Name: Alireza Rajouldezfooly

LNU email: ar223gf@student.lnu.se

GIT HUB LINK: https://github.com/ar223gf/ar223gf_1dv600

## Task 1 - Test Plan

### Objectives

The objectives of testing this iteration is to find bugs, issues with the code that will then be

fixed in the last iteration.

### What to test?

I intend to test the UC1 ("Play Game") because it is the main part of the game. And we want to make sure that the game is doing what it is supposed to do. We are going to specifically test the methods theLine,Winnerchecker and InputChecker. These three methods are important methods that make the application works and therefore the methods have been selected to be tested.

### How to test?

We are going to create manual testing and automated testing (using JUnit) to be able to determine if the application is working according to the requirement.

| TASK | ESTIMATED | ACTUAL |
|---|---|---|
| **Manual Test case** | 30 mins | 45 mins |
| **Junit test** | 1 hour | 2 hour 30 mins |
| **Running test** | 20 mins | 20 mins |
| **Checking the code** | 1 hour | 40 mins |
| **Report of testing** | 1 hour | 1 hour |

## Task 2 – Manual Test Cases using the eclipse consul

### TC1.1 To win the game

Use case tested: UC1(“Play Game”)

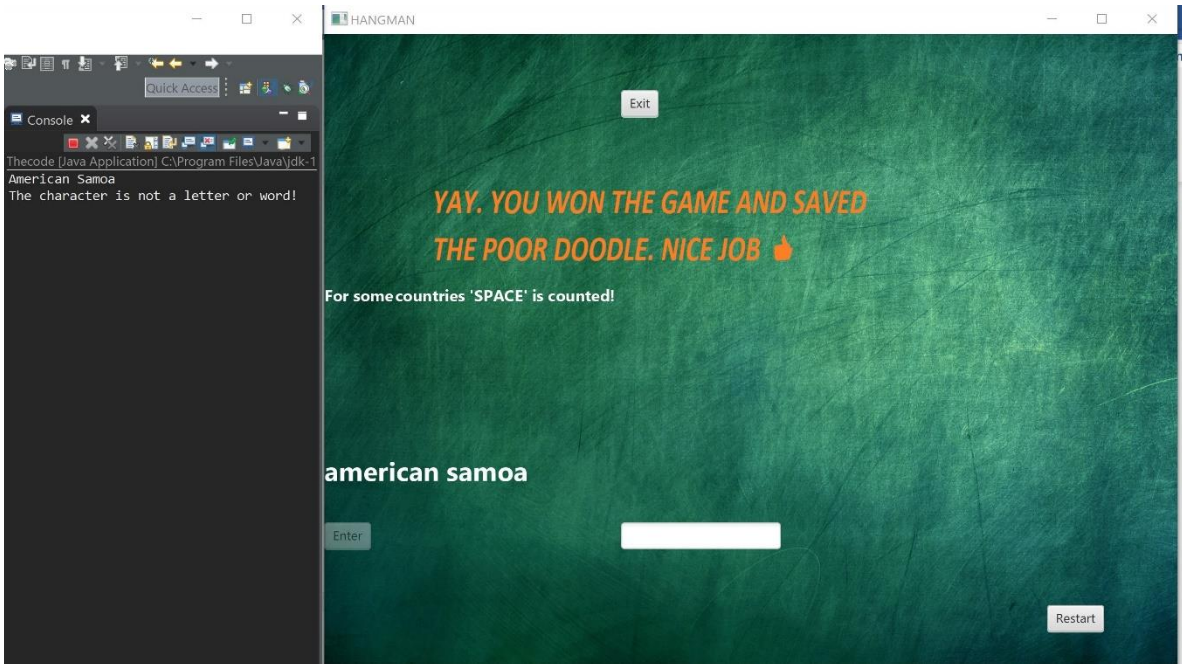
**Short Description:** This test case is testing if the system shows the correct text when the player wins the game for UC1.

### Test steps

- Run Hangman game click on run the code from your IDE.
- Click on start from the Graphical user interface (GUI).
- On text field type “a” and Press the Button “Enter” from your GUI.
- On text field type “m” and Press the Button “Enter” from your GUI.
- On text field type “m” and Press the Button “Enter” from your GUI.
- On text field type “r” and Press the Button “Enter” from your GUI.
- On text field type “c” and Press the Button “Enter” from your GUI.
- On text field type “e” and Press the Button “Enter” from your GUI.
- On text field type “n” and Press the Button “Enter” from your GUI.
- On text field type “i” and Press the Button “Enter” from your GUI.
- On text field type “s” and Press the Button “Enter” from your GUI.
- On text field type “o” and Press the Button “Enter” from your GUI.
- On text field type “ ” (space) and Press the Button “Enter” from your GUI.

### Expected

The system should show the text "YAY. YOU WON THE GAME AND SAVED THE POOR DOODLE. NICE JOB!" on the GUI. THE Restart button and the country name must be visible.



```
Thecode [Java Application] C:\Program Files\Java\jdk-1
American Samoa
The character is not a letter or word!
```

YAY. YOU WON THE GAME AND SAVED
THE POOR DOODLE. NICE JOB

For some countries 'SPACE' is counted!

american samoa

## Results

Did the Test succeed: YES!

## TC1.2 To Lose the game

Use case tested: UC1("Play Game")

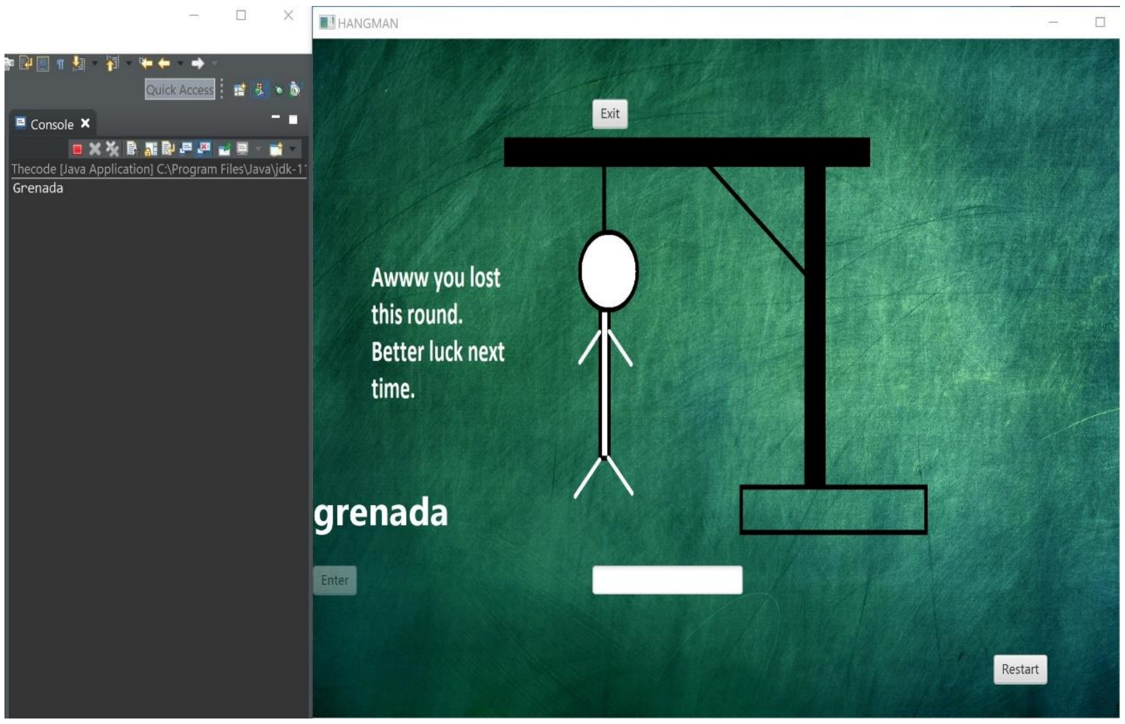
**Short Description:** This test case is testing if the system shows the correct text when the player loses the game for UC1.

## Test steps

- Run Hangman game click on run the code from your IDE.
- Click on start from the Graphical user interface (GUI).
- On text field type "p" and Press the Button "Enter" from your GUI.
- On text field type "m" and Press the Button "Enter" from your GUI.
- On text field type "k" and Press the Button "Enter" from your GUI.
- On text field type "j" and Press the Button "Enter" from your GUI.
- On text field type "u" and Press the Button "Enter" from your GUI.
- On text field type "y" and Press the Button "Enter" from your GUI.

### Expected

The system should show the text ("Awww, you lost, Better luck next time.") THE Restart button, Exit button and the country name must be visible.



```
Thecode [Java Application] C:\Program Files\Java\jdk-1
Grenada
```

Awww you lost this round. Better luck next time.

grenada

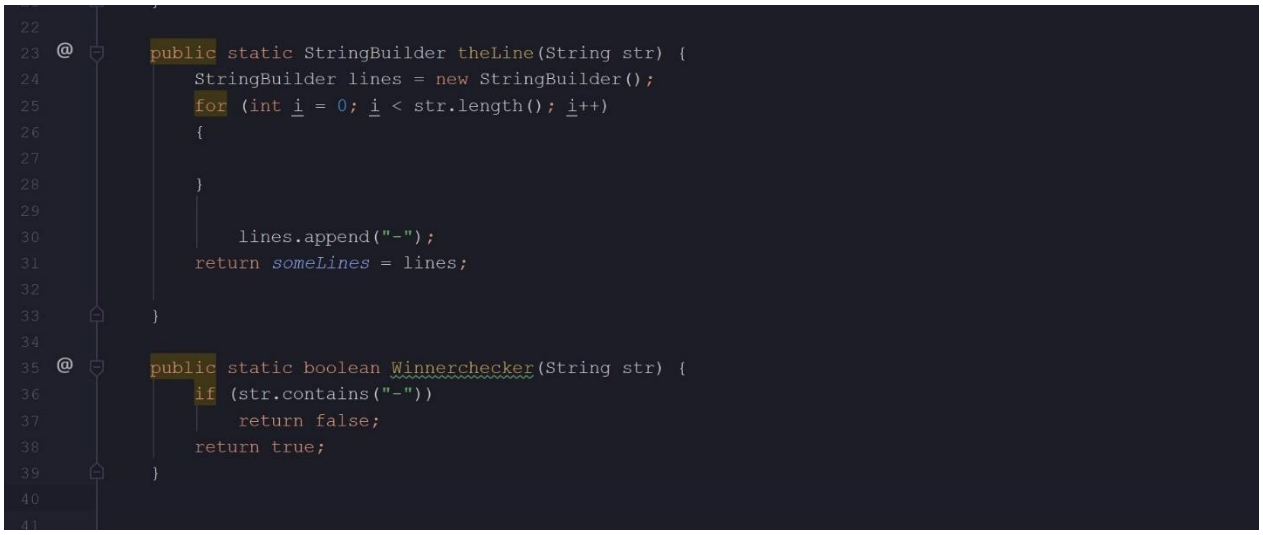
**Results**

Did the Test succeed: YES!

## Task 3, Unit Test

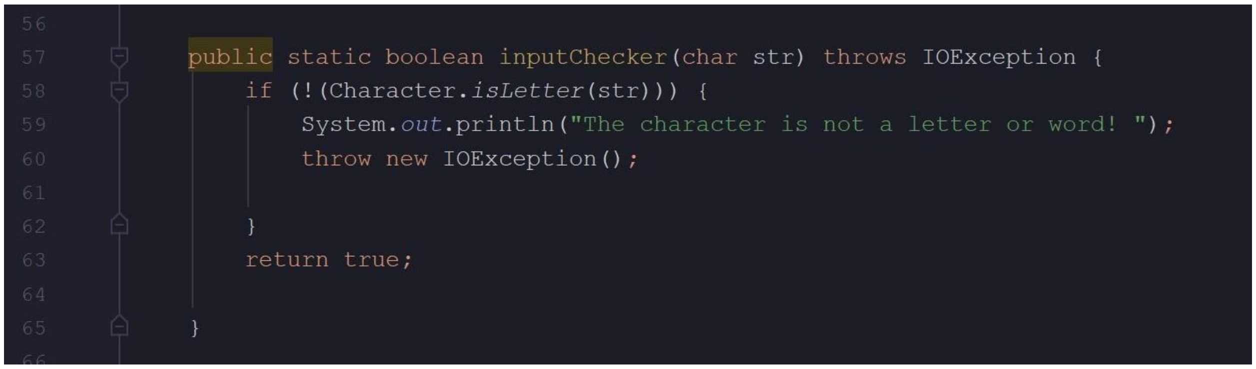
**Methods from the source code that need to be tested:**

**Method theLine and Method Winnerchecker**



```java
    public static StringBuilder theLine(String str) {
        StringBuilder lines = new StringBuilder();
        for (int i = 0; i < str.length(); i++)
        {

        }

            lines.append("-");
        return someLines = lines;

    }

    public static boolean Winnerchecker(String str) {
        if (str.contains("-"))
            return false;
        return true;
    }
```

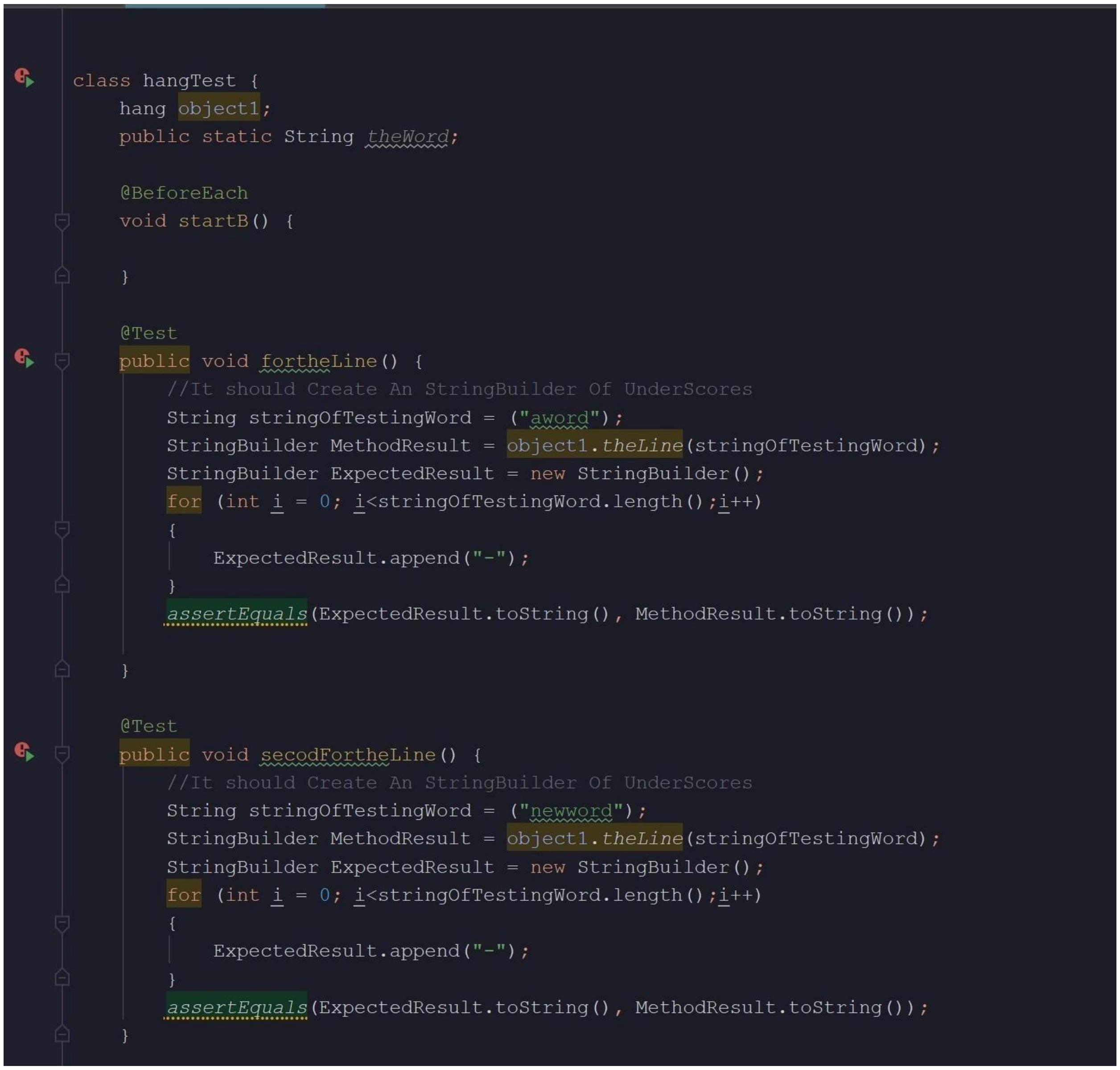
**Method inputChecker**



```java
    public static boolean inputChecker(char str) throws IOException {
        if (!(Character.isLetter(str))) {
            System.out.println("The character is not a letter or word! ");
            throw new IOException();

        }
        return true;

    }
```

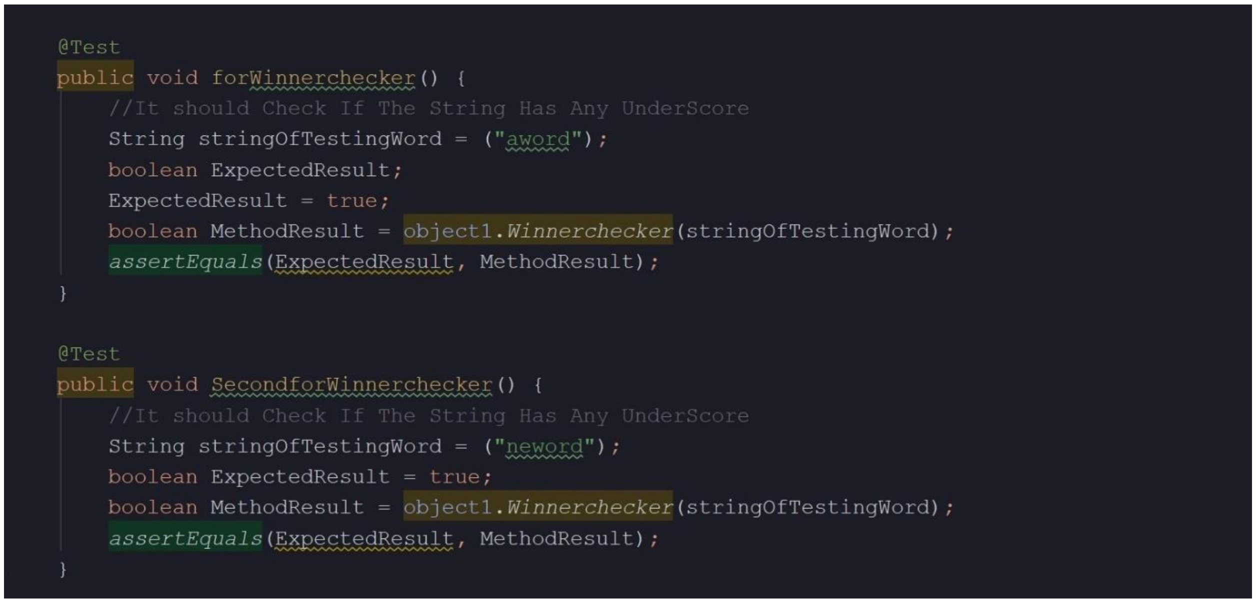
**Methods for the automated tests in Junit 5:**

Junit methods to test the source code Method theLine :¨



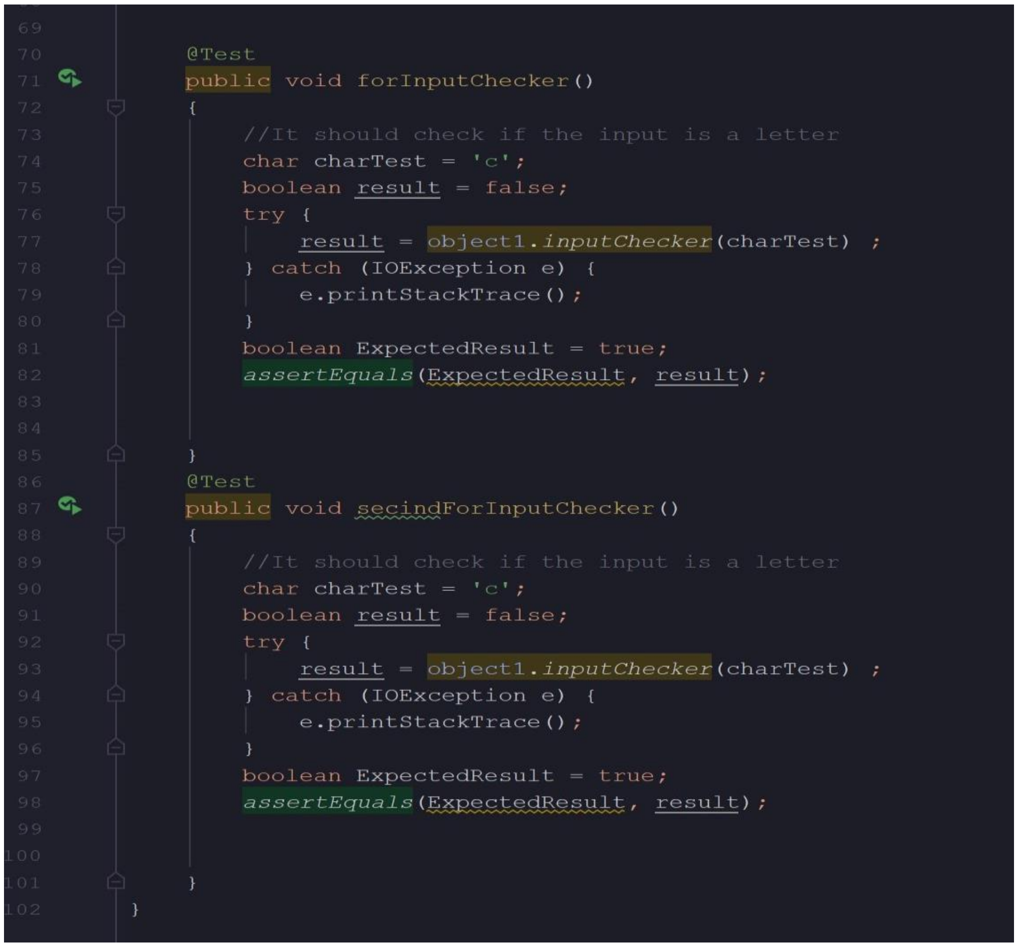
```java
class hangTest {
    hang object1;
    public static String theWord;

    @BeforeEach
    void startB() {

    }

    @Test
    public void fortheLine() {
        //It should Create An StringBuilder Of UnderScores
        String stringOfTestingWord = ("aword");
        StringBuilder MethodResult = object1.theLine(stringOfTestingWord);
        StringBuilder ExpectedResult = new StringBuilder();
        for (int i = 0; i<stringOfTestingWord.length();i++)
        {
            ExpectedResult.append("-");
        }
        assertEquals(ExpectedResult.toString(), MethodResult.toString());

    }

    @Test
    public void secodFortheLine() {
        //It should Create An StringBuilder Of UnderScores
        String stringOfTestingWord = ("newword");
        StringBuilder MethodResult = object1.theLine(stringOfTestingWord);
        StringBuilder ExpectedResult = new StringBuilder();
        for (int i = 0; i<stringOfTestingWord.length();i++)
        {
            ExpectedResult.append("-");
        }
        assertEquals(ExpectedResult.toString(), MethodResult.toString());
    }
```

Junit methods to test the source code Method Winnerchecker :¨



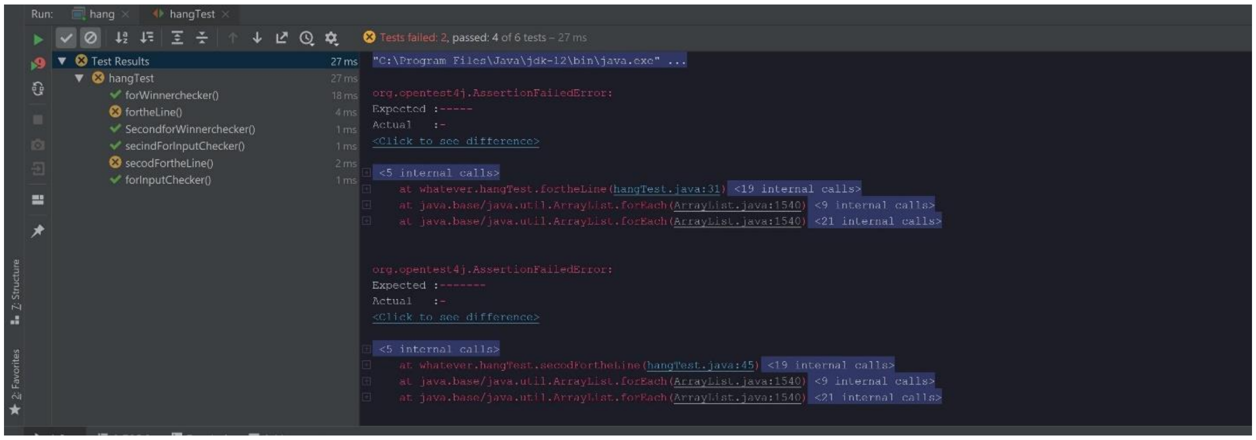
```
@Test
public void forWinnerchecker() {
    //It should Check If The String Has Any UnderScore
    String stringOfTestingWord = ("aword");
    boolean ExpectedResult;
    ExpectedResult = true;
    boolean MethodResult = object1.Winnerchecker(stringOfTestingWord);
    assertEquals(ExpectedResult, MethodResult);
}

@Test
public void SecondforWinnerchecker() {
    //It should Check If The String Has Any UnderScore
    String stringOfTestingWord = ("neword");
    boolean ExpectedResult = true;
    boolean MethodResult = object1.Winnerchecker(stringOfTestingWord);
    assertEquals(ExpectedResult, MethodResult);
}
```

Junit methods to test the source code Method inputChecker :



```
        @Test
        public void forInputChecker()
        {
            //It should check if the input is a letter
            char charTest = 'c';
            boolean result = false;
            try {
                result = object1.inputChecker(charTest)  ;
            } catch (IOException e) {
                e.printStackTrace();
            }
            boolean ExpectedResult = true;
            assertEquals(ExpectedResult, result);


        }
        @Test
        public void secindForInputChecker()
        {
            //It should check if the input is a letter
            char charTest = 'c';
            boolean result = false;
            try {
                result = object1.inputChecker(charTest)  ;
            } catch (IOException e) {
                e.printStackTrace();
            }
            boolean ExpectedResult = true;
            assertEquals(ExpectedResult, result);


        }
    }
```

## Results of the Automated Tests in JUnit:



```
Tests failed: 2, passed: 4 of 6 tests – 27 ms

Test Results                    27 ms
  hangTest                      27 ms
    forWinnerchecker()          18 ms
    fortheLine()                 4 ms
    SecondforWinnerchecker()     1 ms
    secindForInputChecker()      1 ms
    secodFortheLine()            2 ms
    forInputChecker()            1 ms
```

```
"C:\Program Files\Java\jdk-12\bin\java.exe" ...

org.opentest4j.AssertionFailedError:
Expected :-----
Actual   :-
<Click to see difference>

<5 internal calls>
    at whatever.hangTest.fortheLine(hangTest.java:31) <19 internal calls>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1540) <9 internal calls>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1540) <21 internal calls>


org.opentest4j.AssertionFailedError:
Expected :-------
Actual   :-
<Click to see difference>

<5 internal calls>
    at whatever.hangTest.secodFortheLine(hangTest.java:45) <19 internal calls>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1540) <9 internal calls>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1540) <21 internal calls>
```

The last 4 test methods were successful.But the first two “ fortheLine ” and  “ SecondForTheLine “ which were testing the method theLine from the source code did not work properly and as expected.

## Task 4 - Reflection:

This testing helped me to understand how important testing is in the process of coding for a efficient program.

Now I also know how important it is to have proper test cases in order to challenge my code, so I Can improve the code and also, I realised it is such a good practice to do so. I realised that I had some of the code that was not working but it was not visible when running the application.