Name: Loic Galland

LNU email: lg222sv@student.lnu.se

GitHub Release: https://github.com/lg222sv/lg222sv_1DV600/releases/latest

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?

We intend to test the UC2 (“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 isLetterInWord, getStringReprensation and createUnderscoreArray. 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.

Time Plan

Task	Estimated	Actual
Manual Test Case	1h	1h30min
JUnit test	1h30min	2h
Running test	40 min	20min
Checking the code	30 min	25min
Report of Testing	30 min	1h

Task 2 – Manual Test Cases using the client application

TC2.1 Win Game

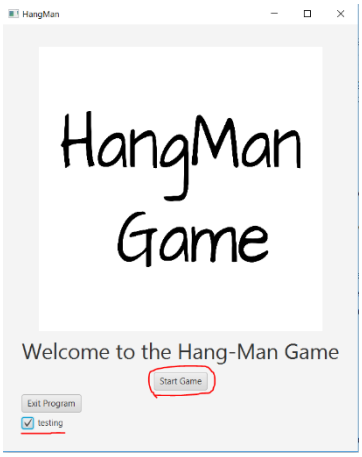
Use Case Tested: UC2 (“Play Game”)

Short Description: This test case is testing if the system shows the correct text when the player wins the game for UC2.

Preconditions: UC1.

Test steps

- Tick the box “testing”.
- Press the “Start Game” Button.



- Write “h” and Press the Button “GUESS”.
- Write “e” and Press the Button “GUESS”.
- Write “l” and Press the Button “GUESS”.
- Write “o” and Press the Button “GUESS”.

Expected

- The system should show the text “h e l l o” under the picture and show the text “Congratulation, You have WON”.

Results

Did the Test succeed: ✓
Comments:……
……
……
……

TC2.2 Lose Game

Use Case Tested: UC2 (“Play Game”)

Short Description: This test case is testing if the system shows the correct text when the player loses the game for UC2.

Preconditions: UC1.

Test steps

- Tick the box “testing”.
- Press the “Start Game” Button.
- Write “q” and Press the Button “GUESS”.
- Write “p” and Press the Button “GUESS”.
- Write “m” and Press the Button “GUESS”.

- Write “w” and Press the Button “GUESS”.
- Write “u” and Press the Button “GUESS”.
- Write “d” and Press the Button “GUESS”.
- Write “a” and Press the Button “GUESS”.

Expected

- The system should show the text “You have Lost” under the picture.
- The system should show the text “Wrong Letters: A D M PQ U W”.
- The system should show a Button “Restart”.
- The image should show the dead hang man.

Results

Did the Test succeed: ✓

Comments:……
……
……
……

Task 3, Unit Test

Method from the source code that need to be tested:

Method isLetterinWord:

```
public boolean isLetterinWord(String c, int pos, String[] word) {
    if(pos>=0 && pos<word.length) {
        if (word[pos].equals(c))
            return true;
        else
            return false;
    }else
        throw new IndexOutOfBoundsException();

}
```

Method getStringRepresentation:

```
public String getStringRepresentation(String[] array){
    StringBuilder sb = new StringBuilder();
    for (int i=0;i<array.length;i++){
        String help = array[i]+" ";
        sb.append(help);
    }
    return sb.toString();
}
```

Method createUnderscoreArray:

```
public String[] createUnderScoreArray(int length){
    String[] help=new String[length];
    for(int i=0;i<help.length;i++) {
       help[i] = " ";
    }
    return help;
}
```

Methods for the automated tests in Junit 5:

JUnit methods to test the source code method isLetterinWord:

```java
package HangManIteration2;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class HangManit2Test {
    HangManit2 sut;
    @BeforeEach
    void startB() {
        sut = new HangManit2();
    }

    @Test
    public void shouldReturnTrueIfTheLetterIsInTheWordAtCertainPosition () {
        //test-input-output definitions
        String[] inputWord = {"h", "e", "l", "l", "o"};
        String inputLetter = "o";
        int inputposition = 4;
        boolean ExpectedResult = true;

        //exercice the sut
        boolean MethodResult = sut.isLetterinWord(inputLetter, inputposition, inputWord);

        //compare actual result with the expected value and report
        assertEquals(ExpectedResult,MethodResult);


    }
```

```java
@Test
public void shouldReturnFalseIfLetterIsNotInTheWordAtCertainPosition(){
    //test-input-output definition
    String[] inputEmptyWord ={"h","e","l","l","o"};
    String inputEmptyLetter ="e";
    int inputEmptyPosition=0;
    boolean ExpectedEmptyResult =false;

    //Exercise the sut
    boolean EmptyMethodResult = sut.isLetterinWord(inputEmptyLetter,inputEmptyPosition,inputEmptyWord);
    //Compare the actual result with the expected value and report it.
    assertEquals(ExpectedEmptyResult,EmptyMethodResult);
}
```

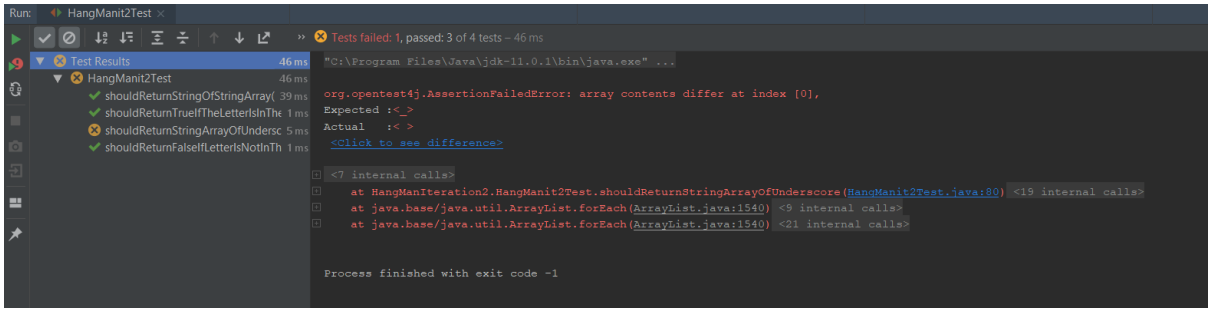
JUnit method to test the source code method getStringRepresentation:

```java
@Test
public void shouldReturnStringOfStringArray(){
    //FIRST TEST
    //Test-input-output definition
    String[] inputStringArray = {"h","e","l","l","o"};
    String ExpectedString = "h e l l o ";

    //Exercise the sut
    String ActualString = sut.getStringRepresentation(inputStringArray);

    //Compare the actual result with the expected value and report
    assertEquals(ExpectedString,ActualString);

    //-------------------------------------------------------------
    //SECOND TEST
    //Test-input-output definition
    String[] inputEmptyArray ={};
    String ExpectedEmptyString="";

    //Exercise the sut
    String ActualEmptyString = sut.getStringRepresentation(inputEmptyArray);

    //Compare the actual result with the expected value and report
    assertEquals(ExpectedEmptyString,ActualEmptyString);
}
```

JUnit method to test the source code method createUnderscoreArray:

```
@Test
public void shouldReturnStringOfStringArray(){
    //FIRST TEST
    //Test-input-output definition
    String[] inputStringArray = {"h","e","l","l","o"};
    String ExpectedString = "h e l l o ";

    //Exercise the sut
    String ActualString = sut.getStringRepresentation(inputStringArray);

    //Compare the actual result with the expected value and report
    assertEquals(ExpectedString,ActualString);

    //-------------------------------------------------------------
    //SECOND TEST
    //Test-input-output definition
    String[] inputEmptyArray ={};
    String ExpectedEmptyString="";

    //Exercise the sut
    String ActualEmptyString = sut.getStringRepresentation(inputEmptyArray);

    //Compare the actual result with the expected value and report
    assertEquals(ExpectedEmptyString,ActualEmptyString);
}
@Test
public void shouldReturnStringArrayOfUnderscore(){
    //Test-input-output definition
    String[] inputArray = {"h","e","l","l","O"};
    String[] Expected = {"_","_","_","_","_"};

    //Exercise the sut
    String[] Actual = sut.createUnderScoreArray(inputArray.length);

    //Compare the actual result with the expected value and report
    assertArrayEquals(Expected,Actual);
}
```

Results of the Automated Tests in JUnit:



The first 3 test methods were successful. However, the test method "shouldReturnStringArrayofUnderscore" did not work as expected.

Task 4 - Reflection:

This assignment made me realise how important testing is in the building of an application. I realised that I had some of the code that was not working but it was not visible when running the application. It also made me realised that was a mess and that it was not implemented in a way where I will able every single part of my code. I will take this into account for the final iteration of this project and for my future projects.