Omar Syed
15-214
October 31, 2013

Scrabble Design Discussion

With this full implementation, some changes had to be made to the core implementation after seeing how they worked with the GUI. First and foremost, errors from Milestone B were changed and Java Docs were added to make the code clearer. Also, large functions such as submitMove in the Game class were made into smaller components. The Hand class was also removed because it wasn't really anything besides a class that held an ArrayList, and so I just gave the Player class that ArrayList. Also I noticed that for keeping the GUI updated, a lot of the methods in the Game class should return a boolean to let the GUI know if the method worked successfully. This way I was able to easily tell if a word was not valid and have a pop up come up alerting the user as such. I also hadn't considered how to end the game in Milestone B. I decided to make the game over when no players can do anything which would be when all players consecutively skip their turns. This also required adding a field in the Game class for game over.

I still kept the initial design of having all the core components interact with each other through the Game class for low coupling but the trade off is that sometimes one action requires calling a method in one class that calls a method in another class, instead of calling the latter's method directly. I believe this is still optimal design because for the initial class calling the method, all it needs to know is that it told the game to do something, it doesn't need to know what other aspect of the game actually did the task as long as the Game class took care of it. Similarly, when a class is receiving a task in the form a method call, it doesn't need to know that another class was requesting the information from Game it just needs to do the task.

I used a similar strategy for the GUI. There are two different viewable screens, the start menu(StartMenu), and the actual gameplay (GamePanel). Each of these add the various components to themselves and then facilitate communication between the components. This again supports low coupling because each GUI component only needs to send information or receive information from the main

JPanel which is either StartMenu or GamePanel. The GamePanel then interacts with Game in the core package through a mediator class which holds the instance of the game.

Extra things I added include fully functional tile and word modifiers (double word/letter and triple word/letter). I also added an exchange tiles button which allows a player to skip their turn and exchange the tiles in their hand. Lastly I added a skip turn button which also determines how the game is ended. If players cannot do anything and cannot exchange tiles because the tile bag is empty, then they have to skip their turn. If all players consecutively do that then they cannot do any moves and so the game ends.

Overall, the design process was a great experience which taught me how important diagraming out a project can be. After actually implementing the GUI I learned that when creating the core implementation, more emphasis needs to be put on how the GUI will find out about changes in the Game's state. I also found that testing for 100% coverage was not completely necessary but more importantly a few run-throughs of the game should be done instead. I was glad that I organized the code very modularly so that adding features was not too difficult.