hw4a Response to Feedback

==========

****IMPORTANT:** Please respond to questions about your design in a file called `/homework/4/response.pdf`. This will count as part of your Milestone B grade!**

Note: I have not made the changes in the design documents yet. I will do so by Milestone C. However I have explained the changes I have made below.

Domain model (10/10)

*** Specialness is a little odd for an attribute, but otherwise it seems fine.**

Specialness was an attribute for if a tile was special or not. i.e would probably hold a value of 0 if not special and 1 if special. However while implementing the core Normal letter tiles and Special tiles are separate classes.

Object model and rationale (38/40)

*** -0.1, It's unclear from your object model how a game is started. Is it just calling constructor? What are the advantages and disadvantages of your approach? Compare this with the code we provided for TicTacToe (recitation 8)**

The game would be started by a call to the constructor of the class Game in the core. I have added implementation which is now similar to the tic tac toe implementation, where there are game listeners which subscribe to notifications made in the core implementation. This would make good sense since there are many elements in the GUI that would depend on stuff happening in the core implementation. For now only an interface has been created so that the GUI know what notifications it can have from the core implementation. I might add some more notifications while creating the GUI if I feel the need to know more from the core.

*** -1, It's unclear how a special tiles can be purchased. Are the tile purchases random? How is the GUI notified how much the tile cost?**

I had not accommodated this in my object model. I have now accounted for it. When a player clicks on a purchase button the purchase( ) method in the Game class gets called which would call the purchaseSpecialTile( ) method in the Player class and based on the player's available points the special tile would then get added to his special tiles rack.

*** -1, `TileBag` should have a method like `isEmpty`, or you cannot know how many tiles you can draw from the bag, or when the game is over.**

This has been changed. TileBag now has a method getTileCount( ) which tells the game how many tiles are remaining

Design Scenario - Move validate (13/20)

*** -2, You should check that there is no empty spot between the head and tail of all placed tiles (i.e. all placed tiles are connected).**

This has now been taken care of. If the player tries to place a tile with no adjacent tiles it would fail. The method hasAdjacentTiles(Location loc) takes care of this.

*** -2, The rule that all tiles in a move must be collinear (Horizontal or Vertical) is not checked.**

This has been taken care of with the methods isCollinearColumn( ) and isCollinearRow( )

*** -1, You did not check if the played tiles belonged to the player. Basically you shouldn't rely on any client (in this case the GUI) having robust code so any reasonable checking you'll need to do for them.**

This has now been taken care of since all players now have attributes of currentTurnPlayedLocations which would store the locations at which the player has played at for the current turn.

*** -2, You should check that at least one placed tile is adjacent to an existing tile on board (or at the center if no tile currently exists on board)**

This has been taken care of with the hasCenterTile attribute and the hasAdjacentTiles( ) method

*** I don't think `isAdjacentEmpty()` is actually doing anything. You should re-think your design with the points given above on how to validate moves.**

A hasAdjacentTiles(Location location) method has been created to validate if a tile is placed on a valid location. This method checks tiles on all adjacent locations.

Design scenario - Actions after player places a word on the board (18/30)

*** -5, Your solution does not place the player’s move on the board at some point after it is made. You need to modify your Board.**

This has now been taken care of the method addTileAt(Location loc) would add the tile to the location on the board.

*** -2, You did not remove the played tiles from the player's hand.**

This has been taken care of. Every time a player takes a tile from the tile rack, it is immediately taken out of the rack . using the removeTile( ) method in the Player class. Even when a tile is placed on the board the tile is removed from the hand of the player using the tileOnHand attribute.

*** -4, You do not have a way to access and modify the score for the move. You will not be able to implement `NegativePoints` tiles.**

This has now been taken care of by sending parameters to the applyEffect(Game game, int score, Location loc) method when a move is submitted. This score can now be changed before updating the player's score.

*** -1, You did not refill the player's hand with tiles after the move was applied.**

This has now been taken care of in the startNewTurn( ) method where the number of tiles to be added are calculated and then added from the tile bag.

Additional Notes

*** Overall this design is decent. One thing you should focus more on is what parameters go into each method. This will help you think about what is necessary to compute everything rather than thinking in general terms like "I'll modify the board".**

*** Having Turn take a turn object in `turnPlayed()` is little weird, so I'm going to assume you meant to pass something else.**

I did mean something else. Now the method is called SubmitMove( ) which does not take any arguments but simply checks the validity of the move by checking what the current attributes of the game possess

*** Keep in mind you can regain large portion of your points in the later milestones, so don't be discouraged!**

Graded by: Andrew Zeng (azeng@andrew.cmu.edu)

To view this file with formatting, visit the following page: https://github.com/CMU-15-214/rvora/blob/master/grades/hw4a.md