A class named GameSystem is created to maintain the whole process of this game. One GameSystem class have one GameBoard and 2 to 5 Player classes here.

The explanation of the classes and some explanation for the Junit test is in the README.pdf file for homework 4b.

Once starting the game, the GameSystem would be initialized by the constructor according to the given player number. When the player chooses to pick a ramdom tile, the system would pick a tile from the left tiles randomly. The system should guarantee the random tile could be placed into the game board validly, if the random tile is not valid, then the system should discard this tile then pick a new random tile until this picked tile is valid. After picking the tile, the user should decide the position for this tile to be placed on the game board by calling the placeTile function in the GameSystem. The user could also choose to rotate this tile to find a valid position by calling the rotate function of the Tile class. The GameSystem should check the placed position is valid or not, if the answer is no, the system would show error message to the user.

After the placement of this tile, all the features, including farm features, road features, city features and cloister features, of this tile should be checked whether they can be added to one or more segments. If the features of this tile cannot be added to any segment, then new segment should be created and the according features should be added. If one tile connected two segments into one segment, these two segments would combine to only one segment, and the followers of these two segments would all be added to this new combined segment.

Then the user need to decide whether to place a Follower on the tile or not, and decide which feature should be covered by this Follower. If the user decide to place follower, the placement would be realized by the placeFollower function. Before the follower is really placed, the GameSystem would check the left followers number of the specific player to decide whether this player could place follower any more. In addition, the GameSystem would also check the segment, which is the follower going to be added, already has a follower or not. If the segment already has a follower, then the player could not place follower any more.

After the placement of the follower, or the player decides not to place follower and

give turn to the next player, the GameSystem would update the score of each player by calling the updateScore function. The system would check all of the segments to see whether they are completed or not, if one of the segments is completed, the score of the host player would updated and the followers on this segment would be returned to the host player. One thing to pay attention is, during the game, the system would only update the score of farm, city and cloister segments. Only when the game is over, the score of the farm segments could be updated.

The left users would repeat the same action one by one until one round completed. Then the game would start over from the first player to repeat the same actions in round one. The game would end then there is no Tile left in the GameSytem.

When the game ends, the system would update the farm score and find the winner who has the most scores at this game.

To improve the code reuse, I set the Segment as interface to represent the different segments on the game board and use City, Road, Cloister and Farm classes to realize this. And TileFeature interface was created, then using CityFeature, RoadFeature, CloisterFeature and FarmFeature classes to realize this interface. The use of interface would decrease the coupling between similar classes, which may share many similar properties or functions.