Discussion:

Due to the precise planning and exhaustive examination of design goals in the UML prototyping period, the design changes in the rest of the project is fortunately minimal; I am able to adhere to the original design and implement most features without much difficulty; However, there are some design changes and potential but reject design revisions that I will elaborate here:

Milestone B:

1. [Proposed but rejected] Making features objects that can store the tiles contained in this feature and perform calculations using its own functions: I thought of this during the implementation of 4B, yet I do believe that this design is inferior according to my evaluations; I choose not to use this design as this would make the scoring process more embedded and coupled, and less likely to be extended and modified for performance; Also, such an approach can have a heavy performance impact, especially if there are many player-meeples and many features to constantly maintain; My approach seeks to represent feature as nothing more than a collection of tiles in a stateless format; This gives great flexibility in both general scoring processing and applying specific scoring rules that will give the most optimal performance; In general, although it makes the code slightly more complex, I do believe that the flexibility and performance is worth the trade off;
2. Renaming the “Score Controller” as the “Game Rule Controller”: I have renamed the score controller as the game rule controller as the latter is more suitable in describing a class that is responsible for scoring meeples on the board and validating meeple placements; alternatively, this class could be named as “Meeple Controller”, but such a naming may not be appropriate as later extensions may add more game rule enforcement requirements other than simply controlling meeples;
3. Added representation class “Tile To Edge Association”: this is a very minor design change that adds a class to aid readability and representation of the code; After making this design change, I will not have to separately pass in tiles and edges for scoring, but can pass the whole association between tile and edges; This does not change any design elements, and is merely an addition to the representation;
4. In the original design, I have intended that the scoring process used not only edges but also corners for scoring, as I initially thought that corners will be necessary to score cities that are disconnected and distincted over tiles; in the implementation process I discovered a clever trick that eliminated the use of the corner segments by using pathfinding over the tiles; however, I still left the corner segment and all fully functional corner segment manipulation functions within the code base as these may become useful in later expansion packs for carcassonne, and may also be very important for the integrity of the framework;

Milestone C:

1. Nothing in the core implementation was changed other than removing the text based gameplay feature in the main method of the top module class; all features are working

as intended from the initial UML design; some patches were made to the scoring algorithms as bug fixes;

2. Added the listener interface that was necessary for the Observer pattern; Also added notification methods that will notify subscribers of the Observer pattern in various parts of the game; the overall game logic is not changed and most of the code was adapted from the text based carcassonne game; due to my design foresight into the observer pattern, nothing much required extensive changing;
3. Added the GUI elements that are subscribers of the core game; these are not considered within the game design and they did not change the game design in anyway;