Alex Antaki

Dr. Rivas

Software Development I

May 8th, 2017

Checkers

Abstract

This paper is my final write-up for Software Development I. This paper will also go into detail about my procedure and thought process throughout the duration of the checkers program and will highlight areas I needed to improve on.

Introduction

My final project for Software Development I is a game of checkers. The reasoning for doing this was to strengthen my skills with arrays, methods, classes, and JavaFX in order to enhance my ability as a computer programmer. This project forced me to utilize these key aspects of programming and will further my skills as a Software Developer in the future.

Procedure

In order to complete this project, I needed to do a lot of studying and reading up on JavaFX. JavaFX is almost as if it were another language in itself. After spending a few days trying to learn the basics of it, I felt ready to attempt to make the checker board. After a lot of intense Googling and Youtube video watching, I made my checker board. It took some time, effort, and a whole lot of patience, but I was able to make it work in the end. My checker board has black and red alternating tiles and each row it alternates how it starts so it truly looks like a real-life checker board.

Next, I tried to add pieces to the board. This part, although not sounding too difficult, was very tough at the end of the day. I had made a new class titled "Piece.java" and used the circle class to make a blue circle with a white perimeter. The piece was going on a black tile on the checker board, therefore, the white perimeter would give it a certain aesthetic to it. I had to place this circle on a pane and then place the pane on a scene to be displayed on the checker board. When I ran my checker program, it would load the checker board in a single window and then the piece itself in its own window. Hours went by and the piece still did not want to cooperate. I decided to scrap that idea and made a circle using Google draw. This allowed me to draw nearly the exact same piece and I had to save it as a .png. I imported this circle and I finally had pieces on the board. There was one problem though, I had pieces on every single tile. In checkers, the pieces need to alternate and the middle two rows are supposed to be free. I spent a few more days simply trying to get the pieces to fill in properly. After thinking I had it figured out to place the pieces in the proper positions, I found out my for loops were coded improperly because the pieces were lining up on the right side instead of the top and bottom. I tried to use the same logic to see if I could make the pieces appear on the left. After valiantly failing, I ended up making what I originally thought was a mistake, but turned out to fix my issue instead. The pieces had finally shown up where I wanted. I had placed an if statement in what I had thought was the wrong place, but had actually solved my problem for me. The next step was to create another piece, therefore we could differentiate player one from player two. I went back into Google draw and made a yellow circle and saved it as a .png as well. Then I had gone back into my code and used the same logic for the blue pieces, but with the proper coordinates for these pieces. Now, after days of work I had a checker board with pieces in the proper places.

After all of the work that was required for the JavaFX section of this project, it was time to move onto the logic and data of the project. I needed to figure out how to make the movement of the pieces. This requires a lot more than one may think. I had to account for no backwards movement (unless you are a king), diagonal movement exclusively, only allowed to move to black tiles exclusively, whether the move is a jump and make sure you remove the jumped piece from the board, check if the game is over, and check whether the move is valid just to name a few. This require more intense Googling and Youtube video watching. I had started to make some progress on the logic of the game, but I kept getting stumped. I did not know where to go, but I did not want to stop working on the game. All of my code for the second movement system compiled, but it didn't do anything. I could not relate the logic to the JavaFX board. I spent days trying to figure out a way to link them but it would not work. I was recommended to try to build a simple game engine and I tried to make that too. I spent a few hours trying to get that to work through debugging, but when one bug was fixed another one popped up and I didn't know how to fix all of them because I was very unfamiliar with game engines. I decided a game engine was too in depth for what I wanted to do and decided to try a much simpler approach. I made a new "Movement.java" file and I called it "Movement2_0.java". This movement file sets positions in the 2D array for the checker board. It used 0s, 1s, 2s, 3s, and 4s. The 0 represents an open place on the checker board. The number 1 represents a blue piece. The number 2 represents a yellow piece. The number 3 represents a blue king, and lastly the number four represents the yellow king. When launched, it only shows 0s, 1s, and 2s, because there are no kings yet. I tried to use a for loop to initialize all of the values, but it wasn't working the way I had intended. This made me manually initialize all of the values. I now needed to print this array so we can see how this looks to the user. I made a printArray method and used nested for loops for this and it worked like a charm.

Conclusion

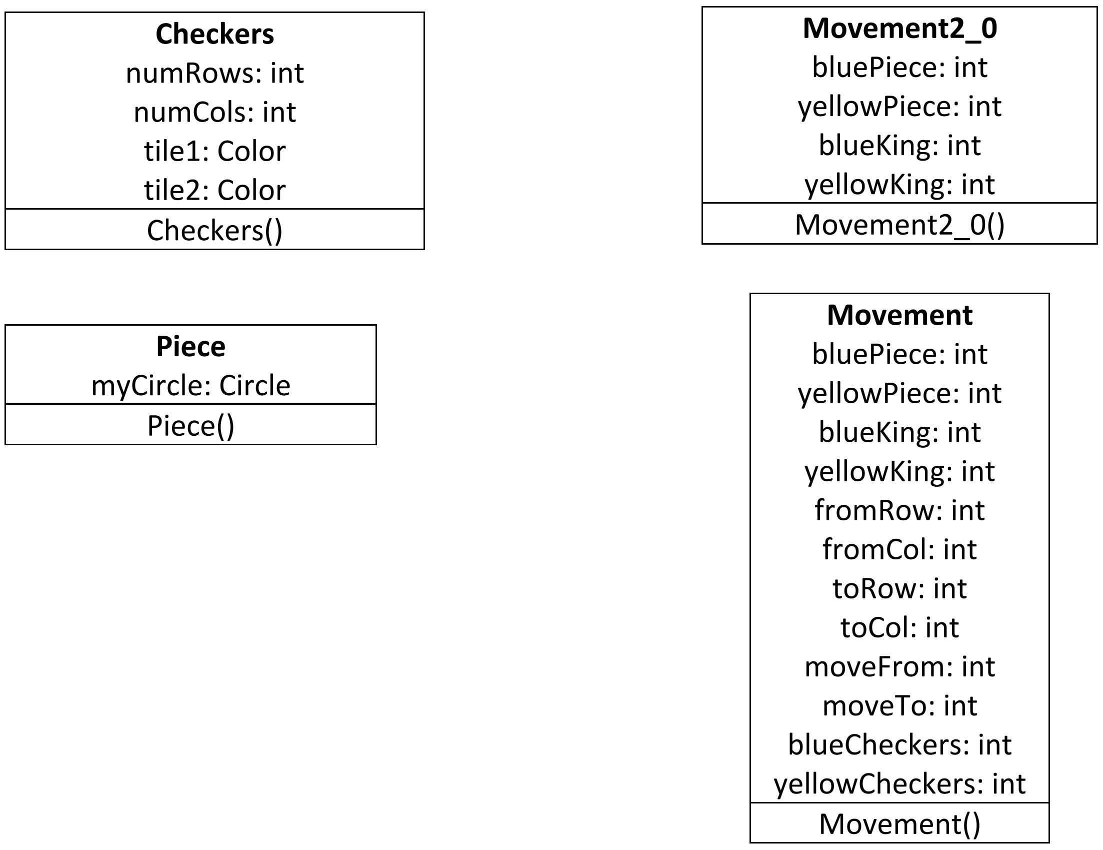
This is how far I was able to make it with this project. I am, however, going to continue this project in the future. I believe that as I progress through my Marist career I will become a better programmer and can finish this project. This project was definitely tough and I have learned an immense amount of new material (JavaFX) by doing so. Also, my abilities with methods, classes, and arrays are now stronger.

I plan on continuing with this project over the summer to bring it to its completion. My end goals are to finish the movement system so the game can be played properly, create an AI to play against so you can play by yourself, create a counter to track the pieces that have been eliminated, highlight the piece that is clicked and show any available moves for that piece, show double jumps that are available for the clicked piece, create a hint button for any person playing against the AI that needs help, and keep track of whose turn it is. (Movement.java does not compile, but I submitted it to show what I had done to try to create movement).

Detailed System Description

The system I created is a game of checkers. At the moment users cannot interact with it the way I had hoped but I do plan on continuing with the application as I had explained above. Also, there are UML diagrams accompanied to illustrate the classes I used to complete this project.

UMLs



Checkers
numRows: int
numCols: int
tile1: Color
tile2: Color
Checkers()
Movement2_0
bluePiece: int
yellowPiece: int
blueKing: int
yellowKing: int
Movement2_0()
Piece
myCircle: Circle
Piece()
Movement
bluePiece: int
yellowPiece: int
blueKing: int
yellowKing: int
fromRow: int
fromCol: int
toRow: int
toCol: int
moveFrom: int
moveTo: int
blueCheckers: int
yellowCheckers: int
Movement()