HANGMAN PROJECT

Name: Khalil Mardini
Linnaeus University
Date:28-03-2021

Contents

Revision History

Date	Version	Description	Author
2020-02-02	01-200	For making the first iteration, on a different level such as the vision, Personal reflection, project plan, the structure of the hangman game and finely the risk analysis	Khalil Mardini
2021-03-25	01-201	Created the state machine diagram, class diagram and use case diagram, All were designed based on the code	Khalil Mardini
2020-05-30	01-202	Making Test plan, manual testing, and finally automated testing.	Khalil Mardini
2021-03-28	01-203	Summing up all the 3 iterations together, reflection and finely doing a general check on all the iterations	Khalil Mardini

General Information

Project Summary	
Project Name	Project ID
Hangman Game	01.200
Project Manager	Main Client
Khalil Mardini	Linnaeus University
Key Stakeholders	
1. Project Manager 2. Developer 3. End-User	
Executive Summary	
The main goal of this project is to construct the hangman game from the scratch, the user needs to select different alphabetical English letters to complete the missing words which is done through multiple attempts of guessing. The player will have to two choices, Play Game or Exit game. If the user wins or lose the score will be displayed based on the correct guesses followed by a congrats message and at the end of the game the player is asked either to play again or exit.	

Vision

Hangman is one the most popular games. It can be created by many different programming languages such as C, python, JavaScript ,etc. since Java language is the most common programming language that majority of developers use in creating such games, so in this project the java language will be used to achieve the goals of the project.

The whole thing behind this game is that after the developers implement all the fundamental features of creating the game, the user should guess a certain word by entering various letters. Varies words will be presented to the to the user to guess. Consequently, the user must select the correct letters in order to win the game with the deserved score. As a result, the game will be more exciting and challenging for the user since there are bunch of difficult words to guess.

The game will be already having a loaded list of words in English language, The wordlist contains various collection of words such as easy words (short word length) and hard words (long word length).
As a result, this will make the user face new challenges every time the game is played.

Personal Reflection:
Creating the hangman game helped me in improving my way of thinking especially when it comes to planning the scenarios of the game as well as my software design and coding skills. I faced some difficulties at the beginning since it's the first time to deal with such kind of project, and due to the lack of time and the project process was not clear for me at the beginning. Hopefully by using beneficial resources such as Udemy, Stack overflow, YouTube, and Java API it helped me in improving my way of thinking and in solving certain issues related to coding and designing my software project.

Project Plan

Hangman project planning will go through 4 different steps. Firstly, game clarification and specification will be illustrated to the reader, by showing how the game works and to provide the rules and the game instruction that need to be followed by the end user. Secondly, its to come up with new ideas and approaches regarding upgrading the game and adding new features apart from the one that already exist in the current version of the game. thirdly, the hangman game will be implemented on Java language, by generating various classes and methods and other programming which leads to accomplish the game basis.
The basic idea of Hangman is that the player is going to guess a word by suggesting letter after letter. The player is presented with the number of letters in the word but for every wrong guess, the system builds an element part of a man getting hanged. The number of wrong attempts that the player can commit is six, the scratch figure for the hangman in the system will be displayed using these description and symbols (ground, vertical pole, horizontal pole, head, body, left arm, right arm, left leg, and right arm or similar).

Moreover, a new feature is invoked so it will enable the player to play another round at the end of the game.

4.1 Introduction

The Hangman game will be created and designed to enable the user to guess the missing correct letters from the pre-defined word list. The player should find the missing letters depending on the luck.

4.2 Justification

The main task of the Software Technology course(1DV600) is to illustrate how the hangman game is being reconstructed and designed as well as to teach the essentials uses of the software development and enhance the planning skills.

4.3 Stakeholders

User: The user will be experiencing all the scenarios of the game and all the provided features by the game developer, to ensure that all the functions are working perfectly
so, the user in either can win or lose.

Developer: The duty of implementing the fundamental features of the game and to test the codes to ensure the stability of the game and to avoid software glitches.

End-User: is the person who utilize the product after it has been modified and developed so he/she can start playing the game and experience it.

4.4 Resources

The game creation will be based on various resources, but mainly it will be Java programming language, since it is easier to create the hangman game using this language and helps in structuring the game events, The platform that will be used is Java API
https://docs.oracle.com/javase/7/docs/api/
Also reading about how the hangman Game works from Wikipedia, give a broad idea about designing the critical poles of the game
https://www.wikipedia.org/

4.5 Hard- and Software Requirements

The foremost valuable programming environment is the IDE. So, the program will be made in IntelliJ IDE additionally the other predefined classes given from Java API.

4.6 Overall Project Schedule

- Iteration 1 – (week 5), 2020-02-02
- Iteration 2 – (week 12), 2021-03-25
- Iteration 3 – (week 21), 2020-05-30
- Final Iteration – (week 12), 2021-03-27

4.7 Scope, Constraints and Assumptions

The primary objective for this task is to make a simple game where the player must fill out the empty blank spaces by picking the missing letters. Therefore, the player will be confronted with two messages toward the start of the program that are "welcome to the Hangman game" and "Are you prepared to start?", that occurs inside the game. Two options will be displayed “Play game” and “Exit game” accessible for client to choose from and to make the game more challengeable and entertaining for the user.

Iterations:

The project in this course will be accomplished in 4 different iterations. The developer in this case is required to update and come up with new features and implement them in the hangman game, All that will be covered in the 4 stages of iterations which are mentioned below.
In the last stage, the developer should merge all the stages together so that then it will be ready to be delivered to the client to try the product.

5.1 Iteration 1

In this iteration will be the start point for the project, the project developer is required to the set a plan for this stage and determine how the game will be created.
The significant classes that will help in creating the game will be implemented. The most essential classes that will be used are the scanner and the array list. Scanner class will enable the user to enter the input for playing the game. The developer will write a code to create welcoming message to the player and ask the user whether to begin the game by selecting in between 2 options (Yes or No). so, to find the missing letters, the developer is required to create a predefined list of words, the predefined wordlist will be implemented by creating a class called (WordList.java). This can be done by establishing array that contains list of words in English language. More features and modification will be added in the next iterations.

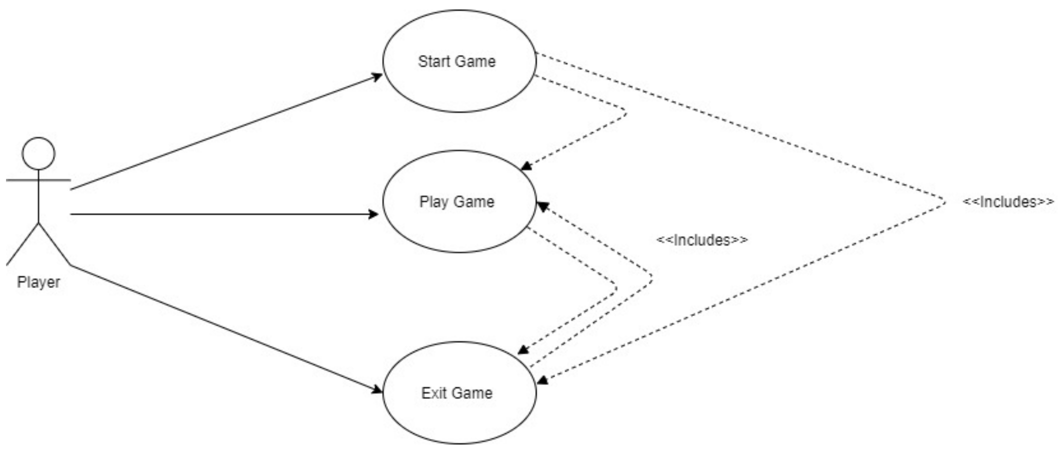
5.2 Iteration 2

In this iteration new concepts and visualization diagrams were added to the project so it will give a general understanding

In the second iteration, all the fundamental features have been added to the game from the first iterations. So the game will start by showing the welcoming message so that the user will choose whether to play the game by selecting between (Yes or No).if the user chooses yes , then the player will start the game by guessing the correct word through inserting the potential letter in the dotted line represented as (_ _ _ _ _ _), the player will only have 6 attempts for guessing the correct letter , if the player exceeded the 6 attempts the hangman figure will be displayed and built completely with a game over message.
Consequently, after finishing from playing the game, the player will be directed to a stage where the result will be displayed and ask if the player wants to play one more round.

To make the Iteration two more clear, I have added the Use cases, state machine and finely the class diagram as follows:

Use Case Diagram:



Dress Use Cases:

Use Case 1: Start Game

Precondition:
N/A

Stakeholder:
The player (End-User)

Main Scenario:
- The game will start the moment the player initializes the game.

- The system will display the welcoming menu to the player with two option to choose “Play Game “and “Exit Game “.
- The player chooses the “Play Game “option.
- The system will convert the player to the next menu (UC 2: Play Game)

Alternative Scenarios:
The player chooses the “Exit game” option and the game system will terminate.

Use Case 2: Play Game

Precondition:
UC 1: (Start Game)

Stakeholder:
The player (End-User)

Main Scenario:
- The player will be trying to guess the correct word by entering possible letter.
- The player will win the game followed by a congrats message and the deserved score.
- The system will ask the player if he/she wants to play one more round or not.
-

Alternative Scenarios:
- The Player will lose the game by guessing the wrong word followed by message stating “YOU LOSE” and the deserved score.
- The system will ask the player if he/she wants to play one more round or not.

Use Case 3: Exit Game

Precondition:
UC 2: Play game

Stakeholder:
The player (End-User)

Main Scenario:

- The player will be trying to guess the correct word by entering possible letter.
- The player types “0”
- The system will display a message to the player stating “Are you sure you want to quit the game? (yes/no): “
- The player enters “yes” then the system will terminate the game.

Alternative Scenarios**:**

- The player will be trying to guess the correct word by entering possible letter.
- The player types “0”
- The system will display a message to the player stating “Are you sure you want to quit the game? (yes/no): “
- The player enters “no”
- The player will continue playing the before the time he/she entered the “0” as an input.

Use Case 3.1: Exit Game

Precondition:
UC 1: (Start Game)

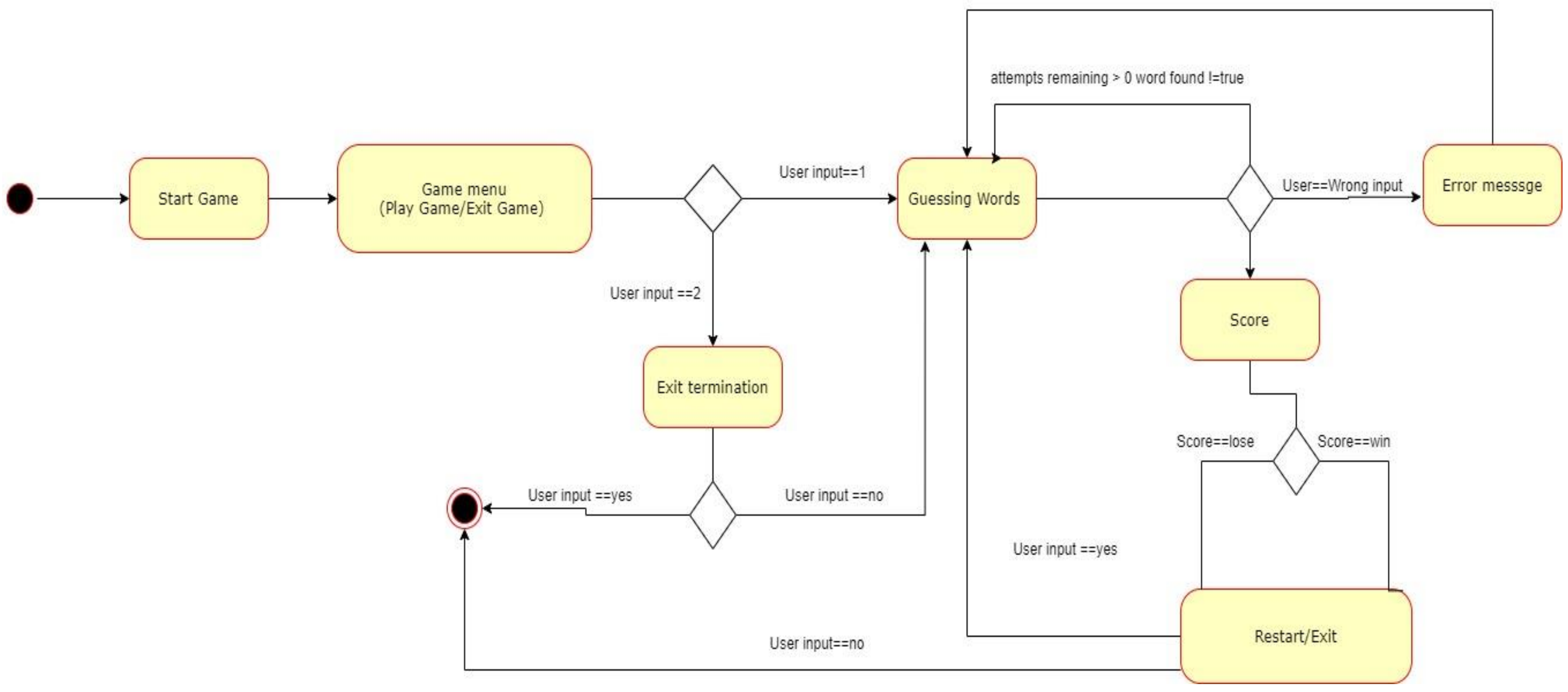
Stakeholder:
The player (End-User)
Main Scenario:

- The system will display the start menu of the game and ask the player either to “Play Game “or “Exit Game “
- The player selects the “Exit Game” option.
- The system will pop up a message for the Player stating “Do you want to leave the game? (yes/no):”
- The player chooses “yes.”
- The game will terminate.

Alternative Scenarios:

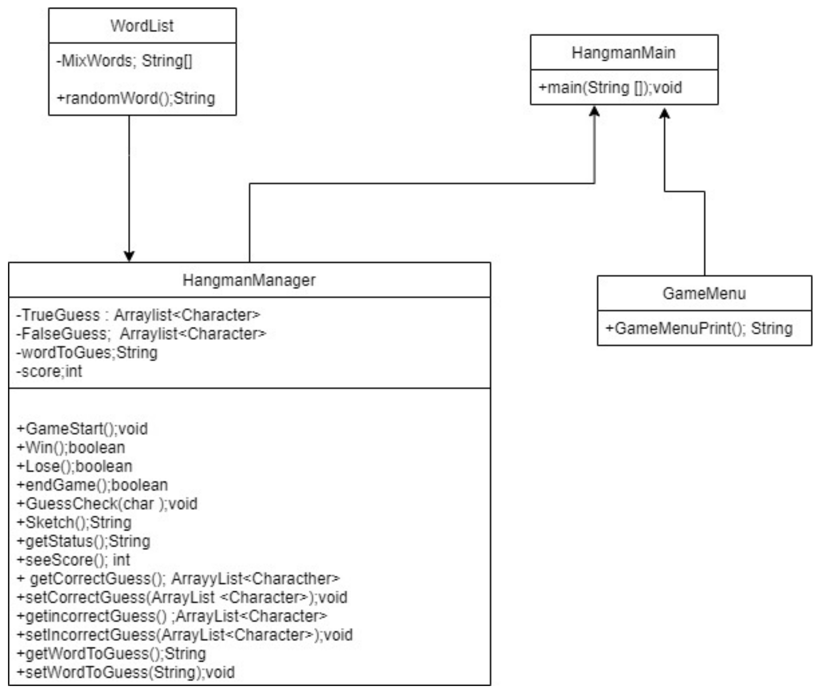
- The system will display the start menu of the game and ask the player either to “Play Game “or “Exit Game “
- The player selects the “Exit Game” option.
- The system will pop up a message for the Player stating “Do you want to leave the game? (yes/no):”
- The player chooses “no”
- The game will start by allowing the player to start guessing the mysterious word.

<u>State Machine diagram:</u>



The State Machine diagram above illustrates the procedure flow of the game, so the moment the game has started, the player gets two options either to “Play Game” or “Exit Game” , Even “Exit termination” and “Restart/Exit” have similar functionality

Class Diagram:



The Class Diagram above illustrates the structure of the code and how different classes depends on each other. The HangmanManager.java is the most significant class that constructs the game, since it contains the useful functionalities and methods.

Type	Actual Time Required	Finish Date
Use Cases	3 hours	2021-03-22
State Machine	6 hours	2021-03-23

Class Diagram	2 hours	2021-03-24
Design & Implementation	4 hours	2021-03-25

5.3 Iteration 3

The main goal of this Iteration 3 is to test the provided code by constructing a test class “HangmanManagerTest.java” that it is task to test the process of guessing the missing letters from a predefined wordlist class.

Test cases and Junit testing were implemented on the code by the developer so it will help in evaluating the project and detecting any potential errors in the logic of the game.

In this Iteration there will be some tests that will pass and some test that will fail by purpose.

Creating the test cases will help the developer of the game to ensure that everything is working properly and to avoid worse scenarios by estimating that everything is implemented perfectly.

Task	**Estimated Time**	**Actual Time**
Automated unit testing	3h	4h

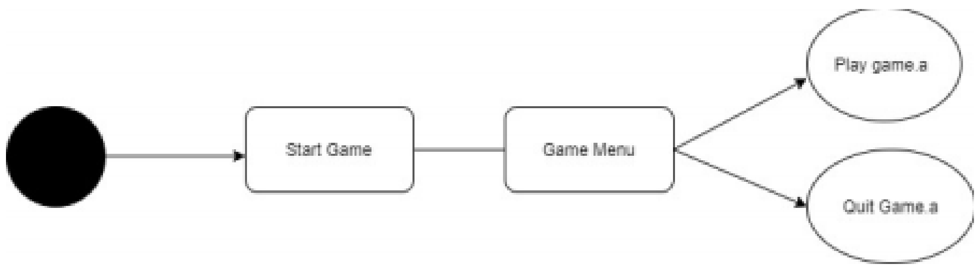
Test plan	2h	6h
Manual test cases	2h	3h
Report and Time log	1h	2h
Upgrade automated unit testing	30m	1h
Upgrade manual test cases	30m	40min
Upgrading Test plan	30m	1h
Code check	1h	30m

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:

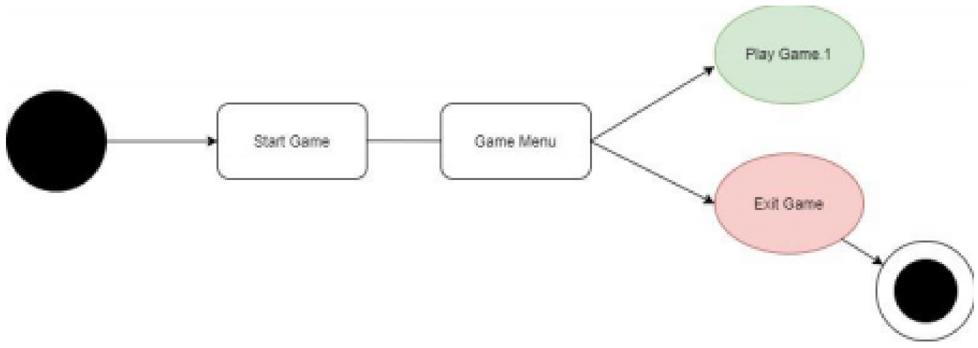
Note: more empty blanks might appear depends on the picked word from the library word list But in this test case I am showing 4 empty blanks

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):”

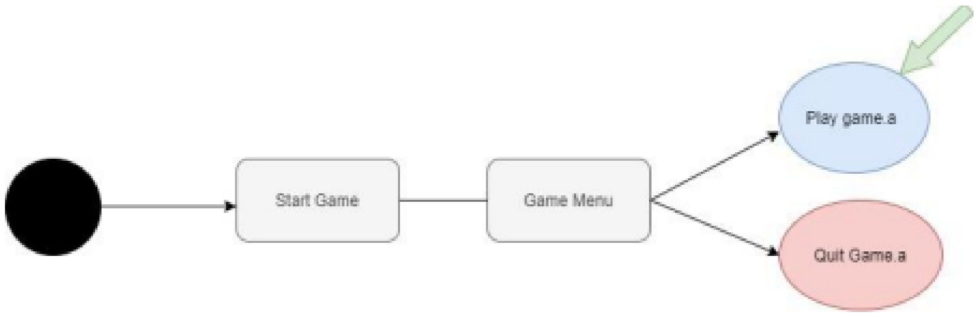
6. press “yes” and enter button.

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:

1. Starting the game
2. The Hangman pole will be displayed and the player will be allowed to play by entering the letters he/she want
3. Five letters will be entered (“h”, ”e”, ”a” ,”v” ,”e”, ”n”)
4. Insert the “h” and press enter
5. 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.
```

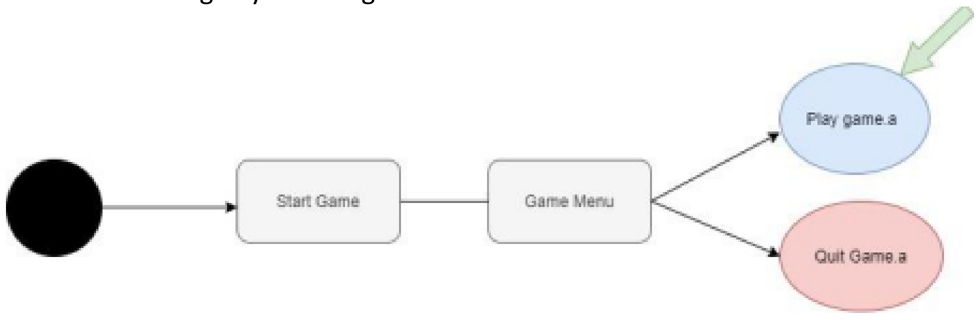
Actual results: The player have won the game successfully by guessing the missing words and a final score is shown

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

Use case: UC3

Scenario: The player chooses 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.

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.

```
  ___
  | |
  | 0
  |
  |
 _|_

 _ _ _ _
 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.

```
  ___
  |  |
  |  0
  | /|\
  |

_|_



_ _ _ _
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

```
  ___
  |  |
  |  0
  | /|\
  | /
```

```
_|_
_ _ _ _
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||”.

```
  ___
  |  |
  |  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:

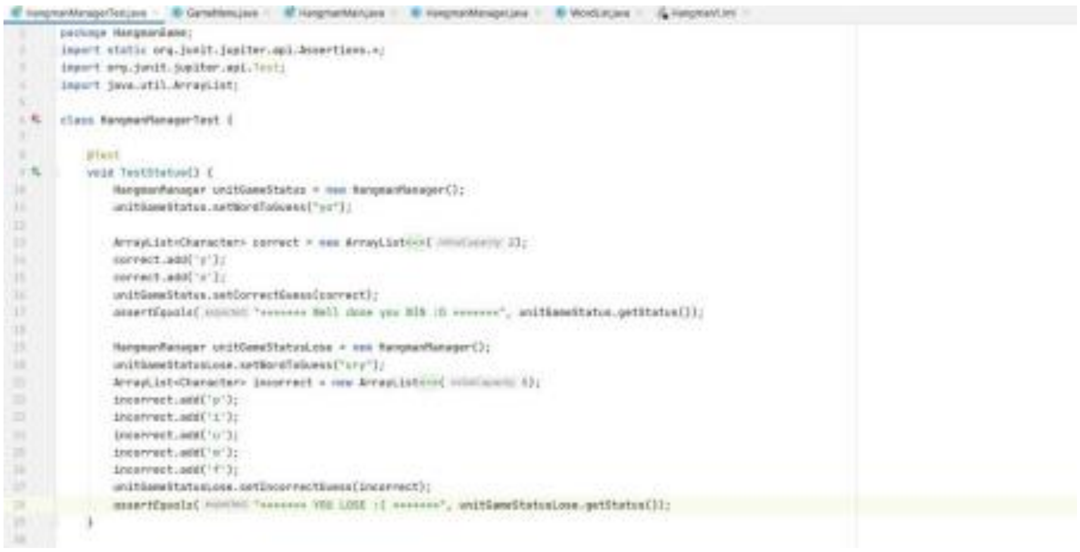
Test	UC1	UC2	UC3
TC1.1	1/Ok	0	0
TC1.2	0	1/Ok	0
TC2.1	0	0	1/Ok

TC2.2	0	0	1/OK
Coverage& Success	1/OK	1/OK	3/Ok

4. Test plan and unit test cases:

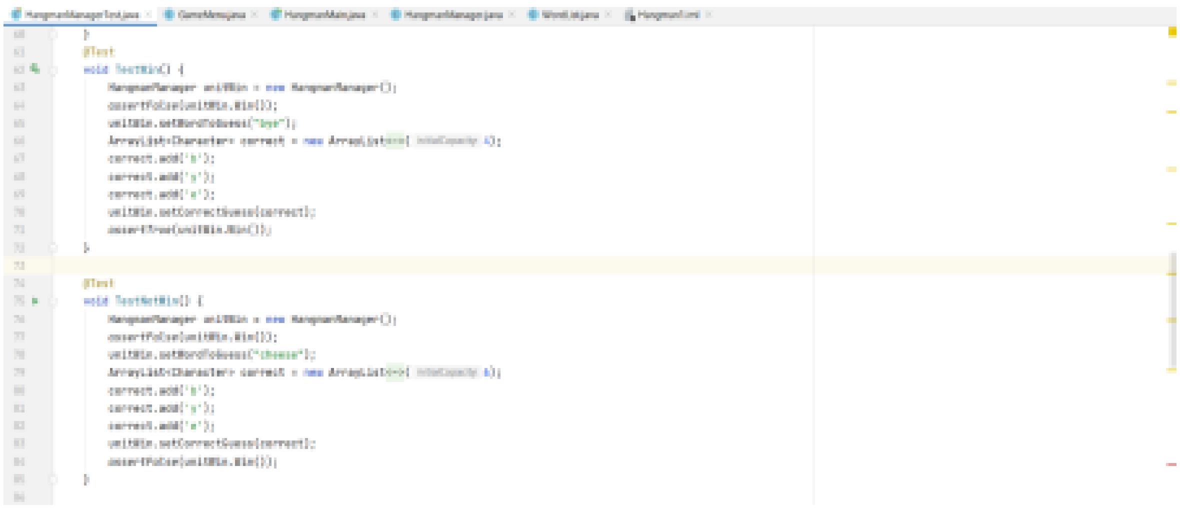
Link to my code:
https://gitlab.lnu.se/1dv600/student/km222pw/assignment-3/-/blob/master/src/HangmanGame/HangmanManagerTest.java

TestStatue() method Test:



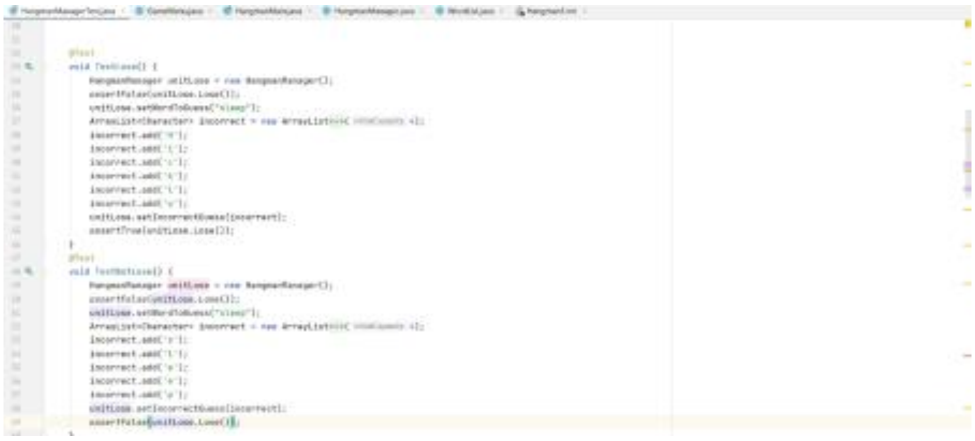
```java
package HangmanGame;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import java.util.ArrayList;

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("sry");
        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:



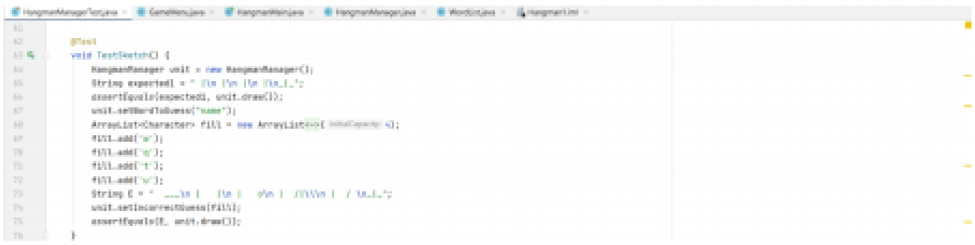
```java
    }
    @Test
    void TestWin() {
        HangmanManager unitWin = new HangmanManager();
        assertFalse(unitWin.Win());
        unitWin.setWordToGuess("bye");
        ArrayList<Character> correct = new ArrayList<>( initialCapacity: 4);
        correct.add('b');
        correct.add('y');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

testLose() and testNotLose() methods:



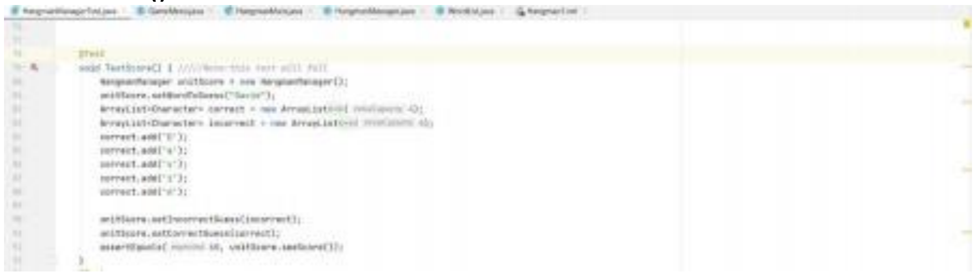
```java
    @Test
    void TestLose() {
        HangmanManager unitLose = new HangmanManager();
        assertFalse(unitLose.Lose());
        unitLose.setWordToGuess("sleep");
        ArrayList<Character> incorrect = new ArrayList<>( initialCapacity: 4);
        incorrect.add('[illegible]');
        incorrect.add('t');
        incorrect.add('c');
        incorrect.add('k');
        incorrect.add('t');
        incorrect.add('v');
        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('r');
        incorrect.add('t');
        incorrect.add('a');
        incorrect.add('v');
        incorrect.add('u');
        unitLose.setIncorrectGuess(incorrect);
        assertFalse(unitLose.Lose());
    }
```

testSketch () method Test:



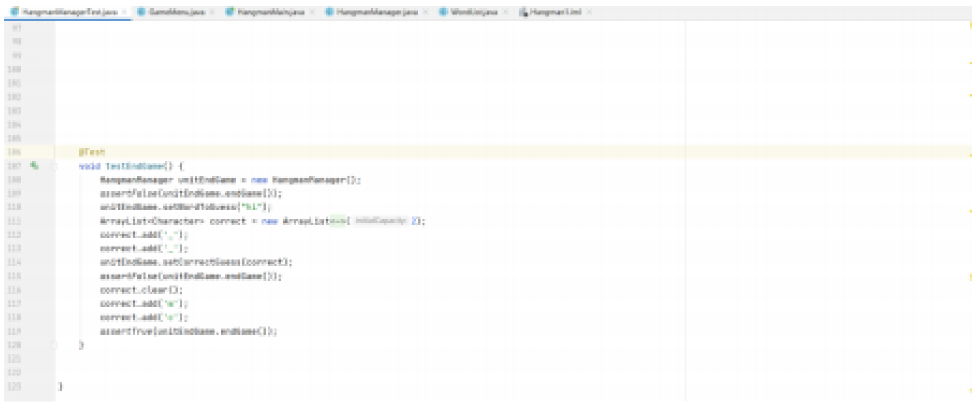
```java
    @Test
    void TestSketch() {
        HangmanManager unit = new HangmanManager();
        String expected1 = " |\n |\n |\n |\n_|_";
        assertEquals(expected1, unit.draw());
        unit.setWordToGuess("name");
        ArrayList<Character> fill = new ArrayList<>( initialCapacity: 4);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

testScore()method Test:



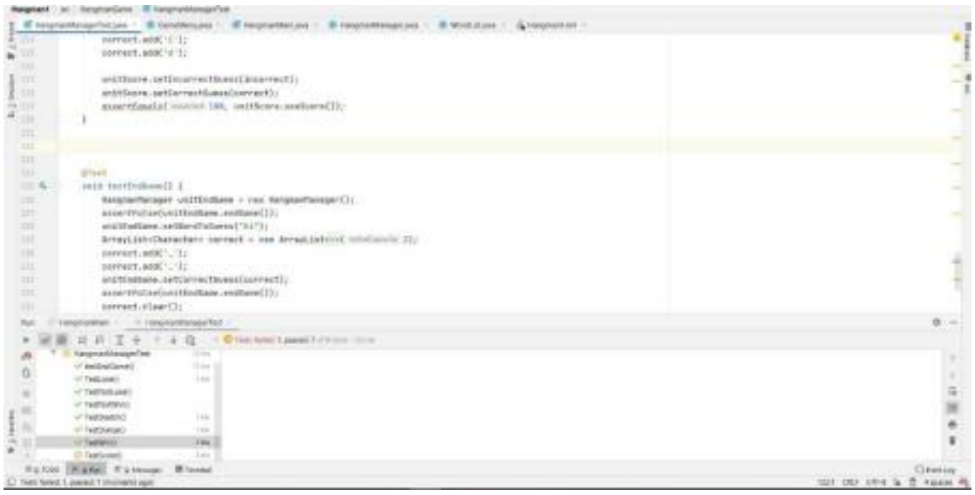
```java
    @Test
    void TestScore() { /////Note:this test will fail
        HangmanManager unitScore = new HangmanManager();
        unitScore.setWordToGuess("Savio");
        ArrayList<Character> correct = new ArrayList<>( initialCapacity: 4);
        ArrayList<Character> incorrect = new ArrayList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

        unitScore.setIncorrectGuess(incorrect);
        unitScore.setCorrectGuess(correct);
        assertEquals( expected: 10, unitScore.seeScore());
    }
```

testEndGame() method Test:



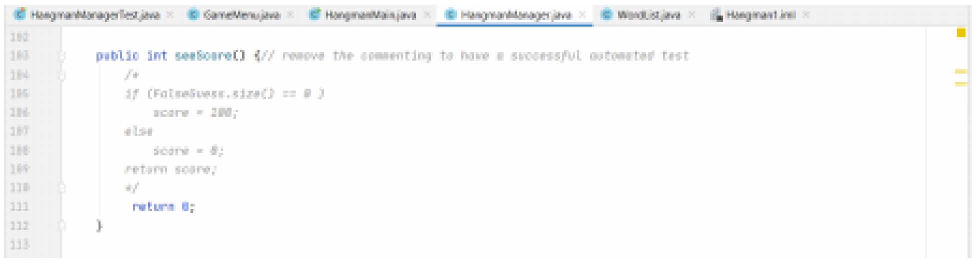
```java
    @Test
    void TestEndGame() {
        HangmanManager unitEndGame = new HangmanManager();
        assertFalse(unitEndGame.endGame());
        unitEndGame.setWordToGuess("hi");
        ArrayList<Character> correct = new ArrayList<>( initialCapacity: 2);
        correct.add('_');
        correct.add('_');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


}
```

Test results:



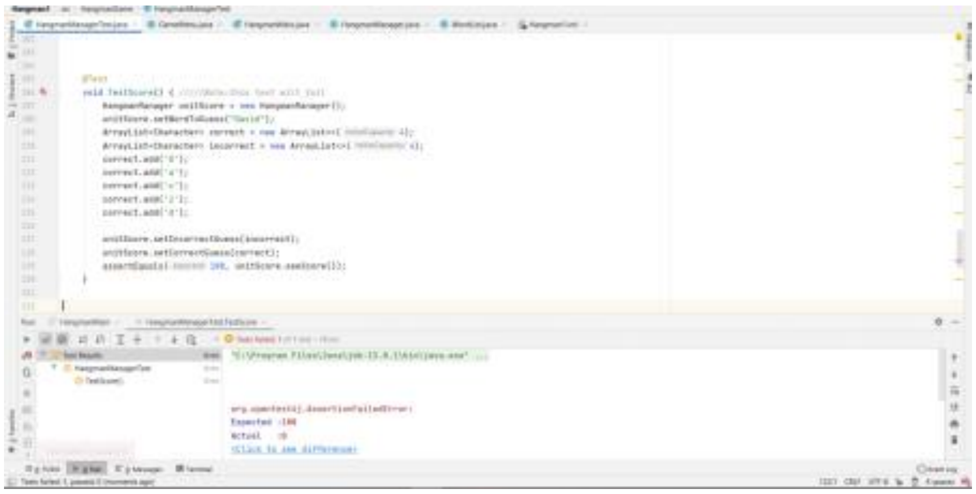
```java
        correct.add('i');
        correct.add('v');

        unitScore.setIncorrectGuess(incorrect);
        unitScore.setCorrectGuess(correct);
        assertEquals( expected: 100, unitScore.seeScore());
    }


    @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();
```

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



```java
    public int seeScore() {// remove the commenting to have a successful automated test
        /*
        if (FalseGuess.size() == 0 )
            score = 100;
        else
            score = 0;
        return score;
        */
         return 0;
    }
```

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



```
@Test
void TestScore() { //[illegible]
    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: 100, unitScore.seeScore());
}
```

```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" ...

org.opentest4j.AssertionFailedError:
Expected :100
Actual   :0
<Click to see difference>
```

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 invoke the 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.

Test type	Time dedicated	Start Date	Date to Finish
Manual testing	10 hours	2020-05-28	2020-05-28
Aumtaed testing (Junit)	6 hours	2020-05-29	2020-05-29

5.4

Iteration 4

The game will be delivered to the client according to the client wishes and the required specifications. In this iteration the main target is to merge all the iterations that were made. So the first iteration was about planning, the second iteration was about diagrams and code implementations and the third one was about testing the some parts of the code. So the main goal of iteration 4 is to ensure about the functionality of the project by debugging the source code and check that everything matches the requirements, An extra feature to the game was added, such as enabling the player to play another round after the game is over. More of that exception handing was added to the code so that will make the game super flow and well structured. Another interesting point is I invoked the score criteria out of 100 which is based on the number of mistakes the player had commit , for instance ,if the player inserted one wrong letter and managed to guess the rest the deserved score will be 75, for 2 and 3 wrong letter guess is 50, for 4 wrong letters 25, for 5 wrong letters 10 and finally 0 if the player failed to guess no of the possible letters.

Task type	Time dedicated	Start Date	Date to Finish
Code check (Debugging)	2 hours	2021-03-26	2021-03-26
Invoking new features	4 hours	2021-03-26	2021-03-26

Risk Analysis

Generally, every project might face risks and obstacles on its way, so the developer should be aware of the worst case, have backup plans and tactics to avoid any potential risk , due to that, the chapter called project management (22) in the Software Engineering book , this chapter illustrate the main risks that might encounters the project and the safety steps that should be followed to avoid the risks, it also provide useful tips to the project managers to minimize the impact of the risk.

6.1 List of risks

Risk identification and management are the main concerns in every software project. Effective analysis of software risks will help to effective planning and assignments of work.

- **Time**: The time required to develop the software is underestimated and can lead to unwanted errors
- **Lack of experience and knowledge**: It is impossible to recruit staff with the skills required, since not all the developers have the same level of knowledge.so that can lead to delay in delivering the project.
- **Tools:** The code generated by software code generation tools is inefficient and Software tools cannot work together in an integrated way
- **Programmatic Risks:** Running out of the fund, changing customer product strategy and priority and Government rule changes all that can badly impact on the development of the project

6.2 Strategies

Prepare for the risks by having strategies for avoiding the risks as well as minimising the impact of them if they do occur.

- **Preparing rescue plan**: saving and uploading all the sensitive data and critical files in a safe particular platform such as (Gitlab), Therefor, it will be easy to retrieve in any worse case.
- **Using Previous experience**: can improve the acceleration of the project process and planning for he project efficiently ,avoiding catastrophic errors and helps in delivering the project at the scheduled time.
- **Defective components:** Replace potentially defective components with bought-in components of known reliability.

Personal Reflection:
The fact that the code is still on its first stages of its implementation so not too many features were included, so based on that, the number of potential risks would not be huge, so the risks and strategies which were mentioned in this report are the most common to occur in the process of building the project. So the way I used to prevent the potential risks is by gaining more experience in the way how to structure the project and to well plan it.

Type	Completion time	Date of completion
Vision	30 min	2020.02.01
Project Plan	5 hours	2020.02.01
Iteration (1)	3 hours	2020.02.01
Risk Analysis	2 hours	2020.02.01
Scope	20 min	2020.02.01

Time Log

	Actual time dedicated	Assumption	Overview	Deadline
Iteration 1	2 days	3 days	The first plan for structuring the skeleton of the code and a general overview about how to initialize the critical steps for creating the hangman game.	2020-02-02
Iteration 2	3 days	4 days	Visualizing the hang man game by representing it as a Use Case diagram, Class Diagram and Finally State machine diagram.	2021-03-25
Iteration 3	4 days	4 days	Completed the manual testing and automated Junit testing	2020-05-30
Iteration 4	2 days	3 days	Combining all the 3 iteration together in one iteration, plus debugging the code and do a general check if all the requirements are fulfilled and finally adding the new features	2021-03-28

Handing in

All assignments have a number of files to hand in. The overall advice is to *keepitsimple*. Make it easy for the reciever to understand what the files are by using *descriptive* file names. Use as *few* separate documents as possible. Always provide a *context*, that is *do not* send a number of diagrams in “graphics format”, but always in a document where you provide the purpose and meaning of the diagrams. Remember that the “reciever” is in reality a customer and as such has very little knowledge of the diagrams and documents – always provide context that make anything you hand in understandable to a non-technical person.

To hand in an assignment, make a git release and hand in the link via Moodle to that release.