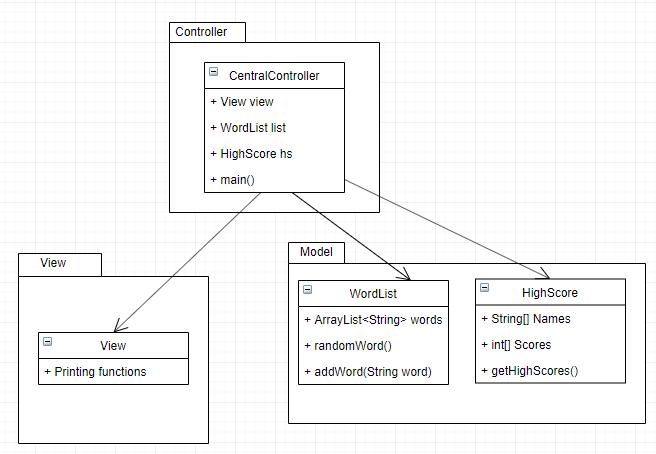
Design document - Second iteration

General plan

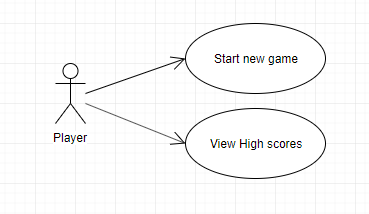
Well, the general plan for this iteration was to make sure that we had all the functionality done for the application. We used the previously created skeleton and implemented the functionality as intended. We had to add a new class to handle the scoreboard as I wanted to keep the separations of concern.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

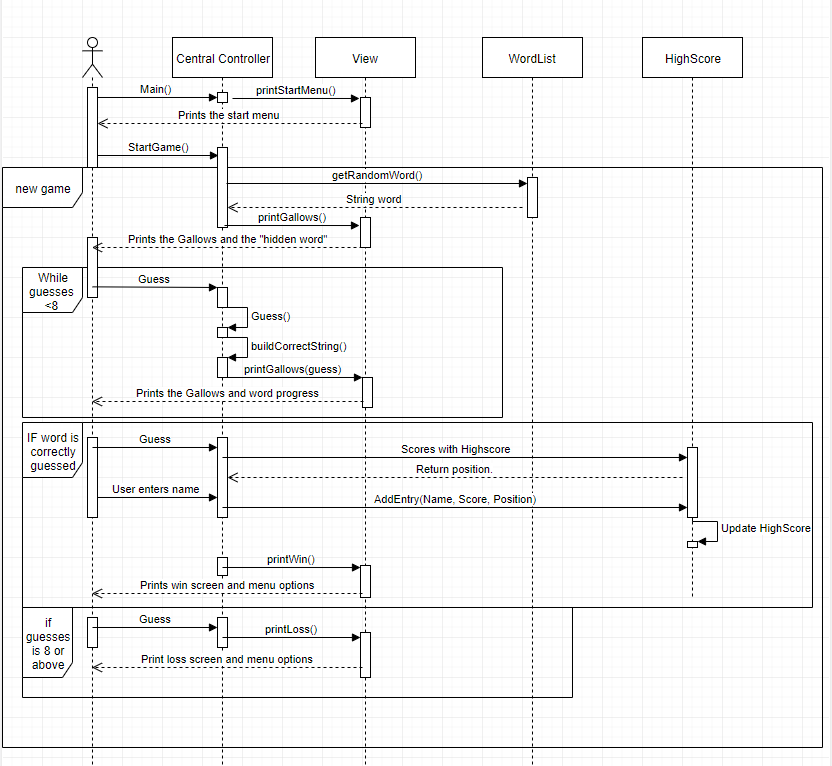
Use cases

There are only 2 use cases for this application. We have a “start new game” which will be the main flow and functionality of this application, and we also have a use case if the user only wants to check the High Scores.

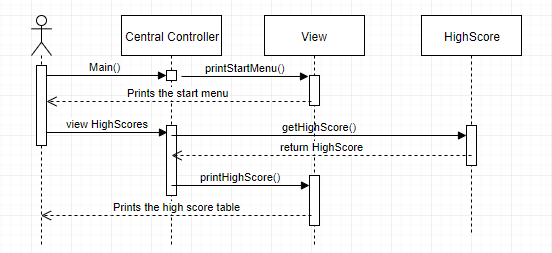


Sequence Diagrams

I created two sequence diagrams for the two use cases which in detail describes the interactions between the user and the application. First we have the “start new game” use cases, which will handle the main functionality of the application.



The second use case is much simpler as its only showing the user the current high scores.



Persistent storage

The storage of the high scores are currently stored in a CSV format in an external textfile. This file is 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.