Khalil Mardini

Km222pw@student.lnu.se

GitLab repo: https://gitlab.lnu.se/1dv600/student/km222pw/assignment-3

1.1Test Plan

Objective:

The main purpose of this assignment is to test the provided code from the previous iteration which provided the player to choose random characters from a predefined word list and at the end it will display the result of the player, The test part goal´s to check the functionality fullify the requirements of the game .Moreover, manual tests will be implemented to verify the system´s functionality for the user.

1.2 What to test?

Starting with this iteration, there will be only three use cases that will be tested by implementing and running manual test cases, Start Game, Play Game and Exit Game. Even though the methods from the HangmanManager class such as Sketch(),Win() Lose(), endgame(), GameStatus(), will all be tested using automated unit testing However there will be one method seeScore() will be tested and using automated testing and it will intentionally fail

1.3 Time Log:

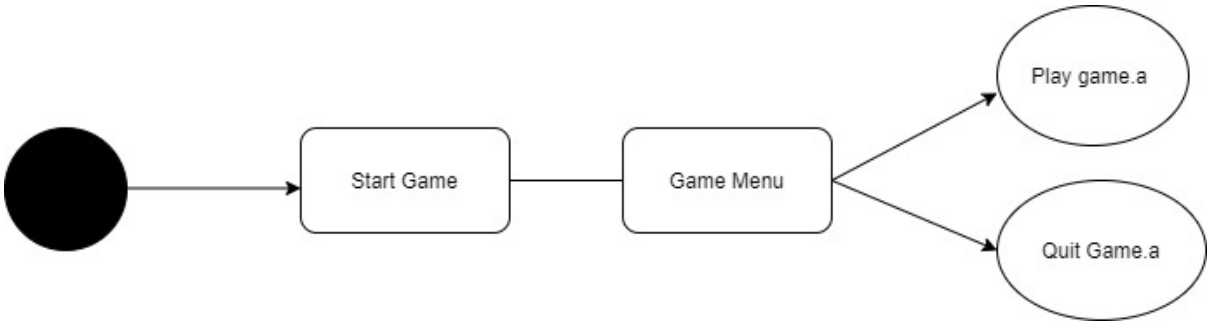
Task	Estimated Time	Actual Time
Automated unit testing	3h	4h
Test plan	2h	6h
Manual test cases	2h	3h
Report and Time log	1h	2h
Upgrade automated unit testing	30m	1h
Upgrade manual test cases	30m	40min
Upgrading Test plan	30m	1h
Code check	1h	30m

2. Manual test cases

Tc.1.1 Starting the game

Use case:UC1

Scenario: The player will Start the game successfully



The main scenario of UC1 is tested where a user starts the game successfully

Test Steps:

1.Starting the game

2 .The game show two options to choose (1) Play game, (2) Quit game

3 . The system will ask the user to user to press 1 or 2 “Press 1 or 2: “

4. Insert 1 and press the enter button

Expected Results: The game will allow the player/tester to see the hangman pole and the empty blanks to type in the letters

```
  |
  |
  |
  |
 _|_
_ _ _ _
```

Incorrect guesses:

type a letter:

Note: more empty blanks might appear depends on the picked word from the library word list

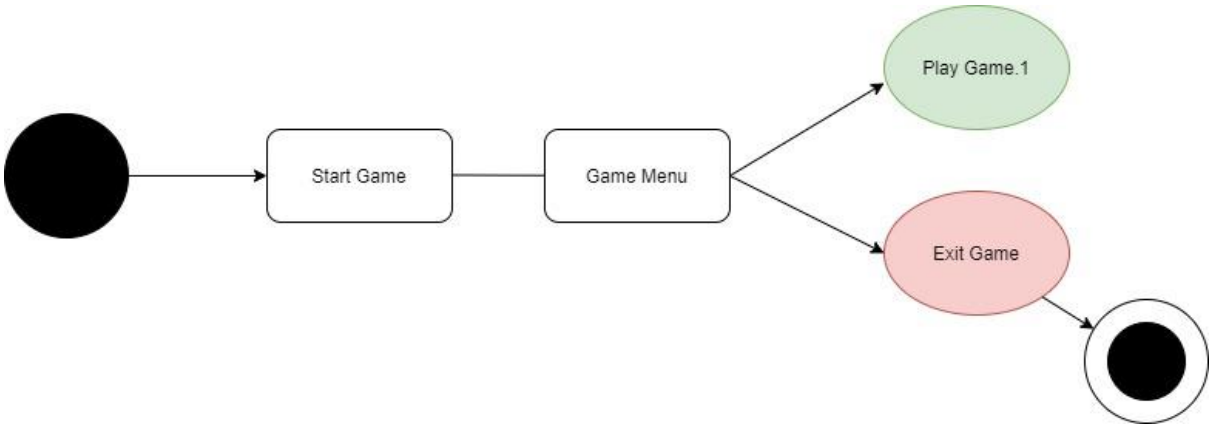
But in this test case I am showing 4 empty blanks

Actual results: The system runs as expected and the game begins working perfectly

Tc.1.2 Leaving the game

Use case: UC2

Scenario: The player will quit the game



The main scenario of UC1 is tested where a user starts the game successfully

Precondition: While the game is already running , the system will

Test Steps:

1.Starting the game

2 .The game menu show two options to choose “1) Play Game”, and “2) Exit Game”

3 . The system will ask the user to user to press 1 or 2 “Press 1 or 2: “

4. Insert 2 and press the enter button

5. the system will show confirmation message asking the user “Do you want to leave the game?(yes/no):”

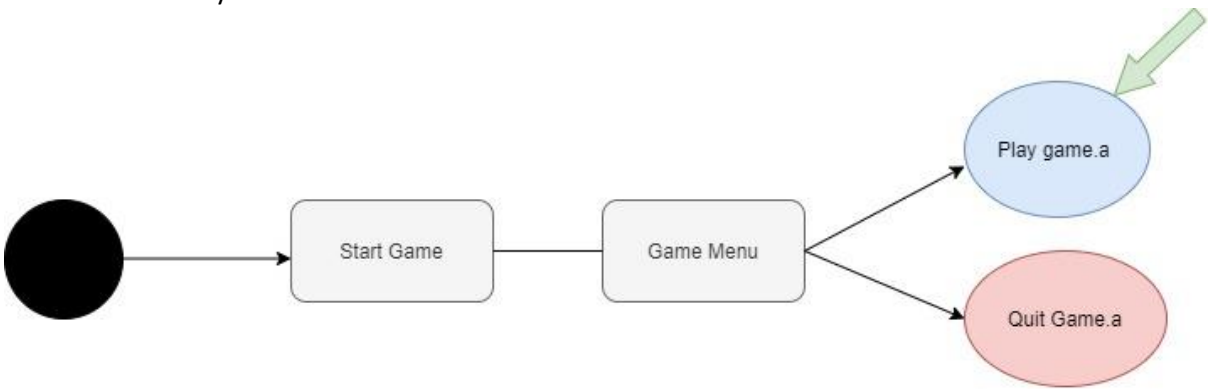
6. press “yes” and enter button.

Expected Results: The system will be terminated

Comment:

TC2.1 Player wins the game without any wrong guesses

Use case: UC3 Play Game



Scenario: The player guesses all the letters correctly without any mistake

Precondition: while the game is running, only one word will be picked from the word list library which is **(“heaven”)** and the rest of the words are commented out.

Test Steps:

- Starting the game
- The Hangman pole will be displayed and the player will be allowed to play by entering the letters he/she want
- Five letters will be entered (“h”, ”e”, ”a” ,”v” ,”e”, ”n”)
- Insert the “h” and press enter
- The hangman pole will be displayed plus the letter that is guessed and the player will be allowed to enter a new letter

```
  |
  |
  |
  |
 _|_
```

h_ _ _ _ _

Incorrect guesses:

type a letter:

2. Insert the “e” and press enter

The hangman pole will be displayed plus the letter that is guessed and the player will be allowed to enter a new letter

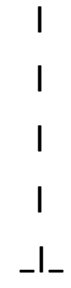
he _ _ _ _

Incorrect guesses:

type a letter:

3. Insert the “a” and press enter

The hangman pole will be displayed plus the letter that is guessed and the player will be allowed to enter a new letter



hea _ _ _

Incorrect guesses:

type a letter:

3**. Insert the “v” and press enter**

The hangman pole will be displayed plus the letter that is guessed and the player will be allowed to enter a new letter

heav_ _

Incorrect guesses:

type a letter:

4. Insert the “e” and press enter

The hangman pole will be displayed plus the letter that is guessed and the player will be allowed to enter a new letter

heave _

Incorrect guesses:

type a letter:

4. Insert the “n” and press enter

The hangman pole will be displayed plus the letter that is guessed and the player will be allowed to enter a new letter

Expected Results: The correct word will be shown, the hangman pole without extra elements and congratulations message with final score

```
  |
  |
  |
  |
 _|_
```

heaven

******** Well done you WIN :D *******

|| your score is: 100 ||

The word is: heaven

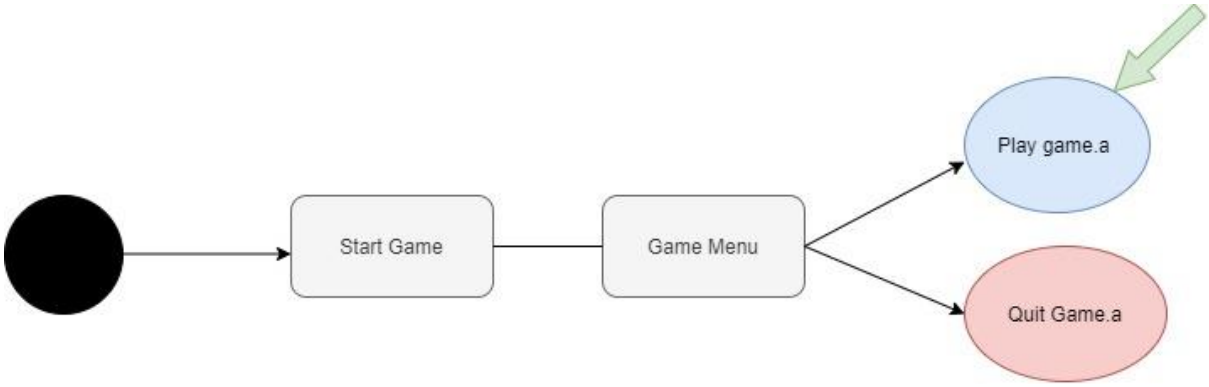
Actual results: The player have won the game successfully by guessing the missing words and a final score is shown

TC2.2 **Player losing the game without any correct guesses**

Use case:UC3

Scenario: The player choose the option Play Game

The main scenario of the UC2.2 is tested where the player loses the game without choosing any correct guesses .



Precondition: while the game is running, only one word will be picked from the word list library is **("hide")** and the rest of the words are commented out.

Test Steps:

1. Starting the game
2. The Hangman pole will be displayed and the player will be allowed to play by entering a letter

```
  |
  |
  |
  |
 _|_
_ _ _ _
```

Incorrect guesses:

type a letter:

- Insert the letter "P" and hit the enter button
- The hangman pole will show up plus the incorrect guess and the system shows "type a new letter " message.

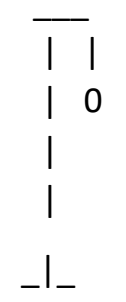
_ _ _ _

Incorrect guess: p

type a letter:

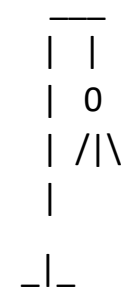
Insert the letter “z” and hit the enter button

- The hangman pole will show up plus the incorrect guess and the system shows “type a new letter ” message.



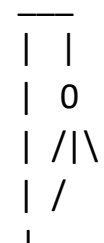
_ _ _ _
Incorrect guess: p z
type a letter:

- Insert the letter “l” and hit the enter button
- The hangman pole will show up ,plus the incorrect guess and the system shows a “type a new letter ” message.



_ _ _ _
Incorrect guess: p z l
type a letter:

- Insert the letter “w” and hit the enter button
- The hangman pole will show up plus the incorrect guess and the player will be allowed to enter the new letter



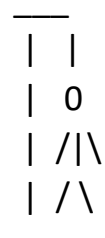
_ _ _ _

Incorrect guess: p z f w

type a letter:

- Insert the letter “w” and hit the enter button

Expected Results: The correct word will be displayed, hangman pole will get extra parts and losing message with the results “||your score is: 0||” .



******* YOU LOSE :(*******

|| your score is: 0 ||

The word is: hide

Actual Result: The player has lost the game, The hangman pole is shown with all the extra parts ,the correct word is displayed and losing message will appear with the final score.

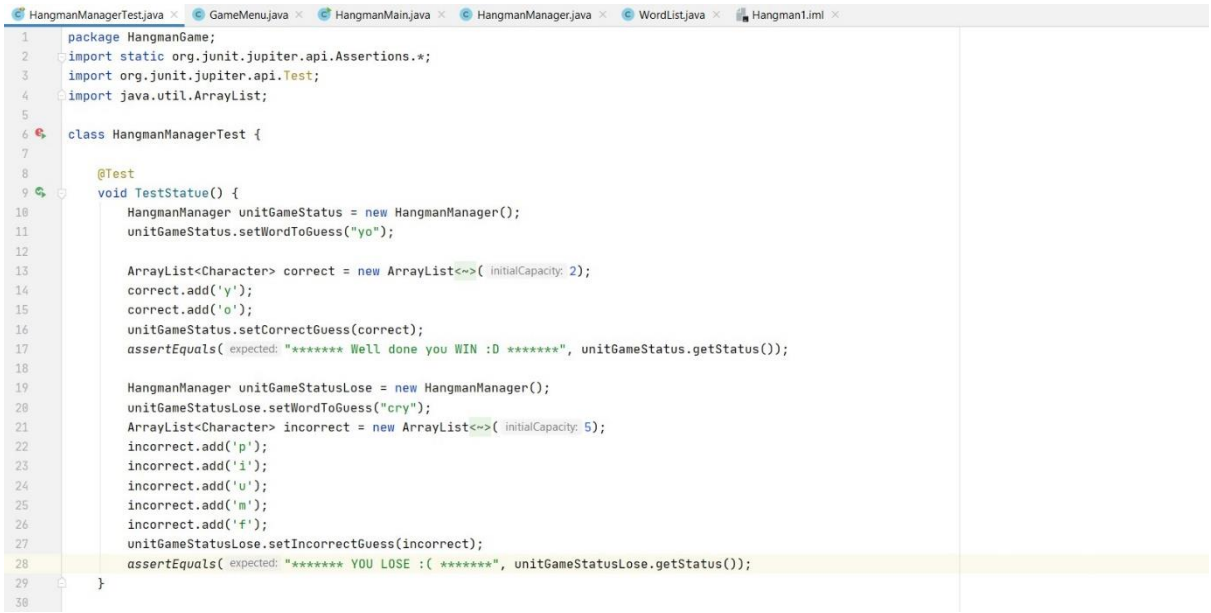
3. Test Report:

Test	UC1	UC2	UC3
TC1.1	1/Ok	0	0
TC1.2	0	1/Ok	0
TC2.1	0	0	1/Ok
TC2.2	0	0	1/OK
Coverage& Success	1/OK	1/OK	3/Ok

4. Test plan and unit test cases:

Link to my code: https://gitlab.lnu.se/1dv600/student/km222pw/assignment-3/-/blob/master/src/HangmanGame/HangmanManagerTest.java

TestStatue() method Test:

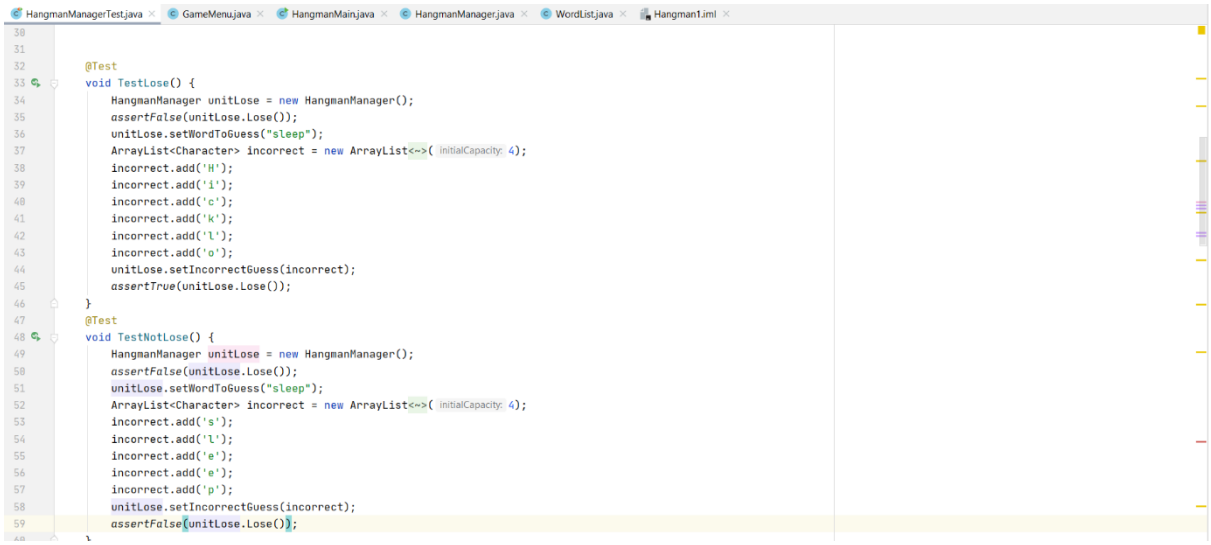


```java
package HangmanGame;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import java.util.ArrayList;

class HangmanManagerTest {

    @Test
    void TestStatue() {
        HangmanManager unitGameStatus = new HangmanManager();
        unitGameStatus.setWordToGuess("yo");

        ArrayList<Character> correct = new ArrayList<~>( initialCapacity: 2);
        correct.add('y');
        correct.add('o');
        unitGameStatus.setCorrectGuess(correct);
        assertEquals( expected: "******* Well done you WIN :D *******", unitGameStatus.getStatus());

        HangmanManager unitGameStatusLose = new HangmanManager();
        unitGameStatusLose.setWordToGuess("cry");
        ArrayList<Character> incorrect = new ArrayList<~>( initialCapacity: 5);
        incorrect.add('p');
        incorrect.add('i');
        incorrect.add('u');
        incorrect.add('m');
        incorrect.add('f');
        unitGameStatusLose.setIncorrectGuess(incorrect);
        assertEquals( expected: "******* YOU LOSE :( *******", unitGameStatusLose.getStatus());
    }
```

testWin() and testNotWin() methods:

```java
    }
    @Test
    void TestWin() {
        HangmanManager unitWin = new HangmanManager();
        assertFalse(unitWin.Win());
        unitWin.setWordToGuess("bye");
        ArrayList<Character> correct = new ArrayList<~>( initialCapacity: 4);
        correct.add('b');
        correct.add('y');
        correct.add('e');
        unitWin.setCorrectGuess(correct);
        assertTrue(unitWin.Win());
    }

    @Test
    void TestNotWin() {
        HangmanManager unitWin = new HangmanManager();
        assertFalse(unitWin.Win());
        unitWin.setWordToGuess("cheese");
        ArrayList<Character> correct = new ArrayList<~>( initialCapacity: 6);
        correct.add('b');
        correct.add('y');
        correct.add('e');
        unitWin.setCorrectGuess(correct);
        assertFalse(unitWin.Win());
    }
```

testLose() and testNotLose() methods:



```
@Test
void TestLose() {
    HangmanManager unitLose = new HangmanManager();
    assertFalse(unitLose.Lose());
    unitLose.setWordToGuess("sleep");
    ArrayList<Character> incorrect = new ArrayList<~>( initialCapacity: 4);
    incorrect.add('H');
    incorrect.add('i');
    incorrect.add('c');
    incorrect.add('k');
    incorrect.add('l');
    incorrect.add('o');
    unitLose.setIncorrectGuess(incorrect);
    assertTrue(unitLose.Lose());
}
@Test
void TestNotLose() {
    HangmanManager unitLose = new HangmanManager();
    assertFalse(unitLose.Lose());
    unitLose.setWordToGuess("sleep");
    ArrayList<Character> incorrect = new ArrayList<~>( initialCapacity: 4);
    incorrect.add('s');
    incorrect.add('l');
    incorrect.add('e');
    incorrect.add('e');
    incorrect.add('p');
    unitLose.setIncorrectGuess(incorrect);
    assertFalse(unitLose.Lose());
```

testSketch () method Test:

```
@Test
void TestSketch() {
    HangmanManager unit = new HangmanManager();
    String expected1 = " |\n |\n |\n |\n_|_";
    assertEquals(expected1, unit.draw());
    unit.setWordToGuess("name");
    ArrayList<Character> fill = new ArrayList<~>( initialCapacity: 4);
    fill.add('w');
    fill.add('q');
    fill.add('t');
    fill.add('u');
    String E = "  ___\n |   |\n |   o\n |  /|\\\n |  / \n_|_";
    unit.setIncorrectGuess(fill);
    assertEquals(E, unit.draw());
}
```

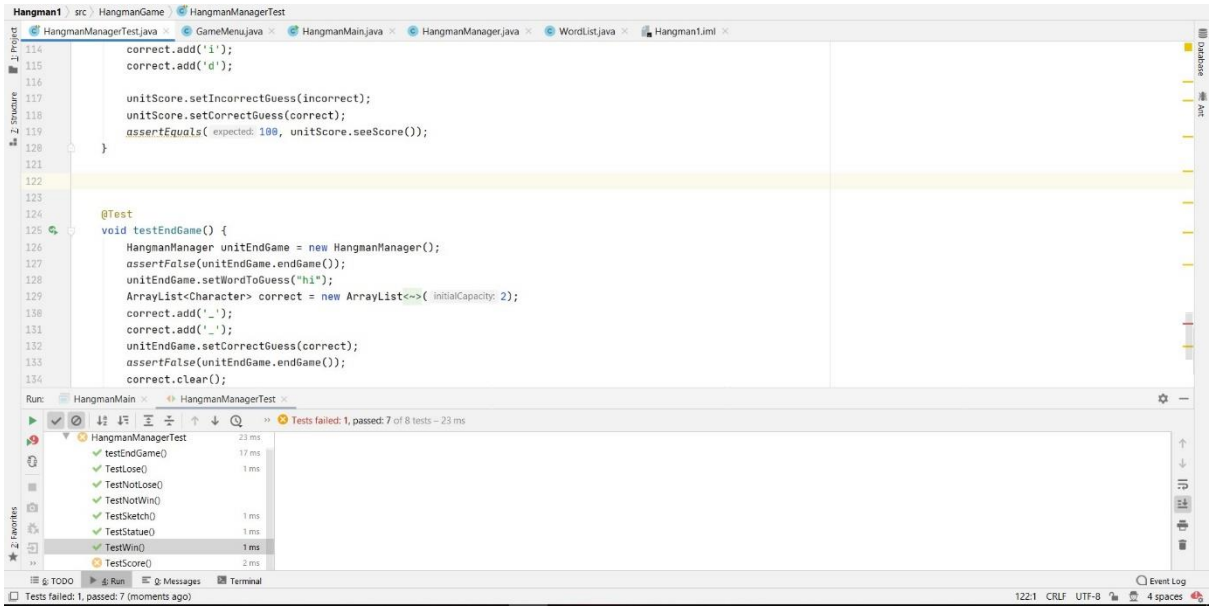
testScore()method Test:


```
@Test
void TestScore() { /////Note:this test will fail
    HangmanManager unitScore = new HangmanManager();
    unitScore.setWordToGuess("David");
    ArrayList<Character> correct = new ArrayList<~>( initialCapacity: 4);
    ArrayList<Character> incorrect = new ArrayList<~>( initialCapacity: 4);
    correct.add('D');
    correct.add('a');
    correct.add('v');
    correct.add('i');
    correct.add('d');

    unitScore.setIncorrectGuess(incorrect);
    unitScore.setCorrectGuess(correct);
    assertEquals( expected: 60, unitScore.seeScore());
}
```

testEndGame() method Test:

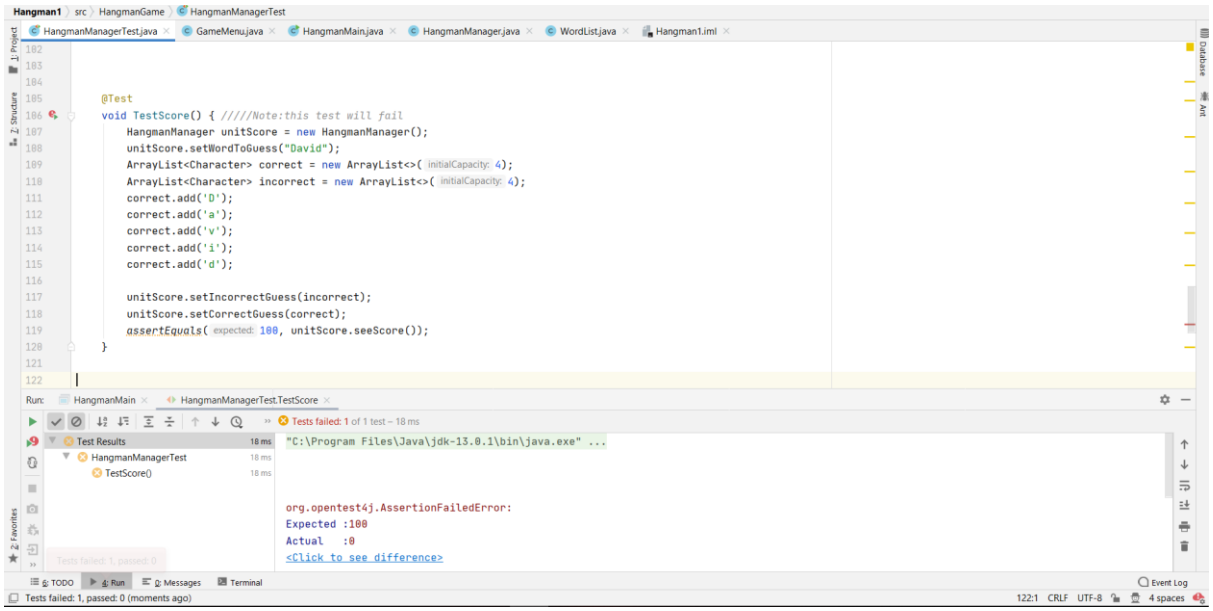

```
@Test
void testEndGame() {
    HangmanManager unitEndGame = new HangmanManager();
    assertFalse(unitEndGame.endGame());
    unitEndGame.setWordToGuess("hi");
    ArrayList<Character> correct = new ArrayList<~>( initialCapacity: 2);
    correct.add('_');
    correct.add('_');
    unitEndGame.setCorrectGuess(correct);
    assertFalse(unitEndGame.endGame());
    correct.clear();
    correct.add('w');
    correct.add('c');
    assertTrue(unitEndGame.endGame());
}

}
```

Test results:



so in this case all the tests will succeeded except for the TestScore() method test fails

The screenshot above show clearly why the testScore() method fails



5. Reflection:

This task gave me the possibility to improve my way of thinking and the logic of using Junit 5 in various cases in coding , although I faced some difficulties in setting a test plan at the beginning but later I realize that it just requires to practice my way of thinking and to try implementing test classes. It took me almost 3 days of planning to draw a test plan and I had to make some minor changes so the test methods will succeed

In general, In this iteration it was totally something new for me since I did not been that much testing compared to now, I totally understood the concept of testing and its benefits in coding, it made me realize how much its significant in all programming life.