Project Name:Hangman

Organisation: Linnaeus university

Project plan

Name: Sundarakrishnan Ganesh

LNU ID: sg222wn@student.lnu.se

Github Repo:

https://github.com/sg222wn/Java-Projects

1. Revision History

Date	*Version*	*Description*	*Author*
10-02-2019	1.0	Vision, Project plan, Iteration 1, Basic Version Of the Game	Sundarakrishnan Ganesh
12-04-2019	2.0	UML diagrams, Modelling, Software Design	Sundarakrishnan Ganesh
16-04-2019	2.0	Vision, Use-case, Test plan, Manual and Unit testing, Reflection	Sundarakrishnan Ganesh
22-04-2019	3.0	Bugs fixed	Sundarakrishnan Ganesh
16-08-2019	3.5	Final product	Sundarakrishnan Ganesh

2. Brief Information

Project Summary	
Project Name	**Project ID**
Hangman	sg222wn_1DV600
Project Manager	**Main Client**
Sundarakrishnan Ganesh	The Gaming community and the programmers
Key Stakeholders	
Customer Project Manager and the End-users	
Executive Summary	
Hangman is a fun guessing game which can be played by one or more players. The player has limited number of chances to guess a word and if the user guesses the correct letter the number of chances remains the same and for every incorrect guess the chances decrease by one. Finally, the user will be greeted upon winning or losing. The purpose of developing this game is to become more familiar with Java. It is a great way to learn programming and we learn new ways of coding all through the process. Testing is indeed a great tool when we work with such projects as it is not only important to deliver the product to the user but also make sure that it is tested and flawless.	

3. Vision

Application to be completed: Hangman game

In the given project, we are required to create a game called “Hangman”. Hangman is a game which started out as a paper and pencil guessing game and eventually made its way to the virtual world. The motive of this game is to make enhance the players’ knowledge in different fields. The game can be played by one or more players. The player has to guess a word based on a given clue and the length of the word. The word or the piece of text to be guessed is in the form of empty blanks or special characters like *, _, etc. If the user guesses the correct letter the number of chances remains the same and the special characters are replaced by letters at its respective positions in the word. The user can guess the letter irrespective of the case. If the player guesses a letter of the word correctly then he progresses further in the game. If he guesses a character wrong then he loses a life. The ultimate vision of the game is to enable the player to know more words in English., so he can apply them in his real life, thus increasing the player’s knowledge in those fields. For example, the word to be guessed is “Hello” and the number of chances the player has is twelve. The word is shown as_____ _. If the player’s first guess is ‘l’. Then it is displayed as _ _ a _ _ a _ and the number of chances is still eight. The other hangman games just give the user certain words to guess which won’t be of much curiosity to the user. This hangman game is a really simpler one. In this game the player has to guess the names of car manufacturers. The players would be allowed to guess the name of few famous car manufacturers.

Reflection: Vision helps the programmer or the developer to stick to his ideas. It was easy for me to understand and know the features that need to be used in the project and the one’s that might be time consuming and unnecessary. The iterations helped develop the game bit by bit.In general, I think that vision can also be varying from time to time depending of the customer requirements and it is a useful tool to help the developer stick to the user requirements and do the appropriate design and test the code. Through the vision, I implemented the idea of difficulty level as well as the high score.

4. Project plan

The main aim of this assignment is to complete the final iteration of the Hangman game and to deliver the finished game without bugs. Assignment 1 was

submitted on 8 February 2019. The assignment is based on creating a game called Hangman. The Software Technology course has three themes. All themes have an assignment attached to it. We will be using the concepts from each theme to complete its respective assignment. Theme 1 is based on Process and Planning. This will be used as a part of the assignment 1. In assignment 1 we should create a documentation of the project and add the first iteration which is to implement an idea to develop the game and some skeleton code for the project to work with.

Then in the assignment 2 which had the deadline on 21 Feb 2019, I have updated my code to JavaFX and also the documentations. State Charts, UML diagrams, Use cases and State Machine diagrams have been made which increased the understanding I had on this project and also simultaneously making it more easier to code certain parts of the program.The assignment 3 which had the deadline on the 8th of March 2019, I tested all the important parts of my code both manually and automated some tests as well.

The assignment 4 is the final iteration which consists of some tests in the final version of the game and also a slight code update. This iteration is mainly focussed on the documentation of whatever we did throughout the process of making this game.

Regarding the game JavaFx has been used to deliver a visually pleasing gaming experience for the user.

The stakeholders are also an important part of this game, they are responsible for informing the user about adding any new features to the game and above all giving constructive feedback which plays a vital role in the process. The overall project is based on the theme of Process and Planning and focussing on documentation. The documentation needs to be completed first so that implementation goals are met.

Reflection: Planning a project like this in advanced could be really useful as we just have to follow the plan in the future while actually working on the project. The project plan is really useful incase someone else wants to design the same game later in the future but also doesn't want to waste a lot of time thinking about the basic logic. The project plan is also very organised which leads to less confusion in selecting tasks during the course of the project.

4.1. Introduction

Hangman is basically a game that allows the player to guess a word based in the given hint. If the user guesses the word in less than a specific amount of trials then the player saves a man from getting hanged. If the player is unable to guess the word then he loses and the man is prosecuted.

4.2. Justification

The project deals with the usage of different themes for different assignments. The theme of Process and Planning will be practically implemented in this assignment which can enhance the learning experience.
The code used is very important to determine the working of the game. The idea is to learn to make high quality software with the course.

4.3. Stakeholders

User*:* Plays the Hangman game designed by the developer. The user is given a limited number of chances to win the game.
Developer*:* Designs the game by constructing the code. Tests the functionality of the code and can adds features like user registration, time limit etc, as well as remove them.

4.4. Resources

The resources available to complete the development process are:

4.4.1. Man- Power: 1 person, about a month to develop using eclipse which runs JAVAFX.

Role: Implementer

Worker: 1

Responsibility: Everything

4.4.2. Tools

JDK version 11.0.1 and Eclipse is used to construct the code and compile it. Study material from MyMoodle.

4.5. Hard- and Software Requirements

4.5.1 Hardware

A personal computer or laptop with at least 2GB Ram, Intel pentium, 50-60MB of HDD space should run this program with ease.

4.5.2. Software

Eclipse, Java Version 1.8.0_181 or higher.

4.6. Overall Project Schedule

The deadline for assignment 1 is 8 February 2019.

The deadline for assignment 2 is 21 February 2019.

The deadline for assignment 3 is 8 March 2019.

4.7. Scope, Constraints and Assumptions

This project plan applies to achieve the following requirements:

1. Construct a by game using optimized graphics that would run in almost all computers.
2. The theme of Process and Planning will be practically implemented in this assignment which can enhance the learning experience.
3. The idea is to learn to make high quality software with the course.

5. Iterations

This heading consists of various iterations done throughout the course of this project. ***Iteration Planning:*** This project is done through various iterations. So basically the plan for the first iteration is to make a simple hangman game with simple logic, that runs in the console, without any user interface. The second iteration consists of making some changes to the game by making it into JAVAFx with some basic gameplay and also making various diagrams in making it. The third iteration is saved for testing, which is one of the more important steps in product development. The final iteration will have the completed user playable game, with complete documentation.

5.1. Iteration 1

The first iteration is this project plan along with some degree of implementation. The documentation should be completed first so that the implementation goals are met in code. An idea and some skeleton code need to be implemented for the project to work with. This is assignment one. In this first implementation, the basic Hangman game is implemented using a word that is defined and the result is displayed in Eclipse. The word defined was “Hello” and the user had to guess that word, which if he did correctly would enable him to win the game.
Reflection: Writing a basic algorithm for the actual game in JAVA helped very well to actually get an idea of how the actual game would turn out in the final iteration. It would also be easier to divide the code into different functions in the final code which will lead to the code looking more neat.

5.2. Iteration 2

This iteration focuses more on developing the game in JAVAFx with a playable user interface. So the code was written in JAVAFx with simple logic. There were made 4 main classes for each window in the game. The Main Menu class contains the JAVAFx code for the Main Menu, the Difficulty menu class consists of code that allows the user to choose difficulty for the game. The Instructions class consists of a window that will open instructions that will tell you how to play the game, in a user friendly way. This iteration also includes drawing various UML diagrams for the sake of better understanding.

5.3. Iteration 3

This iteration mainly focuses on testing the various essential parts of code. Testing includes testing manually for the JAVAFx parts of the code and also some automated JUnit tests for testing the methods in the program. A test report will be generated and required changes will be made to the code to achieve complete test success.

Reflection: By testing this version of the source code, I am now able to point out where the errors are in my code. Testing the essential parts of the source code has made it clear that there were errors that I was later able to fix and make the program run with less glitches or bugs. Now it has occurred to me of how vital a role testing plays in various stages of code development. Manually testing the interface from the client's (end users') perspective have also helped in increasing the user friendliness of the hangman game, by handling exceptions in a way that the layman understands.

Estimating the time duration for each process has increased the accuracy of predicting the time in which a process can be done. Thus to sum it up, this assignment has taught me how important testing a product really is.

5.4. Iteration 4

Note: TO experience the complete game run MainMenu.java and work your way through.

This is the final iteration which is the completed product. This version of the code contains no errors and fulfills all the Use-Cases that were mentioned before. The final version of the game contains a Main Menu (with Start, instructions and quit), a Difficulty selection menu, and the actual completed gameplay screen, where you type a letter in the text box and it shows if that letter is part of the word to be guessed or not. Thus is the final iteration.

The first step in making this game was the working out the basic logic that would compile in the console without the user interface with features like Scanner for user input. Once you are done with the logic, you can start coding the user interface using javafx or scenebuilder or any program of your choice. (I used JAVAFX). Coding the logic in the JAVAFx is the next step. Usage of certain functions is recommended as it will make your code less messier. Then you can

work coding the various menus in the game which will use user interaction. Thus you link the menus and the main game(HangManMain). Now would be the time to start testing your program. It is recommended to both manually and automatically test your code. Once you're done testing you can start to handle errors such that they are displayed in a way that the user is able to understand. Thus is the complete instructions of creating the game using JAVAFx required for the final iteration. I also had to change the Class Diagram a little bit. The unit tests also had different speeds which were taken screenshots and updated in the TestScreenshots folder in github.

Manual Test Cases:

Test Case - 1 (Pass)

This test case will focus on manually testing the Select Difficulty menu of the program. This is tested because it decides the number of lives allocated regarding your difficulty selection and also the further development of the game.

1. Name : Testing Quit game
2. ID : TC_HM_1
3. Reference: UC 5 Quit Game
4. The quit game option is manually tested here to check if the game exits correctly on request.
5. Preconditions for this TestCase: Quit Game button in the Main Menu is clicked and the confirmation menu is shown.
6. ***Test Steps***:

a) Click on the "**OK**" button. (or)
b) Click on the "**Cancel**" button.

7. ***Expected Results***: If the OK button is clicked the control comes out of the game and the session is exited. If the cancel button is clicked then the control goes back into the game.
8. ***Actual Results***:

Clicking OK test(Game Session
Ended) Clicking Cancel(Game
Session Resumes) 9.***Comments:***
a)Nothing peculiar noted during the test. It worked perfectly.

Test Case - 2(Pass)

This test case will focus on manually testing the Start Game button in the Main Menu of the program. This is tested because this is the most important thing in the game. This provides the path for the user to enter the game and play it.

1. Name: Testing Start Game
2. ID: TC_HM_2
3. Reference: UC 1 Start Game
4. The start game option is manually tested here to check if the control enters the game correctly from the Main Menu.
5. Preconditions for this test case: Start Game button in the Main menu is clicked.

6. Test Steps:

a) Click on the Start Game button in the Main Menu.

7. Expected Results: If the Start Game button is clicked the user gets redirected to a Difficulty selection Menu.

8. Actual Results:

Clicking the Start Button(Difficulty Menu is shown with 3 difficulty options)

9. Comments:

a) The test was successful without any error.

Test Case - 3(Pass)

This test case will focus on manually testing the Reload button in the Gameplay Screen of the program. This is tested because it is fairly important to reload a new session after a previous session ended.

1. Name: Testing Reload Button

2. ID: TC_HM_3

3. Reference: UC 6 Reload Game

4. The reload game option is manually tested here to check if the game creates a new session of the game without any errors.

5. Preconditions for this test case: The user must be playing the game.

6. Test Steps:

a) Click on the Reload button in the Gameplay Screen.

7. Expected Results: If the Reload button is clicked the user gets redirected to a new session of the game.

8. Actual Results:

Clicking the Reload Button(The word to be guessed changed)

9. Comments:

b) The test was successful without any error.

Test Case - 4(Pass)

This test case will focus on manually testing the Difficulty selection button in the Gameplay Screen of the program. This is tested because it is fairly important to set the lives remaining for the user.

1. Name: Testing Each Difficulty

2. ID: TC_HM_4

3. Reference: UC 2 Select Difficulty

4. The Difficulty buttons are manually tested here to check if the control enters the game with the correct lives corresponding to the difficulty selected.

5. Preconditions for this test case: The user must click start game.

6. Test Steps:

a) Click on the Noob button in the difficulty menu.

b) Click on the Slightly Average button in the difficulty menu.

c) Click on the Pro button in the difficulty menu.

7. Expected Results: If the noob button is clicked then the control should enter the game with 12 lives and if the Slightly average button is clicked then the control should enter the game with 9 lives and finally if the pro button is

clicked then the control should enter the game with 6 lives

8. Actual Results:

Click on the Noob button in the difficulty menu.(Control enters game with 12 lives)

Click on the Slightly Average button in the difficulty menu.(Control enters game with 9 lives)

Click on the Pro button in the difficulty menu.(Control enters game with 6 lives)

9. Comments:

The test was successful without any error.

Reflection: By making the final iteration of the source code I was able to handle some errors in the code, I was able to make an Instructions screen which was in one of the Use-cases which I forgot to implement. This version of the game has all the

Use-cases fulfilled now. I was able to update the project plan, format the code to make it look more neat and adding comments for reusing the code later in the future. I tested the previous parts just in case, in this iteration to see if something was wrong. I am now sure that the product is complete and it delivers what was promised.

6. Risk Analysis

Product designing always comes with risks in various forms. Considering this project in particular, the risks associated are mainly project risks as there isn't a budget or business in this project nor a team. So some risks to take into account are, the accidental loss of JAVA code, loss of internet connection etc.

1. I might fall sick during the course of the project which may lead to certain delays in actual working time of the project.

• **Effect**: The assignment might become a failure and I would not get the expected grade.

• **Occurrence level**: Medium

• **Solution**: Stick with the deadlines and plan the work.

2.It is difficult to handle all the errors thrown in the program.

- **Effect**: I might overlook a few errors in the code and my code might lack transformation.

• **Occurrence level**: Medium

• **Solution**: Handle the most repeating or the most important errors.

3. I am not very comfortable in JavaFx.

• **Effect**: It might be time consuming to learn the concepts of JavaFx and create a proper user interface for the game.

• **Occurrence level**: High

• **Solution**: Although it is not compulsory to make the project in JAVAFX, I personally thought to actually learn and work in JAVAFX to make a neat user interface that would actually make the user to understand what's going on. The deadline for the course assignments must be considered when planning the study schedule for JAVAFx.

4. What if I accidentally delete the Java code files.

• **Effect**: As all components are of equally high priority, it would take a bit of time to re-code the component and merge it again with the main program.

• **Occurrence level**: Highly unlikely

• **Solution**: I must be more careful if I were to set up a disk cleanup sometime. I would also backup the files to an online cloud service like Google Drive,.

7. Time Log

The time log contains the date, time and the task to be performed. It covers the overall time taken to learn and understand a problem as well as plan, implement and reflect a specific task.

Iteration 1:

Estimated time for writing the Vision: 1
hour Actual time: almost 1;45 hours
Estimated time for planning the basic logic(pen and paper): 10

mins Actual time: 15 mins

Estimated time for coding the logic into JAVA: 15

mins Actual time: 5 mins

Estimated time for writing the Project plan : 2 hours

Actual time: almost 1:15 hours

Iteration 2:

Estimated time for making the layout for main menu in JavaFx : 30

mins Time consumed : 1 hour(6:30Pm - 7:30Pm)

* The main menu has two options as of now, which is the Play game and quit game. So aligning the buttons in the main menu and adjusting the scale and color of the text in the button took a while. And the main menu was filled with a background image.

Thus it took almost an hour to completely design the Main menu. The formatting of the buttons is what took most time.

Estimated time for making the layout for the Difficulty menu in JavaFX: 45

mins Time consumed: 30 mins(7Pm- 7:30Pm approx..)

* The difficulty menu has 4 functions which are linked with the buttons(Noob, Slightly Average, Professional and Go back). This menu didn't need much of the button formatting so it was complete considerably early.

Estimated time for making the basic game layout in JavaFx : 3

hours Time consumed : 4 hours(8Am - 12Pm approx…)

* The basic game layout in JAVAFx took almost 4 hours because it had much logic and connecting those logics to the JAVAFx components were a bit difficult. Handling exceptions also took a considerable amount of time in this process. Thus is the detailed time consumption for making the layout for the game.

Estimated time for writing the usecases: 1 and half

hours Time consumed : 1 hour(10PM - 11PM)

* Thinking of the use-cases from the user's point of view took a bit of time in this task. Writing the collected use-cases was done in almost 20 mins.

Estimated time for making the UseCase diagrams: Half an

hour Time consumed : 15 mins(10AM - 10:15AM approx..)

* Drawing the required use-case diagrams using the online tool Lucidchart,

took about 15 mins because it's basically the use-cases made into the diagram.
Estimated time for making the State machine diagram : 45
mins Time consumed : 30 mins(12 Pm to 12:30 PM approx..)

* This task again required thinking from the user's point of view too and was also a considerably a shorter task.

Estimated time for making the State Chart: 10 Mins

Time consumed : Almost 15 mins(1PM to 1:15 approx…)

* This task also had the almost same requirements so was not that time consuming in the process.

Estimated time for making the Class diagram : 1
hour Time consumed : 1 and a half hours(6PM to
7:30 PM)

* This task required additional knowledge about drawing class diagrams(for example. Which arrow to use to denote inheritance, association,etc..) which kind of confused me a bit at first and then I got the hold of it. And basically collection of information about the class(the constant switching back and forth between Eclipse and Word Doc) was the major part. But if that confusion about the terminologies regarding how to draw the class diagrams hadnt been there, this task wouldve been finished a bit earlier than it has now.

Iteration 3:

Estimated time for completing the testing plan: 6 hours

Actual time taken: 5 hours and 19 mins(2 PM - 7:19 PM approx.)

Estimated time for completing the manual test cases: 2 hours

Actual time taken: 48 mins(12:30 PM - 1:18PM approx.) Estimated time for completing the manual tests: 10 mins

Actual time taken: 25 mins(2:30PM - 2:55PM)

Estimated time for completing the code for JUnit tests: 30 mins

Actual time taken: 1 hours and 30 mins(5PM -

6:30PM)

Estimated time for completing the JUnit report: 1 hour

Actual time taken: Around 30 mins(7:50-8:20) **Iteration 4:**

Estimated time for completing the final manual tests: half an hour

Actual time taken: 25 mins(12:30 PM - 12:55PM approx.)

Estimated time for completing the code for JUnit tests: 30 mins

Actual time taken: 15mins (5PM - 5:15PM)

This is mostly because almost all the components were tested in the previous iterations and I just had to test them again with different values.

Estimated time for completing the documentation: 6 hours Actual time taken: 5 hours (6PM - 11PM approx.)

Estimated time for completing the complete instructions through the four iterations: 1 hour

Actual time taken: 30 mins(5:30 PM - 6PM approx.)

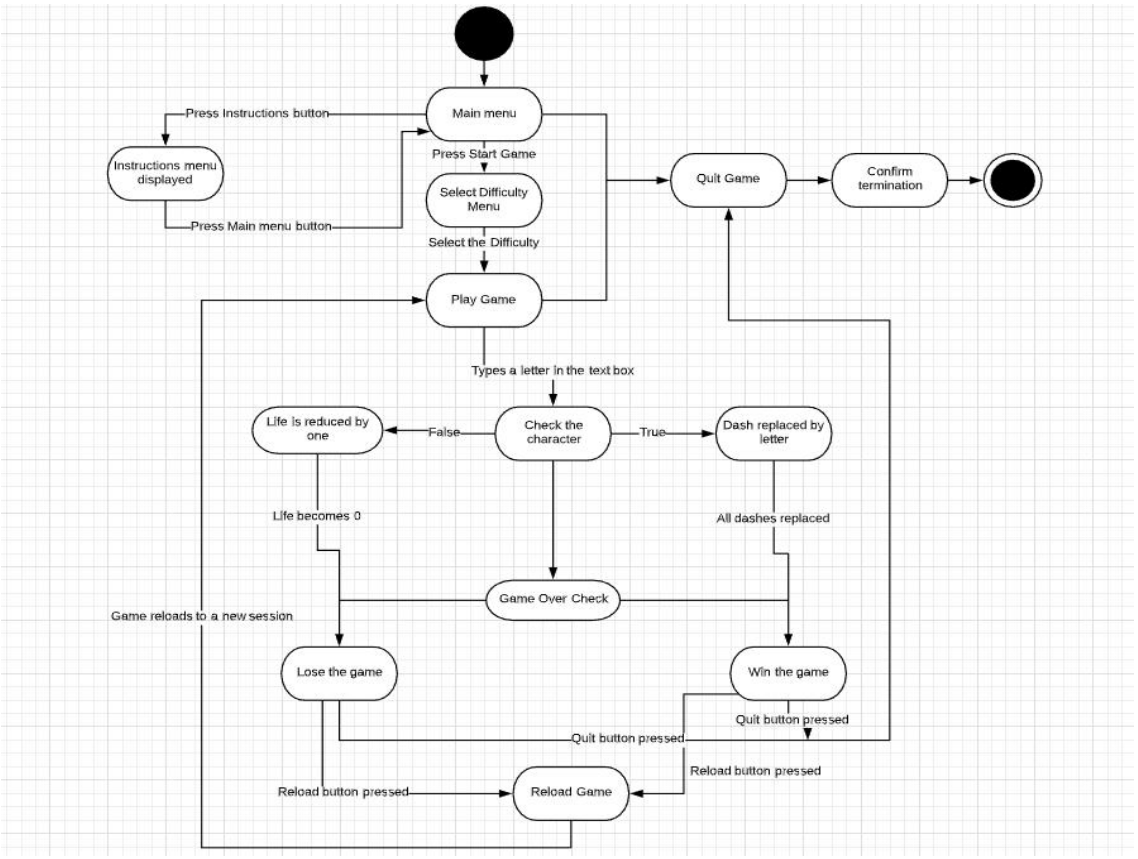
Estimated time for completing the code for the instructions: 15mins

Actual time taken: 10 mins (5:15PM - 5:20PM approx.)

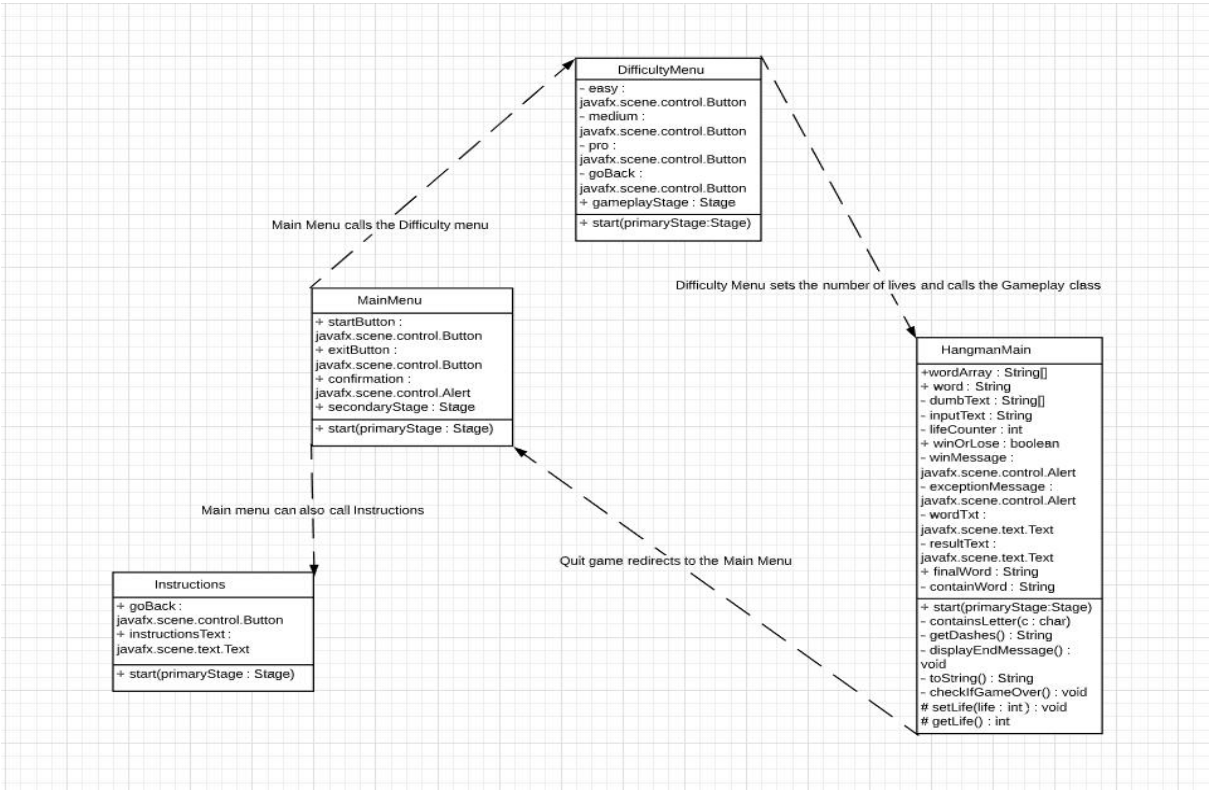
Estimated time for updating the class diagram: 10 mins

Actual time taken: 5 mins(5:20PM - 5:25PM approx.)

9. State Machine Diagram:



10. Class Diagram:



11. Use Case Diagram: