For the design pattern, I change the creation of the special tiles to the factory design, in this way, I can use method to create the different types of special tiles as we want without calling their constructor class. The changes of my design are basically reflected at the I/O level for different classes, like when I determine whether the special tile is effectively triggered, I will import the flag parameter. Move over, I add something to the core classes for the convenient of implementing GUI, like I define different colors for different multiplers, in this way, we can draw the board in the game panel easily.

For the new added special tile, I change a lot for it. First, I change the upper bound number of player's inventory, because my previous design is to set the upper bound to be 7, however, if the player steal the tiles from other users, its inventory will definitely exceed 7 tiles. So after I change the upper bound number of the inventory, I judge the conditions before the game refill the tiles for the current player, like if the player has 14 tiles at this round and he places 4 tiles on the board as the move, then we don't need to refill his inventory. Then I need to determine whether the stealMove tile is current player's when it gets triggered. Moreover, I should think about the condition that the Boom special tile and the StealMove special tile are triggered at the same time, because for my previous design, the Boom special tile will delete the scores of the action and remove all the tiles on the board within the range, but this time I have to directly assign the steal tiles to the beneficiary and victim will not be deleted scores since he/she gets nothing for his/her current move. So after all, I change a lot for the I/O parts of my classes and increase the judgments for the specified action.