For my object model, we start with the ScrabbleGame, which is the main object which will contain all of the components of the game, i.e. players, the board, tiles. Any command from the GUI will interact with this object, namely placing the tiles, challenging, and starting the game which are its public methods.

The ScrabbleGame has the list of players, letters, and board as public fields. The board is an object of ScrabbleBoard, which we use to check if a move we want to make is valid in terms of location (not off the board, connected to already filled in tiles). Adding letter tiles or removing them is dealt here. Each tile of the board is represented by a BoardTile object, of which the board has 225 (15*15) of them.

A BoardTile has either a special tile or a "normy" tile, which are the standard effects from the original game. It also may have a letter tile or not, signified by a Boolean private field. To check for these things, we have public methods isOccupied() for letter tile and hasSpecial() for an effect.

The SpecialTile is an interface with five different implementations (not drawn out in object diagram for simplicity). The Player also will have some amount of them, while the ScrabbleGame handles all of them. The NormyTile is another interface with four different implementations and has a newScore() function for modifying the score of a word with the modifier (double word, triple letter, etc.).

The ScrabbleGame has a BagOfTiles which handles all the letter tiles. We can draw tiles and check if there are no tiles left for players to draw as public functions. The letter tiles themselves are represent as LetterTile objects, with public *final* fields since the score or letter of the tile won't change.

We have a Move object which is used to reverse a move if a move is challenged successfully. The move consists of *final* fields since they shouldn't change after the move is made. We also have a Dictionary object in ScrabbleGame for checking if a word is valid in case of challenges.