Assignment 2 – Modelling

1. Iteration Description

The second iteration should focus on modelling and representing different parts/features of the game using UML diagrams. The first step is to write Use Cases that will represent different features and scenarios of the game. To complete the Use-Case Model, a diagram representing the use cases as well as the actors should be designed.

To help in developing the application, a state machine that describes the functionality of the game needs to be created. Expanding this application to cover all the features could help see all the connections and interactions that will be encountered during the implementing phase.

Before the due date, the game should be in a “playable” state. This could be a good moment to decide on what kind of interface should be used. This state should be represented using a class diagram to have an overview of all the relationships between the classes that are or will be developed.

Task	Description	Estimated Time
Designing Use-Cases	Create some Use-Cases to show different features of the game	1:30 H
Expanding Use-Case-Model	Add a diagram to represent the Use Cases	0:30 H
Design State Machine for “Play Game”	Show the functionality of the code representing the “Play Game” use case	1:00 H
Design Complete State Machine	Show complete functionality of the game	1:30 H
Implementation of base game	Make the game playable (one round of hangman)	3:00 H
Design Class Diagram	Design a diagram to show the relationships between the classes	1:00 H

2. Time log

Task	Date	Time Estimated	Actual Time
Plan 2nd Assignment	19.02.2019	1:00 H	0:30 H
Use Case Model	20.02.2019	1:30 H	2:30 H
State Machine(basic)	20.02.2019	1:00 H	2:00 H
Implement Basic Game	20.02.2019	3:00 H	2:00 H
Class Diagram	20.02.2019	1:00 H	0:30 H
State Machine(extra)	21.02.2019	1:30 H	2:00 H
Fix Class diagram	10.04.2019	0:30 H	0:30 H
Fix State Machines	10.04.2019	2:00 H	2:30 H
Update documentation	11.04.2019	1:00 H	2:00 H

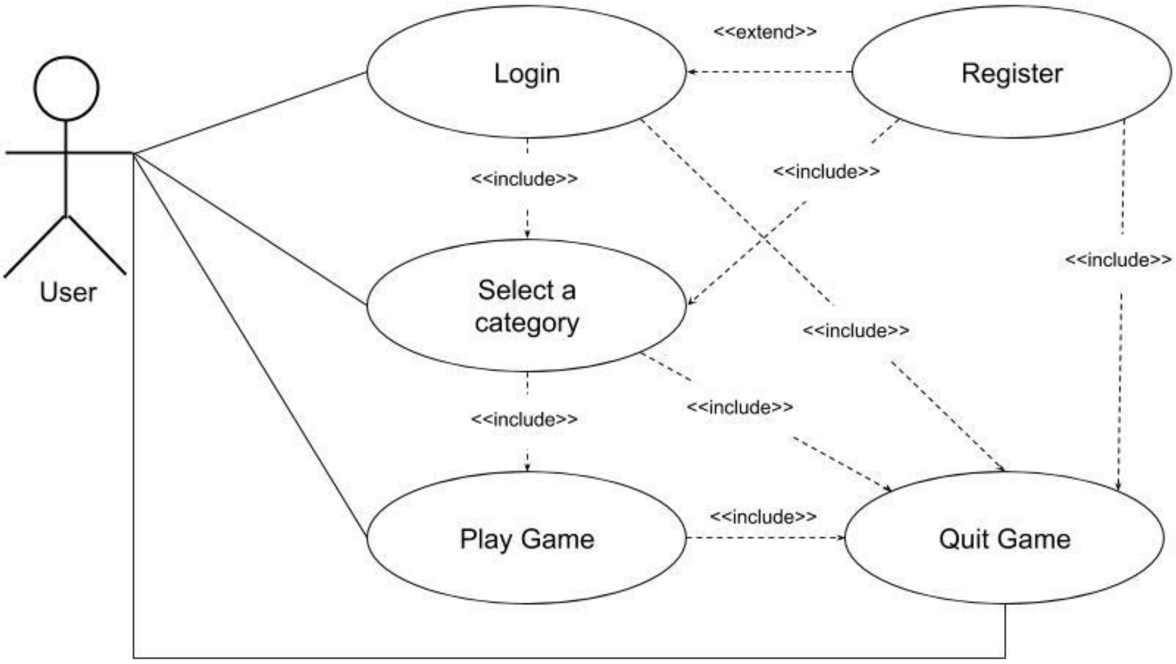
Some of the tasks took more time than estimated as a result of using a bad strategy while designing the diagrams (i.e. trying to learn about all their particularities during their development). This made the actual time be almost two times as large as the expected one. This was taken as a lesson for further projects, never learn how to use an important tool “on-the-go”.

3. Use Case Model

In order to better understand the goals of the users, the required behavior of the system in satisfying these goals and the interactions between the users and the system a use case model was created.

3.1 Use Case Diagram

To showcase the actors and the relationships between the use cases, a diagram was created:



3.2 Use Cases

The system is divided into five different use cases that cover the main scenario as well as the alternative options.

UC 1 Log in

Precondition: none.

Postcondition: the category menu is shown.

Main scenario

1. Starts when the user wants to begin a session of the hangman game.
2. The system shows the options “Login”, “Leaderboard” and “Quit”
3. The user selects the Login option
4. The system asks for the user details and gives the option to login, register a new account, go back to menu or quit.
5. The user inputs the details and selects the login button.
6. The system shows the Category menu (Execute use case 3)

Alternative scenarios

2.1 The user selects the Leaderboard option

- The system shows the and the option to return to menu

5.1 User inputs wrong details

- The system shows an error message and returns to step 4.

5.2 User selects the register option

- Execute Use Case 2

5.3 User selects the back to menu option

- Go back to step 2

2.2/5.4 The user selects the quit option

- Execute Use Case 5.

UC 2 Register

Precondition: the register menu is accessed.

Postcondition: a new user is registered, and the category menu is shown.

Main scenario

1. Starts when the user wants to register a new account.
2. The system asks for the user details and gives the option to confirm, go back to menu or quit.
3. The user inputs the details and selects the confirm button.
4. The system shows the Category menu (Execute use case 3)

Alternative scenarios

2.1 User selects the back to menu option

- Go to Use Case 1

2.2 The user selects the quit option

- Execute Use Case 5

3.1 User inputs invalid credentials

- The system shows an error message and returns to step 2.

UC 3 Select a category

Precondition: The user is logged in/registered a new account.

Postcondition: A round begins.

Main scenario

1. Starts when the user successfully logs in or registers a new account
2. The system presents the available categories with the user progress, the timed-mode checkbox, the option to go back to menu and quit.
3. The user selects a category.
4. The system presents a random word from the category.

Alternative scenarios

2.1 All of the categories are completed

- The system shows a complete game screen and an option to quit.

2.2 User selects the back to menu option

- Go to Use Case 1

2.3 The user selects the quit option

- Execute Use Case 4

UC 4 Play Game

Precondition: A category was selected.

Postcondition: The game has ended.

Main scenario

1. Starts when the user chooses a category.
2. The system presents a random word from the category, the image of the hangman and the options to go back to categories, back to menu and quit.
3. The user guesses a letter from the word
1. Repeat step 3 until all the letters are guessed.
4. The system shows that the word was guessed, increases the user score by one and asks if the user wants to keep playing, select a different category go back to menu or quit.
5. The user selects to quit the game.
6. The system quits the game (Execute Use Case 5).

Alternative scenarios

*. At any point of time, user wants to go back to menu

1. System asks for confirmation

A. 2. User confirms

3. System Goes back to menu

B. 2. User doesn't confirm

3. System returns to previous state

*. At any point of time, user wants to exit

- Execute Use Case 5

3.1 The user selected the Timed Man difficulty (UC 3)

- Same behavior, but in addition, every 10 seconds of inactivity, one "life" point is lost

3.2 The user guesses a letter that's not in the word

- The user loses one "life".

3.3 The user runs out of "lives"

- The system shows a "Game over" message and goes to step 4.

4.1 The Timed-Mode difficulty is selected

- The system increases the score by 3 points per word instead of 1

4.2 The word guessed was the last one in the category

- The user cannot select the "Keep Playing" option

5.1 The user selects "Keep Playing"

- Go to step 2

5.1 The user selects "Back to categories"

- Execute Use Case 3

5.2 The user selects "Back to Menu"

- Execute Use Case 1

5.3 The user selects "Quit"

- Execute Use Case 5

UC 5 Quit Game

Precondition: The game is running.

Postcondition: The game is terminated.

Main scenario

1. Starts when the user wants to quit the game.
2. The system prompts for confirmation.
3. The user confirms.
4. The system shows a message and terminates.

Alternative scenarios

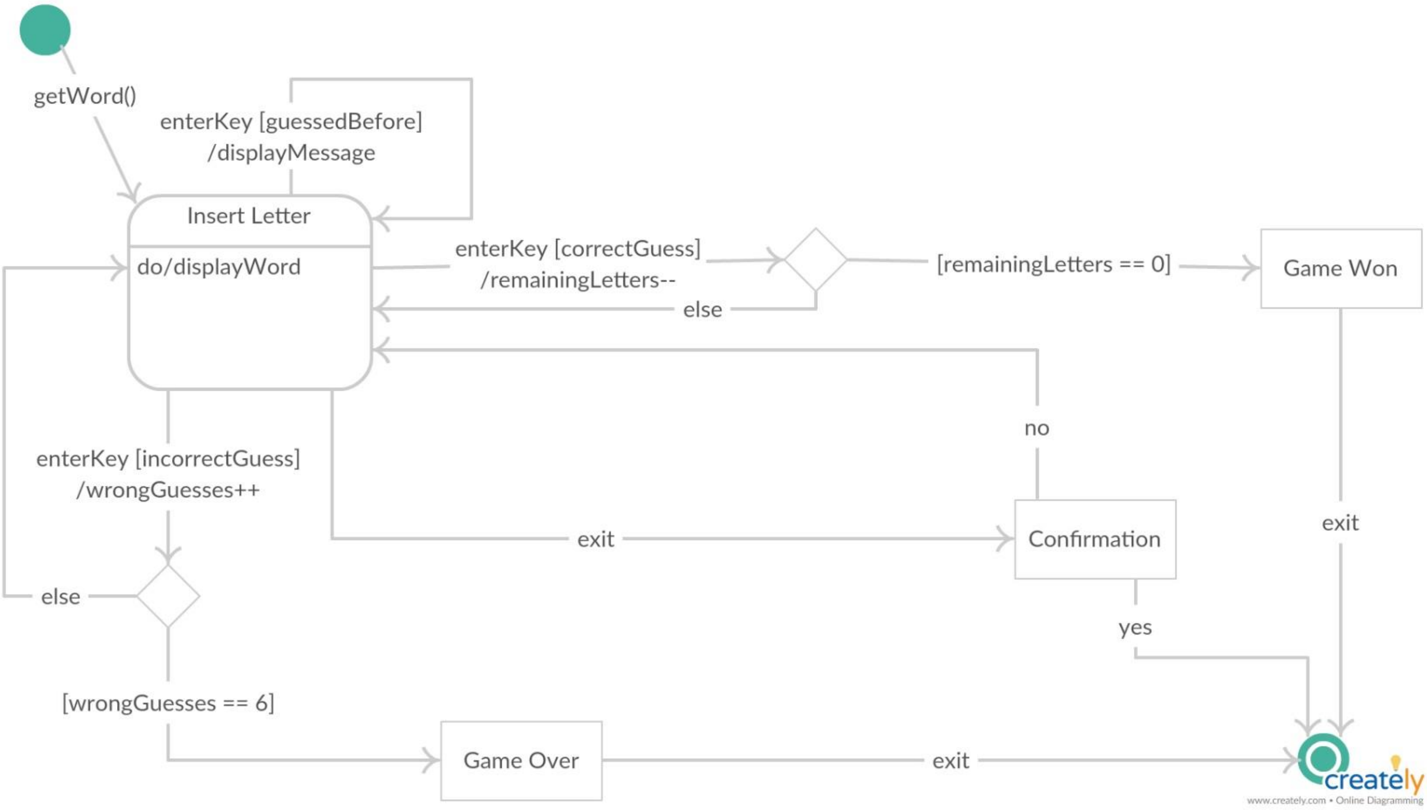
3.1. The user does not confirm

- The system returns to its previous state

4. State Machine Diagram

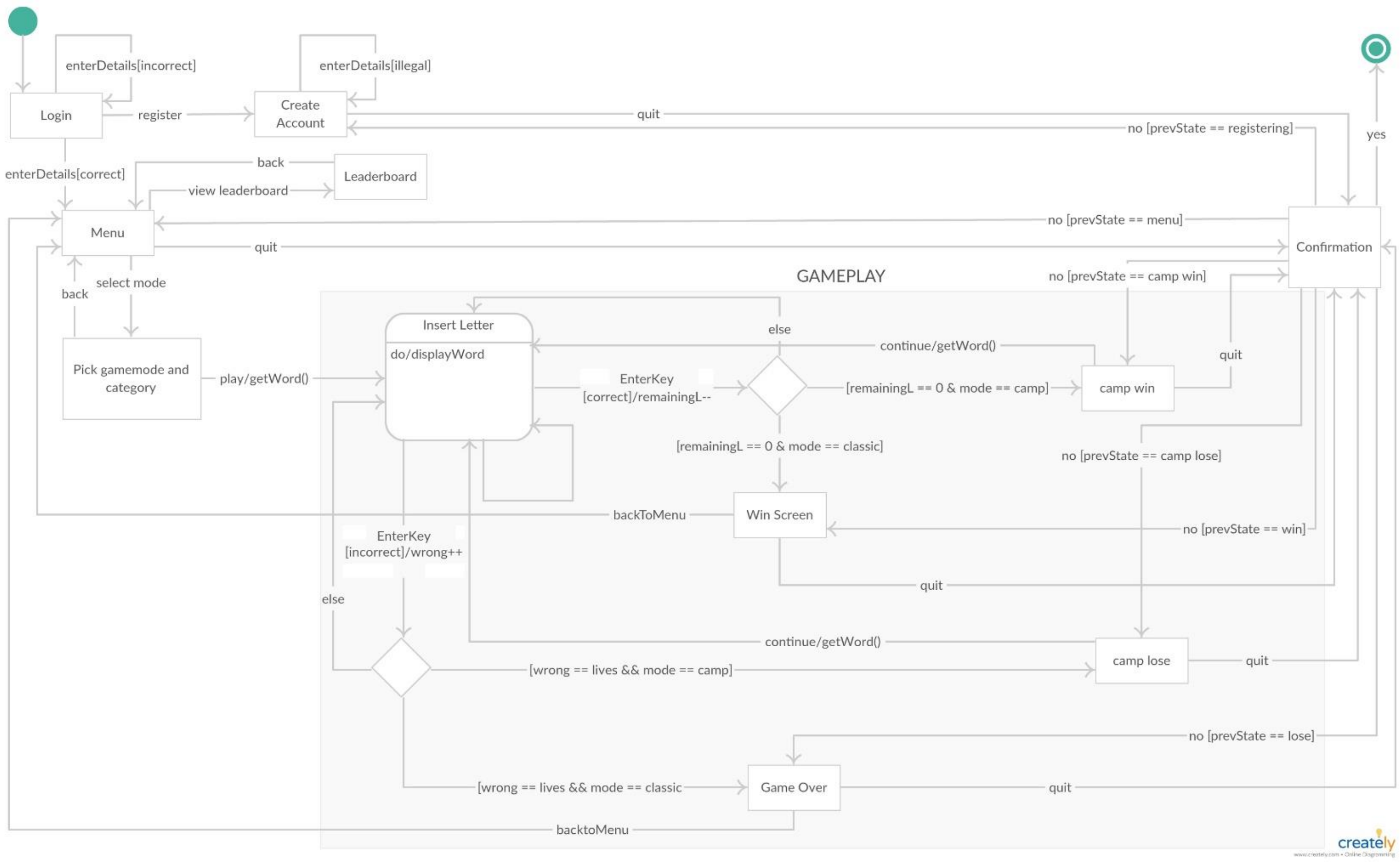
The state machine diagrams are used to help understand the behavior of the system. The first diagram only shows the functionality of the "play game" use case but the second one is extended to the complete system.

4.1 Basic State Machine



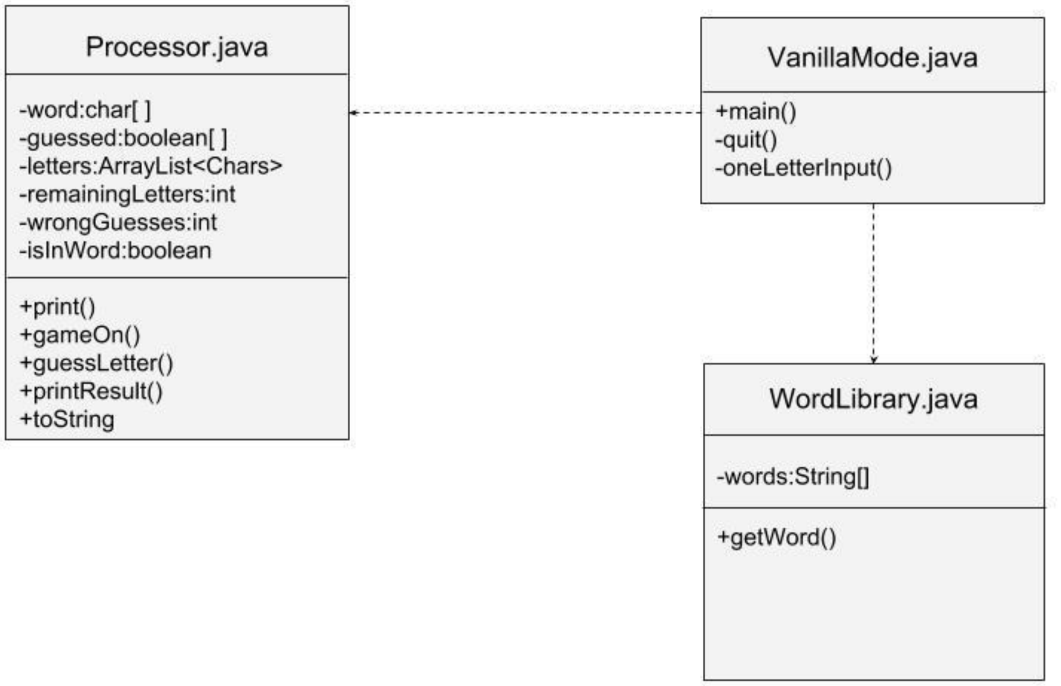
getWord()
enterKey [guessedBefore] /displayMessage
Insert Letter
do/displayWord
enterKey [correctGuess] /remainingLetters--
[remainingLetters == 0]
Game Won
else
no
enterKey [incorrectGuess] /wrongGuesses++
exit
Confirmation
exit
else
yes
[wrongGuesses == 6]
Game Over
exit
creately
www.creately.com • Online Diagramming

4.2 Extended State Machine



5. Class Diagram

The class diagram shows the structure of the system by showing its classes. This diagram was updated, but no additional files were added as they did not contain any code. (They were removed from the implementation as they did not provide any functionality at that point of time)


Processor.java
-word:char[ ]
-guessed:boolean[ ]
-letters:ArrayList<Chars>
-remainingLetters:int
-wrongGuesses:int
-isInWord:boolean
+print()
+gameOn()
+guessLetter()
+printResult()
+toString
VanillaMode.java
+main()
-quit()
-oneLetterInput()
WordLibrary.java
-words:String[]
+getWord()