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”)

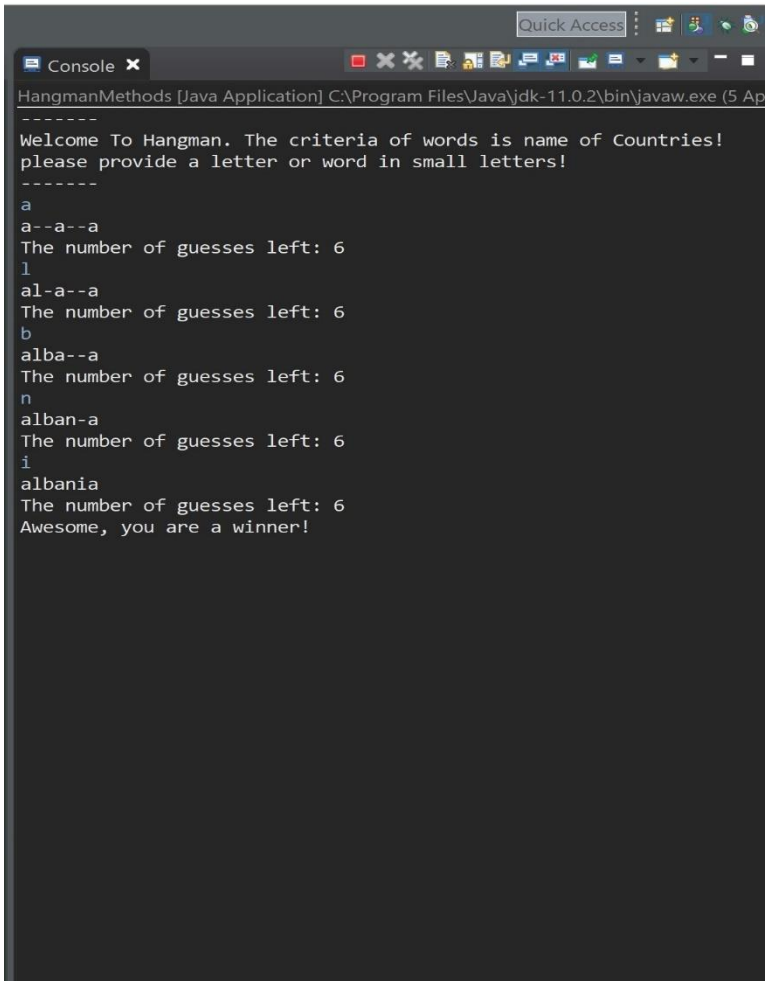
Short Description: This test case is testing if the system shows the correct text when the player wins the game for UC1.

Test steps

- Run HangmanMethods by pressing Ctrl + F11 or F11 from your keyboard
- On Console type “a” and Press the Button “Enter” from your keyboard.
- On Console type “l” and Press the Button “Enter” from your keyboard.
- On Console type “b” and Press the Button “Enter” from your keyboard.
- On Console type “n” and Press the Button “Enter” from your keyboard.
- On Console type “i” and Press the Button “Enter” from your keyboard.

Expected

The system should show the text "Awesome, you are a winner!" at the end of the console.



Results

Did the Test succeed: YES!

TC1.2 To Lose the game

Use case tested : UC1(“Play Game”)

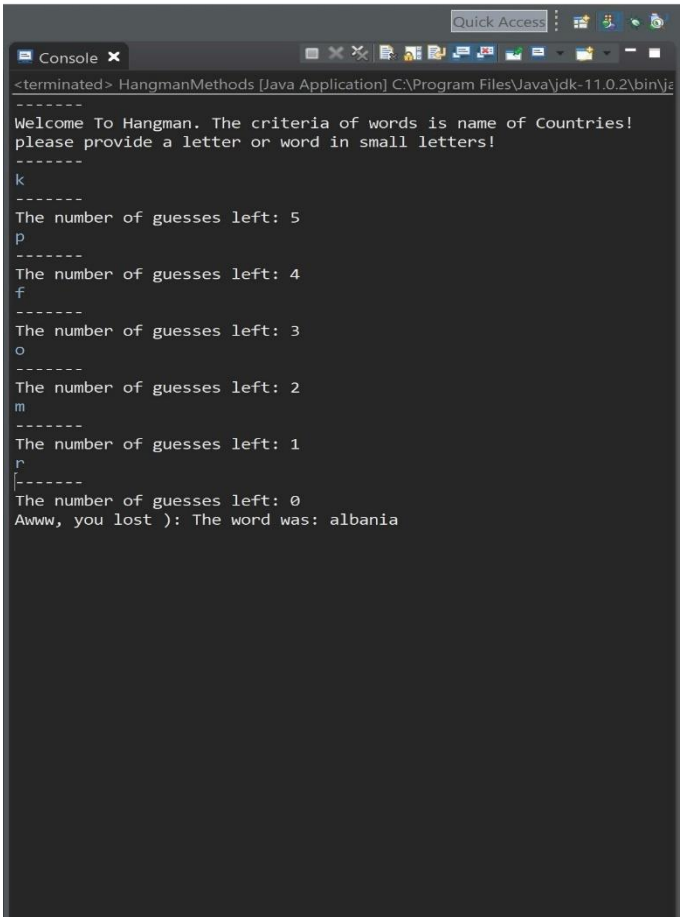
Short Description: This test case is testing if the system shows the correct text when the player loses the game for UC1.

Test steps

- Run HangmanMethods by pressing Ctrl + F11 or just F11 from your keyboard
- On Console type “k” and Press the Button “Enter” from your keyboard.
- On Console type “p” and Press the Button “Enter” from your keyboard.
- On Console type “f” and Press the Button “Enter” from your keyboard.
- On Console type “o” and Press the Button “Enter” from your keyboard.
- On Console type “m” and Press the Button “Enter” from your keyboard.
- On Console type “r” and Press the Button “Enter” from your keyboard.
-

Expected

The system should show the text ("Awww, you lost): The word was: albania "at the end of the console.



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

```
22
23 @    public static StringBuilder theLine(String str) {
24          StringBuilder lines = new StringBuilder();
25          for (int i = 0; i < str.length(); i++)
26          {
27
28          }
29
30              lines.append("-");
31          return someLines = lines;
32
33      }
34
35 @    public static boolean Winnerchecker(String str) {
36          if (str.contains("-"))
37              return false;
38          return true;
39      }
40
41
```

Method inputChecker

```
56
57      public static boolean inputChecker(char str) throws IOException {
58          if (!(Character.isLetter(str))) {
59              System.out.println("The character is not a letter or word! ");
60              throw new IOException();
61
62          }
63          return true;
64
65      }
66
```

Methods for the automated tests in Junit 5:

Junit methods to test the source code Method theLine :¨

```
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 :¨

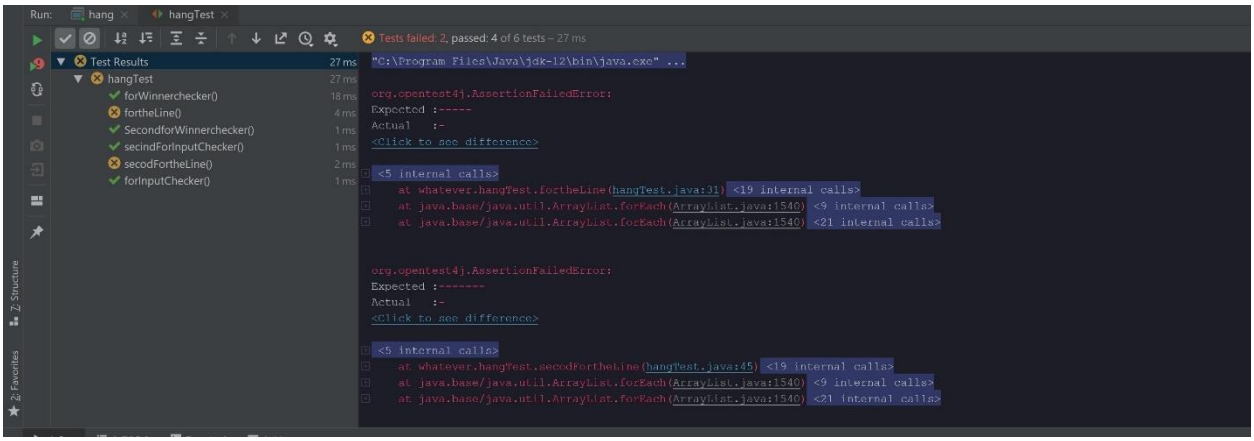
```java
@Test
public void forWinnerchecker() {
    //It should Check If The String Has Any UnderScore
    String stringOfTestingWord = ("aword");
    boolean ExpectedResult;
    ExpectedResult = true;
    boolean MethodResult = object1.Winnerchecker(stringOfTestingWord);
    assertEquals(ExpectedResult,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

```java
69
70          @Test
71          public void forInputChecker()
72          {
73              //It should check if the input is a letter
74              char charTest = 'c';
75              boolean result = false;
76              try {
77                  result = object1.inputChecker(charTest) ;
78              } catch (IOException e) {
79                  e.printStackTrace();
80              }
81              boolean ExpectedResult = true;
82              assertEquals(ExpectedResult, result);
83
84
85          }
86          @Test
87          public void secindForInputChecker()
88          {
89              //It should check if the input is a letter
90              char charTest = 'c';
91              boolean result = false;
92              try {
93                  result = object1.inputChecker(charTest) ;
94              } catch (IOException e) {
95                  e.printStackTrace();
96              }
97              boolean ExpectedResult = true;
98              assertEquals(ExpectedResult, result);
99
100
101         }
102     }
```

Results of the Automated Tests in JUnit:



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.