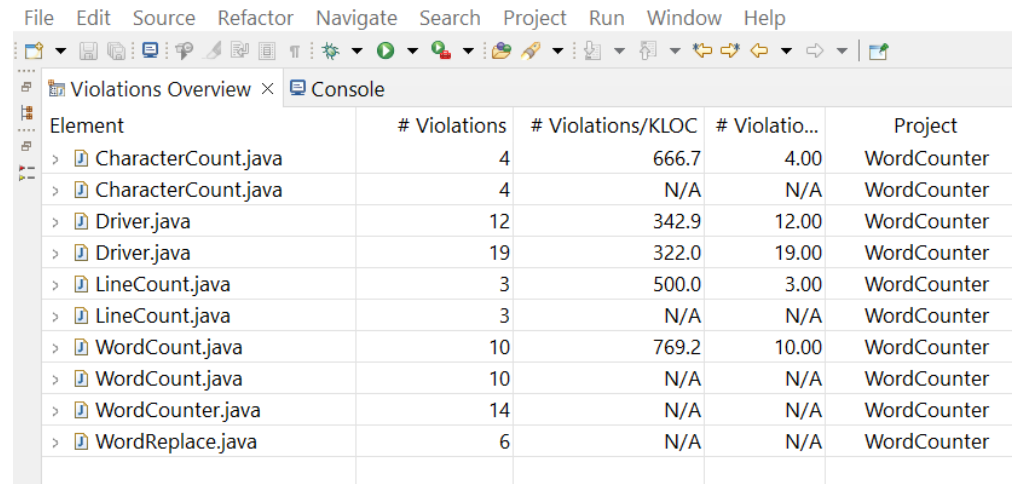
C:\Users\samit\OneDrive\Desktop\projectSE\CS5103_qkp232_Samita-main>

```

Bug Detection:

=====

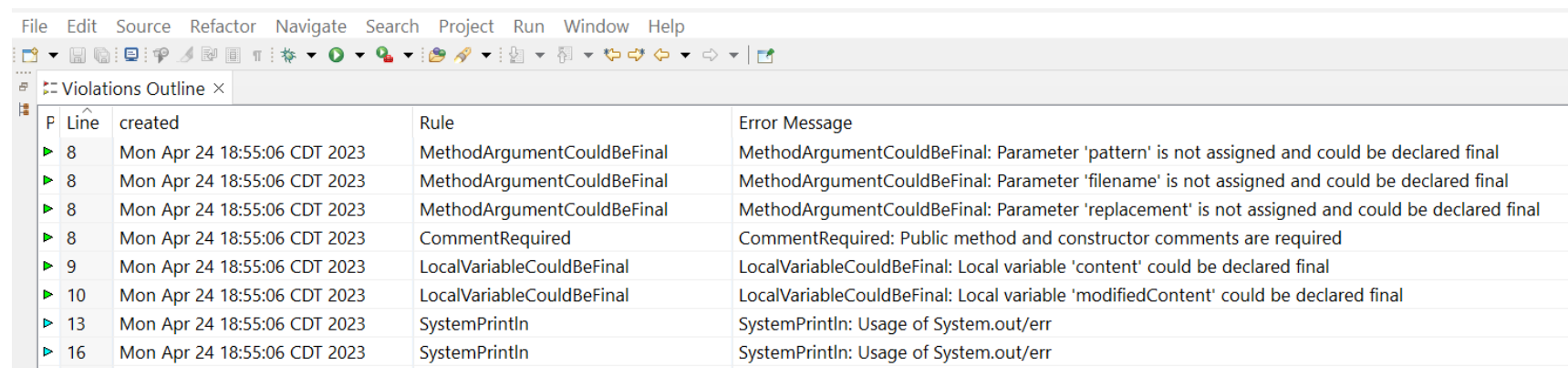
For Bug detection, I have used PMD plugin available from Eclipse Marketplace. The errors found are shown below.



The screenshot shows the Eclipse IDE interface with the 'Violations Overview' tab selected. The table lists violations for various Java files in the 'WordCounter' project.

Element	# Violations	# Violations/KLOC	# Violatio...	Project
> CharacterCount.java	4	666.7	4.00	WordCounter
> CharacterCount.java	4	N/A	N/A	WordCounter
> Driver.java	12	342.9	12.00	WordCounter
> Driver.java	19	322.0	19.00	WordCounter
> LineCount.java	3	500.0	3.00	WordCounter
> LineCount.java	3	N/A	N/A	WordCounter
> WordCount.java	10	769.2	10.00	WordCounter
> WordCount.java	10	N/A	N/A	WordCounter
> WordCounter.java	14	N/A	N/A	WordCounter
> WordReplace.java	6	N/A	N/A	WordCounter

Violation Outline: It provides detailed information of the violation. We can just click on the violation, and it will automatically navigate us to the line of code where violation is detected.



The screenshot shows the Eclipse IDE interface with the 'Violations Outline' tab selected. The table provides detailed information about specific violations, including the line number, creation time, rule name, and error message.

P	Line	created	Rule	Error Message
▶	8	Mon Apr 24 18:55:06 CDT 2023	MethodArgumentCouldBeFinal	MethodArgumentCouldBeFinal: Parameter 'pattern' is not assigned and could be declared final
▶	8	Mon Apr 24 18:55:06 CDT 2023	MethodArgumentCouldBeFinal	MethodArgumentCouldBeFinal: Parameter 'filename' is not assigned and could be declared final
▶	8	Mon Apr 24 18:55:06 CDT 2023	MethodArgumentCouldBeFinal	MethodArgumentCouldBeFinal: Parameter 'replacement' is not assigned and could be declared final
▶	8	Mon Apr 24 18:55:06 CDT 2023	CommentRequired	CommentRequired: Public method and constructor comments are required
▶	9	Mon Apr 24 18:55:06 CDT 2023	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'content' could be declared final
▶	10	Mon Apr 24 18:55:06 CDT 2023	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'modifiedContent' could be declared final
▶	13	Mon Apr 24 18:55:06 CDT 2023	SystemPrintln	SystemPrintln: Usage of System.out/err
▶	16	Mon Apr 24 18:55:06 CDT 2023	SystemPrintln	SystemPrintln: Usage of System.out/err

After removing some Bugs:

=====

I removed some of the violations while developing the project, but some violations were unavoidable, and they were not a critical violations. So, I left them as they are.

Violations Outline ×					Violations Overview ×				
P	Line	created	Rule	Error Message	Element	# Violations	# Violatio...	# Violatio...	Project
▶	44	Sun Apr 30 0...	SwitchDensity	SwitchDensity: A	> Driver.java	3	50.8	3.00	WordCou...
▶	76	Sun Apr 30 0...	ShortVariable	ShortVariable: Av	> WordCount.java	1	76.9	1.00	WordCou...
▶	17	Sun Apr 30 0...	OnlyOneReturn	OnlyOneReturn: ,	> WordReplace.java	1	125.0	1.00	WordCou...
					> LineCount.java	1	166.7	1.00	WordCou...

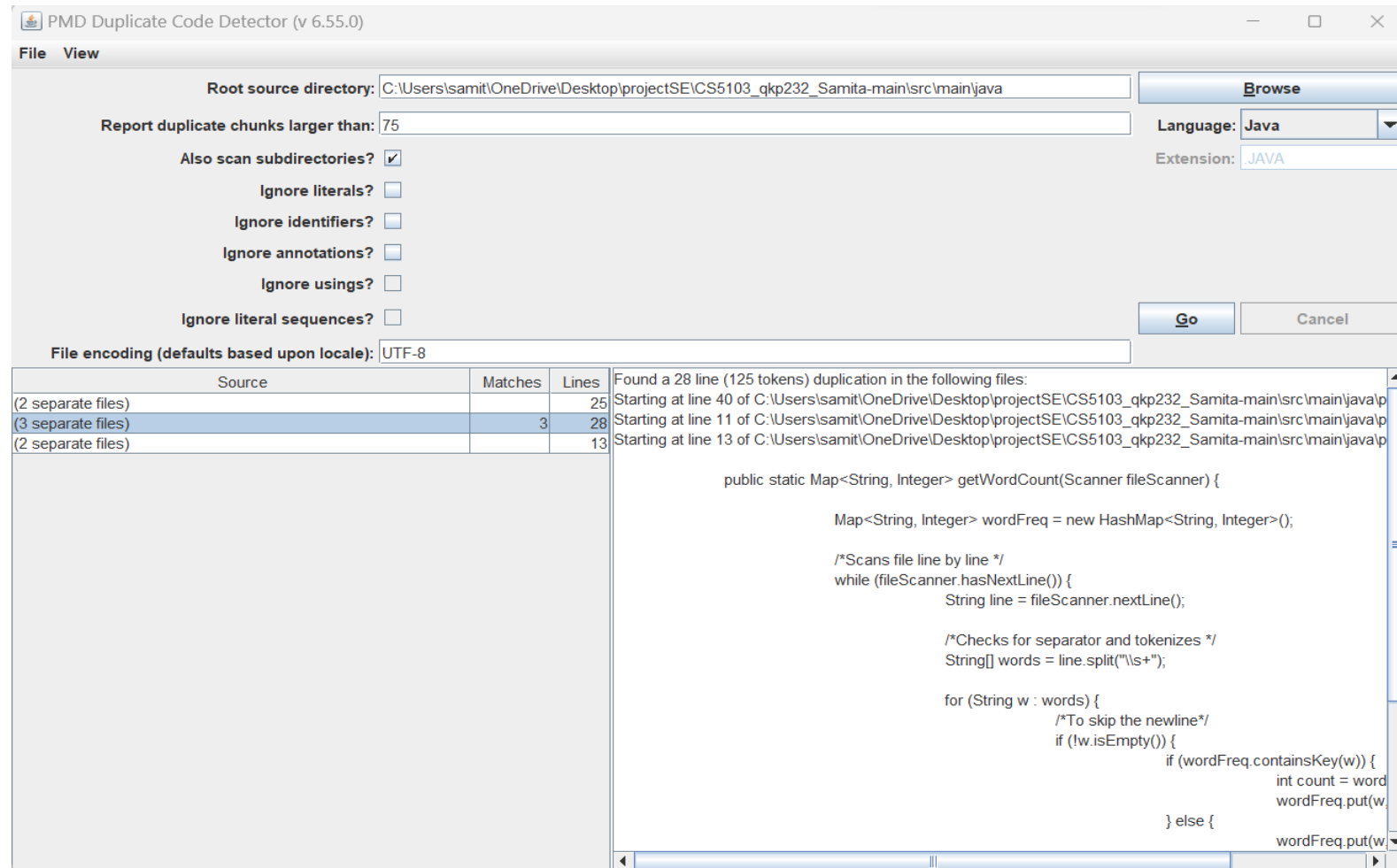
Code Clone Detection:

=====

For Code Clone Detection, PMD's Copy/Paste Detector (CPD) was used. For ease I used the GUI version of it. CPD tools typically analyze the source code and identify blocks of code that have similar syntax or functionality. These blocks are referred to as code clones or duplicate code fragments.

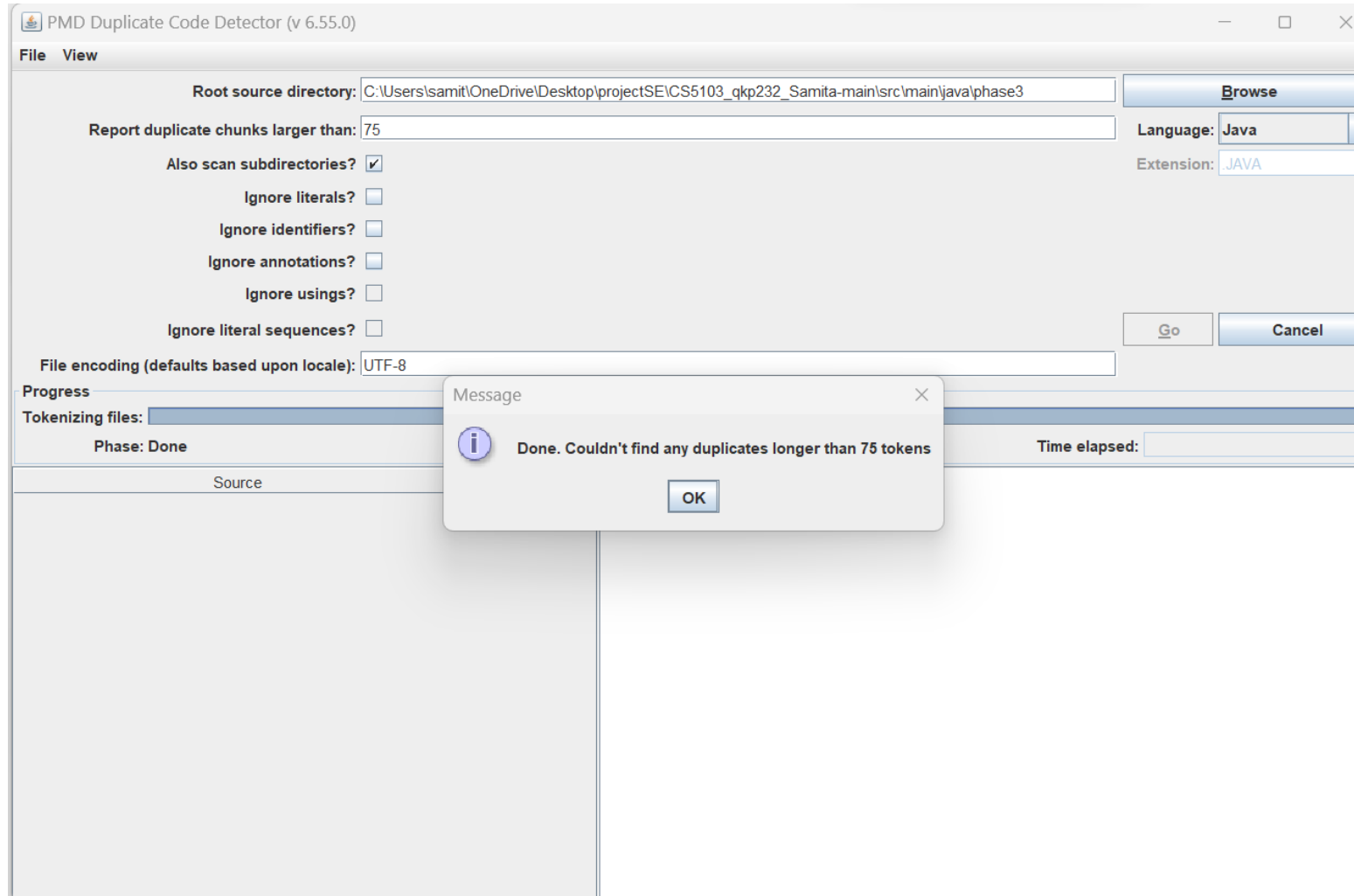
Code Clone Detection Usage Screenshots :

Checking for whole java directory: This will certainly have duplicates. It is because, The ...src/java/ folder contains three packages, Phase1, Phase2 and Phase3. While working on each phase, I have just extended the feature keeping the previous features intact. For example, Phase2 is the subset of Phase3. The WordCount, LineCount etc are still there in Phase3 package which might seem duplicate.



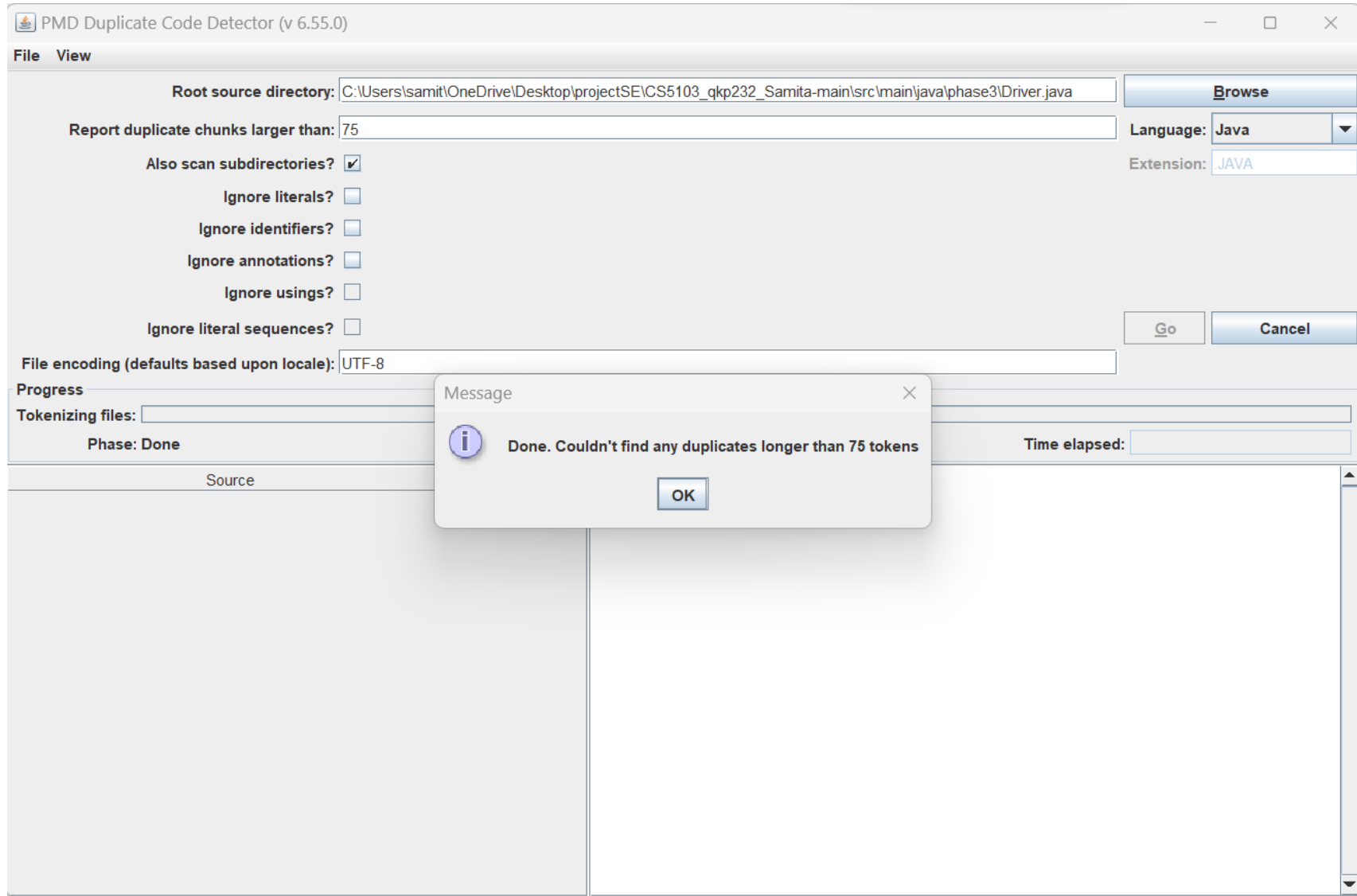
Checking for Phase3 package

This package contains Driver.java, WordCount.java, LineCount.java and WordReplace.java and no code clones were found.



Checking for Driver.java

Similarly, I checked for some files in Phase 3 package for detecting the code clone.



Checking for WordReplace.java

