So for the most part my core implementation of the Carcassonne game did not change and worked surprisingly well. When it comes to the GUI I decided it to design it in 5 different parts of the game: the start panel, the tile panel, the player information table, the meeple placement selector and the game board. I decided to make all of these JPanels and then add them all to one big JPanel that was finally added to the main JFrame.

I that for the start game I would simply use a JText field for the user input and a button that initiates the start of the game. Instead of making the start button and user text field a combined class I choose to implement the functionality inside of the main function. I choose to do this because it will have easy access to the main game, and the rest of the panels in this scope of the program, meaning it can easily update any information in the listener function without having to have a constructor that aliases these panels.

For the rest of the panels I decided to implement them in different classes that all extended the JPanel class so they could easily be added to the main JPanel and then to the JFrame. One problem that I encountered when implementing these frames in their own separate classes was how I was going to update the other panels when something on the current panel changed. This is an easy fix with java swing because the use of the observer pattern and the action performed function. The problem I encountered with this is that there is not an order for what actionlistener function is performed first and for my case sometimes I needed the game data to be updated before some of the text data was updated. So the way I got around this was in each class that has some type of button or listener functionality, I would have the class have a private variable for the other panels, and a setter function so that I could access those panels in the action performed function inside of that main class. I would then immediately set the value of those panels after I initialized them in the main play Carcassonne function.

One last design choice I had to make was how I was going to represent the board in a GUI sense. I decided because it was to slow to load all 144 possible tile locations I would start with a 3x3 board with the main tile in the middle and then when a tile gets placed the board will re-size itself so that a new row or column is added to the board. This ended up working well, the only problem is that there is not a lot of customization for the layout of a JPanel, and so the board was the same size as the rest of the panels, which does make a lot of sense, but I could not figure out how to change this using swing, so it uses a jscrollpane which allows the players to scroll to the location they would like to place the tile.

Overall my GUI design process followed a basic structure that all my panels implemented, and it worked out very well, except for the layout manager in swing. I was not a fan of how swing managed the layout of its elements and offered very little customization, on paper I had a very clean idea of what I wanted the GUI to look like, but this was nearly impossible to accomplish using swings layout management. Maybe in the future or if I were to use a different GUI in another life, I would have looked for a more flexible alternative to swing.