Assignment 3

Osama Zarraa

oz222am@student.lnu.se

https://github.com/osamaz26/1dv600

Linneuniversitet

Date
2019-03-08

Content

1. Test plan 3

1.1 Introduction 3

1.2 Objectives 3

1.3 Team Members 3

1.4 What to test and why 3

1.5 Scope 3

1.6 Time plan 3

1.7 Risks 3

2. Manual Test 5

2.1 TC1 5

2.2 TC2 6

2.3 TC3 7

2.4 TC4 8

2.5 TC5 9

2.6 TC6 10

3. Automated Test 11

3.1 Word class 11

3.2 Player class 14

3.3 Game class 15

4. Test Report 16

5. Reflection 17

1. Test plan

1.1 Introduction

The Test Plan has been created to test hangman game project. It includes the objectives, what we are going to test and way. Also, it includes how to test and the time plan.

1.2 Objectives

The objective is to test part of the code that was implemented the last iteration.

1.3 Team Members

Resource Name	Role
Osama Zarraa	Tester

1.4 What to test and why

It is intended to test use cases (1,2,3) by writing and running dynamic manual test-cases. The reason for testing these use cases is to test the fundamental functionalities of the project.

1.5 Scope

To test the project, manual and automated tests are made and saved.

1.6 Time plan

Task	Estimated	Actual
Manual Test	2h	90m
Unit tests	3h	3.5h
Running manual tests	15m	8m
Code inspection	1h	30m
Test plan	2h	90m

1.7 Risks

#	Risk	Impact	Effect	Mitigation Plan
1	Working alone	High	Delays in implementation during the time frame	Have and follow a strict plan
2	Hardware damage	Low	Loss the project	Upload the project to cloud service

3	Sickness	Medium	Product did not get delivered on schedule	None

2. Manual Test

2.1 TC1

This is the manual test for the use case 1 start game to ensure that game starts without an issue.

ID	**TC1**
Requirement / Use Case Coverage	UC1
Name	Start game successfully
Precondition	• Java 11 is installed • The words file is existed
Test Steps	1. Pass path of words file as an argument 2. Start the application 3. System shows main menu 4. Enter 1 to start the game
Test data	Path for the existed words file
Expected	• The system should show the text "# Game started (Press * if you would like to interrupt the game)"
Actual	As expected
Status	Pass
Date	20190308
Comment	

2.2 TC2

This is the manual test for the use case 1 start game to check if system handles the file not found exception.

ID	**TC2**
Requirement / Use Case Coverage	UC1
Name	Start game unsuccessfully
Precondition	• Java 11 is installed • The words file is not existed
Test Steps	1. Pass a path of non-existed words file as an argument 2. Start the application
Test data	Path for the non-existed words file
Expected	• The system should show print a stack trace message of file not found.
Actual	As expected
Status	Pass
Date	20190308
Comment	

2.3 TC3

This is the manual test for the use case 2 play game to ensure that game has a win scenario.

ID	**TC3**
Requirement / Use Case Coverage	UC2
Name	Play game and win
Precondition	TC1
Test data	Characters
Test Steps	• User keeps entering correct characters individually until the wanted word is completed in less than 10 tries
Expected	• The system should show the text "# You nailed it" with the number of tries
Actual	As expected
Status	Pass
Date	20190308
Comment	-

2.4 TC4

This is the manual test for the use case 2 play game to ensure that game has a loose scenario.

ID	**TC4**
Requirement / Use Case Coverage	UC2
Name	Play game and lose
Precondition	TC1
Test data	Characters
Test Steps	• Player tries entering ten wrong characters
Expected	• A part of the hangman image should be printed each time a wrong character is entered, until the player run out of their tries then the full image should be shown.
Actual	As expected.
Status	Pass
Date	20190308
Comment	-

2.5 TC5

This is the manual test for the use case 2 play game to ensure that game has an interruption functionality.

ID	**TC5**
Requirement / Use Case Coverage	UC2
Name	Interrupt the game
Precondition	TC1
Test Steps	• Player enter * to quit the game • Player enters 1 for confirmation
Expected	• System should return to main menu
Actual	As expected
Status	Pass
Date	20190308
Comment	-

2.6 TC6

This is the manual test for the use case 3 play game to ensure that the game has a functional quit option.

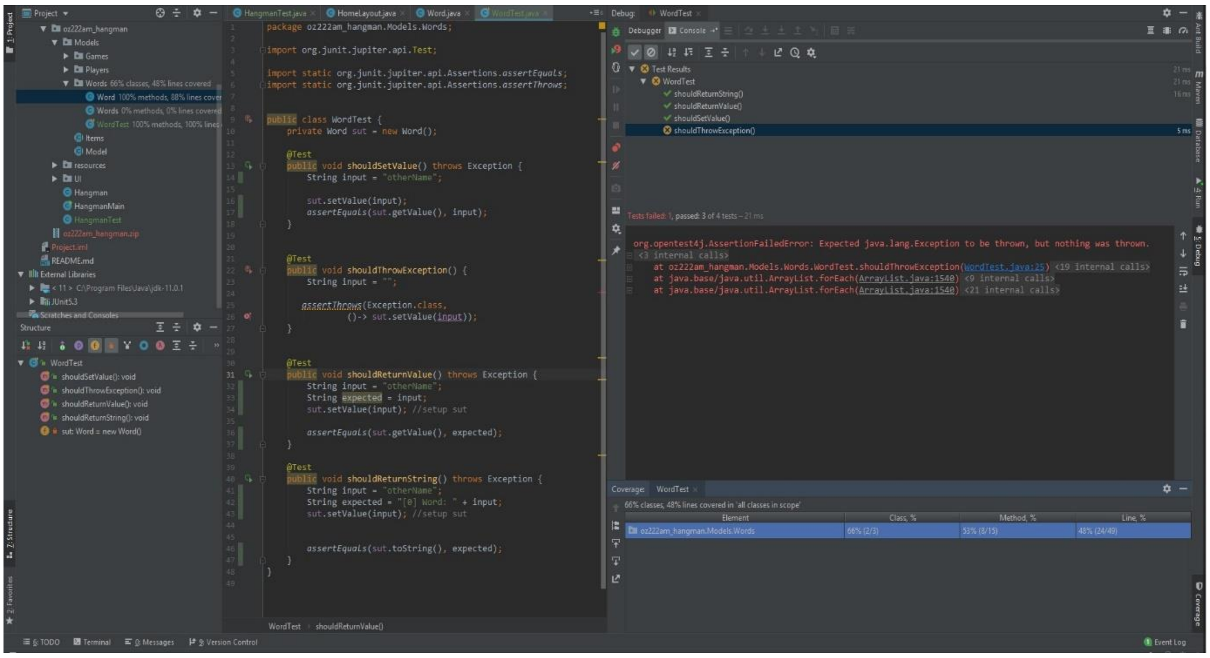
ID	**TC5**
Requirement / Use Case Coverage	UC3
Name	Quit the game
Precondition	TC1
Test Steps	• Player enters 4 to quit the game • Player enters 1 to confirm • System quit
Expected	• System should quit the game
Actual	As expected
Status	Pass
Date	20190308
Comment	-

3. Automated Test

3.1 Word class

The screenshot below shows “WordTest” class with four test methods.

As the screen shows, three methods passed the test and one does not pass due to the non-implemented method “validateValue(String word)” in the word class. Furthermore, It shows the coverage on the left side.



```java
package oz222am_hangman.Models.Words;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;


public class WordTest {
    private Word sut = new Word();

    @Test
    public void shouldSetValue() throws Exception {
        String input = "otherName";

        sut.setValue(input);
        assertEquals(sut.getValue(), input);
    }


    @Test
    public void shouldThrowException() {
        String input = "";

       assertThrows(Exception.class,
                ()-> sut.setValue(input));
    }


    @Test
    public void shouldReturnValue() throws Exception {
        String input = "otherName";
        String expected = input;
        sut.setValue(input); //setup sut

        assertEquals(sut.getValue(), expected);
    }

    @Test
    public void shouldReturnString() throws Exception {
        String input = "otherName";
        String expected = "[0] Word: " + input;
        sut.setValue(input); //setup sut


        assertEquals(sut.toString(), expected);
    }
}
```

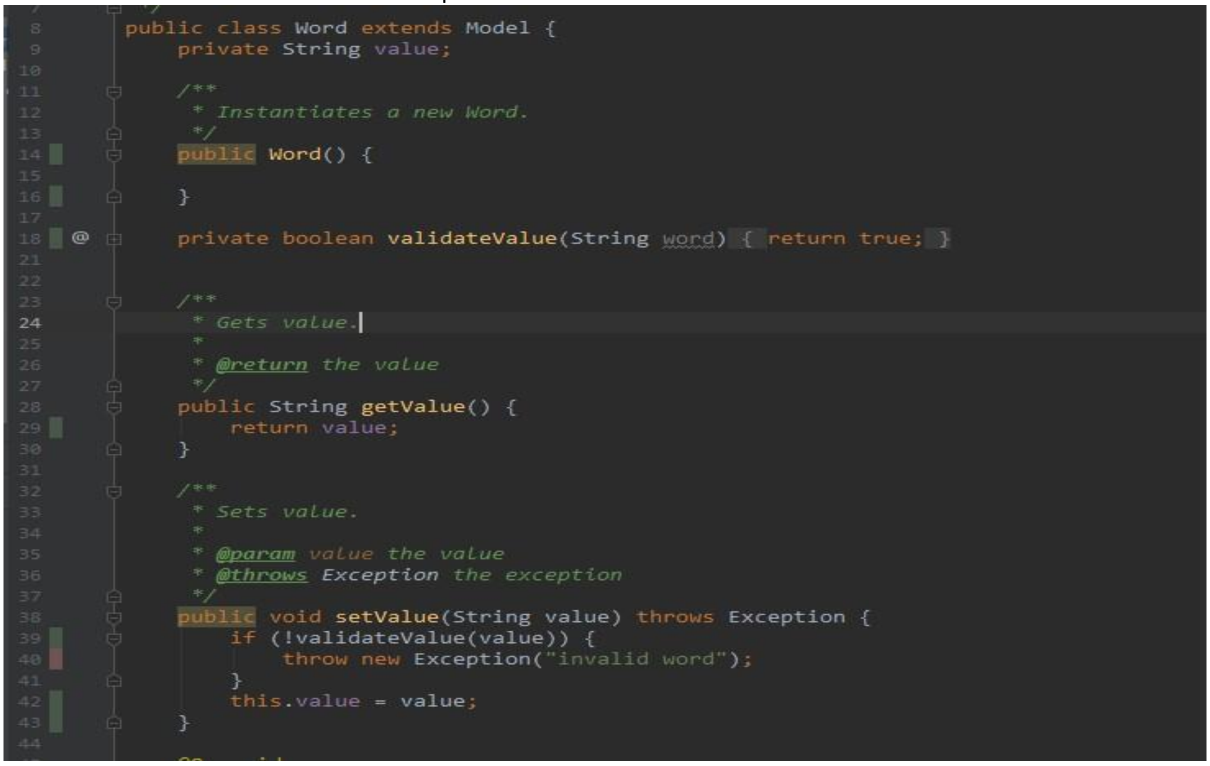
```
Test Results                         21 ms
  WordTest                           21 ms
    ✓ shouldReturnString()           16 ms
    ✓ shouldReturnValue()
    ✓ shouldSetValue()
    shouldThrowException()           5 ms

Tests failed: 1, passed: 3 of 4 tests – 21 ms

org.opentest4j.AssertionFailedError: Expected java.lang.Exception to be thrown, but nothing was thrown.
 <3 internal calls>
    at oz222am_hangman.Models.Words.WordTest.shouldThrowException(WordTest.java:25) <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>
```

Coverage: WordTest

66% classes, 48% lines covered in 'all classes in scope'

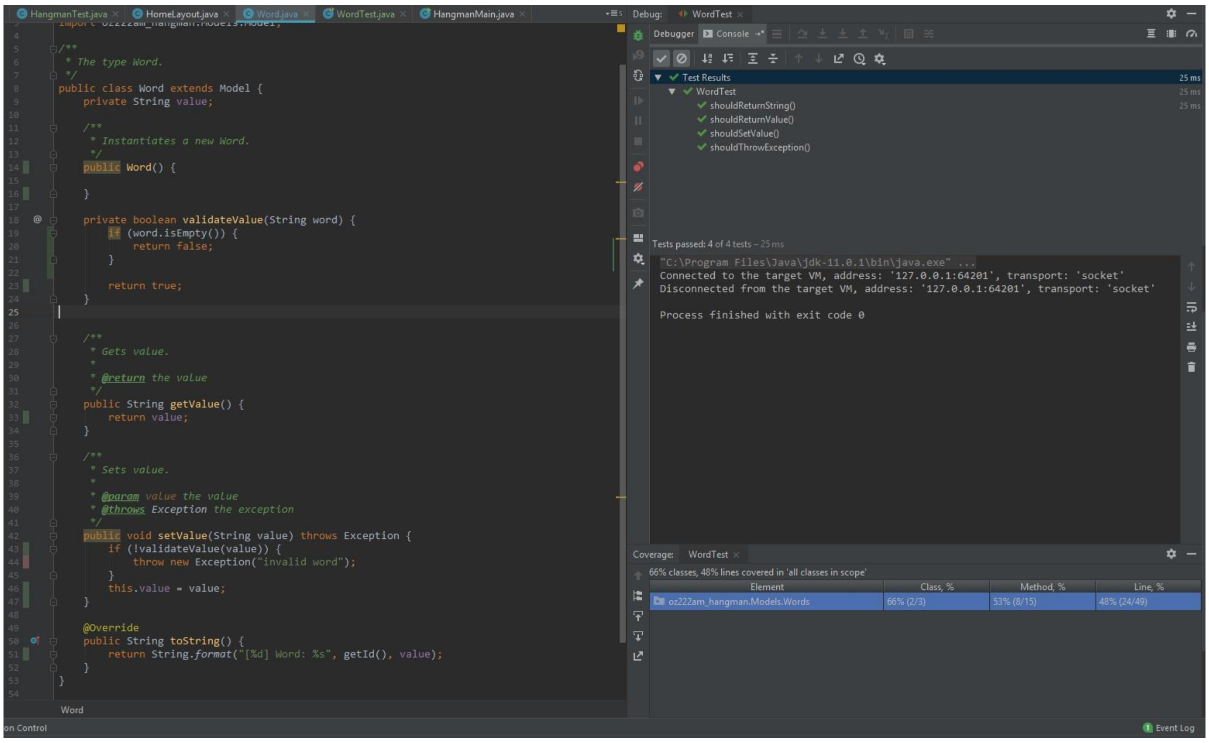
Element	Class, %	Method, %	Line, %
oz222am_hangman.Models.Words	66% (2/3)	53% (8/15)	48% (24/49)

The below screenshot shows the non-implemented “validateValue” method.



```java
public class Word extends Model {
    private String value;

    /**
     * Instantiates a new Word.
     */
    public Word() {

    }

    private boolean validateValue(String word) { return true; }


    /**
     * Gets value.
     *
     * @return the value
     */
    public String getValue() {
        return value;
    }

    /**
     * Sets value.
     *
     * @param value the value
     * @throws Exception the exception
     */
    public void setValue(String value) throws Exception {
        if (!validateValue(value)) {
            throw new Exception("invalid word");
        }
        this.value = value;
    }
```

The below screenshot represents the green phase of the “validateValue” method.



HangmanTest.java HomeLayout.java Word.java WordTest.java HangmanMain.java

```java
import oz222am_hangman.Models.Model;

/**
 * The type Word.
 */
public class Word extends Model {
    private String value;

    /**
     * Instantiates a new Word.
     */
    public Word() {

    }

    private boolean validateValue(String word) {
        if (word.isEmpty()) {
            return false;
        }

        return true;
    }


    /**
     * Gets value.
     *
     * @return the value
     */
    public String getValue() {
        return value;
    }

    /**
     * Sets value.
     *
     * @param value the value
     * @throws Exception the exception
     */
    public void setValue(String value) throws Exception {
        if (!validateValue(value)) {
            throw new Exception("invalid word");
        }
        this.value = value;
    }

    @Override
    public String toString() {
        return String.format("[%d] Word: %s", getId(), value);
    }
}
```

Debug: WordTest

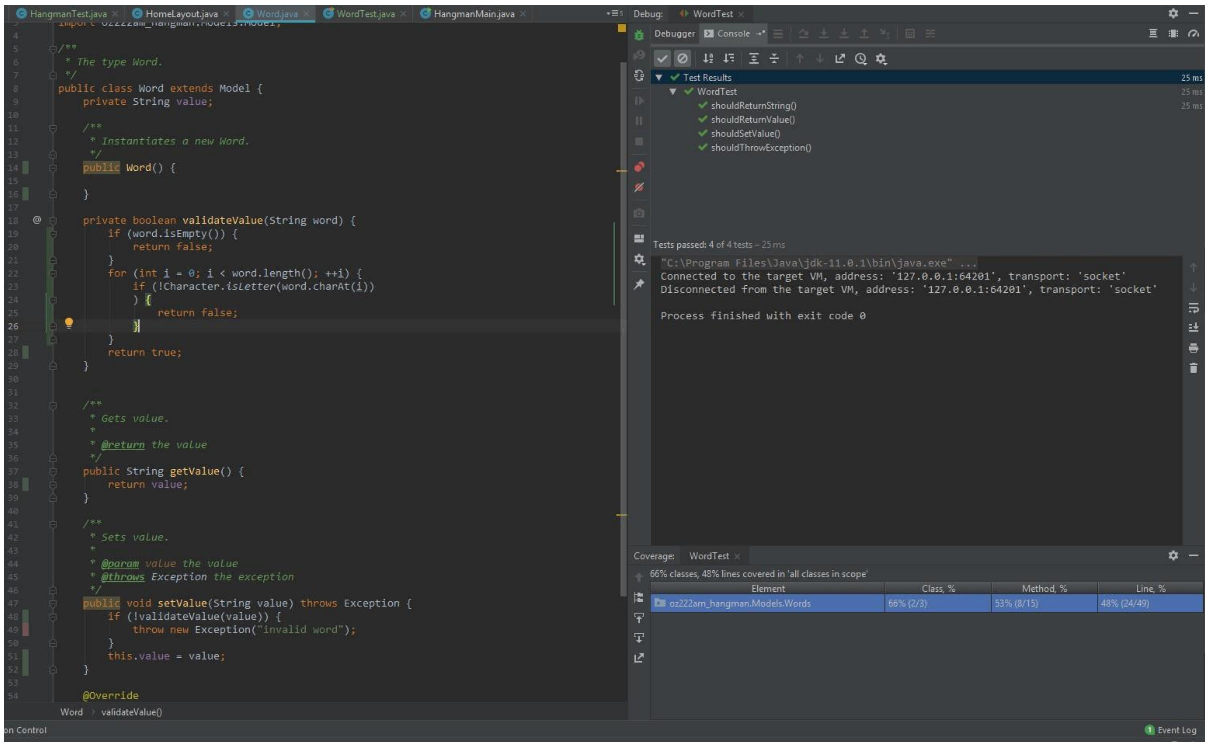
```
Test Results                       25 ms
  WordTest                         25 ms
    ✓ shouldReturnString()         25 ms
    ✓ shouldReturnValue()
    ✓ shouldSetValue()
    ✓ shouldThrowException()

Tests passed: 4 of 4 tests – 25 ms
"C:\Program Files\Java\jdk-11.0.1\bin\java.exe" ...
Connected to the target VM, address: '127.0.0.1:64201', transport: 'socket'
Disconnected from the target VM, address: '127.0.0.1:64201', transport: 'socket'

Process finished with exit code 0
```

Coverage: WordTest

66% classes, 48% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
oz222am_hangman.Models.Words	66% (2/3)	53% (8/15)	48% (24/49)

The below screenshot represents the refactor phase of the “validateValue” method.



HangmanTest.java HomeLayout.java Word.java WordTest.java HangmanMain.java

```java
import oz222am_hangman.Models.Model;

/**
 * The type Word.
 */
public class Word extends Model {
    private String value;

    /**
     * Instantiates a new Word.
     */
    public Word() {

    }

    private boolean validateValue(String word) {
        if (word.isEmpty()) {
            return false;
        }
        for (int i = 0; i < word.length(); ++i) {
            if (!Character.isLetter(word.charAt(i))
            ) {
                return false;
            }
        }
        return true;
    }


    /**
     * Gets value.
     *
     * @return the value
     */
    public String getValue() {
        return value;
    }

    /**
     * Sets value.
     *
     * @param value the value
     * @throws Exception the exception
     */
    public void setValue(String value) throws Exception {
        if (!validateValue(value)) {
            throw new Exception("invalid word");
        }
        this.value = value;
    }

    @Override
```

Word › validateValue()

Debug: WordTest

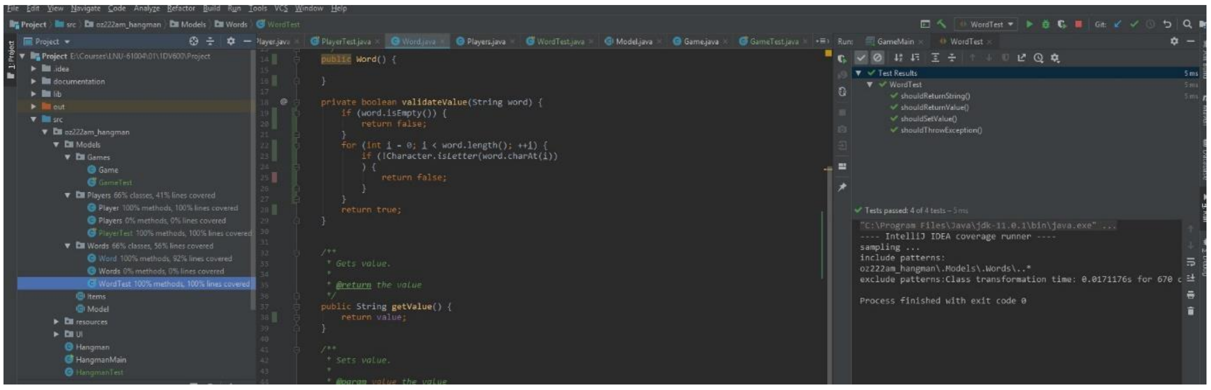
```
Test Results                       25 ms
  WordTest                         25 ms
    ✓ shouldReturnString()         25 ms
    ✓ shouldReturnValue()
    ✓ shouldSetValue()
    ✓ shouldThrowException()

Tests passed: 4 of 4 tests – 25 ms
"C:\Program Files\Java\jdk-11.0.1\bin\java.exe" ...
Connected to the target VM, address: '127.0.0.1:64201', transport: 'socket'
Disconnected from the target VM, address: '127.0.0.1:64201', transport: 'socket'

Process finished with exit code 0
```

Coverage: WordTest

66% classes, 48% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
oz222am_hangman.Models.Words	66% (2/3)	53% (8/15)	48% (24/49)

The below screenshot represents the coverage of the word test class.



```
Project E:\Courses\LNU-61004\01\1DV600\Project
  .idea
  documentation
  lib
  out
  src
    oz222am_hangman
      Models
        Games
          Game
          GameTest
        Players 66% classes, 41% lines covered
          Player 100% methods, 100% lines covered
          Players 0% methods, 0% lines covered
          PlayerTest 100% methods, 100% lines covered
        Words 66% classes, 56% lines covered
          Word 100% methods, 92% lines covered
          Words 0% methods, 0% lines covered
          WordTest 100% methods, 100% lines covered
        Items
        Model
      resources
      UI
      Hangman
      HangmanMain
      HangmanTest
```

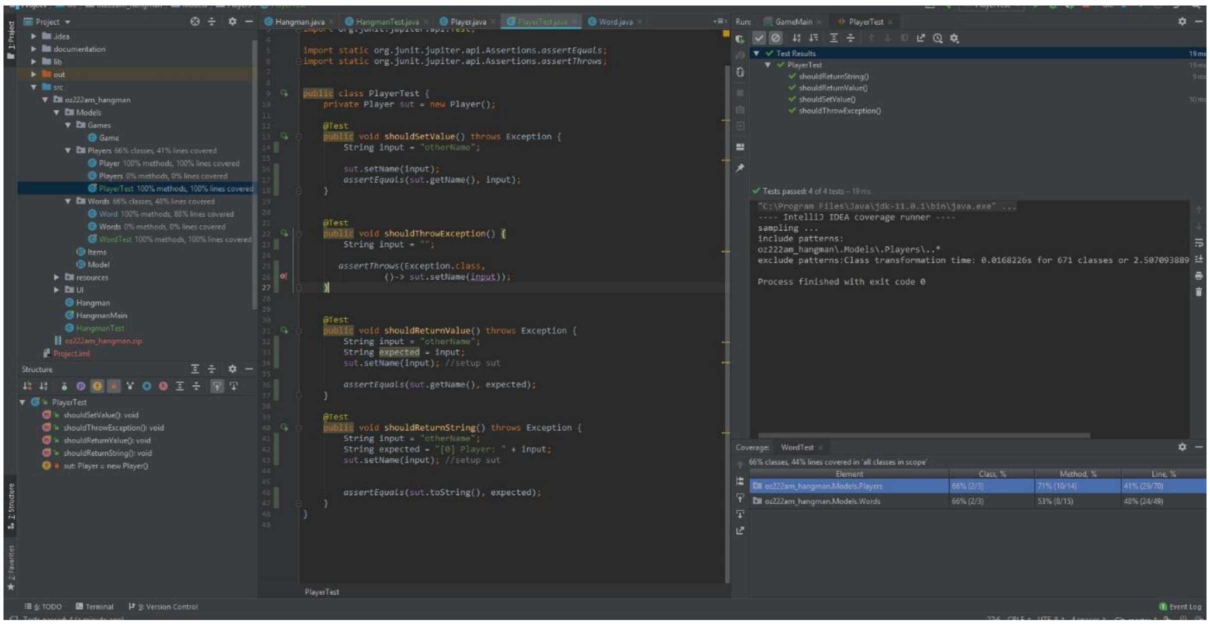
```java
    public Word() {

    }

    private boolean validateValue(String word) {
        if (word.isEmpty()) {
            return false;
        }
        for (int i = 0; i < word.length(); ++i) {
            if (!Character.isLetter(word.charAt(i))
            ) {
                return false;
            }
        }
        return true;
    }


    /**
     * Gets value.
     *
     * @return the value
     */
    public String getValue() {
        return value;
    }

    /**
     * Sets value.
     *
     * @param value the value
```

```
Test Results                         5 ms
  WordTest                           5 ms
    shouldReturnString()             5 ms
    shouldReturnValue()
    shouldSetValue()
    shouldThrowException()

Tests passed: 4 of 4 tests – 5 ms
```

```
"C:\Program Files\Java\jdk-11.0.1\bin\java.exe" ...
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
oz222am_hangman\.Models\.Words\..*
exclude patterns:Class transformation time: 0.0171176s for 670 c

Process finished with exit code 0
```

3.2 Player class

The screenshot below shows “PlayerTest” class, it includes four test methods and the code coverage of this class “100%” on the left side of the screen. All the methods have passed.



```
Players 66% classes, 41% lines covered
    Player 100% methods, 100% lines covered
    Players 0% methods, 0% lines covered
    PlayerTest 100% methods, 100% lines covered
Words 66% classes, 48% lines covered
    Word 100% methods, 88% lines covered
    Words 0% methods, 0% lines covered
    WordTest 100% methods, 100% lines covered
```

```
PlayerTest
    shouldSetValue(): void
    shouldThrowException(): void
    shouldReturnValue(): void
    shouldReturnString(): void
    sut: Player = new Player()
```

```java
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;


public class PlayerTest {
    private Player sut = new Player();

    @Test
    public void shouldSetValue() throws Exception {
        String input = "otherName";

        sut.setName(input);
        assertEquals(sut.getName(), input);
    }


    @Test
    public void shouldThrowException() {
        String input = "";

       assertThrows(Exception.class,
               ()-> sut.setName(input));
    }


    @Test
    public void shouldReturnValue() throws Exception {
        String input = "otherName";
        String expected = input;
        sut.setName(input); //setup sut

        assertEquals(sut.getName(), expected);
    }

    @Test
    public void shouldReturnString() throws Exception {
        String input = "otherName";
        String expected = "[0] Player: " + input;
        sut.setName(input); //setup sut


        assertEquals(sut.toString(), expected);
    }
}
```

```
Test Results                      19 ms
  PlayerTest                      19 ms
    shouldReturnString()           9 ms
    shouldReturnValue()
    shouldSetValue()              10 ms
    shouldThrowException()

Tests passed: 4 of 4 tests – 19 ms

"C:\Program Files\Java\jdk-11.0.1\bin\java.exe" ...
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
oz222am_hangman\.Models\.Players\..*
exclude patterns:Class transformation time: 0.0168226s for 671 classes or 2.507093889

Process finished with exit code 0
```

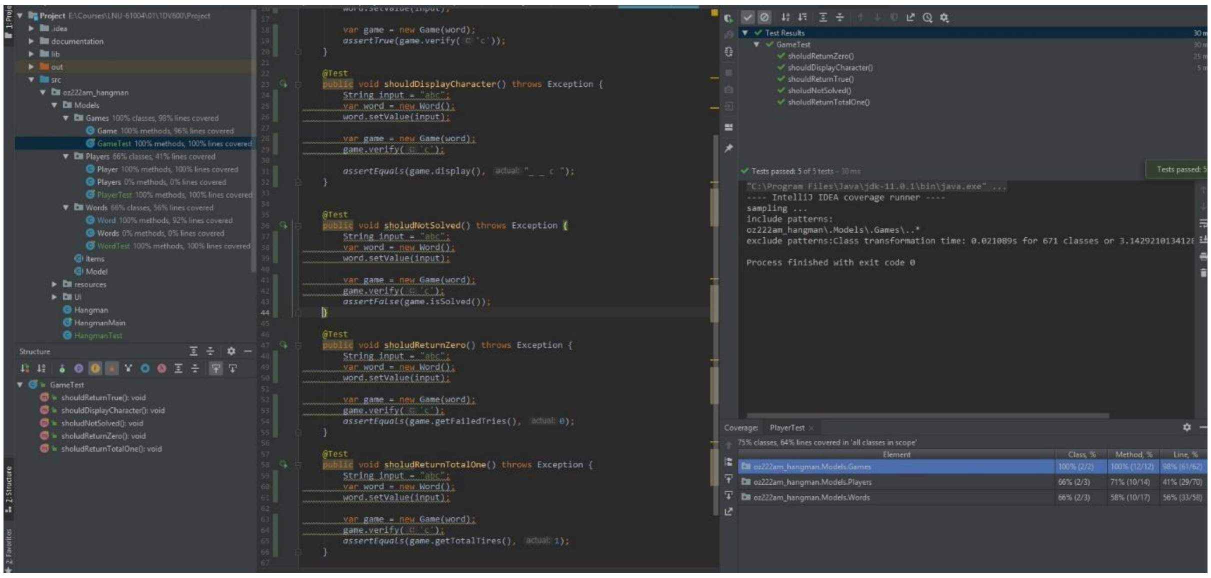
Coverage: WordTest

66% classes, 44% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
oz222am_hangman.Models.Players	66% (2/3)	71% (10/14)	41% (29/70)
oz222am_hangman.Models.Words	66% (2/3)	53% (8/15)	48% (24/49)

3.3 Game class

The screenshot below shows “GameTest” class, it includes five test methods and the code coverage of this class “100%” on the left side of the screen. All the methods have passed.



```
Project E:\Courses\LNU-61004\01\1DV600\Project
  .idea
  documentation
  lib
  out
  src
    oz222am_hangman
      Models
        Games 100% classes, 98% lines covered
          Game 100% methods, 96% lines covered
          GameTest 100% methods, 100% lines covered
        Players 66% classes, 41% lines covered
          Player 100% methods, 100% lines covered
          Players 0% methods, 0% lines covered
          PlayerTest 100% methods, 100% lines covered
        Words 66% classes, 56% lines covered
          Word 100% methods, 92% lines covered
          Words 0% methods, 0% lines covered
          WordTest 100% methods, 100% lines covered
        Items
        Model
      resources
      UI
      Hangman
      HangmanMain
      HangmanTest
```

```
Structure
GameTest
  shouldReturnTrue(): void
  shouldDisplayCharacter(): void
  sholudNotSolved(): void
  sholudReturnZero(): void
  sholudReturnTotalOne(): void
```

```java
        var game = new Game(word);
        assertTrue(game.verify( c: 'c'));
    }

    @Test
    public void shouldDisplayCharacter() throws Exception {
        String input = "abc";
        var word = new Word();
        word.setValue(input);

        var game = new Game(word);
        game.verify( c: 'c');

        assertEquals(game.display(), actual: "_ _ c ");
    }


    @Test
    public void sholudNotSolved() throws Exception {
        String input = "abc";
        var word = new Word();
        word.setValue(input);

        var game = new Game(word);
        game.verify( c: 'c');
        assertFalse(game.isSolved());
    }

    @Test
    public void sholudReturnZero() throws Exception {
        String input = "abc";
        var word = new Word();
        word.setValue(input);

        var game = new Game(word);
        game.verify( c: 'c');
        assertEquals(game.getFailedTries(), actual: 0);
    }

    @Test
    public void sholudReturnTotalOne() throws Exception {
        String input = "abc";
        var word = new Word();
        word.setValue(input);

        var game = new Game(word);
        game.verify( c: 'c');
        assertEquals(game.getTotalTires(), actual: 1);
    }
```

```
Test Results
  GameTest
    sholudReturnZero()
    shouldDisplayCharacter()
    shouldReturnTrue()
    sholudNotSolved()
    sholudReturnTotalOne()

Tests passed: 5 of 5 tests – 30 ms
```

```
"C:\Program Files\Java\jdk-11.0.1\bin\java.exe" ...
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
oz222am_hangman\.Models\.Games\..*
exclude patterns:Class transformation time: 0.021089s for 671 classes or 3.142921013412

Process finished with exit code 0
```

Coverage: PlayerTest

75% classes, 64% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
oz222am_hangman.Models.Games	100% (2/2)	100% (12/12)	98% (61/62)
oz222am_hangman.Models.Players	66% (2/3)	71% (10/14)	41% (29/70)
oz222am_hangman.Models.Words	66% (2/3)	58% (10/17)	56% (33/58)

4. Test Report

Test traceability matrix and success

Test	UC1	UC2	UC3
TC1	1/OK		
TC2	1/OK		
TC3		1/OK	
TC4		1/OK	
TC5		1/OK	
TC6			1/OK
Coverage & Success	2/OK	3/OK	1/OK

Automated unit test coverage and success

Test	Player	Word
PlayerTest	100%/OK	
WordTest		100%/OK
Coverage & Success	100%/OK	100%/OK

5. Reflection

Writing a test plan was not a very hard task since I found many templates on the internet that I can follow. However, the strange thing that I found is that we must submit all the reports (test plan, test report, manual test cases, and personal reflection) in one pdf file. Thus, I had confusion with organizing my final pdf document. I found many templates for the manual and automated test cases. So, I found no difficulties writing them. The estimated time that I sat in my time log was reasonable, well estimated and close to the actual.