Project Hangman

BY Abdalrhman Dabor

Date: 2020-03-20

https://github.com/abodn70/Hangman_java/tree/master/Hang_Man

NORTH
SOFTWARE

Contents

1 / Revision history

Date	Version	Description	Author
2020-02-01	0.2	First draft of project plan	Abdalrhman Dabor
2020-02-07	0.5	Add some features and UML diagrams	Abdalrhman Dabor
2020-02-12	0.8	Finished the testing part	Abdalrhman Dabor
2020-02-16	1	Finished the project	Abdalrhman Dabor

2 / General Information

Project Summary	
Project Name	**Project ID**
Hang_Man	011250
Project Manager	Main Client
Abdalrhman Dabor	Esports CO.
Key Stakeholders	
System Managers, System Testers, System Developers, End Users, Customers	
Executive Summary	
This Project is about developing a game for Esport Co.	

The game shows a welcome message and then will show the menu where a player can set the level or start a new game or see the highest score and exit.

The program will pick a random word. When the program has chosen a word, it will display a set amount of underscores to indicate the length of the word picked. Where The player is guessing the word may, at any time, attempt to guess the whole word. If the word is correct, the game is over, and the guesser wins if the other player makes enough incorrect guesses.

3 / Vision

This vision for this project is to develop a game of hangman to Esport Co. The game depends on good guessing and strategies of knowing the right words through guessing the number of letters in each word. Any right character you enter it will show that the character exists. In the beginning, the program displays the word represented as underscores. And then you will ask you to guess any possible letter you will have 12 chances, each time you type a wrong letter, it will print a part of gallows. There will also be MultiWindow to play with someone else on the same device, simply you are going to have some word, and each one will guess a letter when it turns; each one has 1 minute to type a letter.

My reflection about this vision as a project manager is that the requirements set by the client are realistic with the set time-period that the client has given us. There is no need for new software or hardware to be implemented; the team can use the exciting Java IDE to produce this software. The experience I got from creating this vision is that it takes time and imagination to produce a good product when you do not have so many system requirements from the client. Moreover, I do believe that the game will be well-known, it is an entertaining game with a small size, and it can be downloaded on most devices the old ones and the new ones. However, there are still some challenges to handle, once we overseas the challenges it will raise the percentage of success.

4 / Project Plan

There are four iterations in this project.

The first iteration, with the due to 2020-02-01, will be about the basic structure of the program. System Managers will take control of the documentation in this iteration.

The second iteration, due to 2020-02-07, will be about the project design. It means we should create use cases, fully dressed use case, state machine diagram, class diagram, and implementation. System Developers will take control of the documentation in this iteration.

The third iteration, due to 2020-02-12, will be about software testing, manually and automatically. System Testers will take control of the documentation in this iteration.

The project will be finished by 2020-02-16.

4.1 Introduction

the game depends on good guessing and strategies of knowing the right words through guessing the letters for each word, and any correct letter you enter it will be shown that the letter exists in the word and if not it will print a part of gallows each time until it prints the entire gallows hence you will be lost.

4.2 Justification

It is a suitable game for everyone and enhances their Linguistic skills and their guessing skills as well as having fun and good times with the family and friends.

4.3 Stakeholders

The Stakeholders in this project are :

System Managers: Will use the requirements document to plan a bid for the system and plan the system development process.

System Developers. Will use the requirements to understand what should be developed.

System Testers. Will use the requirements to develop validation tests for the system. And test any new feature and its compatibility before it is released.

End Users. Will use the software.

Customers. Will pay for the software.

4.4 Resources

The main resource that the project has is the Java-IDE Eclipse, JDK-11.0.7, laptop HP core i7-8750H CPU@ 2.2GHz, RAM 8 GB, X64 Operation System.

One Manager.

One Tester.

One Developer.

4.5 Hard- and Software Requirements

The game doesn't require high hard and software requirements as it is textually based. However, it could be changed if JavaFX being used in future updates.

The minimum requirements to launch the game are:

Intel Pentium

Java 8 SDK

128 MB RAM

Java Compiler (NOT MANDATORY SINCE THE GAME COULD BE LAUNCHED USING RUNNABLE JAR FILE)

Windows XP

4.6 Overall Project Schedule

2020/02/01 Creating the project documentation.

2020/02/04 Creating a skeleton of code including essential functions like user interface, Picking up words, display the number of dashes.

2020/02/06 Deadline for delivering release 1

2020/02/07 Modeling the code using UML: use case class and state machine diagrams.

2020/02/11 Deadline for delivering release 2

2020/02/12 start testing the code, manually and automatically.

2020/02/13 Add more features like changing levels and pick up a word based on the level difficulty, and menu bar, and pause option.

2020/02/13 Test the new features, manually and automatically.

2020/02/14 Make sure everything is being tested.

2020/02/15 Review the code and check the document before the final release.

2020/02/16 Deadline for delivering release 3.

4.7 Scope, Constrains and Assumption

Scope:

The application will be published on markets once the final release is delivered. The game will be free up to level 30 after that the user should pay if he/she likes the game and wants to continue and become a customer. There are some other features for customers that will be added in feature release like playing online, share scores, invite friends, and use gems to replace one character.

Constraints:

As one manager and tester and developer are working on the project, time would be crucial; sticking to the plan is required. The structure should be simple and understandable. All the files should have descriptive names. Every iteration should contain a time-log.

Assumptions:

End-user should have Java Virtual Machine environment, with working mouse and keyboard.

4.8 Reflection

A Project Plan is a major document in plan-driven development. The project plan is useful for the developers who are interested in developing the game. Moreover, the scheduled time for Each iteration is essential. All iterations should be done before the deadline, however, in the last iteration, we have some time to review the entire code
and check errors, and if something is missing from a previous iteration, we can add it.

5 / Iterations

The project plan has for four iterations, including this. This is a fine-grained plan on what is to be done in each iteration and with what resources. To begin with, this is a plan of what we expect to do, update this part with additions (never remove anything) when plans do not match up with reality. Also, make time estimates for the different parts. In this course, the overall planning has, in some ways, already been decided, so use the template to provide more details on specific tasks that define your project. Remember that you can plan to add features to any of the phases as long as the main focus is also met. The first assignment is to complete the iteration one.

5.1 Iteration 1

This Iteration will focus on the project plan, make the main documents that help us to know our start point and skeleton code. The plan should have Time logs, Iterations, general information about the project, vision, and reflections. The deadline for this iteration is 6/2/20.

Tasks in this iteration:

- *Write the project General Information about the project. Estimated time: 1h.*
- *Read Software Engineering book edition number 10, chapters 4,5,6,7. Estimated time: 3h.*
- *Write the project plan. Estimated time: 30m.*
- *Write the project vision. Estimated time: 25m.*
- *Write the project list risks. Estimated time: 30m.*
- *Write basic user interface code. Estimated time: 2h.*
- *Check everything. Estimated time: 2h.*
- *delivering release. Estimated time: 20m.*

5.2 Iteration 2

This Iteration will focus on drawing diagrams (UML diagram, Class diagram, state machine diagram) and Play game scenario, Use case scenario, and the logic code of the game. State machine and Class diagram creating must take place, and Use Cases must be described. Also, the implementation should start according to the model. The deadline for this iteration on 11/2/20.

Tasks in this iteration:

- Draw State Machine Diagram. *Estimated time: 1h.*
- Draw Class Diagram. *Estimated time: 20m.*
- Add the logic code of the game. *Estimated time: 10h.*

5.3 Iteration 3

This iteration will focus on testing the logic code that we added in the previous iteration, manually and automatically. The deadline for this iteration is 15/2/2020.

Tasks in this iteration:

- Reading the Java course(1dv506) slides. *Estimated time: 30m.*
- Read Software Engineering book edition number 10, chapters 8, *Estimated time: 1h.*
- Create Use Case Model. *Estimated time: 30m.*
- Write a Test plan. *Estimated time: 1h.*
- Write UC1,2,3. *Estimated time: 1h.*
- Draw Use Case Diagram. *Estimated time: 20m.*
- Write JUnit autotest. *Estimated time: 2h.*
- Write test report, *Estimated time: 2h.*

5.4 Iteration 4

This iteration will focus on complete our code if something is missing and test our code. Moreover, if any bugs are being found, we will fix it. The deadline for this iteration is 16/2/2020.

Tasks in this iteration:

- Check UML Diagrams. *Estimated time: 2h.*
- Check the code. *Estimated time: 2h.*
- Check testing. *Estimated time: 2h.*
- Make a final release. *Estimated time: 1h.*

6 / Risk Analysis

6.1 List of Risks

Risk	**Probability**	**Consequence**
Loss of data	Low	Catastrophic
Hacking threat	Low	Serious
Crash Errors	Medium	Serious
Sickness	Medium	Tolerable

6.2 Strategies

Risk	Strategy
Loss of data.	This risk can be evaded by saving the documentation and code on different devices so that in case one hard drive crashes, you have backup files.
Hacking threat	For avoiding this kind of threat, we need to have SSH protocol. SSH provides strong password authentication and public key authentication, as well as encrypted data communications between two computers connecting over an open network.
Crash Errors	For avoiding this kind of risk, we should have a testing team that can keep testing after each update and detects the bugs.
Sickness	By planning in extra time for the work needed the consequence can be lowered

7 / Time Log

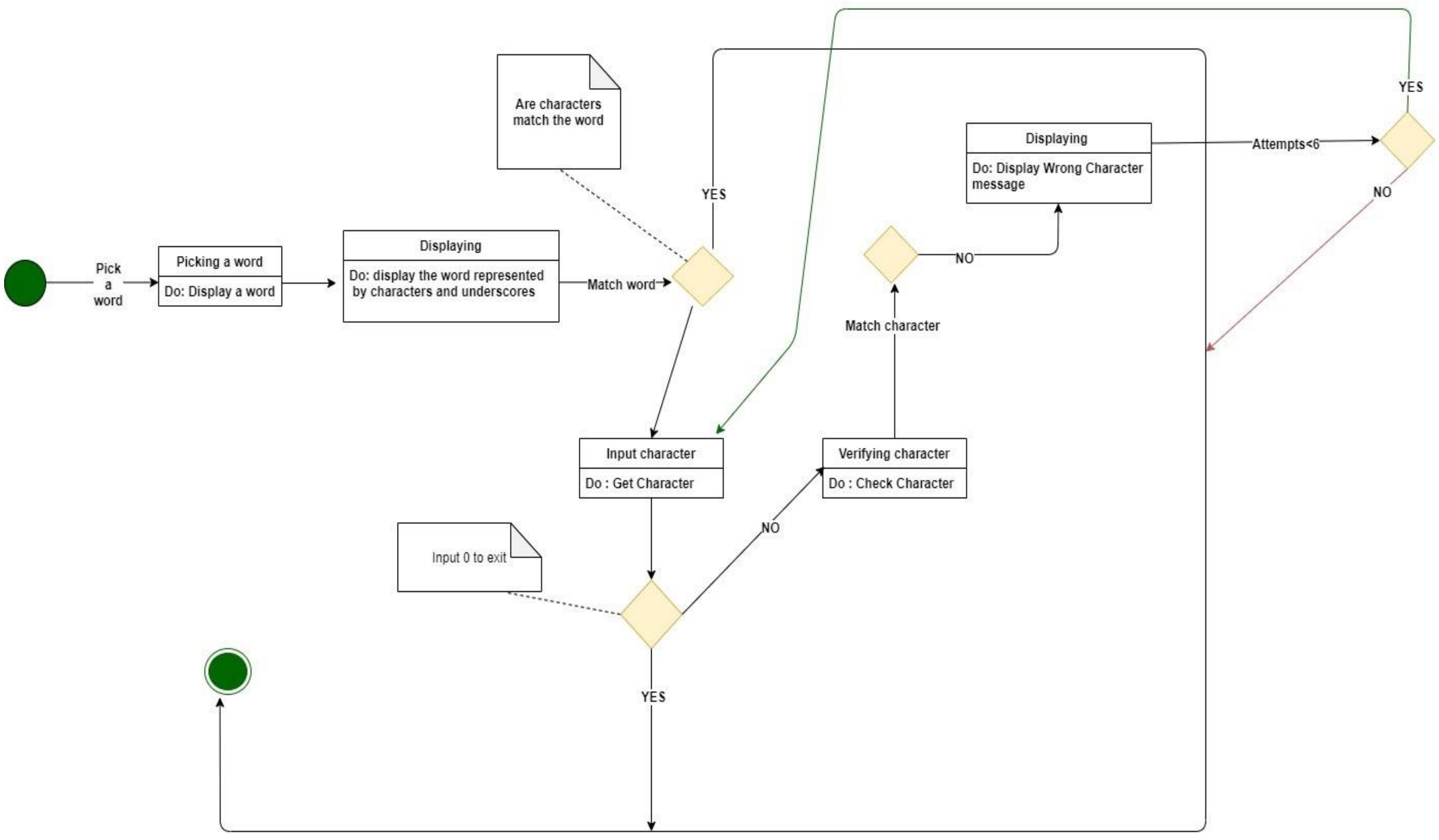
Date	Time needed	Task
2020-02-01	3 hours	Revision history, General information, and Vision
2020-02-04	4 hours	Project plan
2020-03-12	6 hours	Iterations and Risk Analysis
2020-03-16	10hours	Preparing to send all files to GitHub

Diagrams

State Machine Diagram:

Fig 1: state machine that shows the actions and how the game works

Class Diagram:



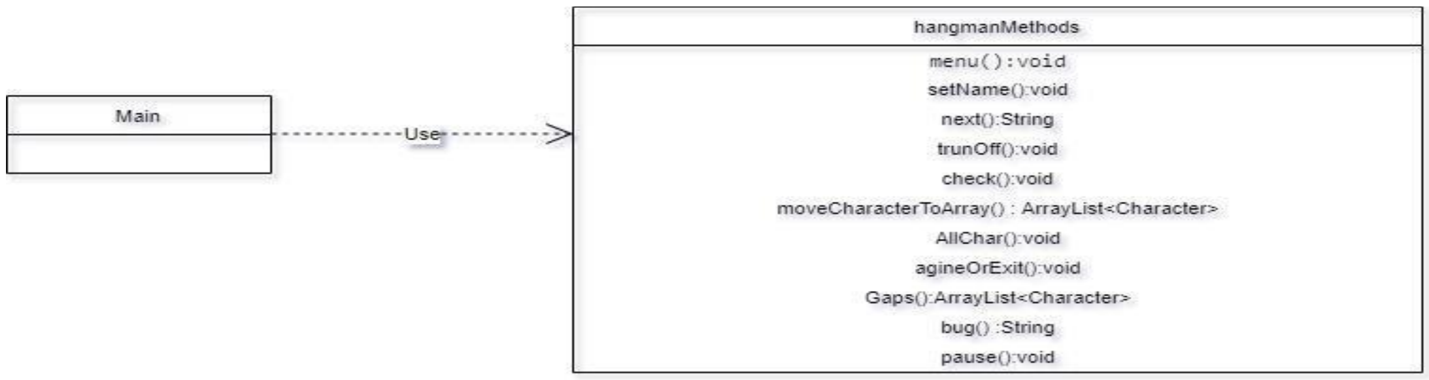


Fig 2: Class diagrams of the game and their structure

■ Use case diagram

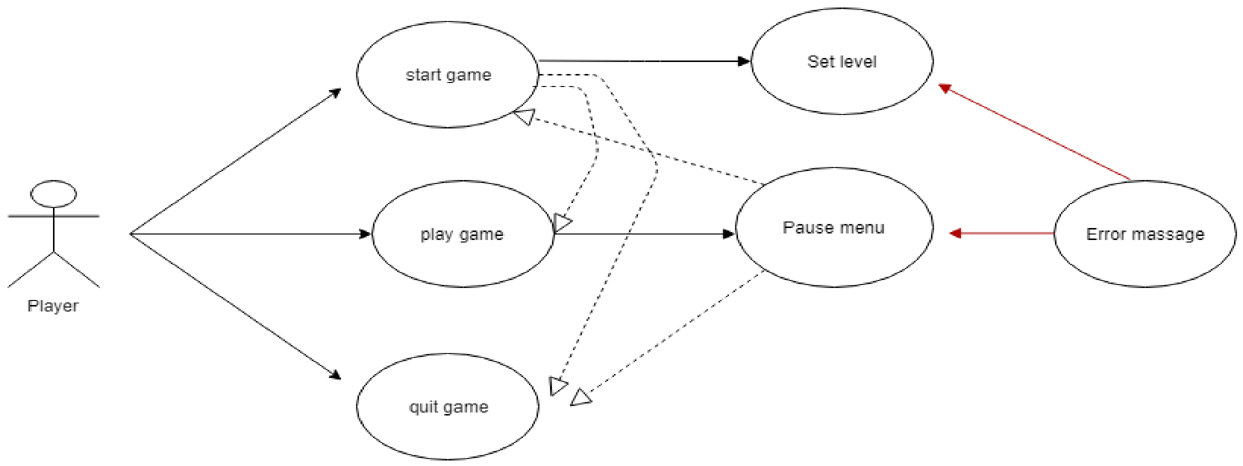


Fig 3: Use case diagram that shows that you can do in the game

■ Use Cases

■ UC1 Store a character

Precondition: None.

Postcondition: a character is entered.

Main scenario

1. The user chooses a new game from the menu.

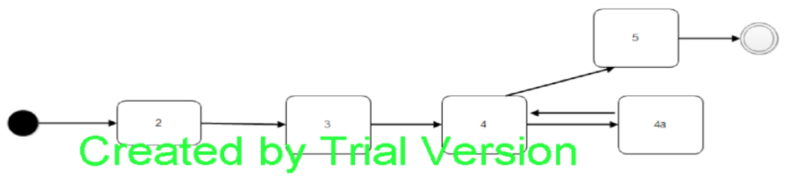
2. The system will present underscores and ask the user to fill them, one by one. Each underscore represents one character.
3. The user provides a character.
4. The system stores the character and shows if the word contains the letter or not.

Alternate scenarios

- 3a The user provides a number.
- The system displays an error message.
- The system will ask the user to retype a character go to UC1 step3, in the Main scenario.

The system will not count this attempt.

Activity Diagram



■ UC2 Change the level

Precondition: none.

Postcondition: The game is up and running.

Main scenario

1-The system presents the menu.

2-The user chooses to Change the level option from the menu.

3-The system will ask the user to enter a level between 0 to 5.

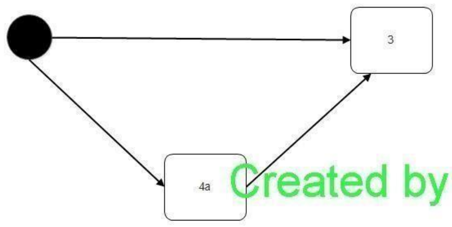
4-The user inputs a level.

5-The system starts the game.

Alternate scenario

- 4a The user has not stored a right level.
- The system shows an error message.
- Go to UC2 Step 3, in the Main scenario.

Activity Diagram

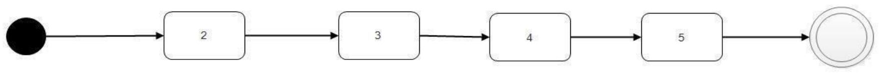


■ Manual Test-Cases

- **TC1.1 Store characters successfully.**

Use case: UC1 Store **a character**.

Scenario: Store a character successfully and shows the characters which are already used.



- Test steps

- Start the game
- The system displays the menu.
- Choose Start a new game.
- The system asks the user to enter a character.
- Enter the character "a" and press enter.
- Enter another character "b" and press enter.

- Expected

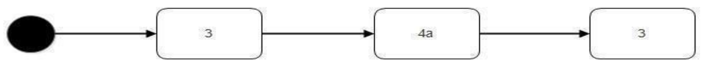
- The system should show that "a" and "b" are already used.

- TC1.2 Input number instead of character

Use case: UC1 Store a character.

Scenario: Store a number instead of a character.

The alternate scenario where the user enters a number and, forced to enter a character.



- Test steps

- Start the game.
- The system shows the menu.
- Choose Start a new game.
- The system asks the user to enter one character, one by one, to fill the underscores.
- Enter "3", and press enter.

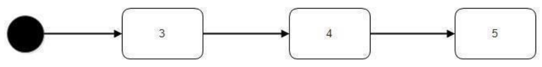
- Expected

- The system should show "Invalid input, try again".
- The system waits for the new input.

■ TC2.1 The system changes the level successful Use case :

UC2 change the level.

Scenario: The user changes the level successfully.



- Test steps

- Start the game.
- The system shows the menu.
- Choose Change the level from the menu.
- Enter 3.

- Expected

- The system should change the level successfully.
- Level 3 should be harder than 1 and 2.
- The system will start the game from the level that is chosen.

■ TC2.2 Input an invalid integer

Use case: UC2 Change the level.

Scenario: Input an invalid integer. The alternate scenario where the user enters an integer that is not between 0-5, is forced to input a valid integer in the range.

- Test steps

- Start the game.
- The system shows the menu.
- Choose Change the level from the menu.
- Enter "9".

- Expected

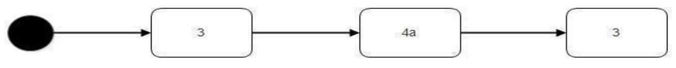
- The system should displays "invalid input. Please try again".
- The system waits for the new input.

.

■ TC2.3 Input a character instead of an integer

Use case: UC2 Change the level.

Scenario: input a character instead of a number. The alternate scenario where the user enters a character is forced to enter a new value between 0-5.



- Test steps

- Start the game.
- The system presents the menu.
- The system will ask the user to enter a level between 0-5.
- Enter "a" and press enter.

- Expected

- The system should show ("A wrong value, please input a valid value).
- The system waits for the new input.

■ Fully dressed UC

■ UC 3 Play Game

Precondition: None.

Postcondition: the game is up and running.

Main scenario

1- Starts when the Player wants to begin a session of the hangman game.
2- The system displays the main menu with a title, the option to play a new game, change level and exit the game.
3- The user chooses to start a new game.
4- The system displays Incomplete word represented by characters and underscores, which are the number of characters that are missing in the word.
5- The user inputs a character.
6- The system checks the entered character.
7- The system replaces the underscores with the guessed character/s.
8- Repeat from step 4 if the entire word is mismatched.

9- The system displays a message that represents that the player won the game.
10- The system will increase the level by one and go to step 4.

Alternative scenarios

- **The user enters miss-matched character to the word.**
    1- The system displays the remaining attempts and asks the user to try again.
    2- Repeat from UC3 step 6 if the entered character is mismatched and there is still an attempt/s left.
    3- The system displays a message telling that the user lost.
    4- The system will ask if the user wants to start a new game repeat from UC3 step 4 or go to the main menu repeat from UC3 step 2.

- **The user wants to terminate the game while playing.**
    1- The user enters 0.
    2- The system displays pause menu with options, continue repeat from UC3 step 5, or back to the main menu repeat from UC3 step 2, or exit the game.

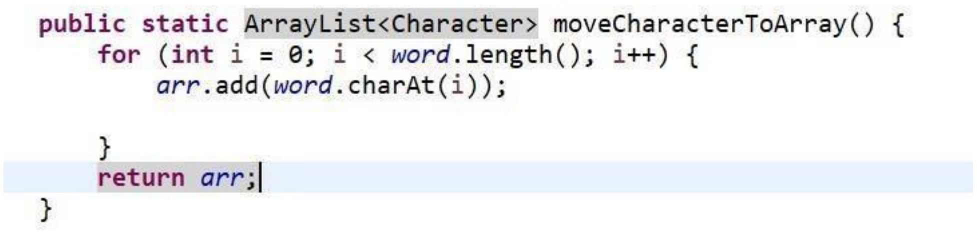
■ Test Report

Test	**UC1**	**UC2**
TC1.1	1/OK	0
TC1.2	1/OK	0
COVERAGE & SUCCESS	2/OK	0
TC2.1	0	1/OK

TC2.2	0	1/OK
COVERAGE & SUCCESS	0	2/OK

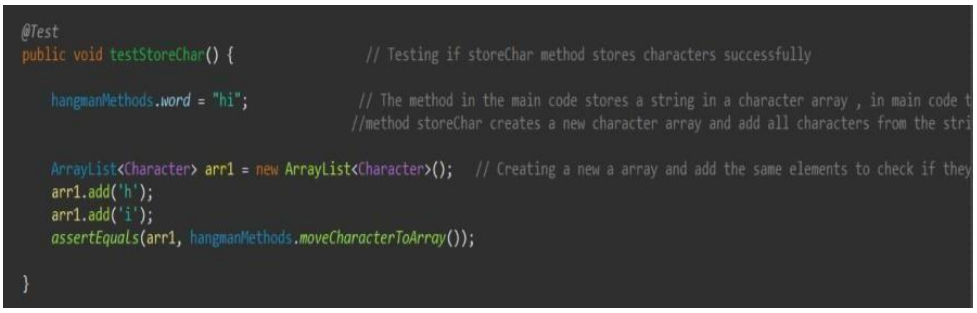
■ Unit Testing

■ Source code 1:



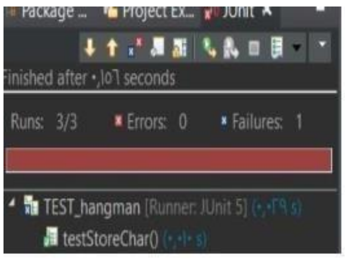
```java
public static ArrayList<Character> moveCharacterToArray() {
    for (int i = 0; i < word.length(); i++) {
        arr.add(word.charAt(i));

    }
    return arr;
}
```

- Unit test for Source code 1:



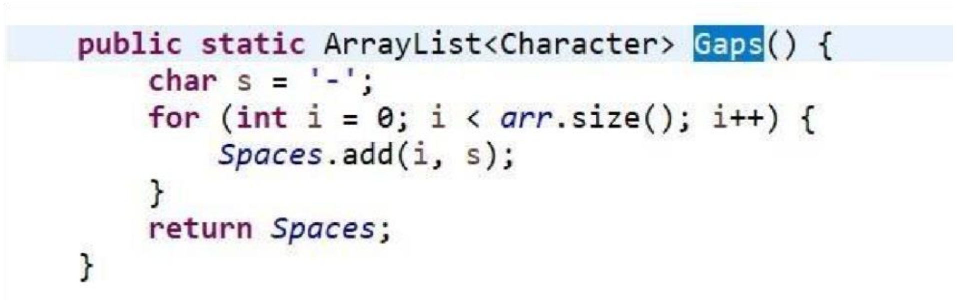
```java
@Test
public void testStoreChar() {                  // Testing if storeChar method stores characters successfully

    hangmanMethods.word = "hi";               // The method in the main code stores a string in a character array , in main code t
                                             //method storeChar creates a new character array and add all characters from the stri

    ArrayList<Character> arr1 = new ArrayList<Character>();   // Creating a new a array and add the same elements to check if they
    arr1.add('h');
    arr1.add('i');
    assertEquals(arr1, hangmanMethods.moveCharacterToArray());

}
```

- Result



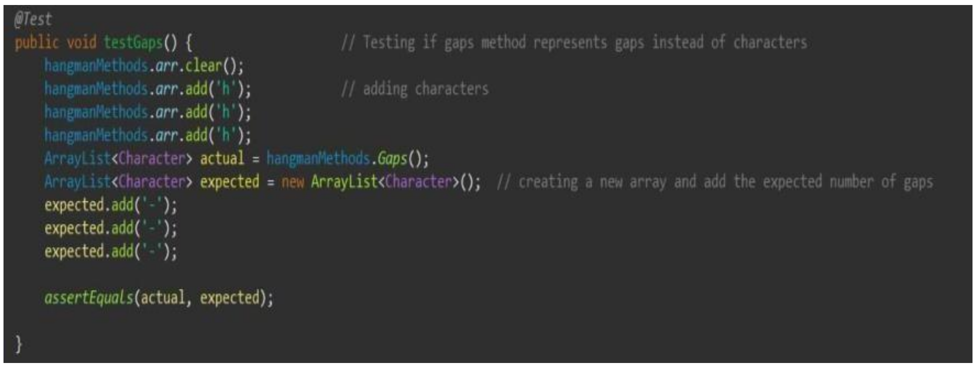
```
Package ...   Project Ex...   JUnit
Finished after ٠,١٥٦ seconds

Runs: 3/3      Errors: 0      Failures: 1

TEST_hangman [Runner: JUnit 5] (٠,٠٣٩ s)
    testStoreChar() (٠,٠١٠ s)
```

Source code 2:



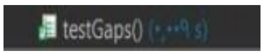
```java
public static ArrayList<Character> Gaps() {
    char s = '-';
    for (int i = 0; i < arr.size(); i++) {
        Spaces.add(i, s);
    }
    return Spaces;
}
```

- **Test source code 2:**



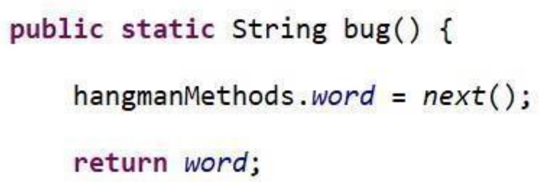
```java
@Test
public void testGaps() {                    // Testing if gaps method represents gaps instead of characters
    hangmanMethods.arr.clear();
    hangmanMethods.arr.add('h');            // adding characters
    hangmanMethods.arr.add('h');
    hangmanMethods.arr.add('h');
    ArrayList<Character> actual = hangmanMethods.Gaps();
    ArrayList<Character> expected = new ArrayList<Character>();  // creating a new array and add the expected number of gaps
    expected.add('-');
    expected.add('-');
    expected.add('-');

    assertEquals(actual, expected);

}
```

- **Test result:**



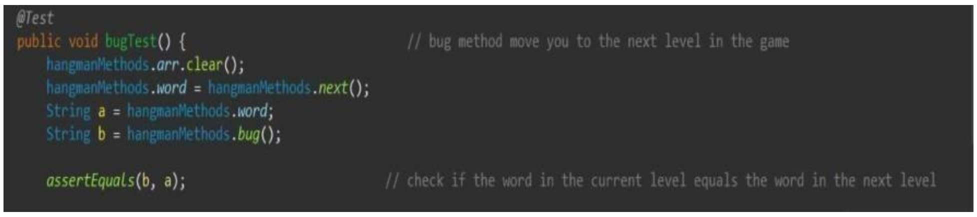
```
testGaps() (٠,٠٠٩ s)
```

Source code 3:



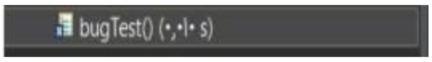
```
public static String bug() {

    hangmanMethods.word = next();

    return word;
```

- **Test source code 3:**



```
@Test
public void bugTest() {                         // bug method move you to the next level in the game
    hangmanMethods.arr.clear();
    hangmanMethods.word = hangmanMethods.next();
    String a = hangmanMethods.word;
    String b = hangmanMethods.bug();

    assertEquals(b, a);                        // check if the word in the current level equals the word in the next level
```

- **Test result:**



```
bugTest() (٠,٠١٠ s)
```

■ Reflection

moveCharcterToArray and next tests pass.

We examined the code, and most classes require mocks to be testable. Although the game works correctly, we need to consider the upcoming updates such as having a real hangman instead of the number of attempts, in order, to engage the user and add the online feature which enables the user to play with online with another user. We need to consider adding new levels as well.

Test	**Console View**	**GameEntity controller**	**Main**	**Game Entity**
Test store character	0	0	0	100%/OK
Test underscores	0/NA	0/NA	0/NA	100%/OK

Test bugs	0	0	0	50%/OK
Coverage&success	0/NA	0/NA	0/NA	75%/OK

■ Manual Test cases

■ TC1 User name :

Precondition: The player starts the game.

Postcondition: Welcome message shown to the player.

Main scenario

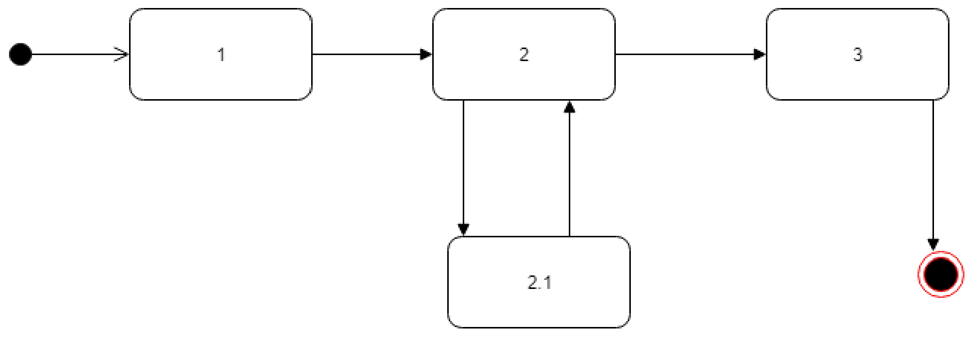
1- The system shows a message to enter a name.
2- The player enters a name.
3- The system shows a welcome message.

Alternative scenarios:

2.1 The player enter number

- The system shows an error message
- system force the player to renter the username

Activity diagram



-

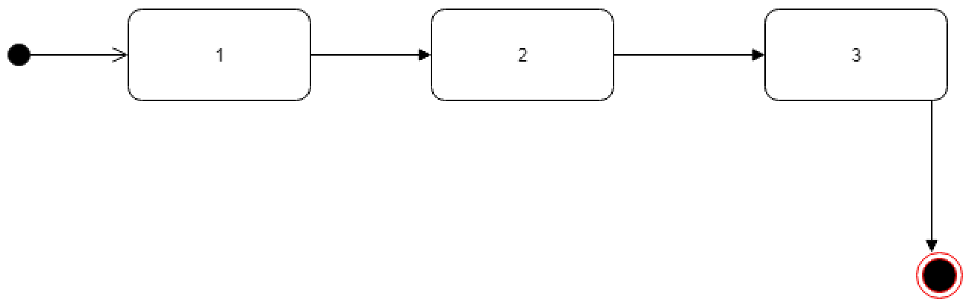
■ TC1.1 User made a valid name :

Precondition: The player starts the game.

Postcondition: Welcome message shown to the player.

Scenario: Welcome message scenario.

The main scenario is tested where a user input a valid name.



Precondition: start the game.

Test steps

- Lunch the game.
- The system shows "Please enter your name".
- Player input "Adam" and Enter.

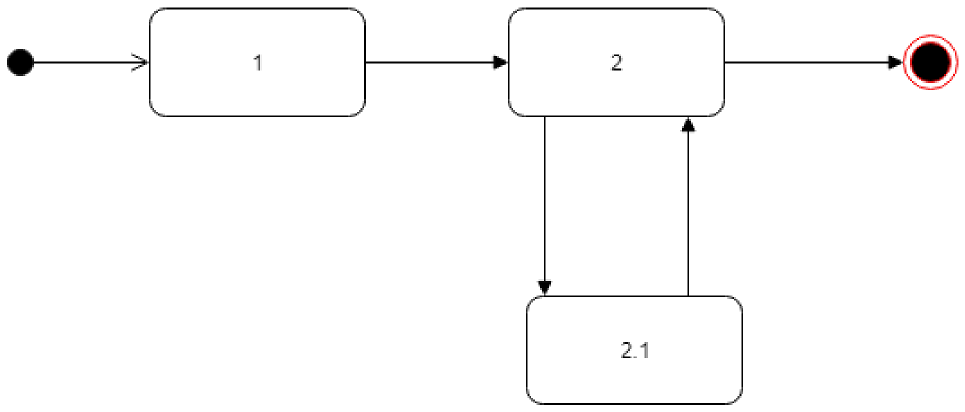
Expected

- The system should show "Welcome Adam ".

TC1.2 User made an invalid name:

Scenario: invalid input force to input again.

The alternate scenario where the user enters a number instead of letters and forced to enter a valid character again.



Precondition: the game already started.

Test steps :

1- Lunch the game.
2- The system shows "Enter your name".
3- Input "100" and press enter.

Expected :

1- The system should show "Error input".
2- The system shows "Please enter your name " and waits for input.

- **Time plan:**

ID	Description	Estimated	Actual	Deadline
D1	User name scenario	3h	1h	22/3/2020
D2	Manual test case	1h	1h	22/3/2020
D3	Unit test	1h	1h	22/3/2020
D4	Test report	0:30	0:45h	22/3/2020
D5	Risk analysis	2h	1h	22/3/2020
D6	Strategies	1h	2h	22/3/2020
Total Time		8:30h	6:45h	22/3/2020