15-214 Homework 4

Muhammad Ahmed Shah
mshah1

Behvioral Contract

When a player tries to place a tile, first it has to request the `GameEngine` to provide it with a `move` object. If the player is not the `activePlayer` or `gameOver=true` a null object will be returned and hence not allowing the player to make moves out of turn or in an inactive game.

If the above conditions are satisfied and a `move` object is porvided, the tile along with its corresponding location is stored in the `move` object temporarily. When the player calls `placeLetterTile` or `placeSpecialTile` to place a tile at the specified location on the board, the `BoardSquare` at that location is checked if its vacant using the `isVacant` function for the `BoardSquare`. So we define an invariant that every tile-location pair in a `move` object refers to an empty `BoardSquare` or one which has a `SpecialTile`.

When the player calls `endMove` the tiles in the `move` are placed on the board and then `board.validate` makes sure that all the tile placements form valid words. In doing so it adds the tiles and their corresponding locations to the `letterTilesOld` hashMap in the `move` object. So here we can define as a precondition that when `executeMove` is called the tile placements form valid words.

`executeMove` checks the locations occupied by the in `letterTiles` have a special tine underneath , if it was then it is added to its `activate` function would be called and the state of the board (in this case the changes would also be reflected in the `move` object) or the behaviour of the game would be altered accordingly. The execution of the `activate` functions happens independently so if multiple `SpecialTiles` were hit then their respective `activate` functions will be called in the order they where hit and their effects would be stacked.

Then the `moveScore` method calculates the total score of the move by using the `getSquareScore` method in each `BoardSquare` referred to in the, now altered, `move` object. The `getSquareScore` method returns the point value of a `LetterTile` multiplied by the defined multiplier (in case of a double or tripple letter score square). If a multiplier has been applied once the `multiplierUsed` attribute of the `BoardSquare` is set to true and all subsequent calls to the `getSquareScore` function will use 1 as the multiplier. Since the locations of the double and triple word score squares are predefined in the board, one scan through the list of locations can tell us if any of then are occupied and then the appropriate multiplier can be applied over the total score. Once again the `multiplierUsed` attribute will be used to keep track of the multiplier squares already used.

The calculated score is stored in the `scorer` by a call to the `updateScore` function.

To Summarize:

1. `@requires: !(game.gameOver) && player.equals(game.activePlayer)`
   `Player.newMove()`

2. `@requires: for every Pair(i,j) in move board.isVacant(player,i,j)`
   `@ensures: true if the tiles in move form a valid word themselves and with the adjacent tiles.`
   `board.validate()`

3. `@requires: board.validate() == true`
   `@ensures: score is the sum of the point value of each tile adjusted according to`

```
the appropriate multiplier, any special tiles hit cause the effect on the game as
intended
board.updateBoard(move)
```