Discussion - Changes Through Milestones

AndrewID: chenxili

Through three milestones, lots of changes were made based on the real conditions' needs. In this document, the actual changes that made comparing to the original design would be discussed.

Change 1: Segment Representation

In the original design, "Strategy Pattern" and delegation was chosen to represent the different variants of the Segment. But when doing the real implementation in the milestone 2, I found that it is unnecessary to construct each variant of the Segment since the only difference of them is the "Type". It would save lots of effort and time if we only use one Segment class with an attribute named "Type" to separate them.

Change 2: Remove the Meeple Class / Redesign the Meeple Implementation

In the original design, Meeple object was designed to be represented by a single class with attributes like Segment, Position and Owner. But after a rethink in milestone 2, if only considering the aspect of calculating scores, there is no need to know where the Meeple is and which Segment it actually sits on. What really matters are how many meeples are there in a feature and it belongs to whom. We can simply use PlayerID to represent a meeple's owner. So we do not need a Meeple class to represent that. We can simply maintain a Integer list to store the PlayerID, which represents the players that have placed their meeples on this feature.

For the aspect of displaying the position of the meeple in GUI, as the player can only place meeple on the current tile, we can simply let the user to choose which segments on this tile does he want to put the meeple on. Then a circle would be drawn on the TileButton. There is no need for a Feature to record which exact segment has a meeple, but a Feature need to maintain a list of Position that has a meeple because after a feature complete, we need to remove the circle on the tile. As there would only be at most one meeple on one tile, so we simply go through this Position list and change the image of the TileButtons on these Positions to the original image without a circle.

Change 3: Applied Polymorphism to differentiate different feature

In the original design, there would be three HashMap that store as "Meeple - CityFeature/RoadFeature/MonasteryFeature" pair. But on the second thought we can see there is no reason to store the Feature on the Gameboard like this since there would be two or more meeples on the same Feature. So in milestone 2, I decided to use three List to maintain these three kinds of Feature variants for score calculating. But in milestone 3, I found that it would be much better if we simply used one single List to store all the features and use polymorphism to differentiate it. So that if the game has a brand new Feature variant in the future, this new type can be easily added into the core without having to change the GameBoard code. So in milestone 3, one single List is used to store all the features, which significantly improves the reusability. Many methods in GameBoard like “placeMeeple” and “checkFeatureComplete” also changed at the same time.

Change 4: Remove Unnecessary Methods in GameSystem class

In the original design, there are several methods like “void startGame()”, “void endGame()”, “nextPlayer(int ID)”, which is totally unnecessary. Since the game will automatically start and end, we can simply use a single method called “void nextTurn()” to perform all the actions like updating the tile and playerid, checking if the game should end and so on.

Change 5: Avoid Scattering the user-visible API across many different classes

It is very important to know that users should only be able to call the GameSystem API as the “controller” class. In the previous design, the user could access the GameBoard or even Tile class which cause user-visible API across many different classes. In order to avoid this, the methods in GameSystem API are re-designed. Some methods like “GameBoard getGameBoard()” and “Player getPlayer(int PlayerID)” were removed to avoid the user to access other classes. Some methods like “void rotateCurrTile()” were added so that the user can perform these actions directly through the GameSystem. In the GUI implementation, there would be only access to the GameSystem class rather than other classes.

Change 6: Modified the Player Class Design and remove unnecessary attributes and methods

In the original design, the Player class has several improper and unnecssary attributes and methods like “currentTile”, “placeTile(Tile tile, Position position)”. We do not need to know what current tile that a player holds since we can always access current tile through GameSystem API. Also, we can place the tile directly through the GameSystem. It is important to clarify it is user to implement with GameSystem API to place tile or place meeple rather than through Player class to do that.

Change 7: Override equals, hashCode, toString method in necessary class

Override equals, hashCode, toString method in Segment, Position, Player, and Tile class for comparing usage and better information displaying.

Change 8: Do Deep Copy of all the Collections object

In order to protect the privacy and security of the object, the accessor of the Collections type attributes in a class needs to return the deep copy that attribute. Some are missing in milestone 2.

Change 9: Correct the score calculation bugs exist in milestone 2

There is miss typing of RoadList to CityList in the “void updateRoad(Tile tile, Position position) which causes score calculating error.

Change 10: Change the representation of the Position

In milestone 2, the second quadrant rule of representation of x and y was applied for core realization. But in milestone 3, I noticed that the GUI Swing JPanel uses the fourth quadrant rule of representation of x and y, which makes the core code wrong. Many methods that implement with a tile’s Position like checking tile placement or merge features or check feature complete are changed due to this.

Change 11: Pre check for the Valid Tile

It is possible that a tile was given and there is no valid Position for it to be placed on. So the GameBoard changed to pre-check the Tile first before updating it to be the new “lastTile”.