# Design Decisions

SYSC 3110 Milestone 1, Group 8

Author: Elyssa Grant

Below is a list of the design decisions made for the first milestone of the SYSC 3110 project.

**Letter**

The Letter class contains all the basic information about the letter tiles that make up the basic gameplay loop of scrabble. This class was created due to a need to track not the actual letter a tile has but also the number of points it is worth when given to the board. Players need to be able to see the values and points of each letter but should not be able to change them in any way, leading to private variables with getter methods.

**LetterBag**

The LetterBag class represents the bag of letters that is present in every Scrabble game, which players draw from when refilling their rack or exchanging letters. This entire class was made static, since we only need a single bag per game, and it can then be shared between all players without them needing to hold onto a reference for it. LetterBag’s createBag() method is called at the start of the game to fill its static ArrayList of letters in a way similar to calling a constructor, but it avoids needing to create in instance of the class. LetterBag makes use of a Dictionary so that it can keep track of all the Letter object types, as well as how many are left in the bag. It also uses an ArrayList of letters to help randomly select a letter from the bag, and a Random element for the same purpose. The ArrayList additionally makes the check to see if the bag is empty simpler.

**Board**

The Board class represents the physical board that players place their letters on. The logic for whether a word placed on the board is valid is here, as said logic needs access not only to the letters that are attempting to be placed on the board, but also to the letters that are currently on the board. Board needs to be able to hold letters in a grid to simulate the board of a scrabble game, and thus a 15 by 15 array of Letters was used. Board additionally contains a HashSet, which is loaded with all the valid words in the game. It uses this to determine if the letters placed on the board form valid words or not.

**Player**

The Player class was created with the intention of containing all the logic related directly to a Player interacting with the letters on their rack, as well as holding their score. These two were combined, as they were both things that each player in the game would need an individual copy of, and they frequently interacted. Player has methods that relate to updating the rack and score, but also contains the playerTurn() method, which has the code for the player to interact with their rack and select letters to play or exchange with the bag. Some of this code will be handled by the Controller when a GUI is implemented.

The gameplay loop is somewhat clunky due to the player needing to select each letter and location separately. However, this was done to make converting the gameplay loop into a GUI version easier. The team envisioned having a board of buttons as well as buttons on the rack representing the tiles, and players would click a tile and then the location they want to place it on the board, before finally hitting submit when all letters have been placed. This cycle is what is being imitated with the text-based version of the game now.

The PlayerTurn returning a Dictionary with two ArrayLists was made due to needing a compact way to return two separate ArrayLists from a single function. Since Tuples do not exist in Java, this was the solution that was settled upon.

**Game**

The Game class is meant to bind the different classes together. This class contains the main method, which holds the basic gameplay loop. Its other methods are simple methods to enhance the main gameplay loop, such as adding players and determining the winning player. The Game class was created mainly to prevent the Player and Board classes from becoming tightly coupled with one another, as well as to organize the main gameplay loop of Scrabble.

**Display**

The display class is a simple class that provides text-based representations of elements that need to be displayed consistently. It is essentially a simplified, text-based form of the View, which will be removed in future updates and transformed into the actual GUI View.

**Words.txt**

This text file contains all the valid words allowed in the Scrabble game. It is loaded into the Board class’s code at the start of the game as a HashSet. This text file is used due to the simplicity of editing it rather than hard-coding in adding every single one of the 10,000 possible words manually into the HashSet.