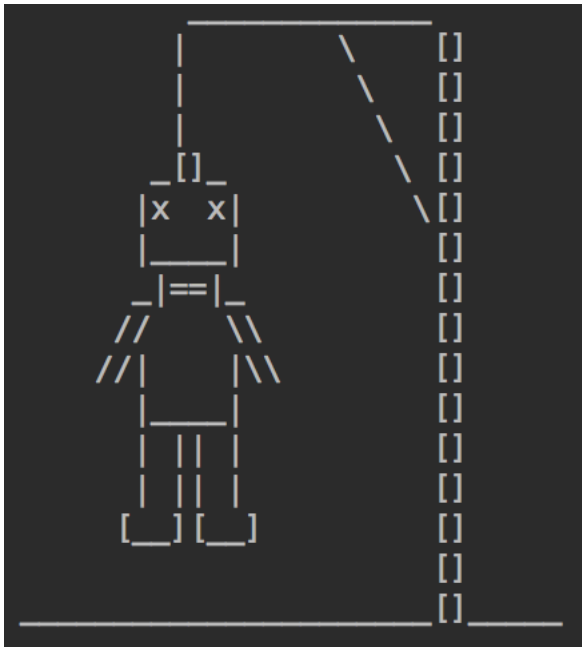
1DV600 - The Hangman Project



Melina Cirverius
Linneaus University

2019-03-22

Contents

1. Revision History _____ **4**

2. General Information _____ **5**

3. Vision _____ **6**

3.1. Vision
3.2. Reflection – Vision

4. Project Plan _____ **7**

4.1. Introduction
4.2. Justification
4.3. Stakeholders
4.4. Resources
4.5. Hard- and Software Requirements
4.6. Overall Project Schedule
4.7. Scope, Constraints and Assumptions
4.8. Reflection – Vision

5. Iterations _____ **10**

5.1. Iteration 1 10
5.2. Iteration 2 11
5.3. Iteration 3 13
5.4. Iteration 4 15

6. Risk Analysis _____ **17**

6.1. List of risks
6.2. Strategies

7. Time log _____ **18**

7.1. Iteration 1 18
7.2. Iteration 2 19
7.3. Iteration 3 20
7.4. Iteration 4 21

8. Use Case Diagram _____ **22**

9. Use Cases _____ **23**

9.1. UC 1 – Start Game 23
9.2. UC 2 – Set up Game 24

9.3. UC 3 – Play Game 25

9.4. UC 4 – Quit Game 27

10. Play Game State Machine 28

11. Class Diagram 29

12. Manual Test Cases 30

12.1. TC1.1 – Start a game successfully 30

12.2. TC2.1 – Set up a game successfully 31

12.3. TC3.1 – Guess a letter successfully 32

13. Automated Unit Tests 33

13.1. Method createUnderscores() 33

13.2. Method evaluateGuess() 34

13.3. Method drawPart() 35

13.4. Unit Tests Reflection 36

1 Revision History

Date	Version	Description	Author
2019-02-08	v.1.0	Finished first version of project documentation	Melina Cirverius
2019-02-19	v1.1	Updated 5.2 – Iteration 2, and 7.2 – Time log for iteration 2	Melina Cirverius
2019-03-08	v1.2	Updated 5.3 – Iteration 3, and 7.3 – Time log for iteration 3	Melina Cirverius
2019-03-10	v1.3	Added updated UML–diagrams and Use Cases	Melina Cirverius
2019-03-18	v1.3	Rewrote project summary and vision, added reflections to time logs for iterations 1-3	Melina Cirverius
2019-03-19	v1.3	Rewrote project plan and updated reflections	Melina Cirverius
2019-03-22	v1.3	Updated 5.4 – Iteration 4 and 7.4 – Time log for iteration 4	Melina Cirverius

2 General Information

Project summary	
Project Name	Project ID
The Hangman Project	mc222ap–1DV600
Project Manager	Main Client
Melina Cirverius	The Lego Enthusiastic Gamer
Key Stakeholder	
Project Manager, Development Team, End user	
Executive Summary	
This project is focused on creating a hangman game for Lego enthusiasts, by using Lego-related words for the guessing as well as a Lego-inspired figure being drawn out as the “hangman”. This game is being created to introduce a different aspect of the classic hangman game by using words from this one specific topic, and in that way target a certain group of gamers. The goal of this project is to create a fun and excited game that is easy to understand but challenging to play.	

3.1 Vision

The goal for this project is to create a Lego-inspired version of the hangman game. This will be a text based game. At the start of the game the player will be greeted by a menu where they can begin a new game. The game will choose a random word from a list each time the player starts a new round. As an extra feature the game will have the words divided into three different difficulties and have the option for the player to choose the difficulty level at the start of the game. An extension of this feature will be the three different list all containing words related to Lego builds and movies, with different guessing difficulty determined by the length of the word as well as the letters present.

The player will be able to give a user name to the game and choose one of the difficulty levels. The game will then present a word from one of the difficulty levels. We want to game to keep the player updated with how may guesses they have and also what letters they have already guessed to give a better overview of where they are at. As the player progresses through the game the goal is for the word to be constantly updated with the correct guesses, and the hangman figure to be drawn out part by part as the player gets letters wrong.

As well as the words being inspired by our target group of Lego–enthusiasts, the hangman figure being shown in the game will also be in the style of Lego. The goal is to create a brick inspired figure instead of the more traditional stick figure to keep in the tone of the game.

The game continues either until the player is out of guesses and the whole hangman is drawn, or the player has guessed the whole word correctly. If the player manages to guess the whole word correctly they will get the option to either continue with a new random word or to end the game. If the player chooses to end the game they will be presented with a result screen where they can see how many words they have guessed correctly in a row, and the least number of guesses they needed to finish a word.

If the player fails to guess the word before the whole hangman has been drawn the game is over and they can start a new game. At this point a result screen will appear and show the player how many, if any, words that they guessed correctly. The result screen will also include a high score list.

3.2 Reflection - Vision

Writing the vision document took some time as it was it was difficult to define the outline of the project. In the end I found that it was easier to just described how I wanted the finished hangman game to work with the different aspects and extra features I want to be implemented in the game if time permits. Having this vision written down in a clear way will help the project to move forward in terms of planning the work to be done. It will also be a huge help when it comes to actually building the game and implement the different aspects of it.

4 Project plan

4.1 Introduction

This project consists of the creation of a hangman game specifically targeted to gamers who are interested in lego. This project plan details the different parts of the project.

4.2 Justification

The purpose of this project is to create a Lego-inspired hangman game for gamers, hardcore or casual, that are also Lego enthusiasts. As hangman is a game that has been around for some time this is a way to interest another group of people to play it be using specific words for them to guess. The goal is therefor to create a game that will be interesting for this particular target group both by the words being used and the visual of the hangman figure.

This project is also an opportunity for the project manager to learn how to properly execute a project according to a plan.

4.3 Stakeholders

The project manager wants to follow the set out plan and have a properly working product to present to the client on time. They also want to documentation to be accurate and up to date so that the process during this project is easy to follow.

The development team wants to create an exciting hangman game with well structured code and zero bugs. They also want to implement extra features as discussed in the vision, of difficulty levels using separate reference materials to generate words, as well as a hangman figure inspired by the target group.

The end user, in this case being the gamer that is also a Lego-fan, wants a fun and creative game to play with words related to their interest.

4.4 Resources

The main resource for this project is the time allocated for the project, which is just under two months with the team working on this half time during this period. The resource in terms of personnel is only one person, as there is a sole developer working on this project and they are also both the project manager and designer of the application.

The resources in terms of research are selected chapters in the book *Software Engineering* by Ian Somerville as well as recorded lectures on subjects that will be helpful to completing the planning and management stages of the project, as well as testing and UML–diagrams.

4.5 Hard and software requirements

The software developed in this project, as well as all documentation and research, will be made using a MacBook Pro with an Intel Core i5 processor. The application will be developed in IntelliJ IDEA CE 2018 and written in Java 1.8. For version control the developer is using Github. The unit tests will be created using JUnit5.

The UML–diagrams are being made using an online tool, and everything is being documented in a word processor.

4.6 Overall project schedule

This project has set deadlines and important dates shown below. The project is divided into four separate iterations each following the completion of the previous iteration and resulting in a completed application by the end deadline in March 2019.

The first iteration including this project plan and some skeleton code for the hangman game, has a deadline on 8 February. In this iteration the complete first version of the plan for the project is to be finished so that the team can get a clear focus on what is to be done during this project. The skeleton code being created should show the basic outline of the hangman game.

The second iteration will focus on modelling, UML and the completion of the main part of the game and has its deadline on 21 February. In this second iteration the team will create the use cases needed to understand what the future application will have in terms of functionality. They will also create UML–diagrams as visual representation for both use cases, states and classes in the application. After the use cases are done, the developer will implement the first working version of the hangman game.

The third iteration will include testing of the application and has a deadline of 8 March. In this iteration the focus will be on testing parts of the application in its current state, by creating both manual test cases and automated unit tests for different parts of the application. The team will also clearly document the testing process and results of the tests made.

The fourth iteration is the main deadline of the whole project. For this iteration the whole hangman game should be completed and handed in together with the completed documentation. This will also include any additional models or tests that has been made, as well as updated from previous iterations. This last and final deadline is 21 March.

4.7 Scope, constraints and assumptions

The scope of this project is to create a working hangman game. The game should start be presenting a menu for the gamer where they can chose to start a new game or quit the game. After this the game will ask the user for a user name and then to chose the difficulty level they want to play at.

The game will show the number of letters to be guessed in a randomly picked word for the list of words at the chosen difficulty level. The word will be displayed as underscores, and the game will also display the number of guesses available. The player can only guess one letter at the time and if the letter guessed is correct, the word will update to show where in the word this letter is situated. If the letter is present at more than one place in the word the game will show all occurrences of that letter. The player then continues by guessing again with a new letter.

If the player's guess is wrong, part of the hangman will be drawn and then they get to guess again. The number of guesses available for the player is determined by the number of parts present in the hangman, which will be ten.

The game continues either until the player is out of guesses and the whole hangman is drawn, or the player has guessed the whole word correctly. If the player manages to guess the whole word correctly they will get the option to either start a new game with a new word, or to end the game. If the player chooses to start a new game they will again be asked at what difficulty level they want the next word to be.

If the gamer loses the game by running out of guesses, the whole hangman will be presented and they will be asked if they want to start a new game. If they do, they will be taken back to the main menu to start over.

Out of scope in this project will be the option to keep track of how many rounds and be extension words the player has managed to guess correctly in a row. Due to time constraints this feature will not be able to be implemented and the player will not be presented with a result screen at the end of each round to show how many rounds they have completed. They will at the end of each round see how many guesses they used in that round, but not how that compared to previous rounds. Another feature that will be out of scope in this project is a high score list, as the development team will focus on implementing the difficulty levels and hangman figure.

The constraints of this project is that the game is created to run in a terminal or command line, which also constraint how visual the application will be. The text and hangman figure will be based on basic terminal text and characters, with no extended graphical user interface or animations.

Assumptions on the user is that they will be at least a casual gamer, even though even a new gamer should be able to understand how to play the game by the prompts given by the game. An assumption is that the gamer is an English speaker as both the game menus and prompts, as well as the words to guess are all written in English. Another assumption due to our target group is that the player has some knowledge of lego and words associated with lego builds and movies. Another assumption is that the player should be able to, by following the provided instructions in the readme-file given with the game, run the game either in a terminal on their computer with the jar-file or if they choose in an IDE.

4.8 Reflection - Project Plan

The first version of the project plan created was a bit to sparse in the details outlining this project. After gaining more understanding on the project as well as how to plan and manage it, the project plan was updated to include a more comprehensive description of the project and what related to it. Updating the project plan made it much easier to get a good overview of this project and the application created through it. In its current state this project plan gives a good explanation on what the goal of this project is and what is included in completing it.

5 Iterations

5.1 Iteration 1

The first iteration of this project include this project plan as well as some skeleton code for the application to show the general outline and design.

To even be able to begin with this project plan the first thing needed is research. This is of course needed to learn how to plan a project from the start and how to write the project plan, as well as to actually begin working on the project.

The next step to this iteration is to start working on the project plan documenting in as much detail as possible what this project includes and how to proceed. The detailed schedule should also be produced taking into account how much time both research and actual writing will take. When the project plan and planning this first iteration is done the next step is to write the skeleton code producing a rough draft of how the application will the organised. The first step to produce the application is to make up a good idea of how this application will look in the end and what features I want to be implemented.

Date planned	Task	Estimated time
2019-01-23	Watch lecture 1 / Introduction to project	2h
2019-01-23	Create Github repository	1h
2019-01-24	Read chapter 2 and 3	6h
2019-01-25	Watch lecture 2	2h
2019-01-26	Plan the time schedule for iteration 1	3h
2019-01-28	Read chapter 22 – Project Management	3h
2019-01-30	Participate in Q and A	2h
2019-01-31	Read chapter 23 – Project Planning	3h
2019-01-31	Watch lecture 3	2h
2019-01-31	Outline the project plan	5h
2019-01-31	Write the vision document	4h
2019-02-01	Complete first draft of the project plan	8h
2019-02-04	Start outlining skeleton code for the application	2h
2019-02-04	Update and finish time schedule	2h
2019-02-04	Write risk analysis	2h
2019-02-06	Finish skeleton code	1h
2019-02-06	Error check and finish up project plan	6h

Deadline for iteration 1: 8 February

5.2 Iteration 2

In the second iteration is time to model the game using UML and then to add features and start making a working game. Diagrams are also to be included in this project documentation and should be implemented in the way modelled.

For this iteration the first step is to create a use case model as well as writing the use cases. The use cases model will show the different use cases available and how they relate to the the gamer. The use cases should be fully dressed cases detailing how the game works.

Next up is to create the state machine for the play game use case. The state machine should clearly show how the game is intended to work. Even extra features that might not be implemented in the game during this iteration should be shown in the state machine.

A basic implementation of the hangman game will be created during this iteration, with extra features to be added in a later state. Lastly, a class diagram is to be created documenting the class structure of the implementation.

Read chapters 4, 5, 6, 7, 15 and 20 and watch lectures related to these subjects.

Date planned	Task	Estimated time
2019-02-06	Watch live lecture on theme 2	2h
2019-02-11	Plan iteration 2	1h
2019-02-13	Expand use case model	1h
2019-02-13	Write fully dressed use case for Play Game	1h
2019-02-13	Write additional use case for Set up Game	1h
2019-02-14	Create State Machine Diagram for Play Game	2h
2019-02-14	Implement functionality to start game and display a word to guess in Hangman game.	3h
2019-02-15	Read chapter 4	2h
2019-02-15	Read chapter 5	2h
2019-02-17	Read chapter 20	2h
2019-02-17	Watch Lecture 4 – Systems and Software Modeling	2h
2019-02-18	Read chapter 7	2h
2019-02-18	Watch Lecture 5 – Modeling with UML	2h
2019-02-18	Read chapter 6	2h
2019-02-20	Watch Lecture 6 – Software Architecture	2h
2019-02-20	Read chapter 15	2h
2019-02-20	Watch Lecture 7 – Software Design	2h
2019-02-20	Watch Lecture 8 – From Software Design to Implementation	2h

Date planned	Task	Estimated time
2019-02-20	Watch live lecture 2 on theme 2	2h
2019-02-21	Expand functionality in game by implementing updated word after guesses and end game scenarios.	8h
2019-02-21	Finish implementing functionality in Hangman to complete a working game.	10h
2019-02-21	Create Class Diagram	2h

Deadline for iteration 2: 22 February

5.3 Iteration 3

The third iteration is mainly focused around testing. In this iteration it is time to plan and execute tests of the application as well as documenting this process.
Read chapter 9 and watch lectures related to this subject.

Objective

The objective is to test some of the code for the hangman game that was implemented in the last iteration.

What to test and how

The plan is to test UC1 and UC3 by running dynamic manual test cases. These two use cases were chosen as UC1 is a short but important part of the application as it is the start of the whole hangman game. UC3 on the other hand is the largest use case and also the most important one as it is the one showing how the game is played.

Automated unit tests will also be written for the methods createUnderscores() and evaluateGuess() in the class Hangman. These methods will be tested with at least two tests each. The reason behind choosing these two methods is since they are a crucial part of how the game should be played according to the project plan. The first method makes sure that the word is displayed in the right way for the player to start guessing and the second is a core part of the game as it evaluates the player's guess be comparing letters in the word with the one guessed.

One more unit test will be created for an unfinished method, drawHangman(), that should succeed once that method has been correctly implemented, but will fail at this stage.

Date planned	Task	Estimated time
2019-02-25	Plan iteration/theme 3	30m
2019-02-25	Create a test plan	1h
2019-02-27	Read chapter 8	2h
2019-02-27	Watch live lecture/workshop on theme 3	2h
2019-02-27	Watch pre-recorded lecture 9	1h 45m
2019-03-01	Watch video on the "Greeter Example"	30m
2019-03-01	Create manual test case for Use case 2	2h
2019-03-01	Create manual test case for Use case 3	2h
2019-03-04	Run manual tests and record the results	2h
2019-03-04	Watch pre-recorded lecture 10	1h 45m
2019-03-04	Watch pre-recorded lecture 10.5	30m
2019-03-06	Watch live lecture/workshop on theme 3	2h
2019-03-07	Create automated unit tests for method createUnderscores()	4h
2019-03-07	Create automated unit tests for method evaluateGuess()	4h

Date planned	Task	Estimated time
2019-03-07	Create automated unit test for unfinished method drawHangman()	2h
2019-03-08	Run automated tests for method (1) and record the results	2h
2019-03-08	Run automated tests for method (2) and record the results	2h
2019-03-08	Run automated tests for method (3) and record the results	2h
2019-03-08	Write reflection	1h

Deadline for iteration 3: 8 March

5.4 Iteration 4

The fourth and final iteration is the completion of the game. In this iteration the project will be viewed as a whole, going through all the steps for any new features implemented into the game as well as make sure that all previous steps are completed and the game is working according to plan. This is also the time to look over the documentation and the planning made at the beginning of the project to see where it has been deviating from the plans and document the reasons for this is well as solutions.

For iteration four the focus will be on completing two features of the game, the first being implementing the drawing of the hangman when the player gets a guess wrong, and the second being implementing the different difficulty levels. In this iteration we will also update previous diagrams to correspond to the new structure of the application as well as new features added. Tests will also be updated and new test written, see the following test plan.

Test Plan for iteration 4

Objective

The objective is to adjust previous tests made to correspond with the finished application, as well as create new test for newly implemented features.

What to test and why

The plan is to now test UC2 with manual test as all functionality needed for UC2 will be implemented in this iteration, mainly the difficulty menu, therefor a manual test will be run to make sure this part of the game works as expected.

Unit test cases will be updated for the previously tested methods as these will be moved to the new class guessHandler() to increase structure in the game. They should still work the same way and succeed when updated. New unit tests will be created for the Hangman class that is responsible for drawing out the hangman figure, these will be created as this is another new feature that needs to be tested to make sure it is working as planned.

Date planned	Task	Estimated time
2019-03-10	Plan iteration 4	30m
2019-03-13	Revise Project Plan and rewrite parts that need clarification	2h
2019-03-13	Add reflections to previous time logs	1h
2019-03-13	Rework iteration plans to be more accurate	1h
2019-03-13	Watch live workshop and participate with questions	2h
2019-03-17	Refactor code - create new classes, rename variables and classes	8h
2019-03-17	Rework code to ensure testability	4h
2019-03-17	Implement the “drawing” of the hangman	4h
2019-03-17	Write three separate text files for the difficulty levels	30m
2019-03-18	Implement extra feature - difficulty levels	6h

Date planned	Task	Estimated time
2019-03-18	Write new manual test for UC2 and run test	1h
2019-03-19	Implement new unit tests for hangman drawing	4h
2019-03-19	Run new tests and record result	2h
2019-03-19	Rework use cases, diagrams and so on	4h
2019-03-19	Create new diagrams for new feature ?	4h
2019-03-20	Participate in Q and A	2h
2019-03-21	Write reflection on time log for iteration 4	30m

Deadline for iteration 4: 21 March

6 Risk analysis

6.1 List of risks

Risk	Probability	Effects
Staff becoming ill and unable to work	Moderate	Catastrophic
Hardware used to complete project failing	Moderate	Tolerable
Time required to complete project being underestimated	High	Serious
Extra features not being included in final product	Moderate	Tolerable

6.2 Strategies

Risk	Strategy
Staff becoming ill	Give staff resources and time for healthy eating and exercise.
Hardware failing	Continuously backup all work made and have the ability to borrow equipment to complete work.
Underestimated time	Creating a realistic schedule and constantly keep track and update time spent on specific tasks, as well as prioritising tasks so the most important parts of the project get done first.
Features not included	Creating realistic goals and not be overly ambitious about what features can be added in the time allocated.

7 Time log

7.1 Iteration 1

Date finished	Task	Scheduled time	Actual time
2019-01-23	Watch lecture 1 / Introduction	2h	2h
2019-01-23	Created GitHub repository	1h	0,5h
2019-01-24	Read chapter 2 – Software Processes	3h	3,5h
2019-01-25	Read chapter 3 – Agile Software Development	3h	3h
2019-01-29	Watch lecture 2	2h	1h 40min
2019-01-29	Plan the time schedule for iteration 1	3h	3h
2019-01-30	Read chapter 22 – Project Management	3h	3,5h
2019-01-30	Participate in Q and A	2h	2h
2019-01-31	Read chapter 23 – Project Planning	3h	4h
2019-01-31	Watch lecture 3	2h	1h 40min
2019-02-01	Outline the project plan	5h	7h
2019-02-01	Write the vision document	4h	4h
2019-02-04	Complete first draft of the project plan	8h	9h
2019-02-04	Start outlining skeleton code for the application	2h	1h
2019-02-04	Update and finish time schedule	2h	1h
2019-02-05	Write risk analysis	2h	2h
2019-02-06	Finish skeleton code	1h	1h
2019-02-08	Error check and finish up project plan	6h	8h
Total time	**Iteration 1**	**54h**	**58h 20min**

Reflection

We found that the main reason for the difference in the time scheduled for this iteration and the time taken was how long the research would take. Reading up on the information about planning and management was a longer process than anticipated as the material needed time and some re-reading to fully understand. This also put more time into the writing of the project plan as this was a new way of planning a project, and we found it useful to go back to the research material during the writing to look up exactly what to write and how the outline this project plan.

7.2 Iteration 2

Date finished	Task	Scheduled time	Actual time
2019-02-06	Watch live lecture on theme 2	2h	2h
2019-02-13	Plan iteration 2	1h	1h
2019-02-18	Expand use case model	1h	1h
2019-02-15	Write fully dressed use case for Play Game	1h	1h
2019-02-17	Write additional use case for Set up Game	1h	0,5h
2019-02-19	Create State Machine Diagram for Play Game	2h	2,5h
2019-02-15	Implement functionality to start game and display a word to guess in Hangman game.	3h	2h
2019-02-15	Read chapter 4	2h	2h
2019-02-16	Read chapter 5	2h	2h
2019-02-18	Read chapter 20	2h	2h
2019-02-18	Watch Lecture 4 – Systems and Software Modeling	2h	2h
2019-02-18	Read chapter 7	2h	2h
	Watch Lecture 5 – Modeling with UML	2h	-
2019-02-20	Read chapter 6	2h	2h
	Watch Lecture 6 – Software Architecture	2h	-
2019-02-20	Read chapter 15	2h	2h
	Watch Lecture 7 – Software Design	2h	-
	Watch Lecture 8 – From Software Design to Implementation	2h	-
2019-02-20	Watch live lecture 2 on theme 2	2h	2h
2019-02-21	Expand functionality in game by implementing updated word after guesses and end game scenarios.	8h	6h
2019-02-22	Finish implementing functionality in Hangman to complete a working game.	10h	8h
2019-02-22	Create Class Diagram	2h	2h
Total time	**Iteration 2**	**55h**	**42h**

Reflection

In this iteration we found that less time was spent on the project than expected. This was partly due to the team having to take time for separate projects. However this did not effect the project in a negative way as we found we had over-estimated how much time was needed for the developer to implement a working version of the game. We found that due to a clearly laid out plan in the beginning of the project the developer knew what was needed to implement the game. What we found there was no time left for was watch all of the research material available.

7.3 Iteration 3

Date finished	Task	Scheduled time	Actual time
2019-02-25	Plan iteration/theme 3	30m	30m
2019-02-25	Create a test plan	1h	45m
2019-03-01	Read chapter 8	2h	3h
2019-02-27	Watch live lecture/workshop on theme 3	2h	2h
2019-03-01	Watch pre-recorded lecture 9	1h 45m	1h 45m
2019-03-05	Watch video on the “Greeter Example”	30m	30m
2019-03-04	Create manual test case for Use case 2	2h	1h
2019-03-04	Create manual test case for Use case 3	2h	1h
2019-03-04	Run manual tests and record the results	2h	30m
2019-03-06	Watch pre-recorded lecture 10	1h 45m	1h 45m
2019-03-06	Watch pre-recorded lecture 10.5	30m	30m
2019-03-06	Watch live lecture/workshop on theme 3	2h	1h 30m
2019-03-07	Create automated unit tests for method createUnderscores()	4h	3h
2019-03-07	Create automated unit tests for method evaluateGuess()	4h	2h
2019-03-07	Create automated unit test for unfinished method drawHangman()	2h	30m
2019-03-08	Run automated tests for method (1) and record the results	2h	30m
2019-03-08	Run automated tests for method (2) and record the results	2h	1h
2019-03-08	Run automated tests for method (3) and record the results	2h	30m
2019-03-08	Write reflection	1h	30m
Total time	**Iteration 3**	**35h**	**22h 45m**

Reflection

In this iteration we found that we had over-estimated how much time would be needed both for creating the tests and running them. Once the tester got the hang of how to write and execute the manual tests, these could be done in a shorter amount of time. The unit tests took a bit more time both to write and test, but this was mainly due to the fact that the code had not been written with testability in mind. Again, once we felt we knew how to test the chosen methods, the actual testing and recording of the results took less time than first estimated. More on this in the test reflection.

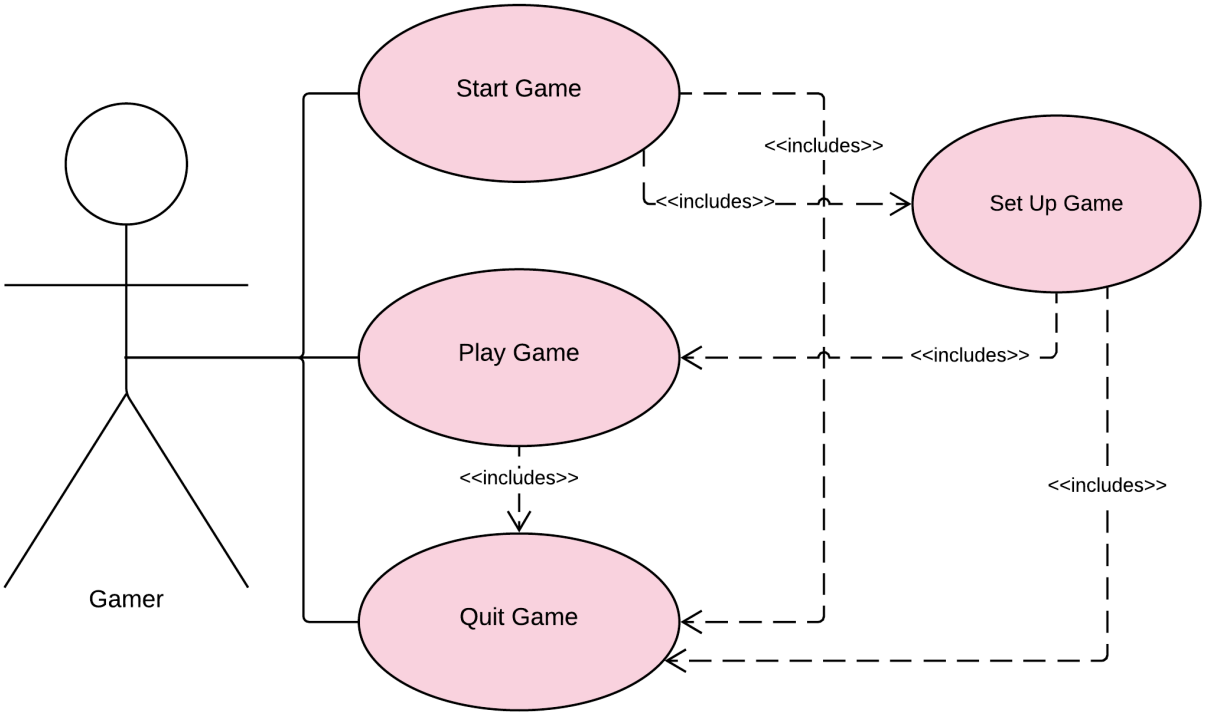
7.4 Iteration 4

Date finished	Task	Scheduled time	Actual time
2019-03-10	Plan iteration 4	30m	30m
2019-03-13	Revise Project Plan and rewrite parts that need clarification	2h	4h
2019-03-17	Add reflections to previous time logs	1h 30m	1h
2019-03-18	Rework iteration plans to be more accurate	1h	2h
2019-03-13	Watch live workshop and participate with questions	2h	1h
2019-03-13	Refactor code - rename variables and classes	2h	2h
2019-03-18	Implement new class to handle everything guess-related	5h	6h
2019-03-13	Rework code to ensure testability	4h	3h
2019-03-17	Implement the “drawing” of the hangman	4h	5h
2019-03-19	Write three separate text files for the difficulty levels	30m	30m
2019-03-19	Implement extra feature - difficulty levels	4h	3h
2019-03-18	Write new manual test for UC2 and run test	1h	1h
2019-03-20	Implement new unit tests for hangman drawing	2h	2h
2019-03-20	Run new tests and record result	2h	1h
2019-03-20	Rework use cases, diagrams and so on	4h	4h
2019-03-20	Participate in Q and A	2h	2h
2019-03-22	Create .jar-file to run game in terminal	2h	4h
2019-03-22	Write reflection on time log for iteration 4	30m	30m
Total time	**Iteration 4**	**40h**	**42,5h**

Reflection

In this last and final iteration of the project what took some more time than anticipated was rewriting parts of the project plan to be more up to date with the status of the project as well as the code left to be implemented. The last features added to the game meant some of the earlier implemented code had to be reworked and moved around a bit which added time for the developer. To in the end also add the functionality of an executable .jar file to be able to run the game in a terminal took time to work out as the file could not access the text-files containing the words for the game.

8 Use Case Diagram



9 Use Cases

9.1 Use Case 1 – Start Game

Precondition: none.
Postcondition: The game menu is shown.

Main scenario

1. Starts when the Gamer wants to begin a session of the hangman game.
2. The system presents the main menu with a title, the option to play and quit the game.
3. The Gamer makes the choice to start the game.
4. The system starts the game (see Use Case 2).

Alternative scenarios

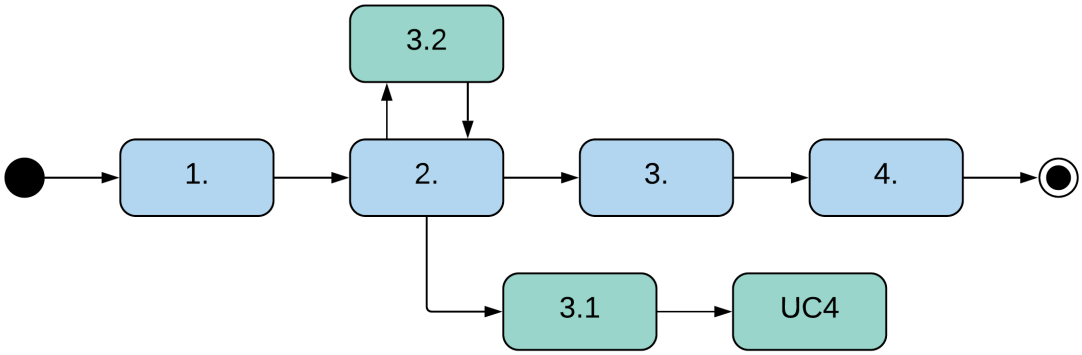
3.1 The Gamer makes the choice to quit the game.

1. The system quits the game (see Use Case 4)

3.2 Invalid menu choice

1. The system presents an error message.
2. Go to 2

Activity Diagram



9.2 Use Case 2 – Set up Game

Precondition: The game menu is shown.
Postcondition: The difficulty menu is shown.

Main scenario

1. Starts when the Gamer makes the choice to start a new game.
2. The system asks for a user name.
3. The Gamer gives a user name.
4. The system present an option menu with three different difficulties of words.
5. The Gamer makes a choice of difficulty.
6. The system starts a round of the game.

Alternative scenarios

5.1 The Gamer makes the choice to quit the game.

1. The system quits the game. (see Use Case 4)

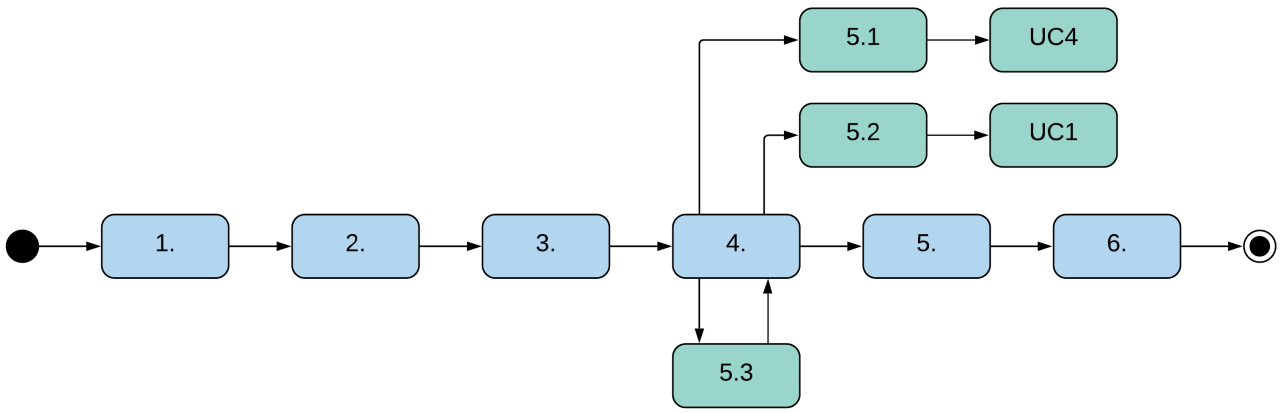
5.2 The Gamer makes the choice to go back to menu.

1. The system shows the game menu. (see Use Case 1)

5.3 Invalid menu choice.

1. The system presents an error message.
2. Go to 4.

Activity Diagram



9.3 Use Case 3 – Play Game

Precondition: The difficulty menu is shown.
Postcondition: The game is running.

Main scenario

1. Starts when the Gamer makes the choice of difficulty.
2. The system present a word to guess with the number of letters in the word, and the number of guesses available.
3. The Gamer makes a guess of a letter.
4. The system presents an updated version of the word with the letter, the number of letters left to guess and the number of guesses left.

Repeat from step 3.

5. The Gamer guesses the last letter and so the whole word correctly.
6. The system presents the finished word and the number of guesses used, as well as the option to start a new round.
7. The Gamer makes the choice to quit the game.
8. The system quits the game. (see Use Case 4)

Alternative scenarios

3.1 The Gamer makes the choice to quit the game.

1. The system quits the game. (see Use Case 4)

4.1 Incorrect guess as the letter is not present in the current word.

1. The system draws a part of the hangman, and presents the number of guesses left.
2. Go to 3.

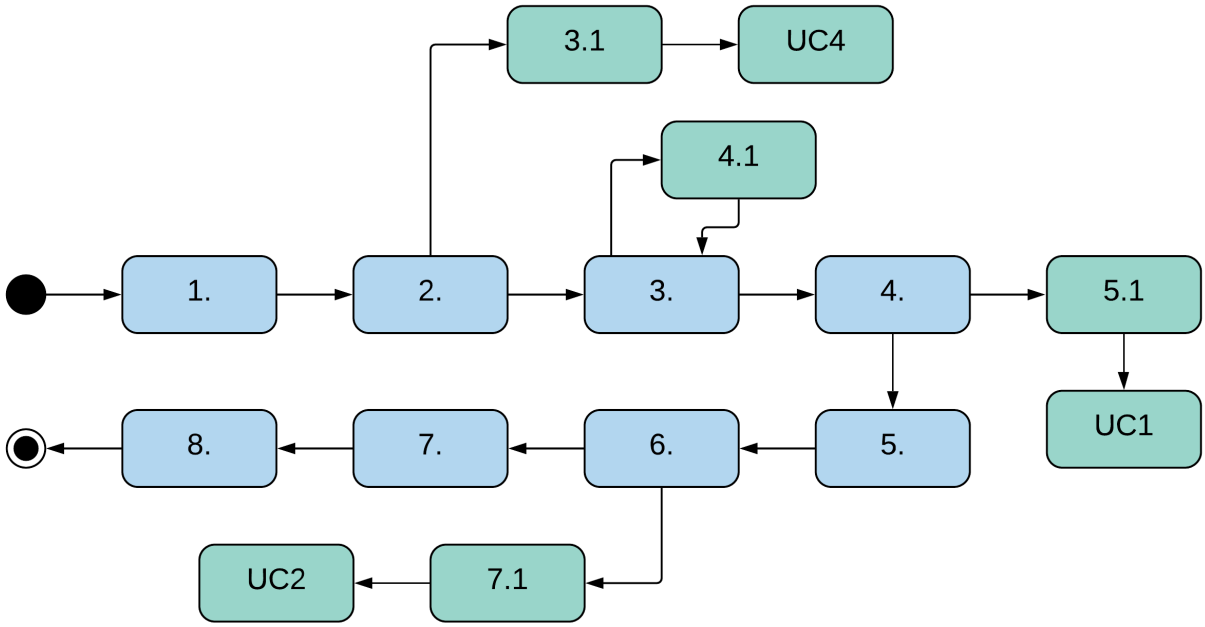
5.1 The Gamer has no guesses left.

1. The system presents a fully drawn hangman and the correct word.
2. The Gamer choses to start a new game.
3. The system shows the game menu. (see use Case 1)

7.1 The Gamer choses to start a new round.

1. The system shows the difficulty menu. (see Use Case 2)

Activity Diagram



9.4 Use Case 4 – Quit Game

Precondition: The game is running.

Postcondition: The game is terminated.

Main scenario

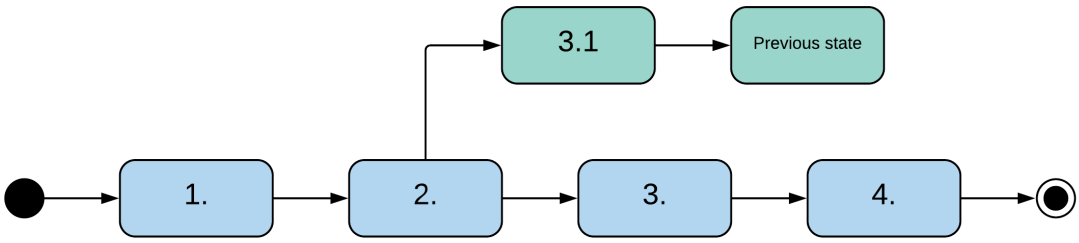
1. Starts when the Gamer wants to quit the game.
2. The system prompts for confirmation.
3. The Gamer confirms.
4. The system terminates.

Alternative scenarios

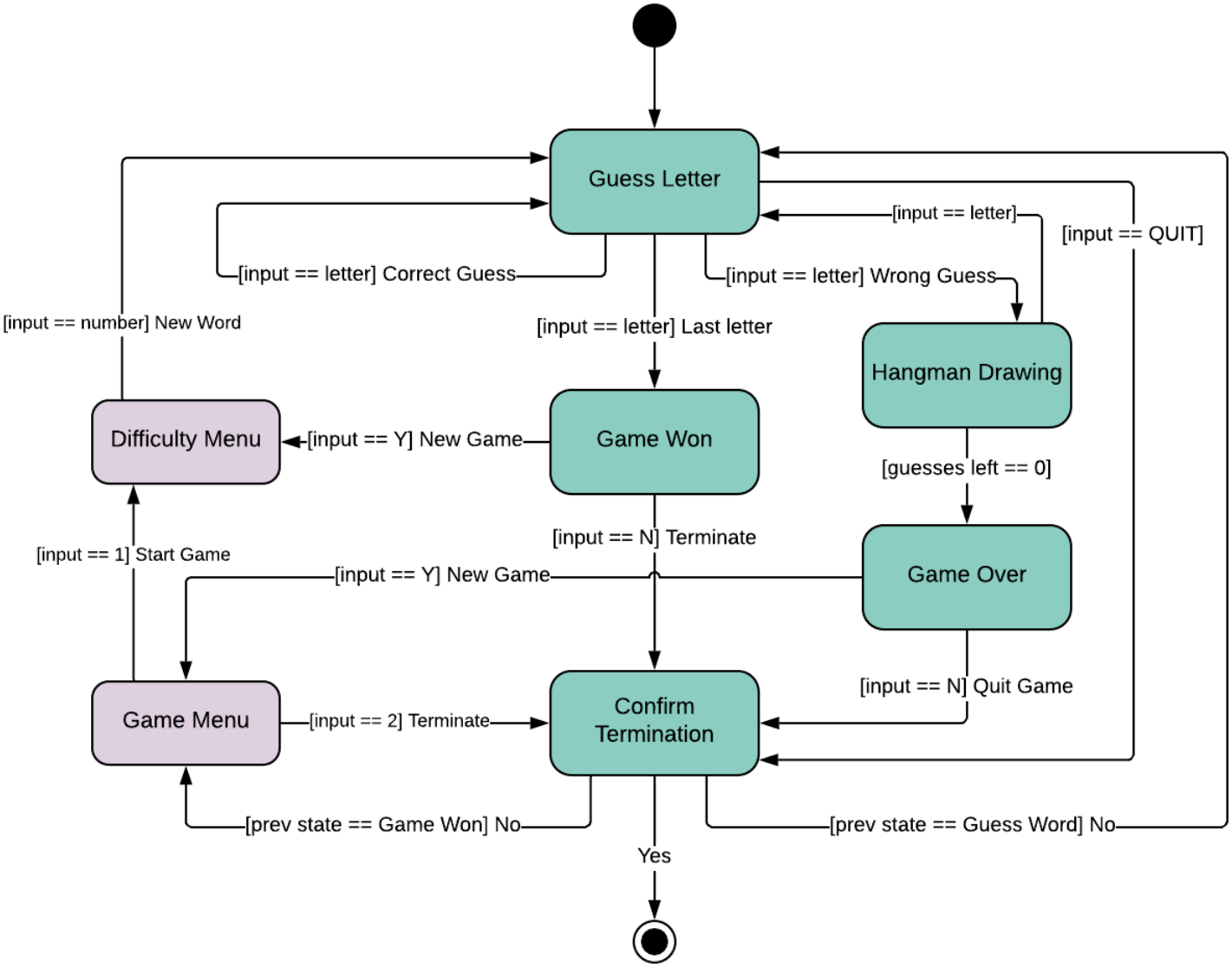
3.1. The Gamer does not confirm.

   1. The system returns to its previous state.

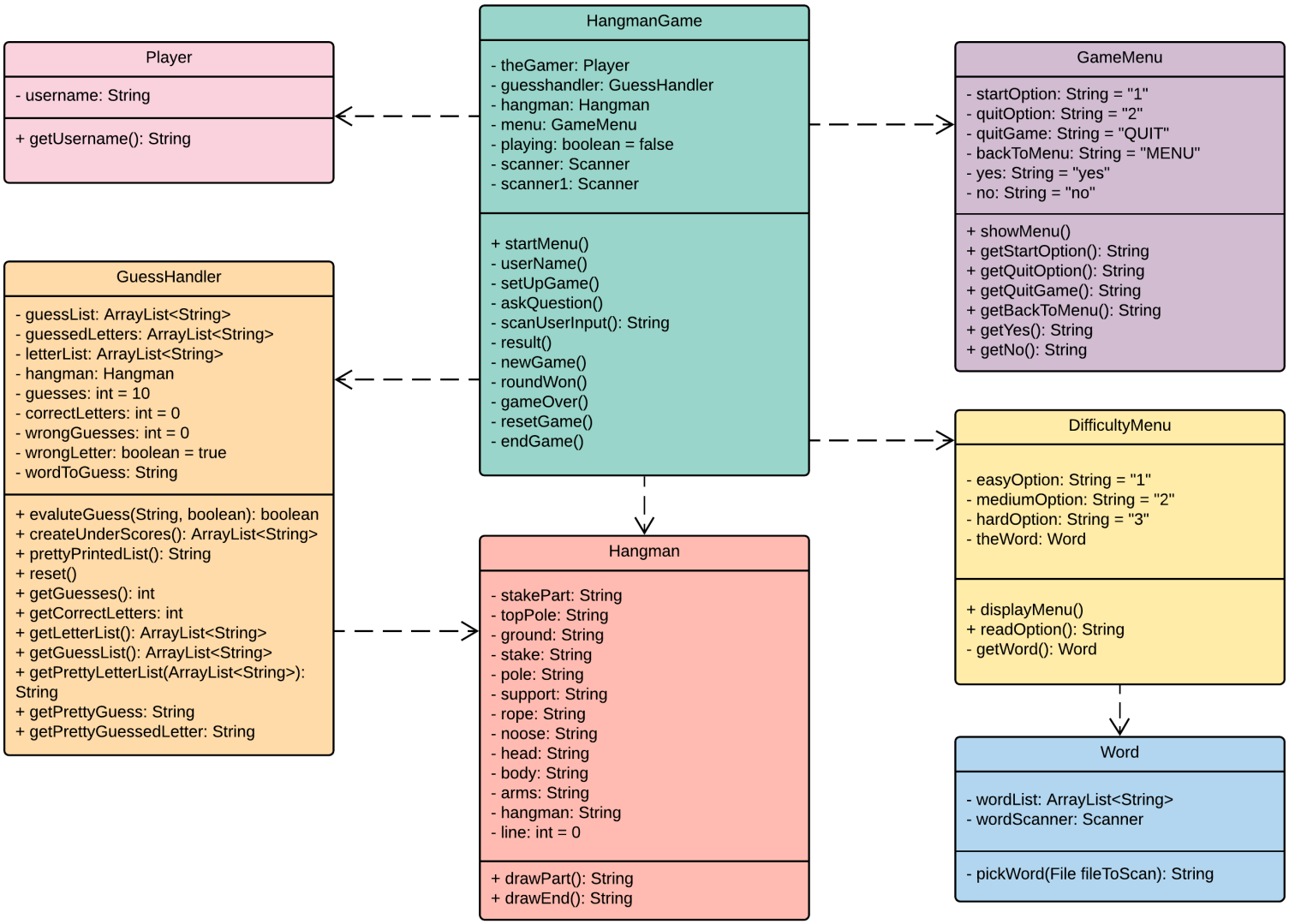
Activity Diagram



10 Play Game State Machine



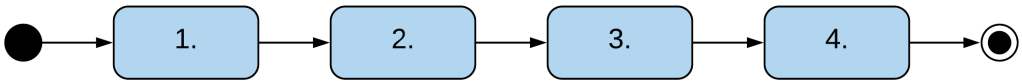
11 Class Diagram



12 Manual Test Cases

TC1.1 Start a game successfully

Use Case: UC1 – Start Game
Scenario: Start a game successfully



The main scenario of UC1 is tested where a user starts a game.

Test steps

- Start the application.
- The system presents “Welcome to Hangman 1.0, Menu, 1. Start Game, 2. Quit Game, Pick a number option:”
- Enter the number “1” and press enter.

Expected

- The system continues to UC2 and asks for a user name.

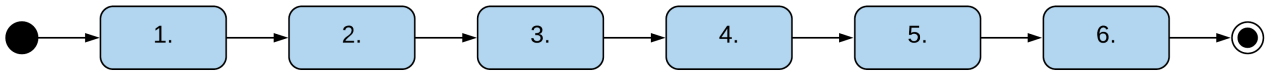
☑ Test successful.

Comments from tester

This test succeeded without any problem.

TC2.1 Set up a game successfully

Use Case: UC2 – Set up Game
Scenario: Start a game successfully



Precondition: UC1 should have been executed correctly, so that the gamer has made a choice to start a game.

Test steps

- Start the application by creating a new instance of the class Hangman with input “LEGO”.
- Execute UC1
- The system presents “Type in a user name:”
- Enter the name “Melina” and press enter.
- The system should show “Hello Melina” and presents a menu with three difficulty options.
- Enter the number “1” and press enter.

Expected

- The system should continue to UC3 and show “Word to guess: _ _ _ _ , Number of guesses left: 10, Guess letter: ”
- The system waits for input.

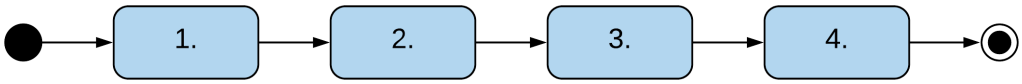
☑ Test successful.

Comments from tester

This test worked as expected, after typing in a user name the system presented “Hello Melina” and continued on to ask for what difficulty level was wanted. After this the system presents the first word in way of showing underlines.

TC3.1 Guess a letter successfully

Use Case: UC3 – Play Game
Scenario: Guess a letter successfully. The first part of UC3 where the system presents a word to guess, the user inputs a letter as a guess and the system presents an updated version of the word.



Precondition: UC1 and UC2 should have been executed successfully, so that the user has chosen to start a game, given a user name and a choice of difficulty. Also a separate instance of the class containing just one word should be used.

Test steps

- Start the application by creating a new instance of the class Hangman with input “LEGO”.
- Execute UC1
- The system presents “Word to guess: _ _ _ _ , Number of guesses left: 10”
- Enter the letter “e” and press enter.

Expected

- The system should show “Correct! Word to guess: _ E _ _ , Number of guesses left: 9”
- The system waits for a new input.

☑ Test successful.

Comments from tester

As the game uses random words generated from a text file, it was necessary to create a separate instance of the class with a pre-determined word for the test.
The result was as expected except “Correct!” is not shown, instead a list of already guessed letters is displayed. In this case, since only one letter is guessed, the list displayed “Guessed letters: E”.

13 Automated Unit Tests

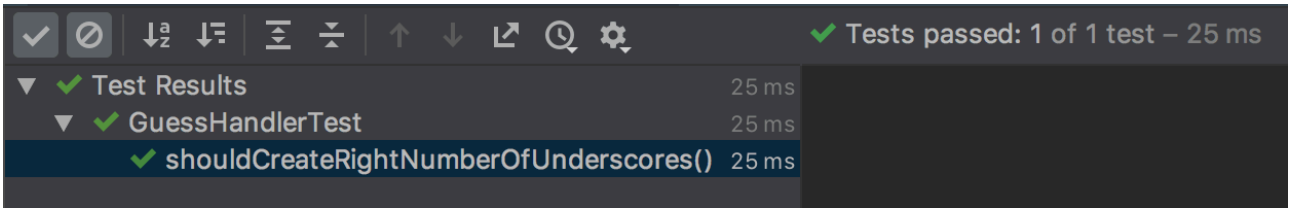
13.1 Method createUnderscores()

Method to be tested: createUnderscores() in GuessHandler class

Test 1 – Check number of underscores

Test method shouldCreateRightNumberOfUnderscores() tests if the number of underscores created by the method corresponds to the number of letters in the word to be guessed.

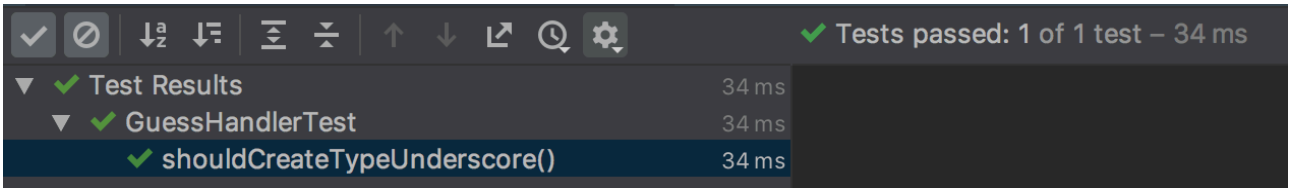
Result:



Test 2 – Check value of string created

Test method shouldCreateTypeUnderscore() tests that the method creates the right type/value of the string, so that it is an underscore created per each letter.

Result:



Comments

These tests proved to be a bit of a challenge to figure out, as how to test this method was not obvious from the start, as well as how to create two different tests for the same method.
The resulting tests created has a great use in making sure this step in the game is handled correctly to that the player knows how many letters are present in the current word. It was also important for this project that the letters are represented with underscores as this is the classic way of playing a hangman game.

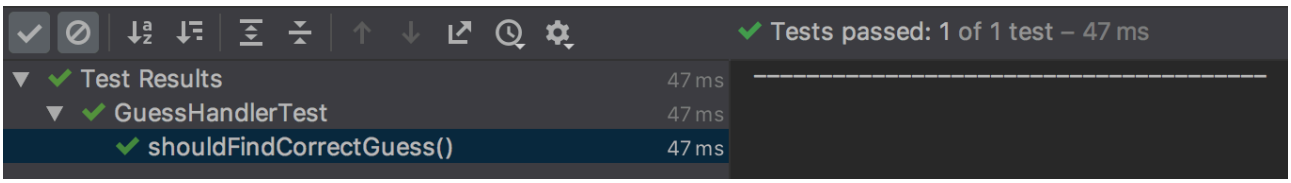
13.2 Method evaluateGuess()

Method to be tested: evaluateGuess(String guess) in GuessHandler class

Test 1 – Find correct guess

Test method shouldFindCorrectGuess() tests that when a letter guessed is present in the current word the method finds this to be true, and the list that displays the word to the player updates with the correct letter.

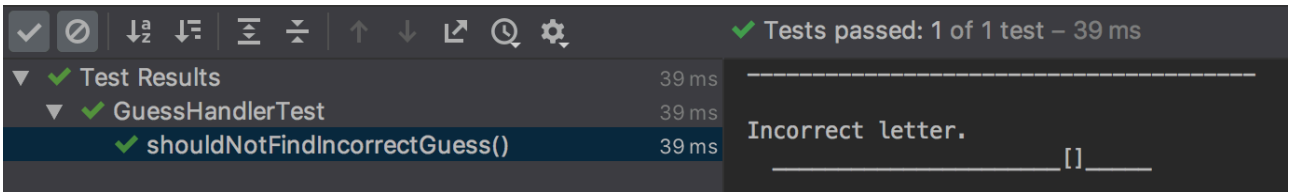
Result:



Test 2 – Do not find incorrect guess

Test method shouldNotFindIncorrectGuess() tests that when a letter guessed is not present in the current word the method does not find a correct guess and does not update the word, but instead displays the word in its previous state again for the next guess.

Result:



Comments

This method is a core part of the application and the game and as such more tests will be needed to check all the functionality of it. These two first unit tests proved useful as they test both scenarios happening after the player has guessed a letter, when the letter is a correct versus incorrect guess. As the game in its normal state is played by a random word being picked from a text file, the tests had to create a new instance of the GuessHandler class with a pre-defined word for the tests to work correctly.

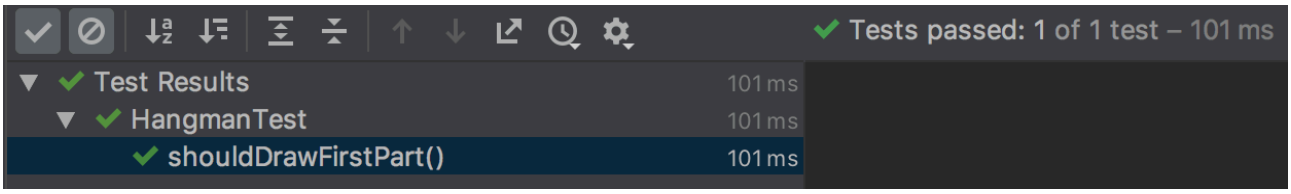
13.3 Method drawPart()

Method to be tested: drawPart() in Hangman class.

Test 1 – Draw first part of hangman

Test method shouldDrawFirstPart() tests that the when the method is called with one wrong guess, the first part (the ground) of the Hangman figure is returned.

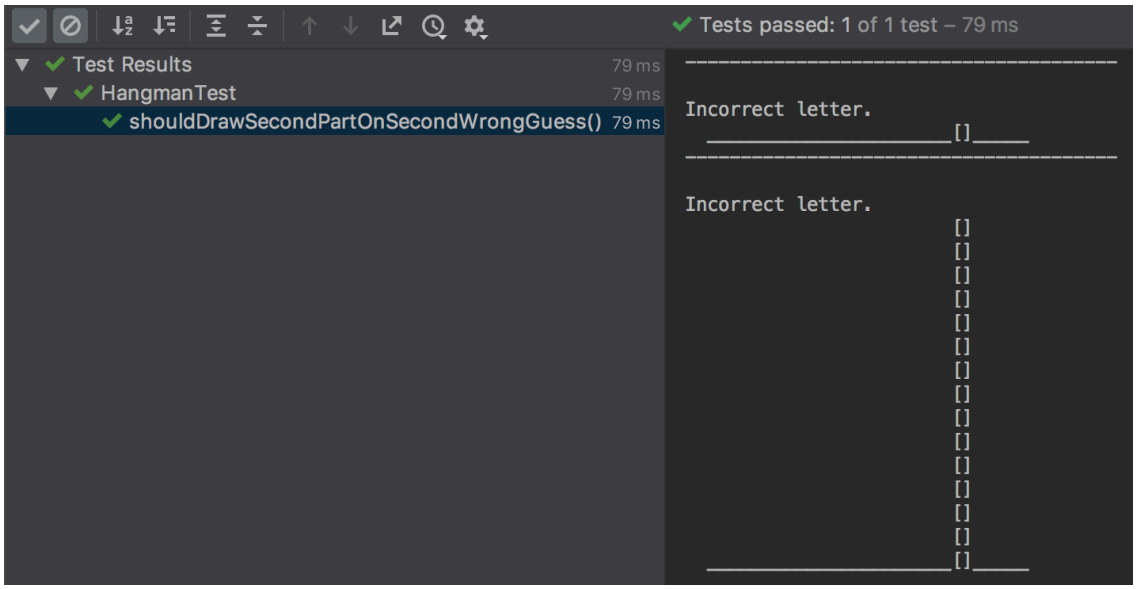
Result:



Test 2 – Draw second part of hangman after two incorrect guesses

Test method shouldDrawSecondPartOnSecondWrongGuess() tests that after two wrong guesses has been made on the word, the second part of the hangman figure will be returned.

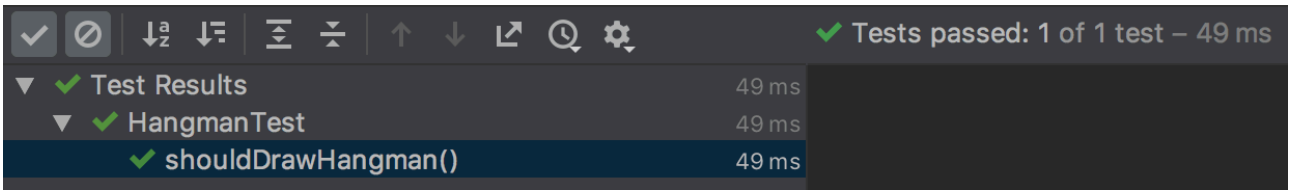
Result:



Test 3 – Draw the whole hangman figure

Method shouldDrawHangman() tests that when the method is called with 10 incorrect guesses the whole hangman figure is drawn out.

Result:



Comments

This method felt useful to test as the drawing of the hangman figure as the player progresses in the game is an important feature. The test took a little bit of time to figure out how to use in a productive way as the method is called upon during the game. The way it is implemented by having the parameter of number of wrong guesses makes it easier to test in a test environment to ensure the correct output.

13.4 Unit Tests Reflection

When starting to write the unit tests I found I had to use quite some time to refactor my code for the Hangman game to increase the testability of the code. The code had not been written with testability in mind and a lot more refactoring and reworking of the code was necessary. Testing the code taught me more about how I want the structure in the application to be to create a more functional game. After the initial unit tests I understood more on how to write tests and how to test this particular application, so the next tests written were easier to make.
All in all, testing the code proved to be a learning and important step in the project.