In our project, we broke the core components - the Board, the Inventory, the Tiles, the Tile Rack, and the Player - into a series of classes. Within these classes are methods which are designed to carry out the required functionality for the game. The first, the Board, forms the game board environment by creating a 20\*20 2-D Array. The Board then contains methods required for adding and removing Tiles during gameplay based on the tiles’ locations. It also includes a method that returns a Tile based on its location.

Meanwhile, our Inventory class functions as the “bag” in an ordinary game of Scrabble. Using an ArrayList, Inventory stores the Tiles that are still contained therein during play, and thus it allows only Tiles that have not yet been played to be drawn. It has a “fillBag” method, which fills it with the appropriate number of each type of Tile: 29 of each vowel worth one point, 15 y’s worth two, and 12 of each remaining letter, worth five points. It also allows for a single, random Tile to be removed using a “removeRandomTile” method, and it can return the value of a Tile at a given point in the ArrayList. It can also return the number of Tiles in the Inventory, thereby revealing its size.

For the Tile class itself, we allowed for each instance of Tile to be created with a specific letter as a Char and a point value as an Int. Both of these traits can be returned using the “getChar” and “getValue” methods respectively. When these Tiles are not in the Inventory or on the Board, they are stored on a Player’s TileRack. The TileRack stores its Tiles in its own ArrayList, and it can refill itself with new Tiles up to a maximum capacity of 12. The TileRack can also remove Tiles, remove Tiles from a specific index, and can return its size using assorted methods.

Our Player class functions as a real player in the game. Each instance of Player contains its own TileRack, and each can record its score after each turn; it adds its previous score with its new points, thereby returning a new total score. The Player can then return the value of its score. Tying each of the classes together is the Scrabble class. Scrabble creates a new instance of Board and Inventory, and also stores a maximum of four Players in an ArrayList.

To test our project, we used a series of JUnit tests which checked the functionality of each method in the implementation code. These tests are organized into classes which correspond to the classes in the implementation code. For example, our BoardTest class tests that the initial Board is empty of Tiles, that Tiles can be added to and removed from a given location, and that, once a Tile is removed, that location is now indeed empty. The InventoryTest class verifies that the initial capacity of the Inventory is 400, that the number of each type of Tile is correct in the Inventory, and that we can pull Tiles from a bag, along with a corresponding reduction in the Inventory’s size.

Our TileTest ensures that the character and value of each Tile can be returned, while the TileRackTest goes over its own functionality. It makes sure that its maximum capacity is no higher than 12, and that Tiles can be removed – and subsequently added automatically – as needed. The PlayerTest class then verifies that the Player’s initial score is 0, and that it can add any new points it earns to its score. To the best of our knowledge, our code meets all of the required functionality.