COMS10009 Object Oriented Coursework Report

By: Samani Mukhtar - 1824200

The Scotland Yard Project has been an amazing project to work on, it provided me with a high enough difficulty that my Java and Object-Oriented Programming knowledge would really be tested, but the steps to follow during the implementation made it a lot easier to approach such a large codebase without feeling overwhelmed, especially as an individual working on this rather than a team.

My work has mainly been in MyGameStateFactory.java where we started off by having to implement the GameState interface with its 8 methods initially set to return null so the game and tests can still run before completing the implementation. Since there were tests provided for all the stages of development for this game, I was able me to use test-driven development since I'd be writing code to make the tests pass, which has definitely been an interesting approach for me.

The first step before implementing the interface methods was to ensure that all the attributes being passed into the constructor were in the correct format and to throw an error if elements were null or if a player has a ticket they shouldn't have. Throughout the code I had to work with Guava which was difficult to understand at the beginning but going through their documentation really helps as it's very well written.

The hardest method to implement was for getting the available double moves for the player. This was because I hadn't played the game enough to know the exact behavior of using double moves, and I wanted to use the return of makeSingleMoves as the first destination and then calculate the second destination from there, but the moves it returned didn't allow me to access the destination since it didn't have a getter method and I didn't want to recalculate the available single moves. So, what I had to do was to use the visitor pattern that was included in the Move object to return the destination of the first move. This took me a while, but I was proud of myself when I figured it out.

I could have definitely improved my implementation of makeDoubleMoves because there was a lot of repeated code from makeSingleMoves, so I could’ve moved that code to an external function and then used it in both functions.