Chess Proposal Revised

Cody Murrer

CIS 201-HYB1

**Project Summary**

The project I have decided to make is a difficult and strategic turn-based game known as chess. When Main.java is ran it will prompt you with a JavaFX based GUI menu screen that will show a home screen (see Main.java) allowing you to pick from the following options Play Game, How to Play, and Exit. When the player hits the play game button the button will send you to the board (see Board.java) where you will see an 8x8 checkered grid with black pieces on top and white pieces on the bottom. With all the pieces already lined up and ready to go you will be allowed to begin playing with a friend or family member.

**Project Background**

The game of chess is not only a game based off strategy and skill, but also a game of luck. The game is a two-player game played on an 8x8 grid with six different piece types per side. Each side has 16 total pieces which consists of 8 pawns, 2 rooks, 2 knights, 2 bishops, 1 queen, and 1 king. Each one of these pieces play a different key role in dominating the game board with each piece’s unique movement styles.

**IPOS Requirements**

The different types of processing needed is the initial setup on the Main.java screen which will need buttons with event handlers dictating where the user will go when the button is clicked. When the play button has been pressed then the gameBoard scene will begin filling a 2D label array to create the 8x8 grid needed to create the board. After the grid is in place the next processing that would be needed is the assignment of the different square with their respective colors within the for loop used to create the grid using a statement that checks even and odd indexes within the 2D array. After the processing of the game board a for loop populates labels within the 2D label array called bwPieces and will begin to fill the first 1x8 and 2x8 indexes on the grid with black pieces and the last two rows will fill with white pieces indexing them at 7x8 and 8x8. After the labeling is produced then the for loop runs through statements checking for indexing and setting images contained within the image folder. The types of inputs that will be required of this project are mainly left clicks as they will be the controlling factor to each piece. The different types of processing requirements that will be needed are the different movements of each individual pieces on the board as well as process of setting turns for each player and how the pieces will interact with each other if they collide with friendly or enemy pieces.

**Conclusion**

In conclusion the choice of creating the game of chess will provide me with a challenge to have to manage all the pieces at once as well as constant validity checking for open spaces and collision events as well. Being my first time using JavaFX to create a GUI I am surely going to run into moments where I don’t fully understand the imported packages API and will have to search through the API for JavaFX 8 to help me continue. This project will be able to show the extent of my knowledge of java by forcing me face the challenges of coding multiple elements with different movement patterns as well as different collision behaviors between all the different pieces on the board.

**Polymorphism**

As of this assignment (Assignment 3) I have been unable to get to a point where I would be able to implement polymorphism into my project but I do however plan on adding it to the main screen as well as within the game as when pawns reach the enemy side of the board they turn into queens and the implementation that would be used on the main menu would be to have a label on each side of the menu selection area and have it change to the different pieces from each players side showing off the different types the player can use during the game.

**Glossary**

Chess Chess is a board game of strategy and skill played by two players on a checkered board

GUI Graphical user Interface

JavaFX A software platform used for creating and developing desktop applications as well as web applications