Hangman Project

Iteration 3

Yetnayet Edeglign Belachew

yb222ce@student.lnu.se

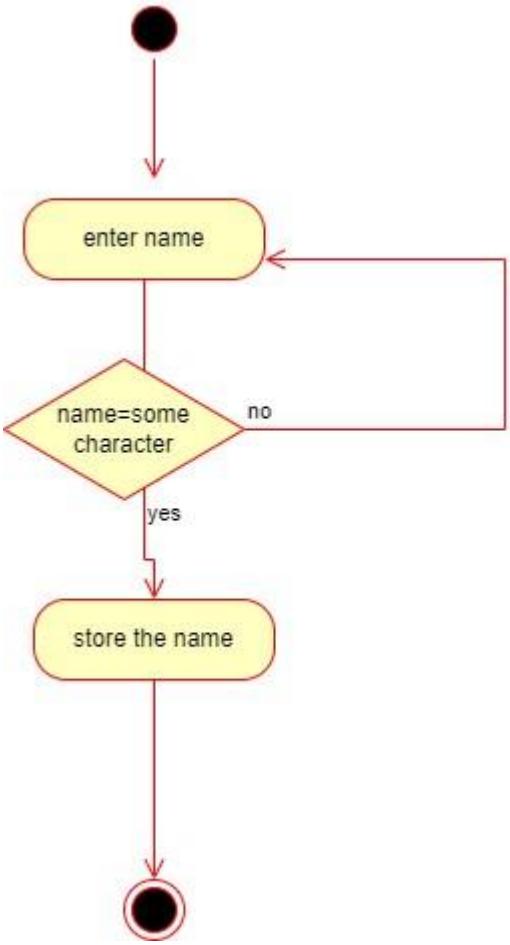
Date 2022

Use-Cases

This is a simple application with two use-cases.

Use Case ID:	UC-1
Use Case Name:	Store username
Primary Actor:	user
Description:	User needs to write his\her name. Store to the system.
Post conditions:	a **username** is sorted
Basic Flow:	• Starts when a user wants to enter her\his username into the game • The system asks for a username • The **User** provides the username • The system stores the username and shows that it has been stored
Alternative Flows:	• the user provides an empty username • The system shows an error message "Enter your name again!"

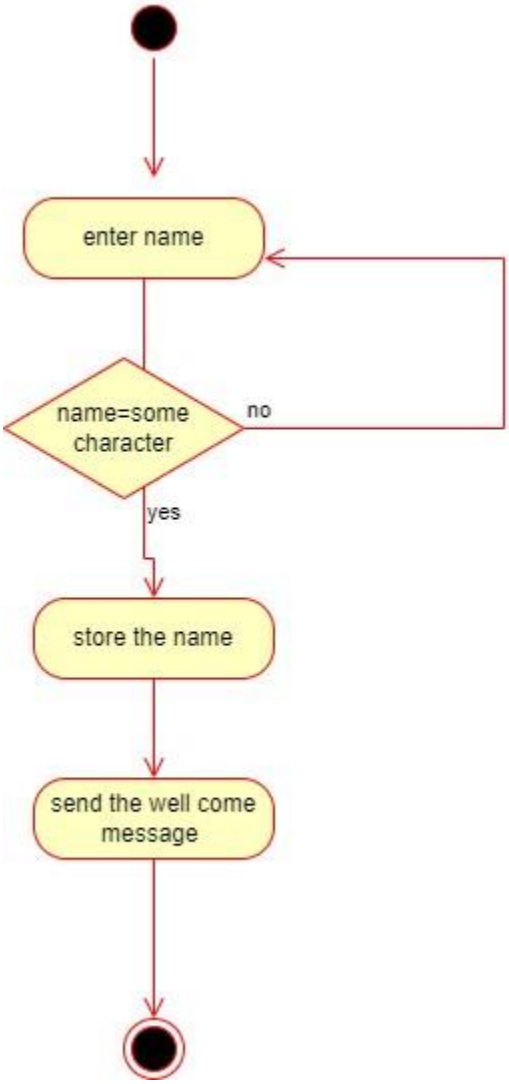
Activity Diagram



Use Case ID:	UC-2
Use Case Name:	Welcomed
Primary Actor:	system
Description:	Store the name and send message to user screen

Post conditions:	a **user** has been welcomed
Basic Flow:	• The user starts Hangman game • The system shows a welcome message including the username "Welcome to Hangman game!"
Alternative Flows:	• the user has not entered a username • Goto "UC1 Store a Username" • Goto Step 2. in the Main scenario

Activity diagram:



Objective

The main purpose of this assignment is to test the code that was implemented in iteration 2 (previous assignment) which gives the player some attempts to guess the letters from a chosen word based on different levels.

What to test and how.

I intended to test the methods that are important to be executed and test to check its validity have been given that run dynamic manual test-cases, so in the beginning, I wrote a test UC1 to check the validity of username, because it is the first step to starting the game, so without entering a username, the game will not be able to start. I also wrote automated unit tests for Hang-man methods in the class Hangman test (e.g., methods to check the level game if the level is easy, medium, or hard, so tests check if the input from the user is correct or wrong, However, as some methods that they do not affect the process of the game that much will not be tested (e.g., the method replace Char, will not do anything. On the other hand, testInvalidName, can be tested with different values) in Junit.

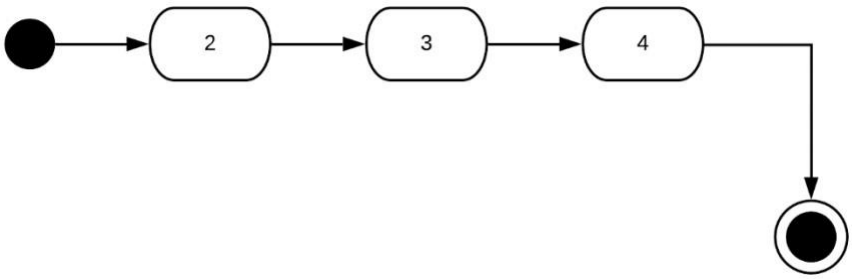
Time plan

Task	Estimated	Actual
Manual TC	4h	3h
Unit tests	3h	2h
Running tests manual	50m	20m
Code inspection	1.5h	1h
Test report	2h	3h

Manual Test-Cases

TC1.1 Enter a valid username "username has been entered" successful

Use case: UC1 Enter a username
Scenario: Enter username successfully "username has been entered"



The main scenario of UC1.1 is tested where the user enters his\her username successfully.

Pre-condition: Start the game **Test**

steps:

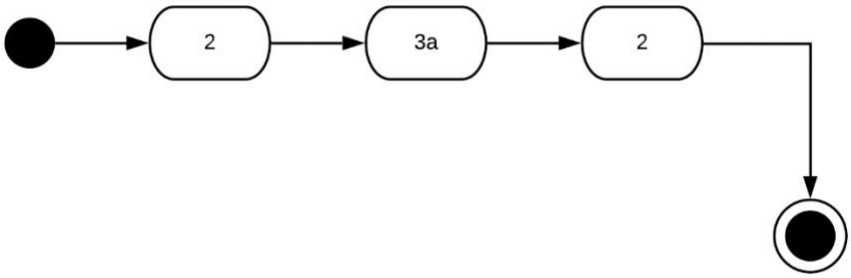
- User starts the game
- The system shows a message "Enter a name to continue:"
- Enter the name "Yetnayet" and press enter

Expected:

- The system will show a welcome message including the **name**" Yetnayet! Welcome to the Hangman game "
- The system shows the player a menu to choose a level: 1. easy level, 2. medium, 3. hard level and 4. quit the game

TC1.2 Enter an invalid username “empty name” force reprompt

The main scenario is tested when the user enters an empty username and system force user to enter a new username.



Precondition: Name must not be entered.

Test steps:

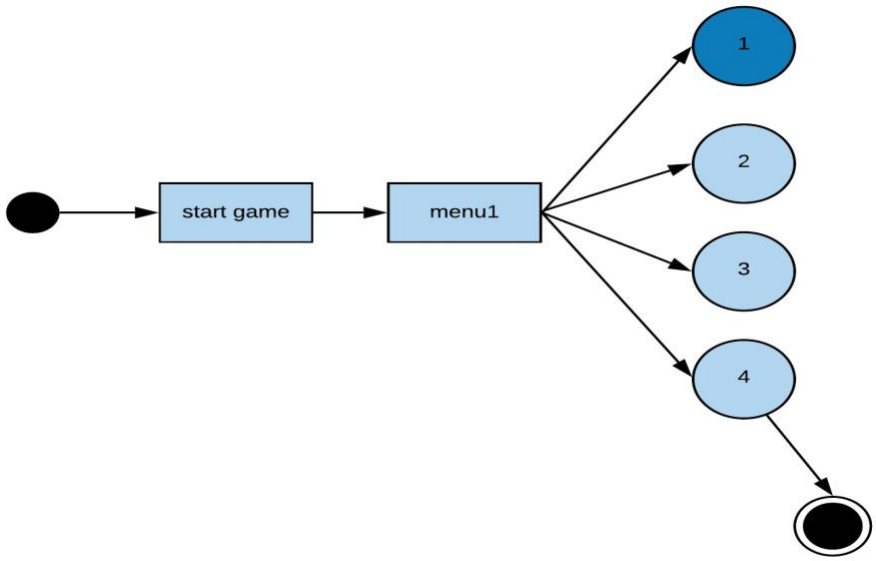
- User starts the game
- The system shows a message “Enter a name to continue:”
- Press enter without entering a name

Expected

- The system should show a message "Enter a name again"
- System shows " Enter a name again:" and waits for input

TC2.1 User chooses level game "Easy"

Scenario: The player chooses option 1 (Easy level) on the menu (menu1).



The main scenario of UC2.1 is tested where a user has been faced with a menu (Menu1) with four options and chooses option 1 to play Easy level.

Precondition: The player must enter nothing but 1 in the menu1 **Test**

steps:

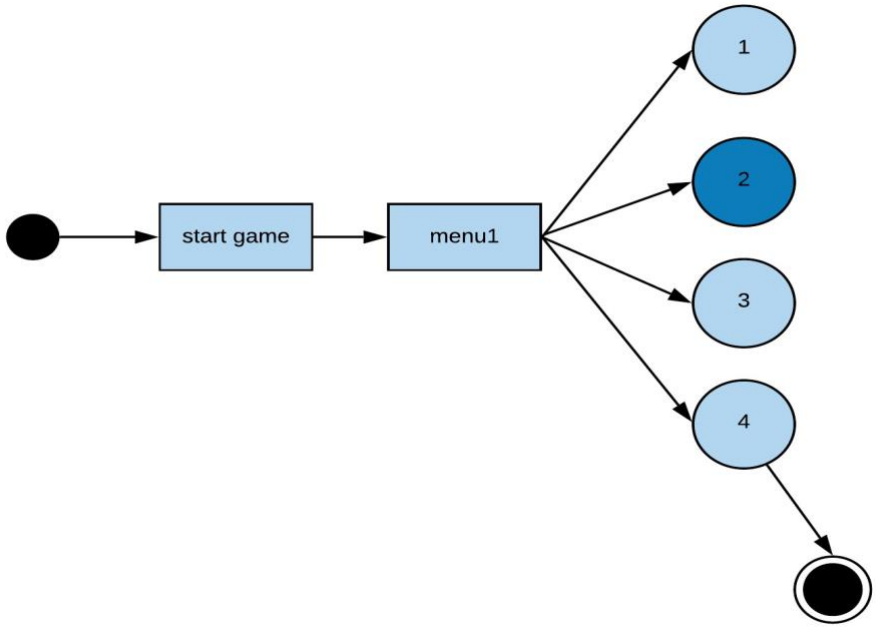
- User starts the game
- User will enter a name to continue
- User will be directed to choose a level menu (menu1)
- Player will choose option(number) 1 to play the easy level
- Game level default will be set as easy

Expected

- The player will be given 7 attempts to find the missing letters.

TC2.2 User chooses level game “Medium”

Scenario: The player chooses option 2 (Medium level) on the menu (menu1).



The main scenario of UC2.2 is tested where a user has been faced with a menu (Menu1) with four options and choose option 2 to play Medium level.

Precondition: The player must enter nothing but 2 in the menu1 **Test**

steps:

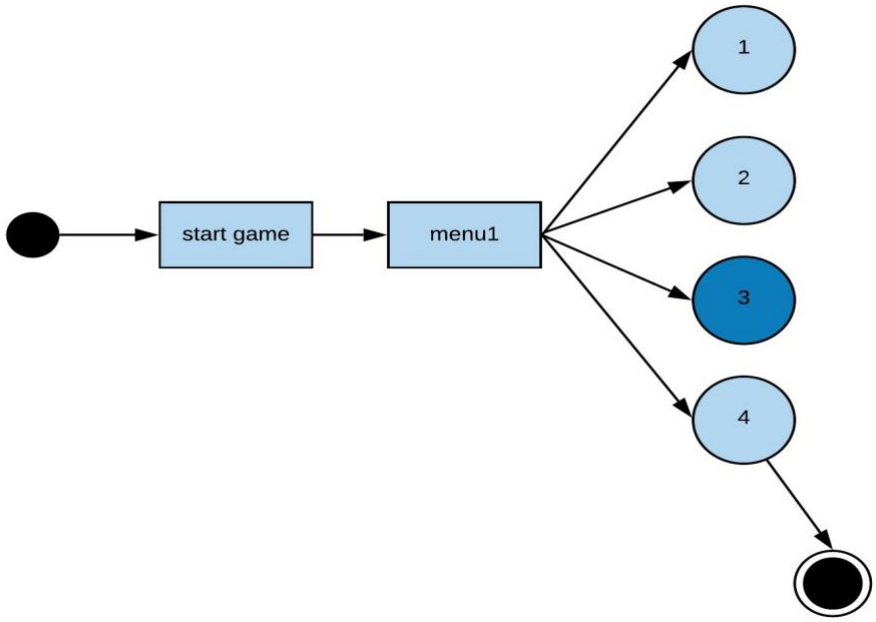
- User starts the game
- User will enter a name to continue
- User will be directed to choose a level menu (menu1)
- Player will choose option (number) 2 to play medium level
- Game level default will be set as medium

Expected

- The player will be given 5 attempts to find the missing letters.

TC2.3 User chooses level game "Hard"

Scenario: The player chooses option 3 (Hard level) on the menu (menu1).



The main scenario of UC2.3 is tested where a user has been faced with a menu (Menu1) with four options and chooses option 3 to play Hard level.

Precondition: The player must enter nothing but 3 in the menu1 **Test**

steps:

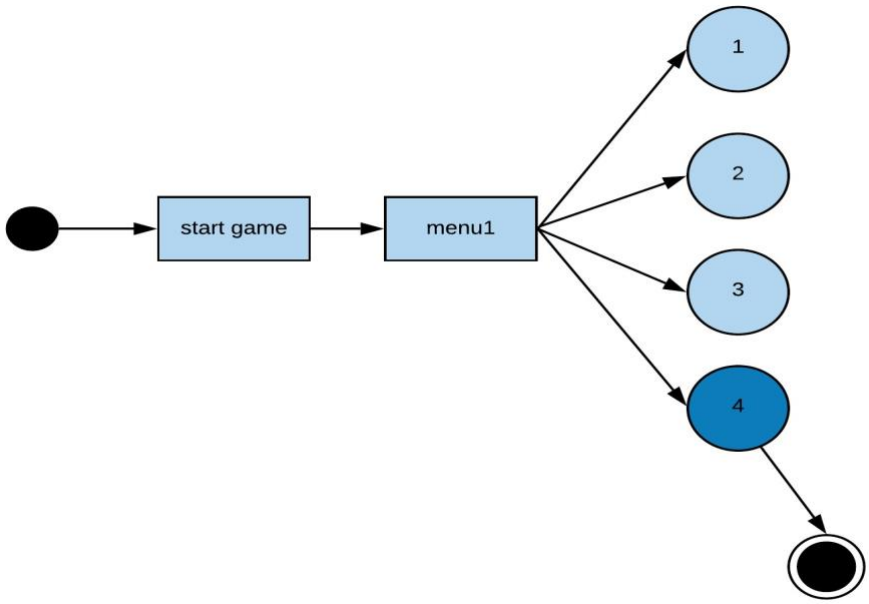
- User starts the game
- User will enter a name to continue
- User will be directed to choose a level menu (menu1)
- Player will choose option (number) 3 to play hard level
- Game level default will be set as hard

Expected

- The player will be given 3 attempts to find the missing letters.

TC2.4 User chooses to quit the game "Quit"

Scenario: The player chooses option 4 (quit the game) on the menu (menu1).



The main scenario of UC2.4 is tested where a user has been faced with a menu (Menu1) with four options and choose option 4 to Quit the game.

Precondition: The player must enter nothing but 4 in the menu1

Test steps:

- User starts the game
- User will enter a name to continue
- User will be directed to choose a level menu (menu1)
- Player will choose option (number) 4 to quit the game
- Player press enter in the keyboard

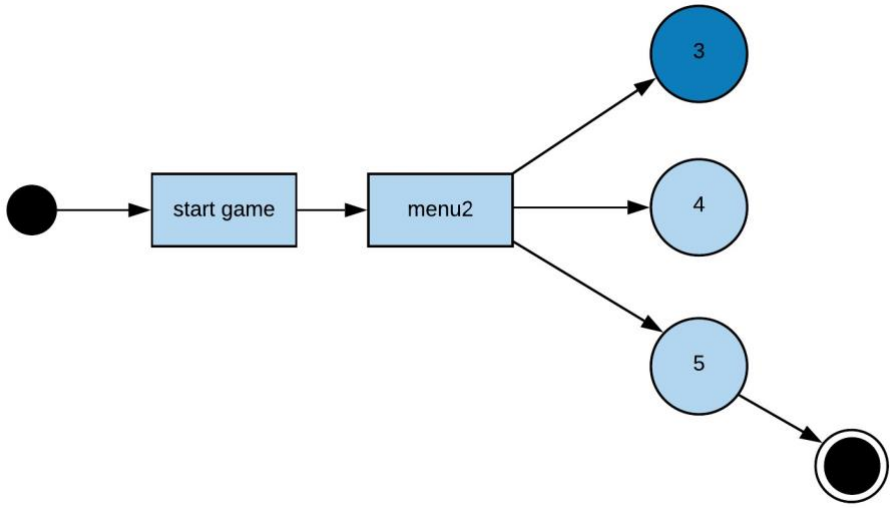
Expected

- The system will show a message “You quit the game, Goodbye”
- Program terminates.

TC3.1

User chooses words categories "Name of the countries"

Scenario: The player chooses the first option (Name of the countries) by pressing 3 in the menu2.



The main scenario of UC3.1 is tested where a user has been faced with a menu (Menu2) with three options and choose the first option (Name of the countries) by entering number 3.

Precondition: The player must enter nothing but 3 in the menu2

Test steps:

- Tester starts the game.
- Users will enter a name to continue.
- Users will be directed to choose a level menu (menu1).
- The tester will choose a level from menu 1 (1. easy, 2. medium, 3. hard).
- The tester will be directed to menu 2 to choose word categories.

Expected

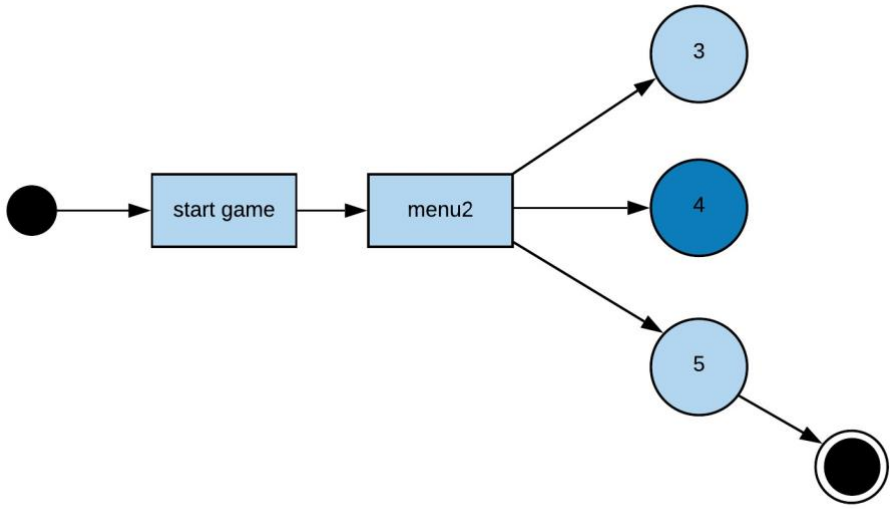
TC3.2

- User will choose option 3 (Name of the countries) from menu2.

- The player will guess the missing letter that is based on the name of countries.

User chooses words categories "Car brands"

Scenario: The player chooses the second option by pressing 4 (Car brands) in the menu2.



The main scenario of UC3.2 is tested where a user has been faced with a menu (Menu2) with three options and choose the second option (Car brands) by entering number 4.

Precondition: The player must enter nothing but 4 in the menu2.

Test steps:

Expected

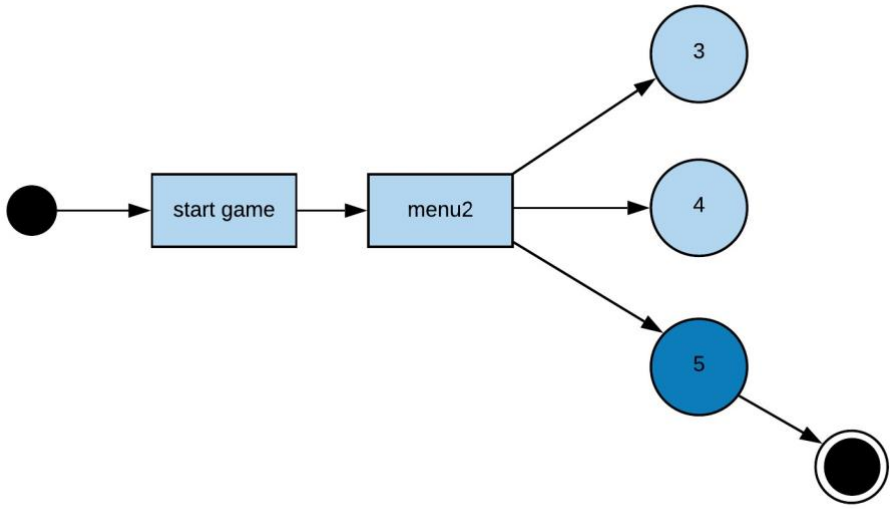
TC3.3

- Tester starts the game.
- User will enter a name to continue.
- User will be directed to choose a level menu (menu1).
- The tester will choose a level from menu 1 (1. easy, 2. medium, 3. hard).
- The tester will be directed to menu 2 to choose word categories.
- User will choose option 4 (Car brands) from menu2.

- The player will guess the missing letter that is based on the car brands.

User quit the game

Scenario: The player chooses the third option by pressing 5 (Quit) in the menu2.



The main scenario of UC3.3 is tested where a user has been faced with a menu (Menu2) with three options and choose the third option (Quit) by entering number 5.

Expected

TC3.4

Precondition: The player must enter nothing but 5 in the menu2.

Test steps:

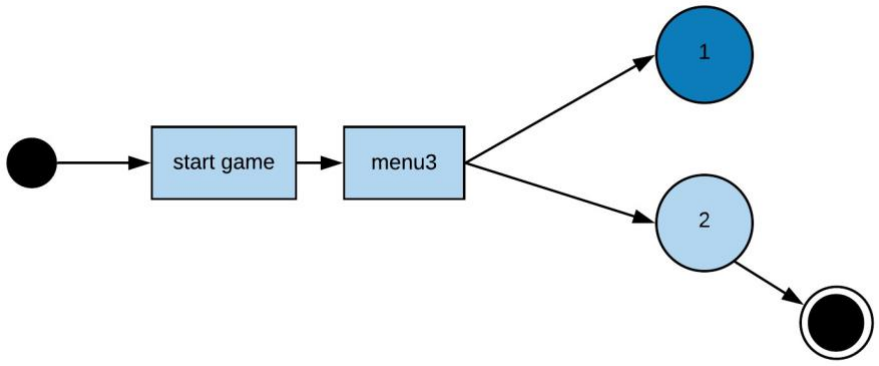
- Tester starts the game.
- User will enter a name to continue.
- User will be directed to choose a level menu (menu1).
- The tester will choose a level from menu 1 (1. easy, 2. medium, 3. hard).
- The tester will be directed to menu 2 to choose word categories.
- User will choose option 5 (Quit) from menu2.

- The system will show a message "You quit the game, Goodbye"
- Program terminates.

Expected

TC4.1 User restart the game

Scenario: The player chooses the first option by pressing 1 (Restart) in the menu3.



The main scenario of UC4.1 is tested where a user has been faced with a menu (Menu3) with two options and choose the first option (Restart) by entering number 1.

Precondition: The player must enter nothing but 1 in the menu3.

Test steps:

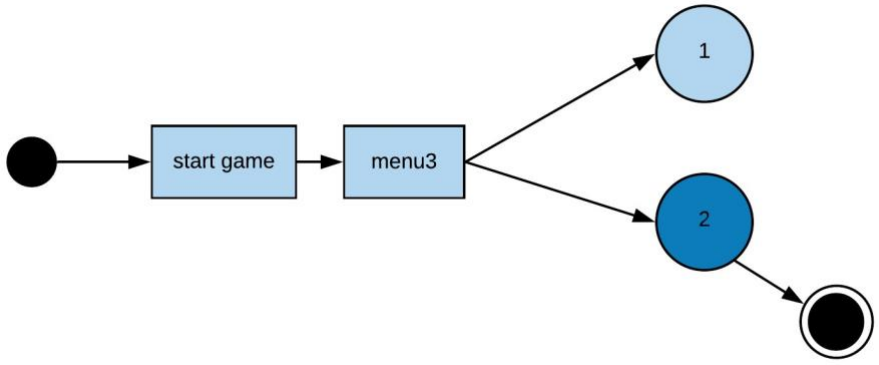
- Tester starts the game.
- User will enter a name to continue.
- User will be directed to choose a level menu (menu1).
- The tester will choose a level from menu 1 (1. easy, 2. medium, 3. hard).
- The tester will be directed to menu 2 to choose word categories (3. Name of countries, 4.Car brands).
- The user will guess the missing letters.
- The user will be directed to the menu 3.
- User will enter 1 in keyboard to restart the game.

Expected

- The system will direct the user to the menu 1.

TC4.2 User quit the game

Scenario: The player chooses the second option by pressing 2 (Restart) in the menu3.



The main scenario of UC4.2 is tested where a user has been faced with a menu (Menu3) with two options and choose the second option (Quit) by entering number 2.

Precondition: The player must enter nothing but 2 in the menu3.

Test steps:

- Tester starts the game.
- User will enter a name to continue.
- User will be directed to choose a level menu (menu1).
- The tester will choose a level from menu 1 (1. easy, 2. medium, 3. hard).
- The tester will be directed to menu 2 to choose word categories (3. Name of countries, 4. Car brands).
- The user will guess the missing letters.
- The user will be directed to the menu 3.
- User will enter 2 in the keyboard to quit the game.

Expected

- The system will show a message “Thank you and Goodbye”.
- Program ends

Test report -TC 1.1, TC1.2

Test traceability matrix and success

Test	UC1
TC1.1	1/OK
TC1.2	1/OK
Coverage and success	2/OK

Test	UC2
TC2.1	1/OK
TC2.2	1/OK
TC2.3	1/OK
TC2.4	1/OK
Coverage and success	4/OK

Test	UC3
TC3.1	1/OK
TC3.2	1/OK
TC3.3	1/OK
Coverage and success	3/OK

Test	UC4
TC4.1	1/OK
TC4.2	1/OK
Coverage and success	2/OK

Automated unit test coverage and success

Test	Main	Hangman Test	**Hangman**
Username Test	0	0	100%OK
Coverage and success	0/NA	0/NA	100%OK

All the tests successfully passed.

Test plan and unit test cases

```
    @Test
    void testvalidInputToEnd_Valid() {
        // it shoud provides or gives postive numbers, so it will show you the test
        // without any error
        assertEquals(true, man.validInteger(1));
        assertEquals(true, man.validInteger(5));
        assertEquals(true, man.validInteger(43));
    }
//  This is short test to show you Hangman test the correct test and UserName Test short test

    @Test
    void testvalidInputToEnd_Valid1() {
        // user shoud provides or gives postive numbers that why it gives(returns) false in the test or even when we change from true to fals
        assertEquals(false, man.validInteger(-1));
        assertEquals(false, man.validInteger(-5));
        assertEquals(false, man.validInteger(-43));
    }

    @Test
    void testEnterValidName_Valid() { // this is for right right input user name , english letter or integers
        assertEquals(true, man.enterValidName("talha"));
        assertEquals(true, man.enterValidName("jack"));
        assertEquals(true, man.enterValidName("mora"));
    }
```

```
@Test
void testEnterValidName_Valid() { // this is for right right input user name , english letter or integers
    assertEquals(true, man.enterValidName("talha"));
    assertEquals(true, man.enterValidName("jack"));
    assertEquals(true, man.enterValidName("mora"));
}

@Test void testEnterValidName_NotValid() {
    assertEquals(true, Hangman.provideValidName("0'*==/)"));
    assertEquals(true, Hangman.provideValidName("+4K"));
    assertEquals(true, Hangman.provideValidName("-.*"));
    assertEquals(true,Hangman.provideValidName("!#&.+"));
}
```

Reflection

In this iteration, 4 use cases have been used to write a manual test case, the results were promising since the tests passed. However, I have written 12 test use cases.
Regarding the automated test case in my program, there are 12 tests in total which all will pass except one of them will fail the test method called Test That Fail, so I did not give the tester choice to write any name for example tester can provide a name like My moon then the test will pass, but if tester writes name Bela as I did in this iteration then the test will fail.

I have spent much time on this iteration to come up with the test that fails on purpose. In the beginning, my idea about this test was to make the user write the name only in English but now I updated my documentation and codes.
This iteration is the hardest part, but I have improved my skills at writing tests in Junit and from this iteration, a tester can make sure either the methods are going to work or not.