Project Reflections: Alpha Merge

Sirisha Bhupathi

Introduction:

I was not super enthusiastic to start the project as I still could not fully grasp the concept of Classes. Used to programming in a linear fashion, Classes seemed unnecessarily complex. Ironically, as things often do, this ended up being the most fun project for me so far. And not as complex as I thought it would end up being.

Design:

Determining classes:

While board/tile and user classes seemed quite straightforward, I had debated quite a bit on whether Moves class was necessary. In the end, it was the number of methods/attributes in a given category that determined if I turned something into a class or not.

Display:

I spent a majority of my initial time on figuring out how to display the board. Figuring out colored displays on windows terminal, ASCII art for alphabets took a lot of time. In the end I designed my own alphabets art to achieve the look of board tiles. I learned quite a bit on Python inbuilt functions, character encoding.

Code:

This was the best part in the whole project. I felt at home as soon as I started coding.

Tiles & Board:

Tile and Board classes were straight forward. Having done Galton's box code in the homework, this felt similar. Initially, I planned on deleting and adding new tile instances whenever merges happened. And moving the same tile instance on the Board to different positions when tiles slide across.

But I quickly realized, considering there are always fixed number of tiles, it is simpler to update the value of the tile. So I decided to treat empty spaces on the board as tiles with empty/dummy character (z). This meant I could initialize the requisite number of tiles upfront at the time of board creation. I then added methods in the Tile class to update the value of the tile letter when changes occured.

Moves:

Moves class was a fun one to code. Here, my aim was to keep the number of lines of code as low as possible. I spent a fair amount of time playing around with loops. At first it seemed too complex to do all the moves at a time. Once I started to see board moves as confined changes within rows and columns, it fell into place. I could breakdown the moves into slide all tiles first, merge where possible and slide all again-row by row or column by column.

User & Overall game:

User class and overall game was the easiest. I was even able to go back and add more complexity to the game in the end by adding difficulty levels.

Conclusion:

I am not new to breaking down a complex algorithm into small sequential tasks, and that's what helped me in writing this algorithm as well. However, my biggest learning is the comfort with Classes this gave me. With classes, I could visualize components as almost tangible objects as opposed to sequences of instructions like I would in regular code. It made coding feel cooler (if that were possible).