Rationale

By Chao Li (chaol1)
Updated March 27, 2014

My design changed gradually during my implementation the core classes and the GUI stuff. I still use the basic domain that I finished during the milestone A, but I changed the object model much because I found many details ignored during the milestone A, but they definitely made difference when I write the core classes. In the following paragraphs, I will first introduce my domain model and then emphasize my design referring to my object model and my design strategies.

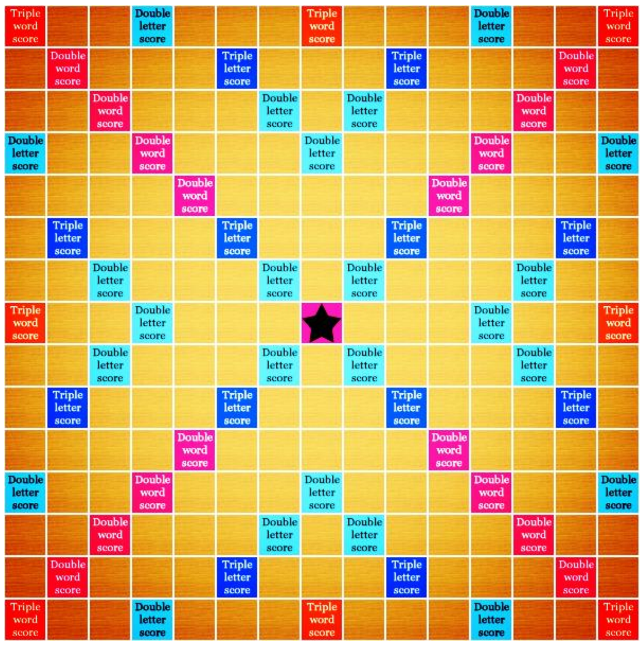
Domain Model

Since I have never heard about this game before, I first referred to the Wikipedia page and videos. Then I tried to abstract the key concepts according to the rules of the game. Game is the framework that we need to hold all other objects and control the overall process of the game. When we add the GUI in the future, the GUI will mainly talk to every object through game. So this is the fundamental object in our project.

Then we need Player object, which is the participant for this game. The player object will provide all variable and methods that a real player need during the game. In addition, since the tiles are stored in player's rack and the rack also some executes some command the player, I made the Rack as an independent object. Every player has a rack, and every rack holds 0 to 7 seven tiles.

What's more, the game also needs a tile bag, which contains 0 to 100 regular tiles. So I abstracted these two as two objects, the tile bag stores the number of tiles and return a tile for the player. And the regular tile has a letter and corresponding points.

Next, I abstracted the board that holds 225 board content like the following figure. This board is like a container so that we can put features, special tiles and regular tiles on it.



In addition, to check if the player move right, I set a board referee class, which plays a role to record new words, new tiles and new locations and make a djugement to the behavior of the player.

Object Model

During the milestone A, I just considered the two scenarios of interaction diagram and informal questions after, and then added methods needed for every object in the Domain Model. Finally, the object model is finished.

Although it basically defined all my classes, they were not enough for all. So I added new methods and classes to my model when I implemented the program. In the first phase, I started from classes which were shown in the "leaf" of model. When I say leaf, I mean such classes do not have much associations such as aggregation with other classes. For example, I started from Tile and Location class because they have little connections with others.

In my design, I tried best to design for the division of labor, so I separated the GUI from core, and also divided labor within the core folder. The Scrabble Game class controls the game overall and it holds the Player, Board, Board Referee and Tile Bag within it. To increase the cohesion within each object, I abstract every object and made each object just did what they must. To be specific, Player class needs to draw tiles, use tiles, buy special tiles, exchange tiles for the user. And each player also needs to control its rack, that's why I separate the rack class from player, in which way the player will decouple from Tile bag.

What's more, I have a board class which holds a 15*15 board content class, like the square class in class. The board is responsible of placing tile on board, removing tile, knowing what tiles are on board now. In addition, the board also needs to tell the game if the location is empty.

Another important part in this game is to judge if the play made a player is valid. Since it is a big task, I create a referee which holds a board to check the activity of the players in the game. And referee knows all rules in the game, have a lot of methods to check the play.

To further divide the labor of the board, I create the board content class to hold all the tiles, special tiles and board feature on the board. This class is only a container to put tiles.

For the special tiles, since we may have many more kind of special tiles in the future, I make the special tile an abstract class so that it act as a framework to allow more special tiles to be added through extension.

My design emphasize the division of labor and extensibility in the future, and the high cohesion and low coupling is also shown in my object model.