Data Structure Explanation

Board Class:

The Board class uses a 2D array, which is Tile[][] to represent the 15x15 Scrabble grid/table. This gives O(1) access for all operations, when using indexing. Board[row][column]. The fixed board size creates a static array which is great for a grid of 225 spaces. HashMap and ArrayList were considered but they were too complex for the case of a grid where direct access is important in order to validate placement of tiles and calculate scores.

TileBag Class:

The TileBag makes use of an ArrayList to store 100 game tiles, this gives O(1)) amortized time when drawing and also returning the tiles. We also shuffle once by using Collections.shuffle(), then we use remove(size()-1) to draw from the end in O(1) time. The fact that the ArrayList allows dynamic resizing, shuffling ability and the base to allow random drawing of tiles is why we picked it over other alternatives such as stack or queue.

Player Class:

The player class makes use of an ArrayList to store the players' hand. Even though removal of tiles is O(n), it is still good because n <= 7 which is not bad in our case. The ArrayList also gives the option of iteration to display tiles and easy addition when we draw new tiles. We also thought about HashMap so that removal of tiles is O(1) but the high work is not efficient in the case of a small pack of tiles. Which is why we picked an ArrayList.

Dictionary Class:

The Dictionary class makes use of a HashSet to check words. It has O(1) time when we want to look into it which is important for good and efficient game flow. In this case, it is way faster than an ArrayList of O(n) to search through the dictionary. So here, the fact that words are validated pretty quickly prevents any delay that might occur with the use of an ArrayList.

ScrabbleGame Class:

The ScrabbleGame class makes use of an ArrayList to store 2 to 4 players in order to rotate between them. We also make use of an index and modulus to track the current turn and cycle through players. The ArrayList gives O(1) access to get the current player and also a simple iteration to display the scores. There were other choices such as an Array with fixed size or a LinkedList which has no benefit in this case with a small n. Queue was also another choice but it was scrapped since it was unnecessarily complicated to rotate within it in contrast to the ArrayList.