**CSSE 230 Milestone I:**

**Scrabble Project**

Nobleman Charles Eubanks

Honorable Comrade Devon Timaeus

Lord Dharmin Shah

Minion Tyler Nuanes

***GUI Enhancements:***

A menu bar will be used for loading and saving.

In order to play a letter, the player will have to select a letter by clicking on it and then select a tile to play on by clicking on it.

Except for these changes and implementing the currently unimplemented buttons, no other changes are planned for the GUI.

***Algorithms and Data Structures:***

To load the dictionary, a Trie structure created from modified Node class will store every legal word and look up legal words quickly.

To keep track of human and computer players, there will a ScrabbleGame class which determines the gametype, whose turn it is, and human player’s movies.

A ScrabbleTile class will hold the character value of the letter, the point value of the letter as a short, any point modifier the tile is on (stored as a character), the position on the two-dimensional gameboard (stored as a pair of shorts), and a boolean value keeping track of whether letters can be added anywhere beside it.

The tiles in a player’s hand will be stored in a ScrabbleTile array of length 7.

A ScrabbleAI keeps track of which tiles a player has in their hand, stored in an array of characters, as well as the best word, stored as a string, that has been found for computer players as well as that word’s score, stored as an integer. In addition, the ScrabbleAI will hold a priority queue of ScrabbleWords that can be created from the ScrabbleTiles in the player’s hand where priority will be determined based on the point-value of the words without any bonuses. Finally, the ScrabbleAI will hold a second priority queue of ScrabbleWords based on words that can legally be played on the gameboard; again, priority will be determined by point value, though the points will include bonuses as well.

To store the gameboard, a fifteen-by-fifteen, two-dimensional array of ScrabbleTiles will be implemented.

A ScrabbleWord class will hold an ArrayList of the scrabble tiles, an integer of the value, any word point modifiers the tile is on, and an integer for the length, as well as a string for the word.

To determine the highest-scoring play, the program stores words that can be created formed with letters in the player’s hand in a priority queue, keeping track of the maximum score each word creates. When the hand tiles runs out of possible words to create, the algorithm will traverse the gameboard, searching for letters that are contained in the highest-scoring word. If a position is discovered, the algorithm will store the ScrabbleWord with ScrabbleTiles containing location in a priority queue based on the ScrabbleWord’s score with bonuses. The program shall then continue through the board, searching for more valid locations to place the highest-scoring word. After exhausting these searches, the priority queue will poll the highest-ranking SrabbleWord and the ScrabbleTiles will be added to the gameboard. In the event that the highest-scoring word cannot be placed, the computer will search letters contained within the second highest-scoring word on the gameboard. More specifically, our code will loosely follow the description found in Steven A. Gordon’s 1994 paper “A Faster Scrabble Move Generation Algorithm” (East Carolina University).

***UML Diagram:***

The