SOFTWARE DEVELOPMENT PROJECT TEMPLATE

Marcos Gómez de Quero

21/05/2020

Contents

1 Revision History

Date	Version	Description	Author
03/02/2020	1	Assignment 1	Marcos Gómez de Quero
23/02/2020	2	Assignment 2	Marcos Gómez de Quero
08/03/2020	3	Assignment 3	Marcos Gómez de Quero
19/03/2020	4	Final Assignment	Marcos Gómez de Quero
17/04/2020	5	Assignment 3 Resubmission	Marcos Gómez de Quero
20/05/2020	6	Assignment 3 Resubmission	Marcos Gómez de Quero
21/05/2020	7	Final Assignment Resubmission	Marcos Gómez de Quero

2 General Information

Project Summary	
Project Name	Project ID
Hangman	HelloDarknessMyOldFriend
Project Manager	Main Client
Marcos Gómez de Quero	Linnaeus University
Key Stakeholders	
Developer: Marcos Gómez de Quero Project Manager: Marcos Gómez de Quero Costumer: Linnaeus University Tester: Marcos Gómez de Quero	
Executive Summary	
In this project I will try to reach what we are asked to make, the hangman game. At first probably I will just make the basic game, but as the assignments are being made, I will try to improve my game by adding things like a table score, timer, multiplayer…	

3 Vision

The objective of this project is to create a hangman where the words that we have to find out are different capitals of the world. The idea is to make a game that will work, and anybody can play without even knowing anything about the program. I will make a menu that will start the game with a button to start the game and another one to quit. After playing the game for once and winning or losing, you can play again or quit the game. The capital that the player will have to find out is picked randomly from a txt where I have all the capitals and where I can add more capitals if I wish to. At first it will be a basic hangman game, but as time comes for next assingments, I will try to improve the program with new settings by adding new features, for example a timer, a score table, the option to add users, maybe a multiplayer option…

4 Project Plan

My idea is to make the program for the first assignment, so later I will just have to make new features like I said before, some examples, that probably will be more in the future could be, multiplayer option, a timer for inserting words, user list, leaderboard… To make the program more clear, I decided to make some txt, one for the capitals that is selected randomly later from there for the game, and other 8 for the errors where I made the different steps of the errors for the hangman.

4.1 Introduction

This project is a program where you can play a game called Hangman.

4.2 Justification

I have a client that is asking for a program where he can play hangman.

4.3 Stakeholders

Developer: Marcos Gómez de Quero
Project Manager: Marcos Gómez de Quero
Client: Linnaeus University
Tester: Marcos Gómez de Quero

4.4 Resources

I used NetBeans to create the program.

4.5 Hard- and Software Requirements

I used NetBeans to create and program the Hangman. To run the program you can use anything with a Java compiler.

4.6 Overall Project Schedule

The dates where I have to submit the assignments.

4.7 Scope, Constraints and Assumptions

The scope of the project is to create a game that anybody can play. I think every part that I made is inside the project.

5 Iterations

5.1 Iteration 1

For first assignment 1, I decided to make the template (as we don't know yet too much about the project, probably this first documentation will be quiet short and with not much details), the whole program of hangman, with the options to play and to exit. The list of capitals that I made, I decided to make a .txt and add the file in the code, so I can always add more capitals or anything I want. Also the errors (the scheme of the hangman), I decided to make it in .txt aswell, with 8 .txt of errors from 0 to 7 where the hangman is being created with each error.

5.2 Iteration 2

For assignment 2, I decided to let the code as in first assignment, but I added a confirmation of exit menu. After doing this I started with the Use Cases. Then the Use Case Diagram, after it, created the State Machine and to finish with it the Class Diagram.

5.3 Iteration 3

For assignment 3, I started making the Manual Tests using the Use Cases of assignment 2 and after it, I made the Unit Tests and the report of these. To finish this assignment I wrote the Test Plan with the time plan and edited this template. As my testing wasn't working, I just did it again and it is working fine now! So I pushed it and it is already working. I also had to reupload the Test.pdf with some changes on the manual test cases as they weren't done as it was working, and I had to be more specific, so I also uploaded that.

5.4 Iteration 4

For the final project, I will finish the game, maybe upgrading it or adding more things. As I had to make the retake, I improved the code that had some bugs, the testing from iteration 3 and upgraded quiet much the project documentation adding, for example, a reflection. I also merged the 2 pdfs that I had in one.

6 Risk Analysis

As in any project, there will always be risks, but as it is a project for the University and it is decided from time ago, probably things will not change, so there are not many risks that I will have to figure out with. I also think this part is important, as we can manage almost every problem that we can have in the future by predefining strategies for problems that we may have in the future. As I am a newbie in projects, I will probably not have imagination to predefine many problems that I may have to deal with, but I can try to figure out some of them.

6.1 List of risks

Product competition. If we take the other students' projects, this could be a risk.
Requirements change. Maybe the teachers decide to make changes in the final project.
Size underestimate. Maybe the project is quiet big and it can't be run or sent.

6.2 Strategies

To solve the product competition I could manage it making staff that anyone else think in, like new features that are original.
To solve the requirements change there is not much things that I can do, but I could make my code to be flexible or improve things and features in it.
To solve the size underestimate I can compress the project or modify the project and remove functionality.

7 Time log

Date	Expected Time	Real Time	Task
02/02/2020	3 hours	1 hour 30 min	Code of Hangman
02/02/2020	15 min	15 min	General Information
03/02/2020	20 min	22 min	Vision
03/02/2020	30 min	43 min	Project Plan
03/02/2020	30 min	28 min	Iterations
03/02/2020	40 min	51 min	Risk Analysis
15/02/2020	5 min	5 min	Confirmation of Exit
16/02/2020	20 min	23 min	Use Cases
20/02/2020	15 min	13 min	Use Case Diagram
22/02/2020	40 min	48 min	State Machine
22/02/2020	30 min	34 min	Class Diagram
23/02/2020	20 min	24 min	Edit the template

The time I spent making the code, was quiet less than I thought because the program was easier than I thought at first.

The vision has moreless the same time as I expected.

The project plan was quiet harder, because at first I started writing the same as in vision task, so I had to change it later.

For iterations was moreless as I expected.

The risk analysis was quiet different because I had to think in many risks and as I am a newbie, it is difficult to find some for a project for me.

8 Reflection

To be honest, this project has helped me a lot with some aspects and I have learned some helpful things for future projects like looking for information, explaining things correctly, making it readable for the rest, in this case for the client. I have learned also to test, and it seems to be useful whenever you don't know where something is failing and don't find the error. I also learned about diagrams, like Use Case Diagram, State Machine, Class Diagram… and I think it is quiet useful to learn how to make a diagram whenever is needed. I aksi saw the manual test cases and the use cases like a big part of the project, so you, your client, and everyone can understand how the program works with a really good explanation.

From my point of view there were a couple things that were difficult at first, as the testing part, because I never used it, and the diagrams part. But I got much help with it and was able to understand and make it! Also the project template was a good idea for us to document everything that we do and don't lose ourselves in the project, so we could follow like a guide.

Most of the course has been great, the only bad thing that I can say is about the doubts that we have. Even though I think the people that corrected our assignments and came to the question sessions helped a lot to make us finish the project properly.

Task 1: Test Plan

The objective of this is to test the code that I implemented for this iteration and look for anything wrong and update it. By testing it, we are asked to create two unit tests for 2 different methods plus one unit test that fails, so at the final part I will have 5 unit tests, 2 for each method, and 1 that fails. If I see any method failing, I will update it. I intend to test UC1 and UC2 by writing and running dynamic manual test-cases.

Time plan

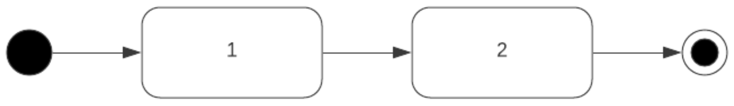
Task	Estimated time	Actual time
Manual TC	2h	2h 30 min
Unit tests	2h	30 min
Running manualtests	10 min	10 min
Code inspection	30 min	10 min
Test report	10 min	10 min

Task 2: Manual Test Cases using the client application

TC1.1 Select an option

Use case: UC1 Start Game.

Scenario: Select an option successfully.



The main scenario of UC1 is tested where a user selects play game option successfully.

Precondition: There is no precondition.

Test steps

- Start the app.
- System shows the menu.

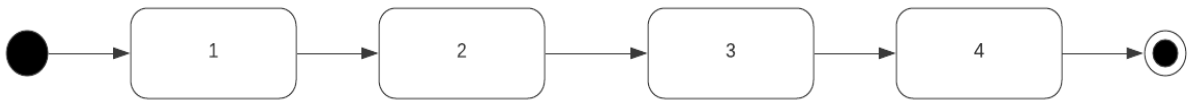
Expected

- System shows a menu and waits for input.

TC1.2 Select Play Game option

Use case: UC1 Start Game.

Scenario: Select 1 to play the game.



Precondition: Run the program.

Test steps

- Start the app.
- System shows the menu.
- Player selects option 1 (Play Game).

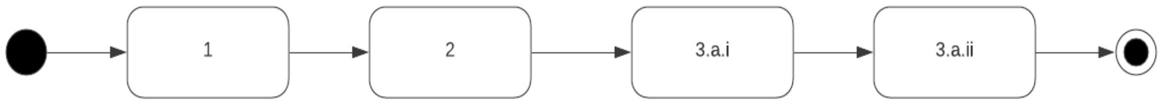
Expected

- System selects a random word from the file capitals.txt.
- Game will start.
- User guess the missing letters.
- System shows underscores that represent the number of letters the capital has.

TC1.3 Select Quit Game option

Use case: UC1 Start Game.

Scenario: Select 0 to quit the game.



The alternative scenario of UC1 is tested where a user selects Quit Game option successfully.

Precondition: Run the program.

Test steps

- Start the app.
- System shows the menu.
- Player selects option 0 (Quit Game).

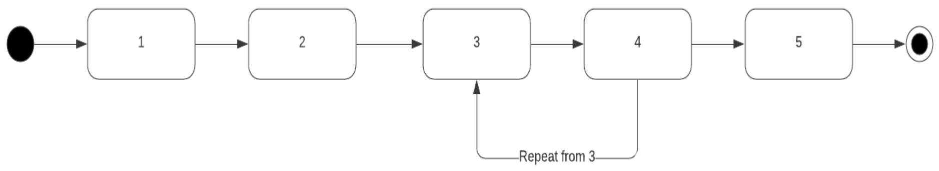
Expected

- System shows a confirmation menu and a message: “Are you sure you want to quit?”
- User chooses 0 in the menu
- Program will end

TC2.1 User wins a game

Use case: UC2 Play Game.

Scenario: Win a game successfully.



The main scenario of UC2 is when the user wins the game.

Precondition: Select Play Game option in the menu.

Test steps

- Run the program.
- Select 1 (Play Game option) to start the game.
- System prints 4 underscores to guess the capital.
- Player introduces letter “r”.
- System changes the first underscore with the letter “r”.
- System asks to enter another letter.
- Player introduces letter “o”.
- System changes the second underscore with the letter “o”.
- System asks to enter another letter.
- Player introduces letter “m”.
- System changes the third underscore with the letter “m”.

- System asks to enter another letter.
- Player introduces letter "e".
- System changes the fourth underscore with the letter "e".
- System shows the message "You win!!!"

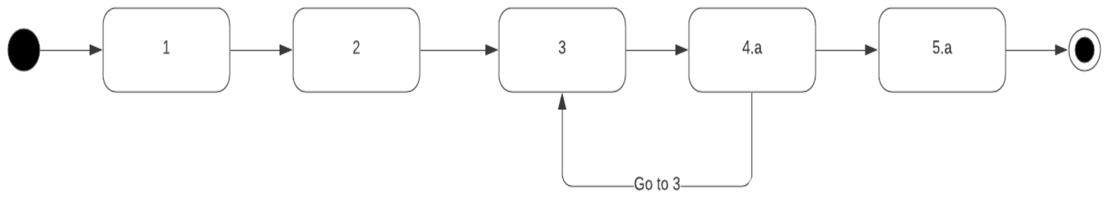
Expected

- System will ask the user 4 questions (Name, LastName, Gender and Feedback).
- System will show the Play option menu.

TC2.2 User loses a game

Use case: UC2 Play Game.

Scenario: Lose a game successfully.



Precondition: Select Play Game option in the menu.

Test steps

- Run the program.
- Select 1 (Play Game option) to start the game.
- System prints 4 underscores to guess the capital.
- Player introduces letter "q".

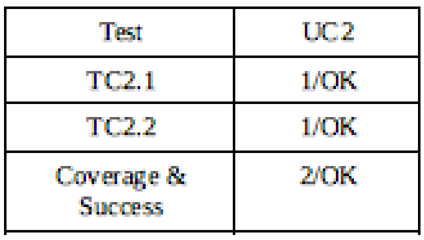
- System prints the base of the hagman (error1).
- System asks to enter another letter.
- Player introduces letter “w”.
- System prints the drawing above but adding the head (error2).
- System asks to enter another letter.
- Player introduces letter “t”.
- System prints the drawing above but adding the body (error3).
- System asks to enter another letter.
- Player introduces letter “a”.
- System prints the drawing above but adding the left arm (error4).
- System asks to enter another letter.
- Player introduces letter “s”.
- System prints the drawing above but adding the right arm (error5).
- System asks to enter another letter.
- Player introduces letter “n”.
- System prints the drawing above but adding the left leg (error6).
- System asks to enter another letter.
- Player introduces letter “x”.
- System prints the drawing above but adding the right leg (error7).
- System shows the message “You lose!!!”

Expected

- System will ask the user 4 questions (Name, LastName, Gender and Feedback).
- System will show the Play option menu.

Test reports

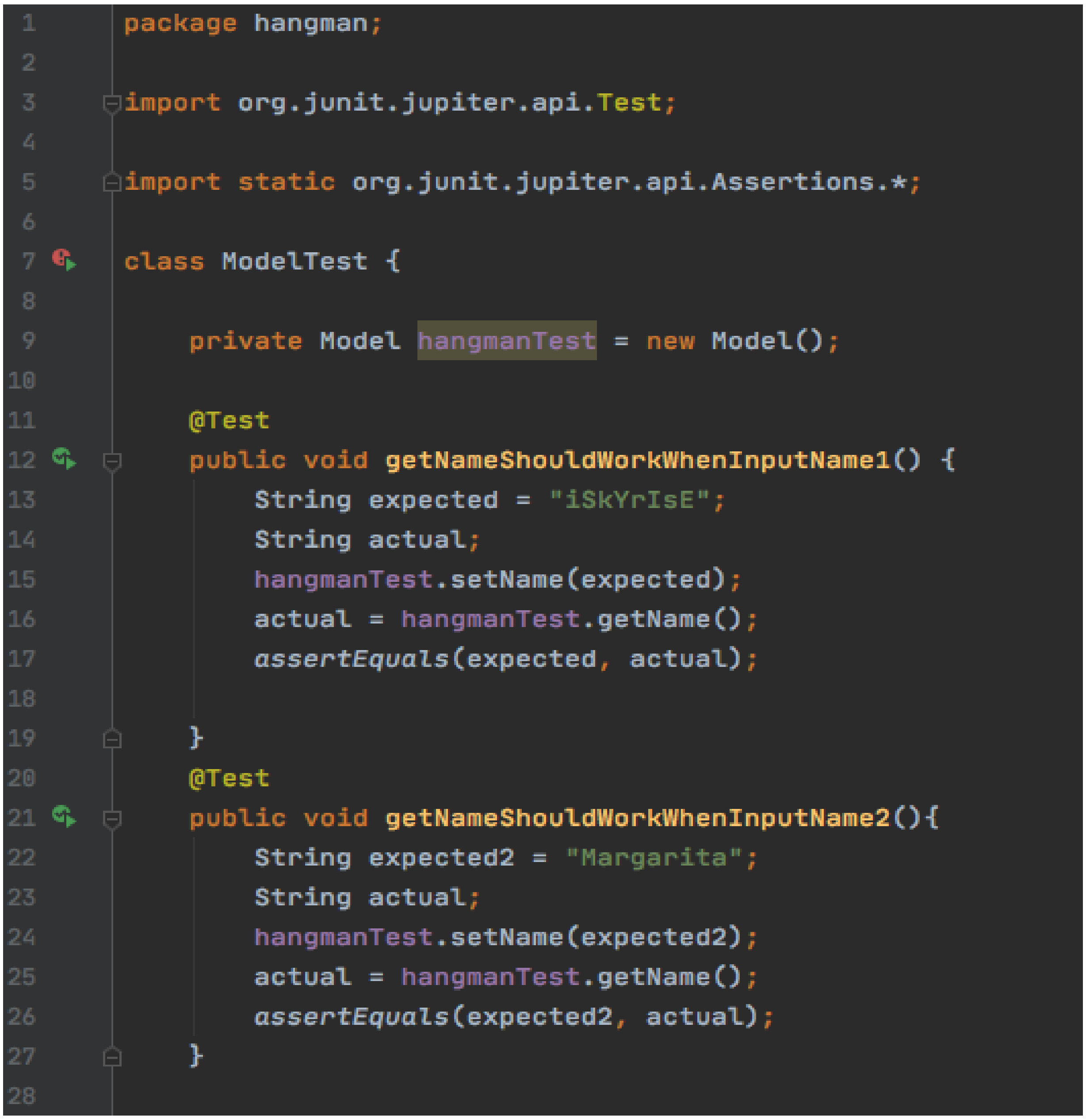
Test	UC1
TC1.1	1/OK
TC1.2	1/OK
TC1.3	1/OK
Coverage & Success	3/OK



Test	UC2
TC2.1	1/OK
TC2.2	1/OK
Coverage & Success	2/OK

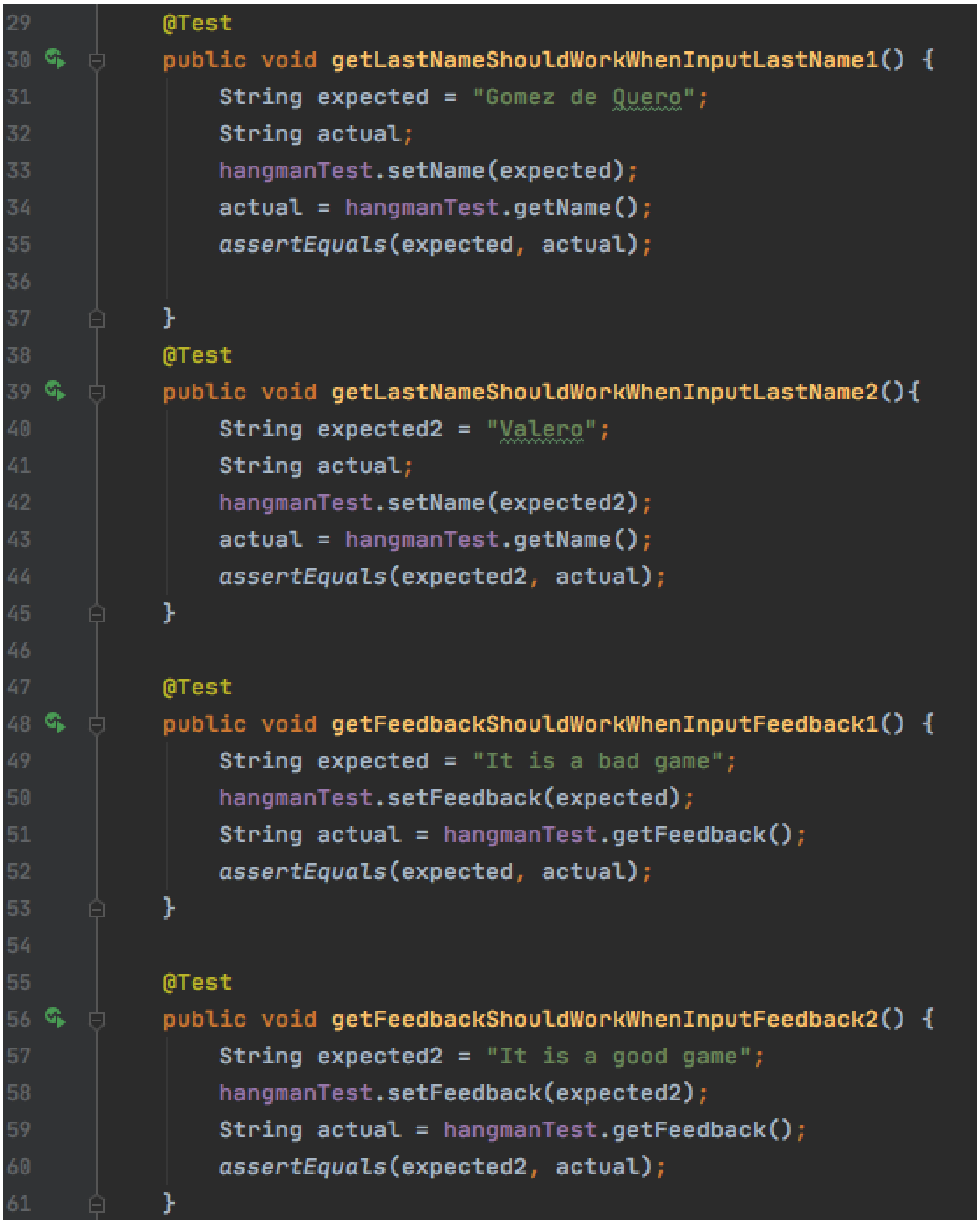
Link to my testing class called ModelTest:
https://gitlab.lnu.se/1dv600/student/mg223mg/assignment-3/-/blob/master/Program/Code/Test/ModelTest.java

Here are the screenshots of my testing class, where 7 work and 1 fails, the one failing will work if you uncomment the lines 248, 249 and 250 in Model class.This screenshot shows the 2 tests for Name:



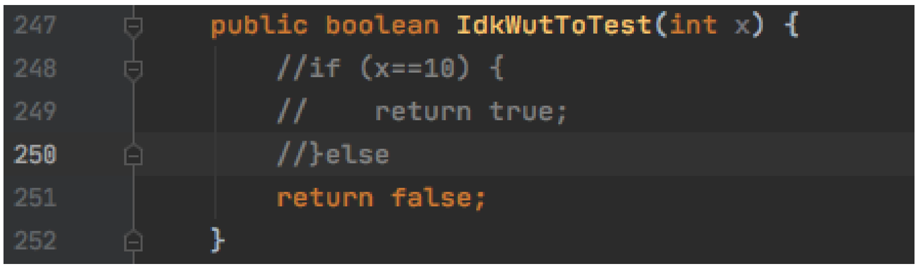
```java
package hangman;

import org.junit.jupiter.api.Test;

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

class ModelTest {

    private Model hangmanTest = new Model();

    @Test
    public void getNameShouldWorkWhenInputName1() {
        String expected = "iSkYrIsE";
        String actual;
        hangmanTest.setName(expected);
        actual = hangmanTest.getName();
        assertEquals(expected, actual);

    }
    @Test
    public void getNameShouldWorkWhenInputName2(){
        String expected2 = "Margarita";
        String actual;
        hangmanTest.setName(expected2);
        actual = hangmanTest.getName();
        assertEquals(expected2, actual);
    }

```

This screenshot shows the testing for the lastName and for Feedback which both will pass aswell:



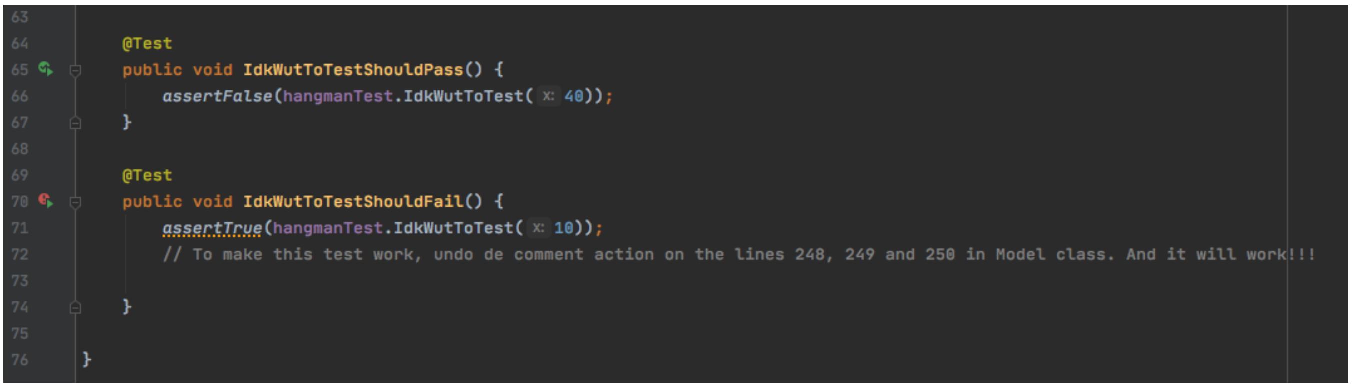
```
    @Test
    public void getLastNameShouldWorkWhenInputLastName1() {
        String expected = "Gomez de Quero";
        String actual;
        hangmanTest.setName(expected);
        actual = hangmanTest.getName();
        assertEquals(expected, actual);

    }
    @Test
    public void getLastNameShouldWorkWhenInputLastName2(){
        String expected2 = "Valero";
        String actual;
        hangmanTest.setName(expected2);
        actual = hangmanTest.getName();
        assertEquals(expected2, actual);
    }

    @Test
    public void getFeedbackShouldWorkWhenInputFeedback1() {
        String expected = "It is a bad game";
        hangmanTest.setFeedback(expected);
        String actual = hangmanTest.getFeedback();
        assertEquals(expected, actual);
    }

    @Test
    public void getFeedbackShouldWorkWhenInputFeedback2() {
        String expected2 = "It is a good game";
        hangmanTest.setFeedback(expected2);
        String actual = hangmanTest.getFeedback();
        assertEquals(expected2, actual);
    }
```

This screenshot shows the testing of a method that I created at the bottom of Model class, I will also add a screenshot of the method:



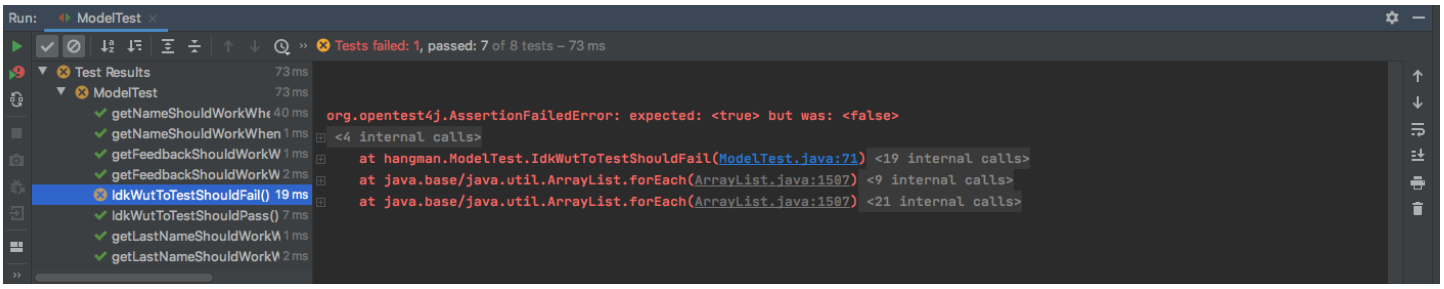
```
247    public boolean IdkWutToTest(int x) {
248        //if (x==10) {
249        //    return true;
250        //}else
251        return false;
252    }
```

IdkWutToTestShouldPass will always pass even with commented code. Not the same happening with IdkWutToTestShouldFail, it will only work if the code in the method is not commented:



```
    @Test
    public void IdkWutToTestShouldPass() {
        assertFalse(hangmanTest.IdkWutToTest( x: 40));
    }

    @Test
    public void IdkWutToTestShouldFail() {
        assertTrue(hangmanTest.IdkWutToTest( x: 10));
        // To make this test work, undo de comment action on the lines 248, 249 and 250 in Model class. And it will work!!!

    }

}
```

Here I add the screenshot of the tests that pass and the one that fails because the commented code:



```
Run: ModelTest
Tests failed: 1, passed: 7 of 8 tests – 73 ms

Test Results                      73 ms
  ModelTest                       73 ms
    getNameShouldWorkWhe          40 ms
    getNameShouldWorkWhen          1 ms
    getFeedbackShouldWorkW         1 ms
    getFeedbackShouldWorkW         2 ms
    IdkWutToTestShouldFail()      19 ms
    IdkWutToTestShouldPass()       7 ms
    getLastNameShouldWorkW         1 ms
    getLastNameShouldWorkW         2 ms

org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
<4 internal calls>
    at hangman.ModelTest.IdkWutToTestShouldFail(ModelTest.java:71) <19 internal calls>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1507) <9 internal calls>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1507) <21 internal calls>
```

Task 1. Requirements

1. 1. Use Cases:

UC 1 Start Game

Precondition: there is no precondition.

Postcondition: the game menu is shown.

Main scenario

1- Gamer: Runs the hangman's program starting the game.

2- System: Shows the menu with the option with two options: Play the game and Quit the game.

3- Gamer: Selects the Play game option.

4- System: Selects a random capital of capitals.txt and starts the game.

Alternative scenarios

3.a.i - Gamer: Selects the Quit game option.

3.a.ii - System: Shows the confirmation exit menu.

3.b.i - Gamer: Invalid menu choice.

3.b.ii - System: The system presents an error message: "ERROR! The option does not exist."

UC 2 Play Game

Precondition: the game is running main menu screen.

Postcondition: the game is over and the main menu screen reopens.

Main scenario

1- Gamer: Selects the Play game option.

2- System: Prints underscores.

3- Gamer: Attempts to guess a letter.

4- System: Analyse if the letter is contained in the capital. The letter is correct, then it prints the letter in the position that corresponds.

Repeat from 3.

5- System: Prints the whole word, and a message "You win!!!".

Alternative scenarios

4.a- System: Analyse if the letter is contained in the capital. The letter is wrong, then it prints a part of the hangman.

Go to 3.

5.a- System: Prints the whole hangman, and a message "You lose!!!".

UC 3 Quit Game

Precondition: The game is running.

Postcondition: The game is terminated.

Main scenario

1- Gamer: Select the option Quit the game

2- System: Sends a message "Are you sure you want to quit?" and the system opens a new menu that asks for confirmation.

3- Gamer: Selects Yeah.

4- System: Sends a message "See you soon!" and the game terminates.

Alternative scenarios

3.1. Gamer: Selects Nope.

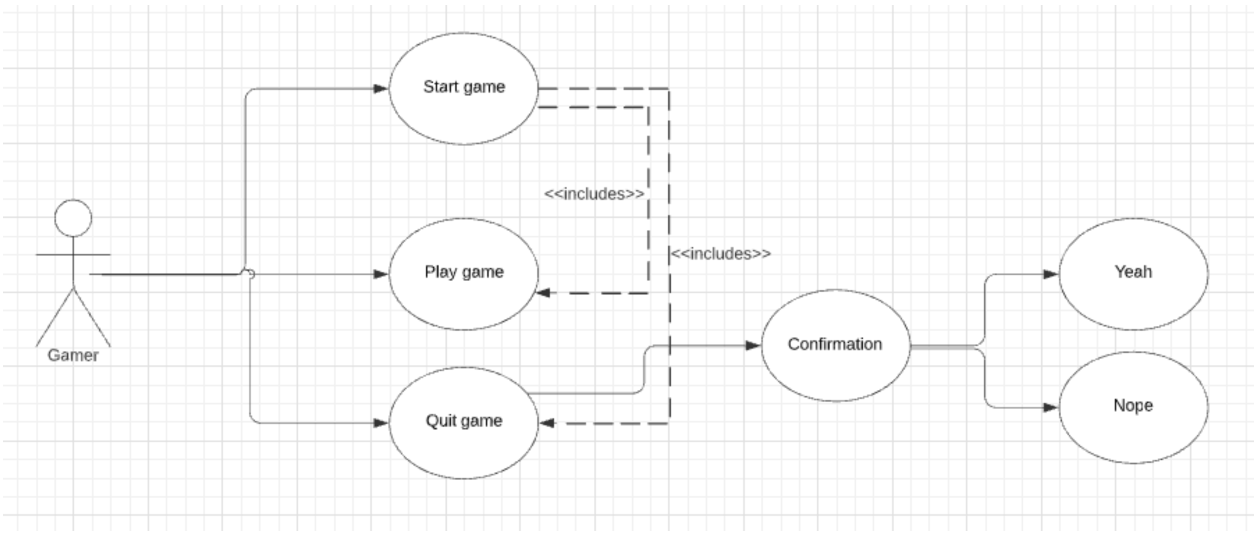
1. System: Sends a message "So lets play!!!" and shows the main menu again.

3.2. Gamer: Selects other option that does not exist.

1. System: Sends a message of error "ERROR! The option does not exist" and returns to the menu of confirmation.

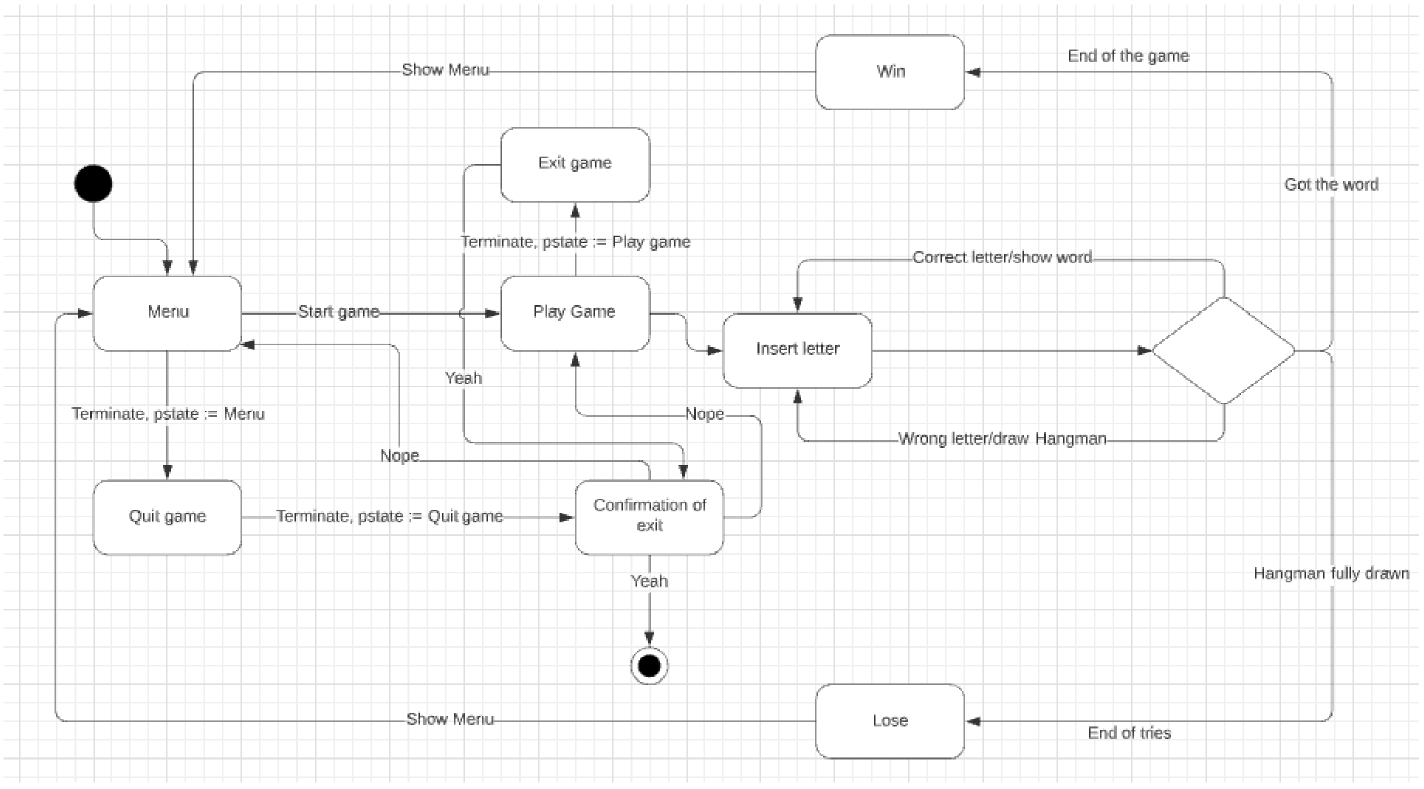
1.2. Use Case Diagram:

The Use Case Diagram is a diagram that shows the actions between the system and its environment. This are used to describe the interactions between users and systems in the system being designed (like sequence diagrams). Use cases describe interactions between a system and external actors. Also, it is a UML diagram type that is used to identify use-cases and graphically depict the users involved. It must be supplemented with additional information to completely describe use-cases. It is good for the user to watch this diagram because he can understand how the program works with a not difficult diagram to understand.



Task 2. State Machine:

Which shows how objects change their state in response to events. These are represented in the UML using state diagrams. This diagram helps the user to get a full knowledge of where can the program go selecting an option or another and what can it do.

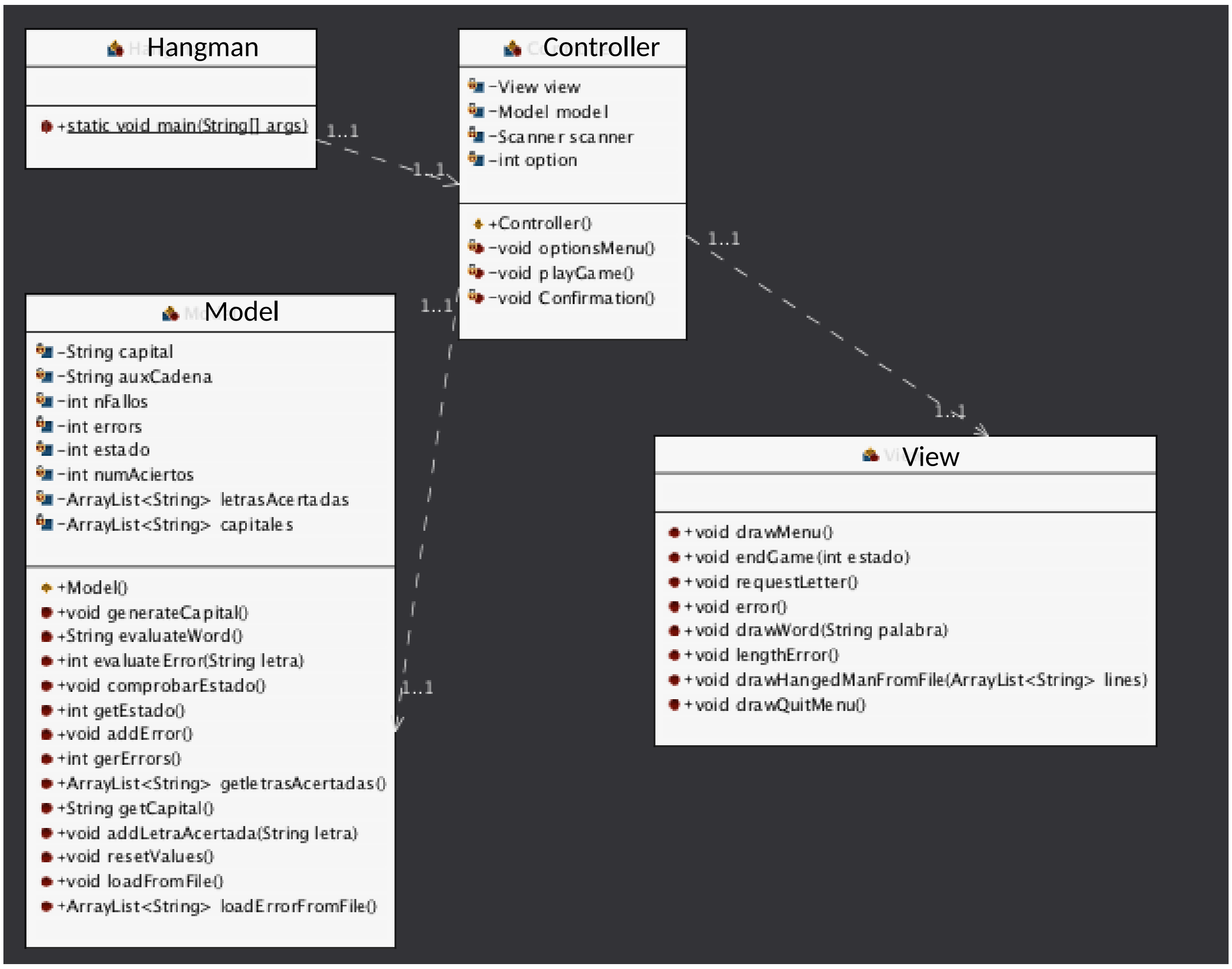


Task 3. Code:

Code is already updated as .java in Program/Code/src.

Task 4. Class diagram:

Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes. The purpose is to model the static view of an application. This is helpful to the user so he can know how is the program designed and what can it do.


Hangman
+static void main(String[] args)
Controller
-View view
-Model model
-Scanner scanner
-int option
+Controller()
-void optionsMenu()
-void playGame()
-void Confirmation()
Model
-String capital
-String auxCadena
-int nFallos
-int errors
-int estado
-int numAciertos
-ArrayList<String> letrasAcertadas
-ArrayList<String> capitales
+Model()
+void generateCapital()
+String evaluateWord()
+int evaluateError(String letra)
+void comprobarEstado()
+int getEstado()
+void addError()
+int gerErrors()
+ArrayList<String> getletrasAcertadas()
+String getCapital()
+void addLetraAcertada(String letra)
+void resetValues()
+void loadFromFile()
+ArrayList<String> loadErrorFromFile()
View
+void drawMenu()
+void endGame(int estado)
+void requestLetter()
+void error()
+void drawWord(String palabra)
+void lengthError()
+void drawHangedManFromFile(ArrayList<String> lines)
+void drawQuitMenu()
1..1
1..1
1..1
1..1
1..1
1..1