Discussion

These are some major changes I made from milestone A to final implementation.

Change 1: In milestone A, I separate place tile and place meeple into two functions, in milestone B, I combine them together by adding meeple Position parameter in place tile function. This is because it is more natural that player considers the tile placement and meeple placement together and finally makes the move. Also, this change makes my code logic for placing a tile clearer.

Change 2: In milestone A, game system class only communicates with game board class to handle all the necessary functions (ex. Place tile, get score, get returned meeple). Later, I found that game system should be able to call other classes for some functionality. When comes to scoring, game system asks for complete features from game board, and then game system calls directly to these complete features to get winners and scores. This changes make me easier in program development.

Change 3: In milestone A, I am not very clear how each feature is represented and how can the program check whether it is complete. Later, I made the design that each feature is consisted of a set of edge features, each edge feature has a position and an orientation (and for sure they belong to one feature type). To check whether the feature is completed, all the edge features in this feature can be grouped into pairs (pairs have conditions for positions and orientations) and no edge feature should be left.

Change 4: In milestone B, each feature contains a list of tiles and a list of meeples, but during GUI testing, I found that in some cases there are some bugs in scoring features and returning meeples. I found that when combining two features together, list type can cause duplicates for tiles and meeples, so in milestone C, I change list to set, so there should no duplicate in set and this change turns out to be great.

Change 5: In milestone A design, my Player class owns a list of tiles. Later I found that this is not necessary because there is no directly communication between player class and Tile class. When comes to scoring, its depends directly on meeples not players. The responsibility of Player class is to store scores and number of used meeples.

Change 6: In milestone A, in game board class, when checking for whether this placement is valid or not, I only consider whether it matches its adjacent edge features. Later, I realized that it should first check whether there is at least one abut tile, then it checks whether this tile matches the adjacent edge features, it should also check whether the specified meeple position is valid (this is the bug I found in milestone C).