GREETER APPLICATION

Assignment 3

MARCH 7, 2019

SANA HAMEED
Sh223nw@student.lnu.se

The Greeter application

This application is designed for hangman game. In this application I will briefly describe the use case scenarios of my game. This is the simplest application and help the user how to play this game step by step and we also test all the modules either all modules are working correctly or not. This game have basic idea for guessing words if user guess wrong word the application shows an error if user proceed with correct words then user got score.

Vision

Our application is user friendly application. On our platform we provide a user friendly interface in which user enjoy and play game with interactively. Our application also provide some hints which is helpful for the user and user enjoying the game with the winning game. In the game we have some other options which is also helpful for the user.

Use Cases

UC 1:

Start a new game.

Precondition:

1. System support the game configuration
2. The file has been run and game screen appeared.

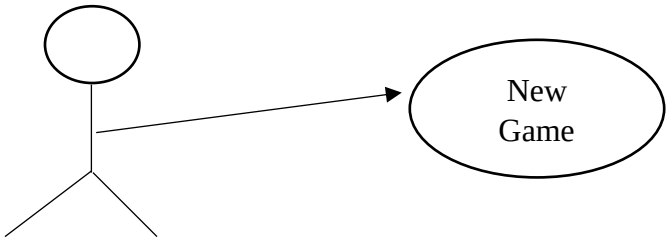
Main Scenario:

1. Go to the new game button and click on it.
2. New game is loaded on system.

Alternative Scenario:

Press start new game button and play a game.

Activity Diagram:



UC 2:

Resume Game

Precondition:

1. Game was played before.
2. Game support to have a check point to start from.

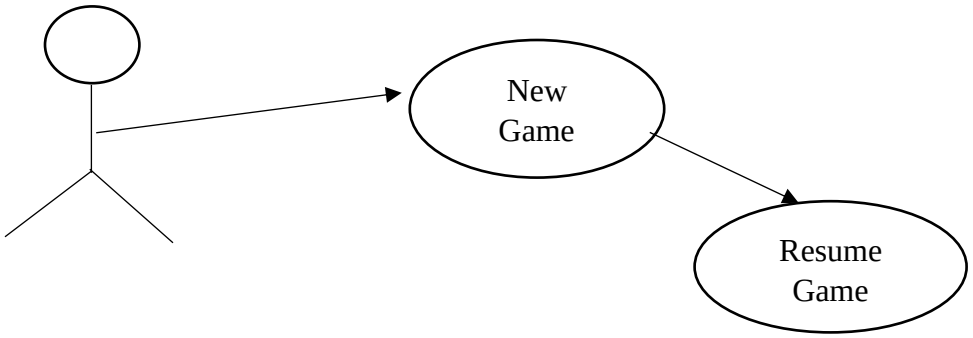
Main Scenario:

1. Go to the Resume game button and click on it.
2. Game is loaded on system
   from the previous stop point.

Alternative Scenario:

Press the Resume button and start your game from the same place where you stop last time.

Activity Diagram:



UC 3:

Select Level

Precondition:

1. Required level has been
unblocked.
2. Game supports loading level.

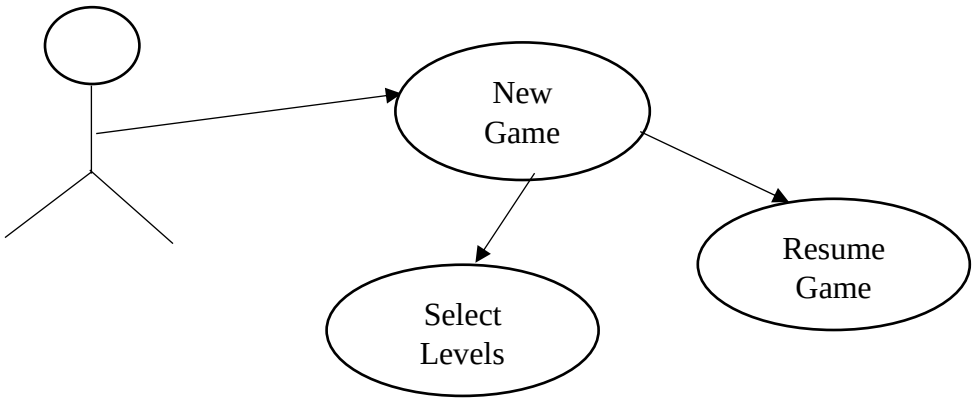
Main Scenario:

1. Go to the new game and click to the select level.
2. Select level and load the select level.

Alternative Scenario:

Click on select level and directly select the level.

Activity Diagram:



UC 4:

Exit Game

Precondition:

A game level is being played.

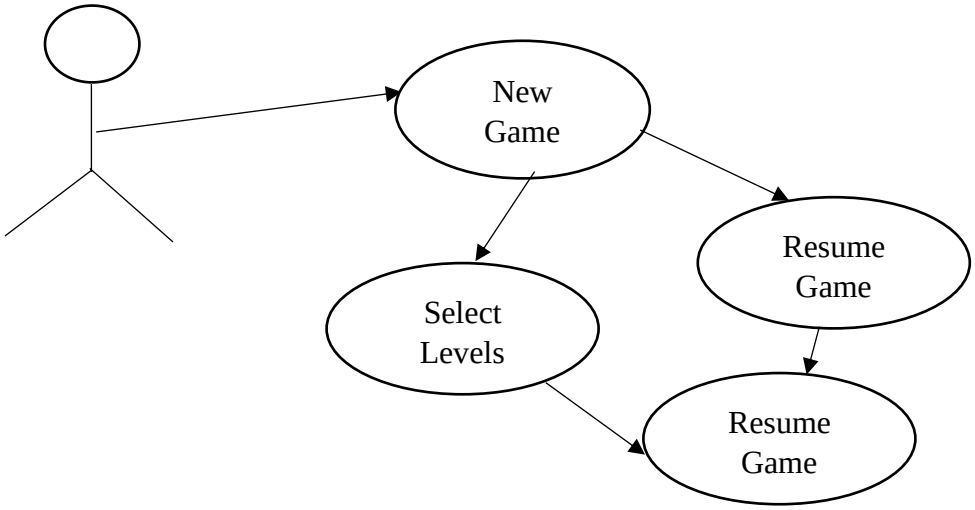
Main Scenario:

1. If user guess wrong words more than 3 time.
2. Game is automatically come in start menu.

Alternative Scenario:

Click on exit game.

Activity Diagram:



Test Plan:

Objective:

The objective of testing is to test all the small parts of the game that are free of errors or not.

Time Plan:

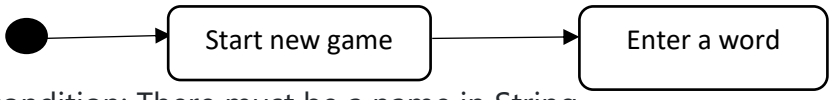
Task	Estimated	Actual
Manual TC	1h 30m	1h
Unit TC	1h	45m
Running manual tests	25m	5m
Test Inspection	2h	1h 38m
Test report	1h	50m

Manual Test cases:

TC1: Enter a word.

Use case: UC1 Enter a word.

Scenario: Enter word successfully.
The main scenario of UC1 is tested where a user Enter a word successfully.



Precondition: There must be a name in String.

Test steps

- Start the game.
- System shows "Enter a word"
- Enter the word "Sweden" and press enter.

Expected

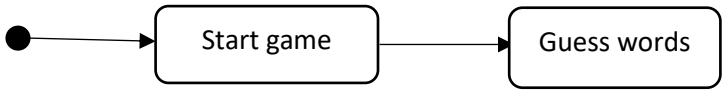
- The system should show the text "Word Sweden Entered"

```
import static org.junit.Assert.*;

import org.junit.Test;

public class EnterLetterTest {

	@Test
	public void EnterLetterTest() {
		Hangman test = new Hangman();
		String words = "Rakesh";
		char[] enteredLetters = new char[2];
		System.out.print("Testing word is:-"+words+" and any
letter of this word \n");
		int result = test.enterLetter(words, enteredLetters);
		assertEquals(1,result);
	}

}
```

TC1.2 Guess missing characters from word

Use case: UC1 Enter a word

Scenario: Store word. The alternate scenario where the user enters an empty word and is forced to guess a new username.



Precondition: There must be a word in String.

Test steps

- Start the game.
- System shows "guess the missing characters"
- press enter after guessing words.

Expected

- System shows "Enter guessing word" and waits for input
- The system should show the text "A word must have character, try again"

```
import static org.junit.Assert.*;

import org.junit.Test;

public class FindEmptyPositionTest {

	@Test
	public void FindEmptyPositionTest() {
		Hangman test = new Hangman();
		char[] enteredLetters = new char[5];
		enteredLetters[0] = 'r';
		enteredLetters[2] = 'k';
		enteredLetters[3] = 'e';
		enteredLetters[4] = 's';
		int result = test.findEmptyPosition(enteredLetters);
		assertEquals(1,result);
	}
}
```

Test Report:

Test	UC1	UC2
TC1.1	OK	OK
TC1.2	OK	OK
Coverage &Success	OK	OK

Automated unit test coverage and success

Test	Console view	Main	Greet controller	Name DAL
Enter word test	**0% error**	**0% error**	**0% error**	**100% ok**
Guessing word test	**0% error**	**0% error**	**0% error**	**100% ok**