Rationale

Zhilin Huang – zhilinh

First, I will introduce the domain model to give a whole picture of the program. We have a Scrabble class for the control of the game. It has some players, a tile package to store tiles, a playing board which will be represented by GUI and a counter as timer. When the game starts, we get multiple players with their names entered. They will be several classes with name, score, the number of tile, an inventory and move as attributes to record their behaviors and the effects of them. All these attributes will determine the status and result of the game.

Second, we are going to introduce the move method. Players can achieve many stuffs by the Move class. For instance, players can get tiles from the inventory, set tiles they have to the board, buy special tiles from the shop and place them on the board, trigger special tiles to have some effects on the game, challenge others' tiles validation by dictionary and gain scores by placing tiles, etc. For the Board class, it has a lot of methods to make sure the game is on the right track. For examples, it stores the location of tiles and special tiles by a HashMap (the data structure could be discussed), places and removes tiles from the board, collects the word that tiles make up and so on. It passes the game status to players, dictionary, tiles and special tiles to check validation, make a decision and gain information.

Third, we have SpecialTile and Multiplier as two interfaces to achieve the special game effects. Because these two stuffs have many similar sub-classes to implement, it is a good chance to use polymorphism to achieve rather than writing respective but repetitive classes. Moreover, we will add unknown effects and self-defined effects on them so we use observer pattern to implement. It makes sure that when the game is going on or adding other stuffs to the game, the game and the interfaces won't change each other.