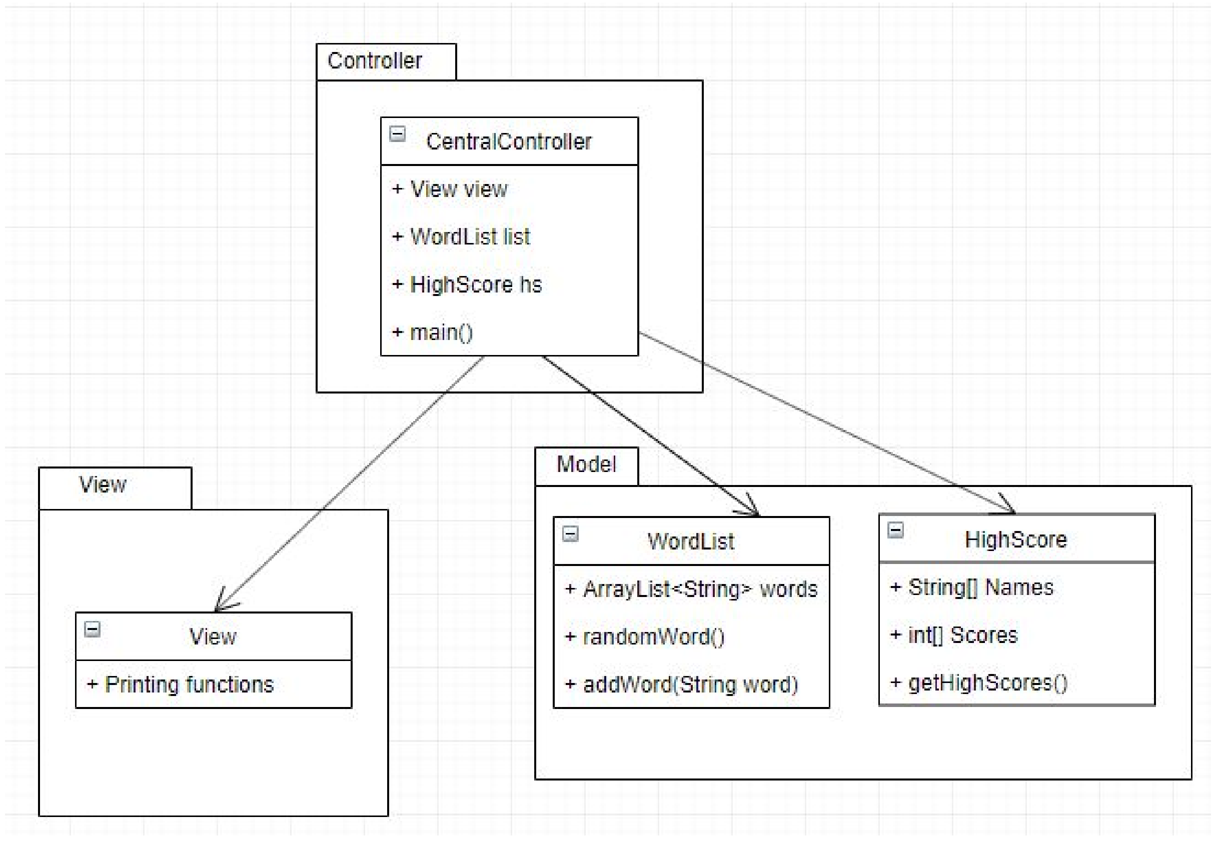
Design document

General plan

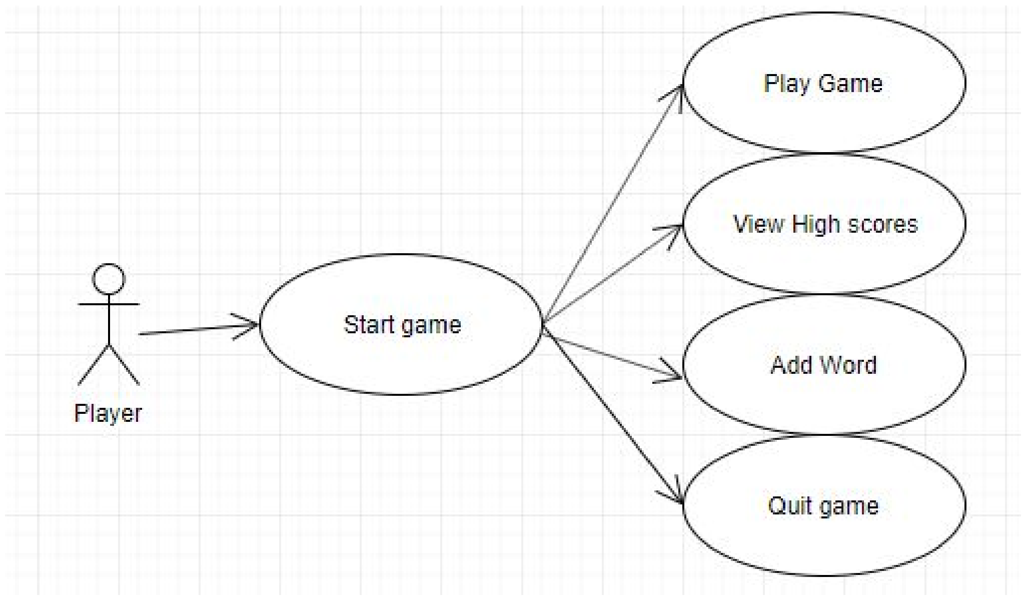
Well, the general plan for this iteration was to make sure that we had all the functionality done for the application. We used the previously created skeleton and implemented the functionality as intended. We had to add a new class to handle the scoreboard as I wanted to keep the separations of concern. Therefore the class diagram had to be modified, and is now looking like this:



So the CentralController class will handle all the interactions between the user and the other classes, while the model will hold the words, the highscore table etc. The View will only handle the printing of menus and other options such as printing the screen for winning etc. The view will not do any of the logic, but simply print the requested commands. All the logic for updating the wordlist or highscore will be contained in their separate classes. The central controller will just handle the interaction between these classes and the user interaction.

Use cases

There are 5 use cases for this application. We have a “start game” which will be launching of the application. The second use case is the “new game” which the main flow and functionality of this application. The third use case is to “View High Scores” where we will list the score of the fastest wins for this application. We also have a use case if the user wants to quit the application and lastly, we will have a use case for adding new words to the application. Below you find the use cases more in detail.



UC 1 Start Game

Precondition: none.

Postcondition: the game menu is shown.

Main scenario

1. Starts when the user wants to begin a session of the hangman game.
2. The system presents the main menu with a title, the option to play, show highscores and quit the game.

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 Invalid menu choice

1. Goto 2

UC 2 Play Game

Precondition: User started the application and chose “start new game”.

Postcondition: Successfully played a game of “hangman”

Main scenario

1. The application gets a random word and displays a gallows and an underscore for each letter in the hidden word.
2. The player guesses a letter or a whole word.
3. If the word contains the guessed letter, all instances of the guessed letter will be revealed. If all the letters have been revealed, the player has won.

Repeat from step 2

Alternative scenarios

3.1 The player guessed incorrectly

2. The application will show an added item to the gallows.
3. If the player guessed wrong 8 times, the player has lost the game.
4. The application will show the start menu. (Use case 1)

4.1 The user made a new high score

2. The system prompts the player to enter his name.
3. The player enters his name
4. The high scores will be updated with the player name and at one of the 5 places in the high score table.

UC 3 View Highscore

Precondition: The game is running.

Postcondition: The high score table is shown

Main scenario

1. Starts when the user enters chooses “show high score” from the start menu.
2. The system presents the high score table and the user is returned to the start menu (Use case 1).

UC 4 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. User selects the “Quit game” option in the menu.
3. The system terminates.

UC 5 Add Word

Precondition: The game is running.

Postcondition: A new word is added to the word list

Main scenario

1. Starts when user chooses “add new word” in the start menu.
2. System prompts user for a word and the user enters a word.
3. The word list will add the word to the list of available words and save them externally.
4. User is returned to the main menu.

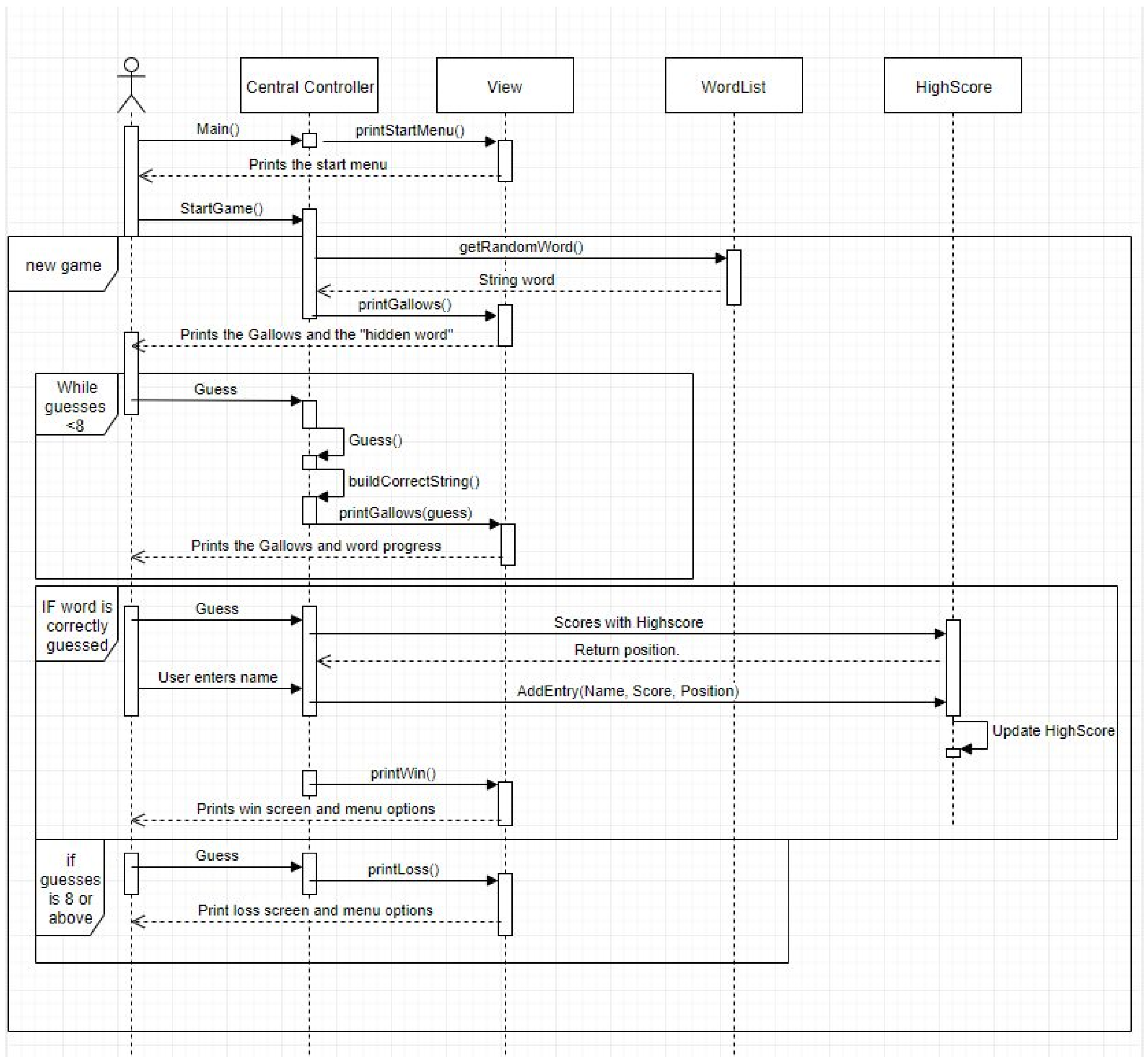
Alternative scenarios

2. The player enters “00”.

1. The application will return the user to the main menu.

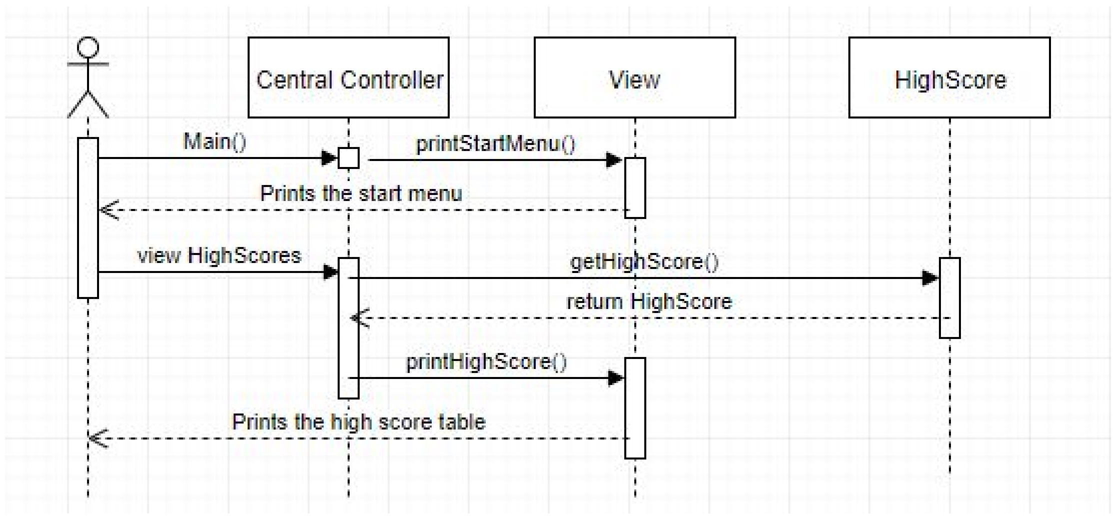
Sequence Diagrams

Play game



First we have the "play game" use cases, which will handle the main functionality of the application.

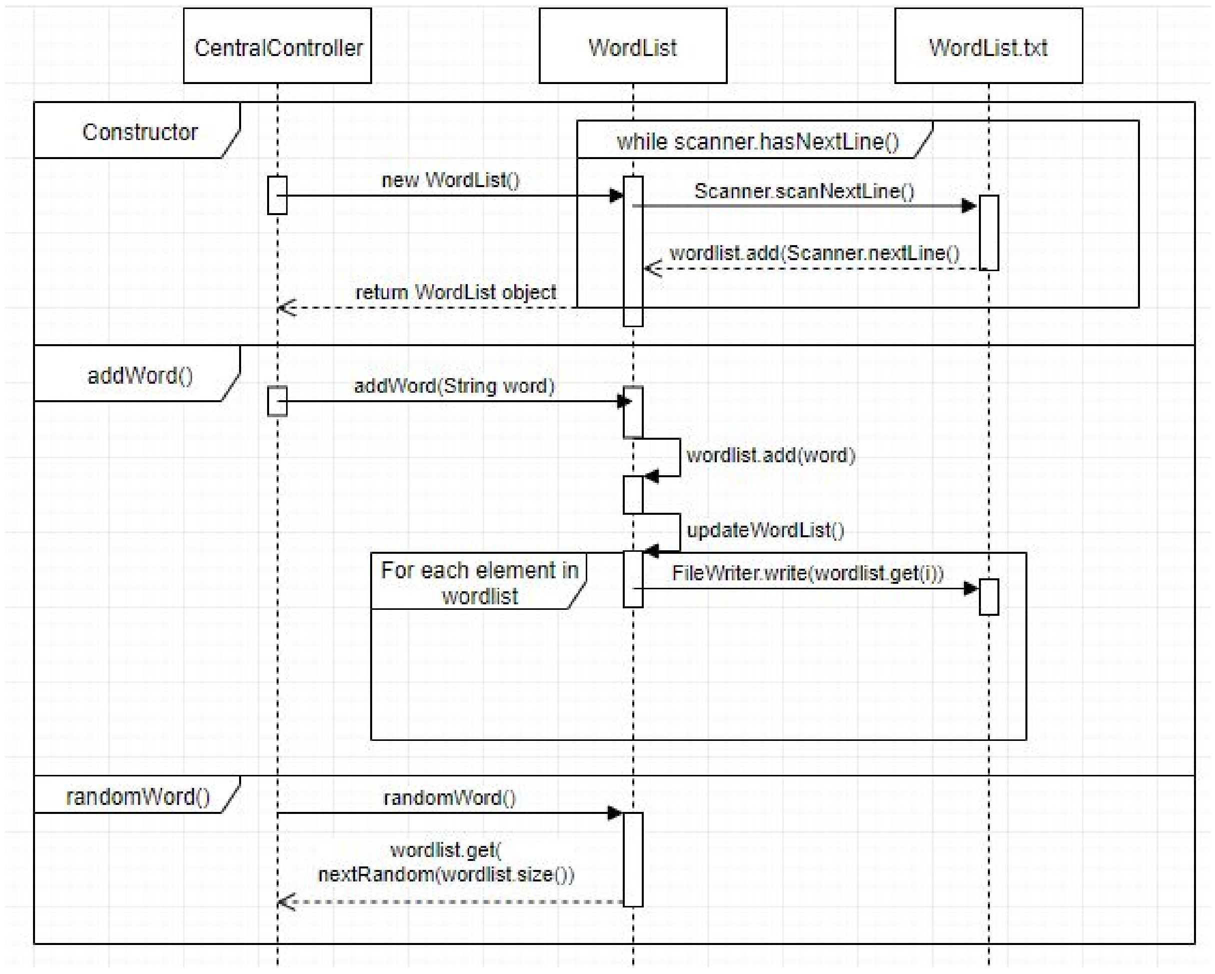
View High scores



This diagram shows the simple flow how the classes interact during the “view high scores” use case.

Add Word and the reworked WordList class

This sequence diagram will cover all the functionality of the reworked WordList class.



So the big changes to this class is the persistence. We added some menu selections for the controller for adding new words to the wordlist, and they are now saved properly in the .txt file. Also added a new print to the View class in order to display the add word option.

Persistent storage

The storage of the high scores and the word list are currently stored in a CSV format in external textfiles. These files are read every time the application starts and all the handling of the high scores such as updating, reading and adding new high scores is handled in the HighScore class. The WordList class will have the same type of functionality and all the functionality will be contained inside its own class.