HANGMAN PROJECT

Mohammadali Rashidfarokhi
Linnaeus University

2020-01-31

Contents

1. Revision History

Date	Version	Description	Author
2020.01.31	0.1	For creating the first iteration, different tasks such as vision, project plan and skeleton of hangman project were covered.	Mohammadali Rashidfarokhi
2020.02.24	0.2	Creation of use case, state machine, class diagram, runnable version of game code.	Mohammadali Rashidfarokhi
2020.03.09	0.3	Test plan, reflections, manual testing, automated testing	Mohammadali Rashidfarokhi
2020.03.20	0.4	Combining all iterations, planning and reflection.	Mohammadali Rashidfarokhi

2. General Information

Project Summary	
Project Name	Project ID
Hangman project	0003
Project Manager	Main Client
Mohammad Ali Rashidfarokhi	Linnaeus University
Key Stakeholders	
• User • Developer • End-user	
Executive Summary	
This project aims to create a hangman game base on java programming language. The user must choose different letters to complete the missing words in several attempts. There are different modes for playing the game. If the user wins, a list of information is necessary for user to fill out.	

3. Vision

Hangman is a very popular game. It can be created by different programing languages (e.g. JavaScript, C++, phyton etc.). Since java is one of the programing languages that enables the developers to create such games, this project will use java programming language to fulfill the purposes of the project.

The general idea behind this game is that after the developer has carried out all essential actions to create the game, the user is supposed to guess a specific word by entering different letters. The player is presented with different words. Consequently, the user must choose correct letters to win the game. The game will be based on different modes. Therefore, playing the game will become challenging and fun at the same time.

There will be an already loaded list of words in English that will be used for the game so that the game will become fun to play for users because each time playing the game, the user will face with a new challenge (new word to guess).

Personal Reflection:

The creation of this game is a way to improve the skills of programming and test the planning skills at the same time. A useful website that I used a lot and helped me along the process was the java API, which was unclear for me at the beginning, but thanks to the hangman project not completely but it became clear for me. However, there were some difficulties along the way. Due to the shortage of time, it was difficult for me to think about the whole project since I was unfamiliar with the process, but I tried to cover all the necessary features that their existing is essential for creating this game.

4. Project Plan

The developer aims to create a game called hangman to challenge the player to guess some letters from specific words. Once the player has started the game, the screen will display a message and ask the user to state whether ready to begin the game or not.

Since the game is not an ordinary game, the game has different modes of playing. That is, if the user decides to begin the game (the user will be asked if he/she is ready to start the game), the game will provide the user with 2 modes of playing (hard and medium). As it is obvious winning the game in hard mode is challenging (number of attempts to choose the missing letters will be reduced to 6). For helping the player, an option that helps the player with the game will be in the menu of game.

If the user wins the game, there should be a form that asks the user to fill out some data. Then, a farewell message will be displayed. On the other hand, there will be an option in the game menu that will enable the user to restart the game. Also, two new options in the second menu of the game that are hint and exit will be added.

4.1 Introduction

A game called hangman will be created to enables user to play the game by choosing the missing letters from pre-defined words in the code. The player should find the missing letters based on the chosen mode for playing.

4.2 Justification

Apart from the fun experience of creating the hangman game, one of the tasks of software technology course (1DV600) is to make a hangman game in detail and learn new techniques about software development and improve the planning skills.

4.3 Stakeholders

User: Logically the player is going experience the game and all provided features in the game by a programmer, the user can either win or lose the game.

Developer: The responsibility of generating, implementing all essential features and deciding how to test the provided code is developer responsibility.

End-user: End-user is the individual who uses the product after it has been fully developed based on the user experience.

4.4 Resources

Since the game will be written in java language programing, one of the most useful platforms to use will be Java API.
https://docs.oracle.com/javase/7/docs/api/

4.5 Hard- and Software Requirements

One of the useful programing environments is IDE. Therefore, the program will be created in Intellij IDEA and using information provided in java Api.

4.6 Overall Project Schedule

First iteration: 03/02/2020

Second iteration: 24/02/2020

Third iteration: 09/03/2020

Final iteration: 20/03/2020

4.7 Scope, Constraints and Assumptions

The main goal for this project is to create a simple game where the player must fill out the blanks by choosing the missing letters. Consequently, the player will be faced with two messages at the beginning of the program that are “welcome to the hangman game” and “Are you ready to begin?”, that happens inside the game. In order to spice up the game, different modes for playing will be available for user to choose from.

Then hopefully, the game will continue its flow and the player can enjoy the game.

Personal Reflection:

As I have mention before, this project can become challenging for someone who has not previous knowledge about this kind of projects and to create a flawless project which requires knowledge in both programming and planning. If I had previous knowledge in programming, it would be better and easier to plan a better project (more details).

5. Iterations

The course 1DV600 required to complete the project in 3 different stages. That means, the developer is required to update and add extra features to the hang man game at each stage. The final stage, the developers is required to combine all stages together and at that point the game will be ready for the user to play and enjoy the hangman game.

5.1 Iteration 1

This stage is the start of the project. The developer is required to plan perfectly and decide how the game will be created.

The essential classes that are required for the hangman project will be implemented. Obviously one of the important classes that is required for the project is the scanner class which allows user to enter the inputs for playing the game. Therefore, after creating the scanner class, the developer will write a code to print a welcome message to the user and ask the user to start the game with approval (yes/no). In order to guess to missing letters, the developer is required to provide a list of pre-defined words inside the code. This goal can be accomplished by using an array that is filled with list of words in English. For choosing random words, a

random class will be created.

More features will be added in the next iteration.

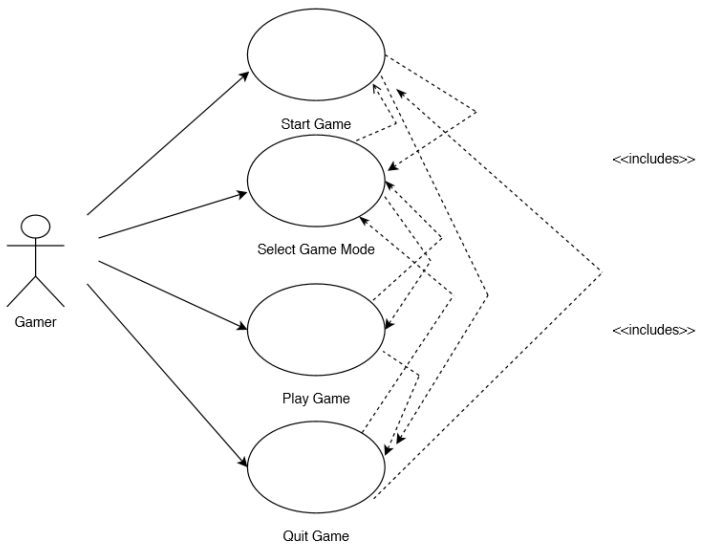
Category	Actual time dedicated	Assumption for finishing	Date of start	Date of finish	Deadline
Project plan	4 hours	3 hours	2020-01-31	2020-01-31	2020-02-03
Risk analysis	55 minutes	2 hours	2020-01-31	2020-01-31	2020-02-03
Iteration 1	1 hour	2 hours	2020-01-01	2020-01-01	2020-02-03
Vision	2 hours	3 hours	2020-01-31	2020-01-31	2020-02-03
Code implementation	45 minutes	1 hour	2020-01-02	2020-01-02	2020-02-03

5.2 Iteration 2

In the second iteration, a significant feature has been added to the game. As usual, the player will be welcomed by a welcome message. Then the user should state if he/she is ready to begin the game. If the player choice is yes, the second message will be displayed which will ask the user to choose a level between medium and hard. The difference between moderate and hard is that in the moderate the player has 8 attempts to choose the missing letters. However, only 6 attempts will be given to the user in the hard mode. Then, the user will begin the game. After finishing the game, the player will be directed to a stage where the result of the game will be displayed. Finally, the user will be directed to a stage where the game will give the user two options two choose from. Consequently, two options namely restart, and exit will emerge. If the user decides to play again, by choosing the restart option, the player will be directed to the same stage (if the user is ready to continue or not)

to repeat the same procedure. However, after winning the game, the final message will show up where it will ask the user to fill out his/her personal information along with a survey that asks for user feedback. After going through all these stages, the program will terminate.

Use case diagram:



Dress use cases:

UC 1 Start Game
Precondition: none
Postcondition: the game menu is shown
Stack holder: player (who is going to experience the game)
Main scenario
It will start when the player wants to run the game.
The system will show the player a menu with titles Continue, Exit
The player chooses to continue.
The system takes the player to the next menu (see use case 2).
The player will play the game.
Extensions:
The player will see the result.
Go to user case3.

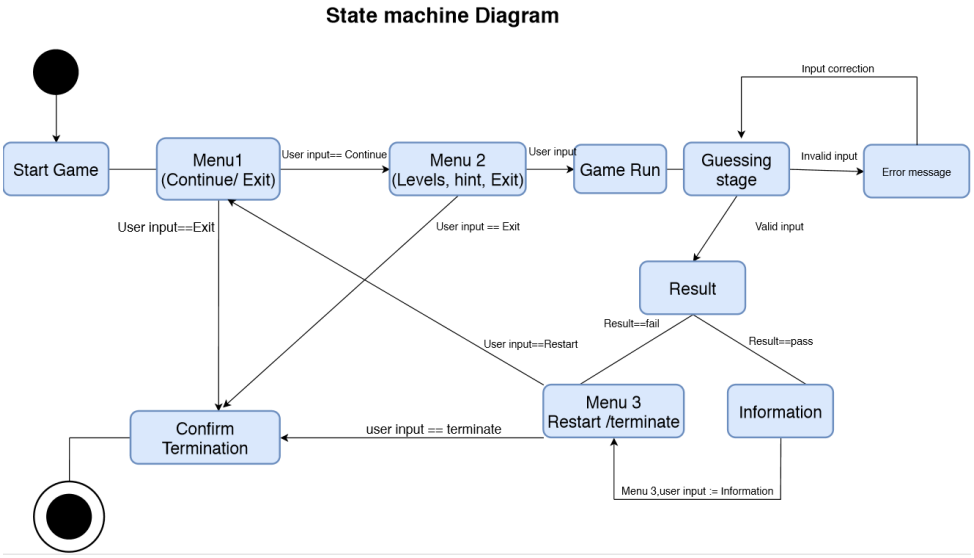
Use case 2 Select game mode
Scope: Hangman game.
Stack holder: player (who is going to experience the game).
Precondition: Start game.

The system will provide the player with different modes for playing and two more options. (Hint and exit).
Postconditions: Game gives users different attempts based on the level.
Main scenario:
Player choose one of the levels (hard or medium).
Go to use case 3.
Extension:
The player runs out of attempts.
Game ends. show “Out of attempts”.
Go to use case 3.

Use case3 play game:
scope: Hangman game.
Stack holder: player (who is going to experience the game).
Precondition: select game mode.
Postconditions:
Player wins or not the game.
Main scenario:
The player tries to guess the missing word.
The player wins the game, system show “yon won”.
The system shows a screen where the user enters data and fill out a survey form.
System show a new menu.
Go to use case 4

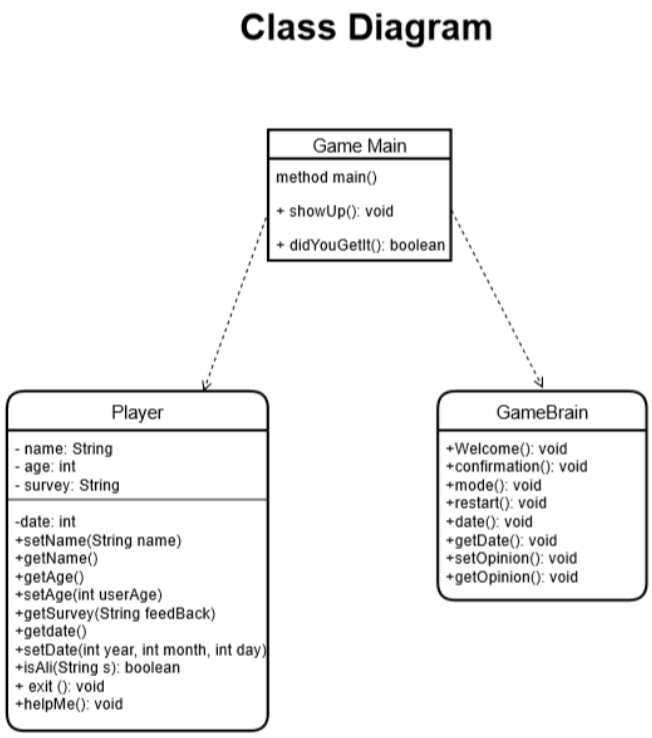
Use case 4 Quit Game
Scope: Hangman game
Stack holder: player (who is going to experience the game)
Preconditions: play game
The system will provide the player with two more options. (Hint and exit).
Postcondition: the program terminates.
Main scenario:
The player will be shown a menu to either leave or start over.
Player choose to leave
Program ends.
Extension:
Player chose restart
Go to the user case1

State machine diagram:



This is a simple diagram which explain the procedure of the game. The user chooses 2 different modes for playing. Also, 3 menus are included in the game which the first and last one has similar functionality. On the other hand, Menu 2 has more functionality.

Class diagram



Category	Actual time dedicated	Assumption For finishing	Date of start	Date of finish	deadline
Class diagram	5 hours	3 hours	2020-02-24	2020-02-24	2020-02-24
Iteration2	1 hour	2 hours	2020-02-23	2020-02-23	2020-02-24
State machine diagram	5 hours	2 hours	2020-02-23	2020-02-23	2020-02-24
Use case diagram	2 hours	4 hours	2020-02-23	2020-02-23	2020-02-24
Implementation	7 hours	3 hours	2020-02-24	2020-02-24	2020-02-24

5.3 Iteration 3

Creation of test cases that can help the developer to evaluate the game.

In this iteration the developer will create test cases and create a test that fails on purpose.

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 chosen word based on different levels.

The developer intended to test the methods that are suitable enough to be executed and some test to check its validity have been provided that run dynamic manual test-cases. At the beginning the TC1 will be evaluated. I also write automated unit tests for player-methods in the class player test. However, as some methods cannot be used to write tests (e.g. the methods that will ask the player to provide some information and the ones that are not type void) in j unit the methods in the class player will be used to write test cases.

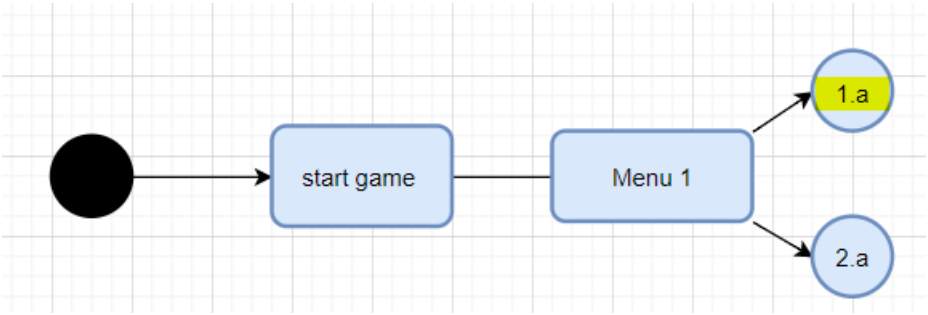
Time plan

Task	Estimated	Actual
Manual Tc	5 h	3 h
Unit tests	2 h	1:30 m
Running manual tests	1h	15 m
Code inspection	2 h	1 h
Test report	1 h	45 m

Manual Test-cases

Tc1.1 user choose to play the game

Scenario: the player chooses to continue (1.a) in Menu1.



The main scenario of this UC1.1 is tested where the user is faced with a menu (Menu 1) with two options and choose 1.a to continue.

Precondition: The player should not enter anything but 1.a

Test steps

• The player will run the game in terminal.

• User press 1.a in the keyboard.

• User press enter in the keyboard.

Expected

• Menu to choose the level appear.

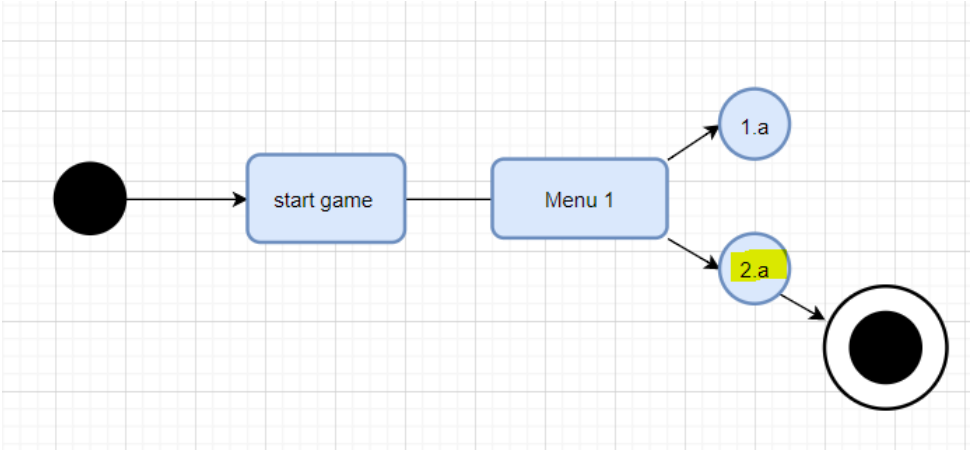
• The terminal shows the message “choose the level”.

• Press 1 for medium.

• Press 2 for hard

Tc1.2 User choose to leave the game.

Scenario: User will choose option 2.a (Exit game) in the menu1.



The main scenario of this UC1.1 is tested where the user is faced with a menu (Menu1) with two options and choose 2.a to Exit the game.

Precondition: The player should not enter anything but 2.a

Test steps

- The player will run the game in terminal.
- User press 2.a in the keyboard.
- User press enter in the keyboard.

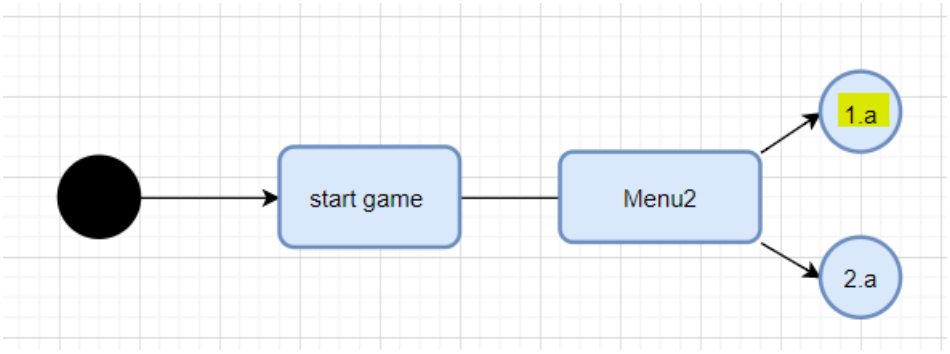
Expected

• The system will show a goodbye message.

• The program ends.

TC2.1 user choose mode moderate

Scenario: The player choose option 1.a (moderate mode) in the menu2.



The main scenario of this UC2.1 is tested where the user is faced with a menu (Menu2) with two options and choose 1.a to set the game mode as moderate.

Precondition: The player should not enter anything but 1.a

Steps

• The player will run the program.

• User enters in keyboard to continue the game.

• User will be directed to choose level menu (menu 2).

• User will choose option 1.a in keyboard for moderate mode.

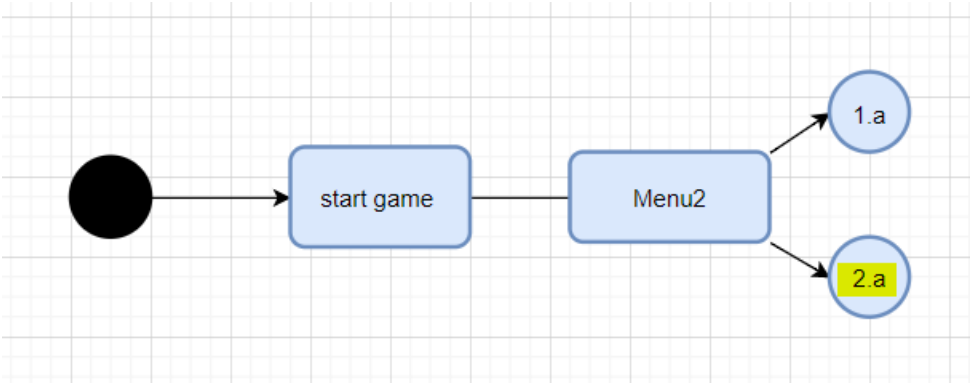
• User press enter in keyboard and set the game mode as moderate.

Expected

• The player will be given 8 attempts to find the missing letters.

TC2.2 user choose mode Hard

Scenario: The player choose option 2.a (Hard mode) in Menu2.



The main scenario of this UC2.2 is tested where the user is faced with a menu (Menu2) with two options and choose 2.a to set the game mode as Hard.

Precondition: The player should not enter anything but number 2.

Steps

• The player will run the program.

• User enters 1.a in keyboard to continue the game.

• User will be directed to choose level menu(menu 2).

• User will choose option 2.a in keyboard for hard mode.

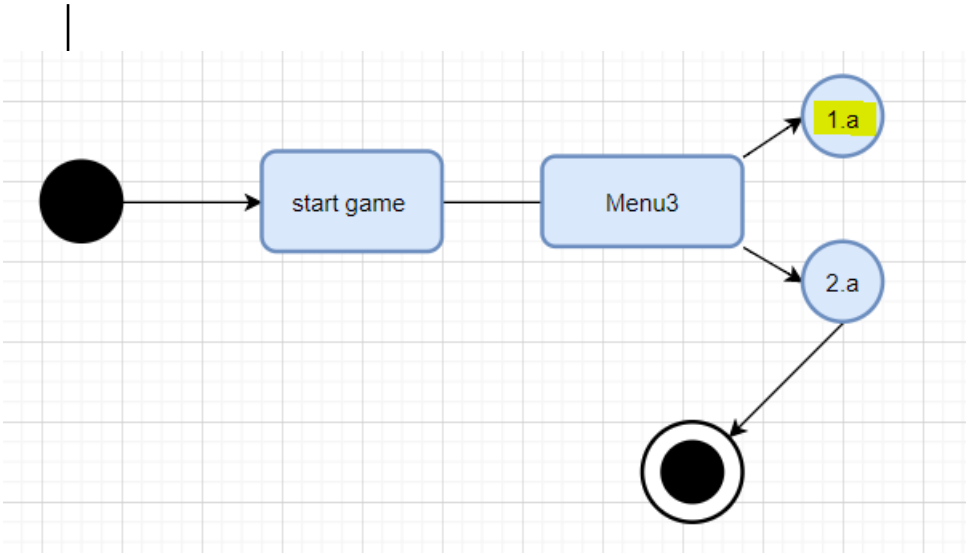
• User press enter in keyboard and set the game mode as hard.

Expected

• The player will be given 6 attempts to find the missing letters.

TC3.1 user choose restart

Scenario: The player choose option 1.a (Restart game) in the Menu3.



The main scenario of this UC3.1 is tested where the user is faced with a menu3 with two options and choose 1.a to restart the game.

Precondition: The player should not enter anything but 1.a

Steps

- The player will run the program.
- User enters 1.a in keyboard to continue the game.
- User will be directed to choose level menu (menu2).
- User will choose options 1.a or 2.a in keyboard for hard and moderate mode.
- User press enter in keyboard and set the game mode as hard or moderate.

• User will guess the missing letters.

• User will be directed to menu3.

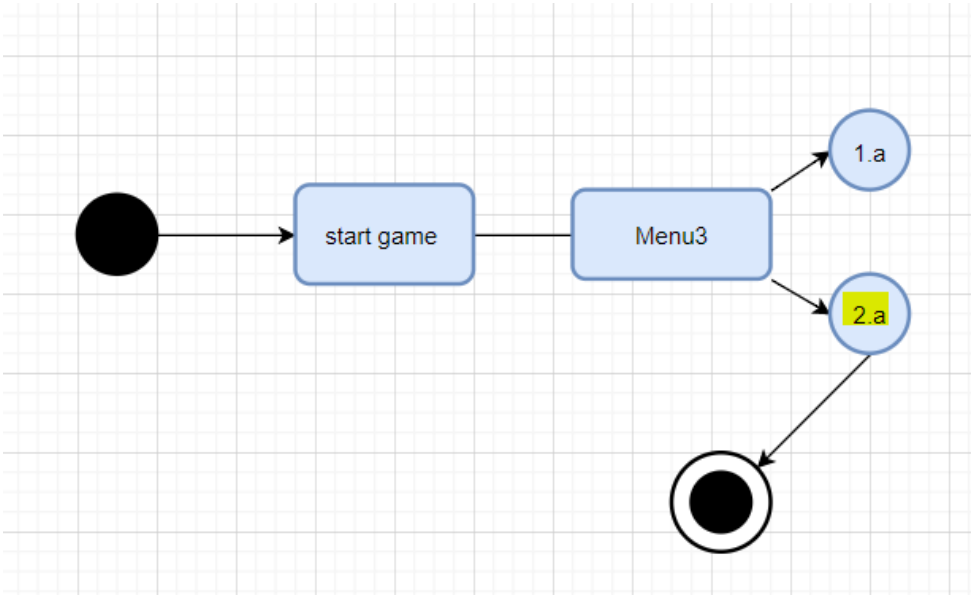
• User will enter 1.a in keyboard to restart the game.

Expected

• The System will direct the user to the game menu stage.

TC3.2 user Exit the game

Scenario: The player choose option 2.a (Exit game) in the game menu3.



The main scenario of this UC3.2 is tested where the user is faced with a menu (menu3) with two options and choose 2.a to exit the game.

Precondition: The player should not enter anything but 2.a

Steps

- The player will run the program.
- User enters 1.a in keyboard to continue the game.
- User will be directed to choose level menu (menu2).
- User will choose options 1.a or 2.a in keyboard for hard and moderate mode.
- User press enter in keyboard and set the game mode as hard or moderate.
- User will guess the missing letters.
- User will be directed to the result menu (menu3).
- User will enter 2.a in keyboard to exit the game.

Expected

- The system will ask the user to enter some information.
- The system will provide user with few messages.

• The program ends.

Test reports

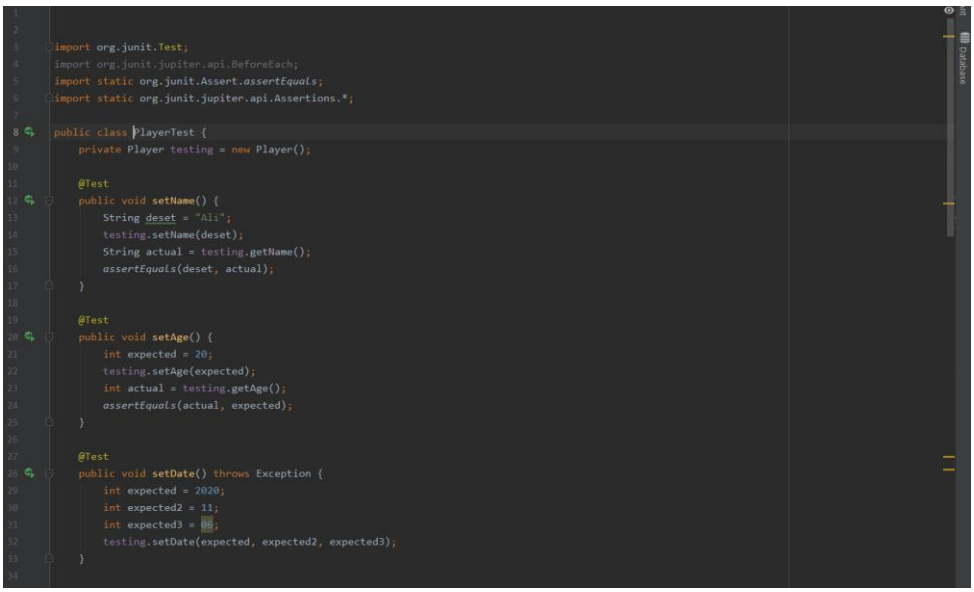
Test	UC1
TC1.1	1/OK
TC1.1	1/OK
Coverage & Success	2/OK

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

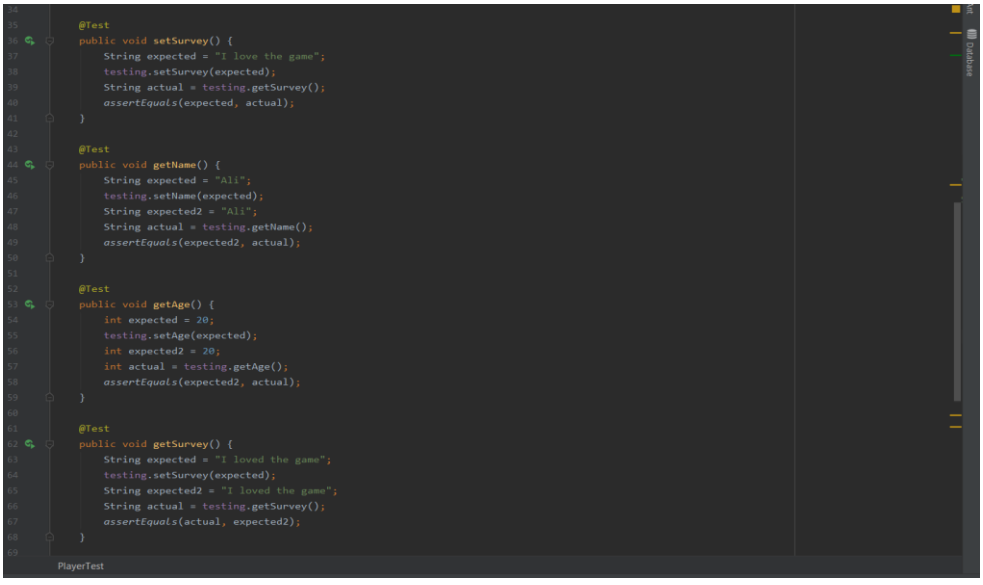
Test	UC3
TC3.1	1/OK
TC3.2	1/OK
Coverage & Success	2/OK

Test plan and unit test cases

Link to my code: https://gitlab.lnu.se/1dv600/student/mr223jp/assignment-3

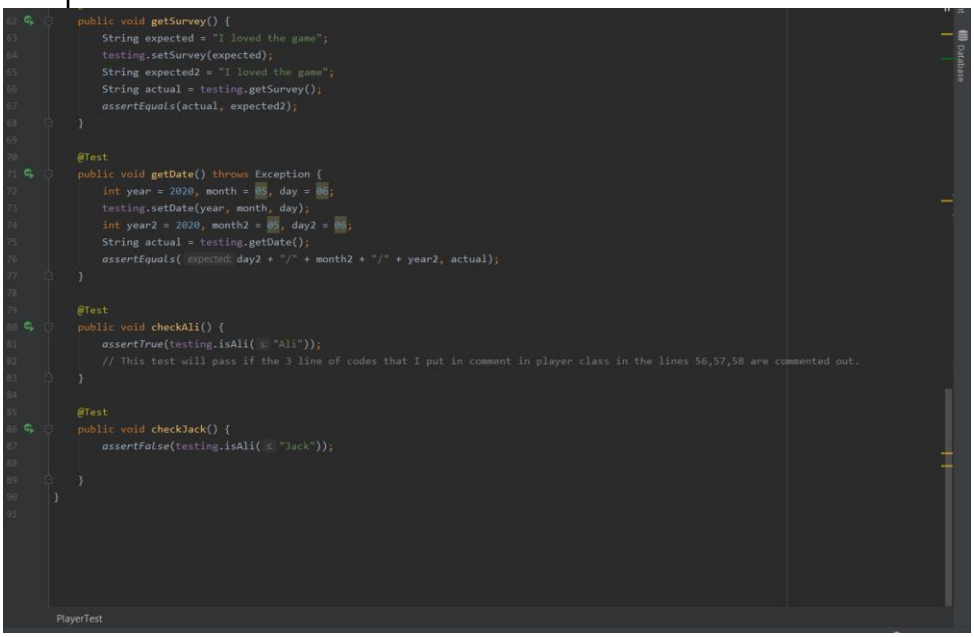


```java
import org.junit.Test;
import org.junit.jupiter.api.BeforeEach;
import static org.junit.Assert.assertEquals;
import static org.junit.jupiter.api.Assertions.*;

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

    @Test
    public void setName() {
        String deset = "Ali";
        testing.setName(deset);
        String actual = testing.getName();
        assertEquals(deset, actual);
    }

    @Test
    public void setAge() {
        int expected = 20;
        testing.setAge(expected);
        int actual = testing.getAge();
        assertEquals(actual, expected);
    }

    @Test
    public void setDate() throws Exception {
        int expected = 2020;
        int expected2 = 11;
        int expected3 = 06;
        testing.setDate(expected, expected2, expected3);
    }
```



```java
    @Test
    public void setSurvey() {
        String expected = "I love the game";
        testing.setSurvey(expected);
        String actual = testing.getSurvey();
        assertEquals(expected, actual);
    }

    @Test
    public void getName() {
        String expected = "Ali";
        testing.setName(expected);
        String expected2 = "Ali";
        String actual = testing.getName();
        assertEquals(expected2, actual);
    }

    @Test
    public void getAge() {
        int expected = 20;
        testing.setAge(expected);
        int expected2 = 20;
        int actual = testing.getAge();
        assertEquals(expected2, actual);
    }

    @Test
    public void getSurvey() {
        String expected = "I loved the game";
        testing.setSurvey(expected);
        String expected2 = "I loved the game";
        String actual = testing.getSurvey();
        assertEquals(actual, expected2);
    }
}
```

PlayerTest



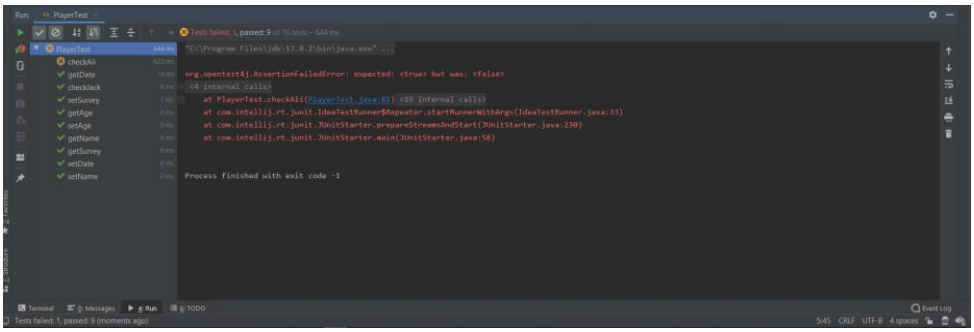
Automated unit test code screenshots

The below pictures illustrate two test codes like each other but one of them will pass and the other will fail when executing at the same time.

The test method called checkAli(), this Test can pass if the codes in lines 56,57,58 in class player are commented out.

```
    }

    @Test
    public void checkAli() {
        assertTrue(testing.isAli( s: "Ali"));
        // This test will pass if the 3 line of codes that I put in comment in player class in the lines 56,57,58 are commented out.
    }

    @Test
    public void checkJack() {
        assertFalse(testing.isAli( s: "Jack"));

    }
}

```

As I have mentioned before the Test checkAli will fail and the Test checkJack will pass.



Comments

All the tests passed, but more test for data user case can be written to test if it works or not. However, two methods called checkAli() and checkJack() are the prominent examples that will demonstrate one test will fail on purpose and the other will pass (checkJack).

Reflection

In this iteration, 3 use cases have been used to write a manual test case, the results were promising since the tests passed. However, more use cases can be written to make sure if they can pass or fail. An example would be writing a use case along with 2 test cases for the data stage in my code.

Regarding the automated test case in my program, there are 10 tests in total which all will pass except one of them (fail on purpose).

A noticeable point about the test that fails (checkAli) is that there is another test called checkJack() which is similar to the checkAli() test, but one will fail and the other will pass.

I have spent more than four hours to come up with the test that fails on purpose. In the beginning, my idea about this test was to make some changes to the date method in the player. But I came with the idea of creating two methods that fail and pass at the same time. Consequently, I spent more time on creating these two methods.

Thanks to this assignment I have improved my skills at writing tests in junit and from this assignment, a developer can make sure either the methods are going to work or not. But it would be better there was more time to spend on this project. Therefore, more precise tests would have been written.

Iteration3

Category	Actual time dedicated	Assumption for finishing	Date of start	Date of finish	Deadline
Manual testing	8 hours	5 hours	2020-03-07	2020-03-07	2020-03-09
Automated testing	3 days	2 days	2020-03-06	2020-03-08	2020-03-09

5.4 Iteration 4

It is finally the time for presenting the final project which was created in different iterations. The main purpose in this iteration is to make sure about the functionality of the project. Many implementations have been done in the previous iterations. However, there were some mis functionality along the way. The developer has fixed those problems. As an example, in the game there is a stage where the system asks the player to enter some information. Date is one of the information that will be asked from the player. In previous iterations, the system would except invalid inputs, but this problem has been solved. Moreover, 2 options have been added to the game menu in this iteration that are hint and exit. As it is obvious, the player can get help about the procedure of the game if the option hint has been selected.

Category	Actual time dedicated	Assumption for finishing	Date of start	Date of finish	Deadline
Planning	3hours	2hours	2020-03-18	2020-03-18	2020-03-20
Debugging	8 hours	5 hours	2020-03-19	2020-03-19	2020-03-20
implementations of new features	2 hours	2 hours	2020-03-20	2020-03-20	2020-03-20

The main idea for this iteration was to present a flaw less project which can lead to player's amusement and satisfactory of the game.

Personal reflection:

The debugging part was the most time-consuming part in this iteration, since my attention was to correct my mistakes in previous iterations as much as possible.

6. Risk Analysis

Logically, every project may face some risks. Therefore, a developer should have a backup plan or be prepared to overcome those risks. Due to the importance of this matter, there is a chapter called project management (Chapter 22), that introduce some chances to reduce and avoid the risks. In order, to minimize the risk of the project, there are some steps that project manager should consider to be ready, avoid and reduce the impact of the risk.

6.1List of risks

It is very important to identify the possible threats to reduce their impacts on the project. There are some prominent threats to the project:

- **Lack of time** (since the time is limited, it can result in some errors).
- **Lack of human resource** (Only one developer is responsible for code implementation).
- **Lack of knowledge** (It can be challenging for a developer who is the first time).

- **System failure** (There may be a possibility that the system crash in one stage).
- **Delay in delivery** (The project may not be ready by the deadline because of loads of works).

6.2 Strategies

There are some approaches that can reduce the chance of project risk that are:

- **Using popular libraries and websites** (e.g. stack overflow is popular community of developers where a developer can consult with others and get help).
- **Using previous knowledge** which can result in acceleration of the process and planning of the project.
- **Having a backup plan** (e.g. uploading the files in google one drive) can help the developer for retrieving the files.
- **Not developing a difficult project** which eventually may increase the chance of risks.

Personal Reflection:

Since I consider my code simple and it does not contain many features, the chance of risks will eventually reduce. However, working on complicated projects and implementing hard and extra features in the game will result in the creation of more risks.

I have decided to mention the above risks and strategies because there are very common to happen during my project especially the strategies. Also, the given risks may happen in my project due to my lack of knowledge in programming and planning.

7. Time log

Assignment	Actual time dedicated	Assumption	Overview	Deadline
Iteration 1	3 days	4 days (for finishing)	A very first plan (skeleton) for the code structure and an overview of the first steps of a game was created.	2020-02-03
Iteration 2	2days	3 days (for finishing)	A significant feature called game mode has been added. It is a feature that allows the user to play the game in moderate or hard modes. Also, personal information will be asked from the player	2020-02-24
Iteration 3	3days	4 days	At this stage two main purposes that are manual testing and automated testing have been completed.	2020-03-09
Iteration 4	3days	5 days	All the iteration has been placed in one file. Debugging and adding new options in menu are promising in this iteration.	2020-03-20

8. 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.