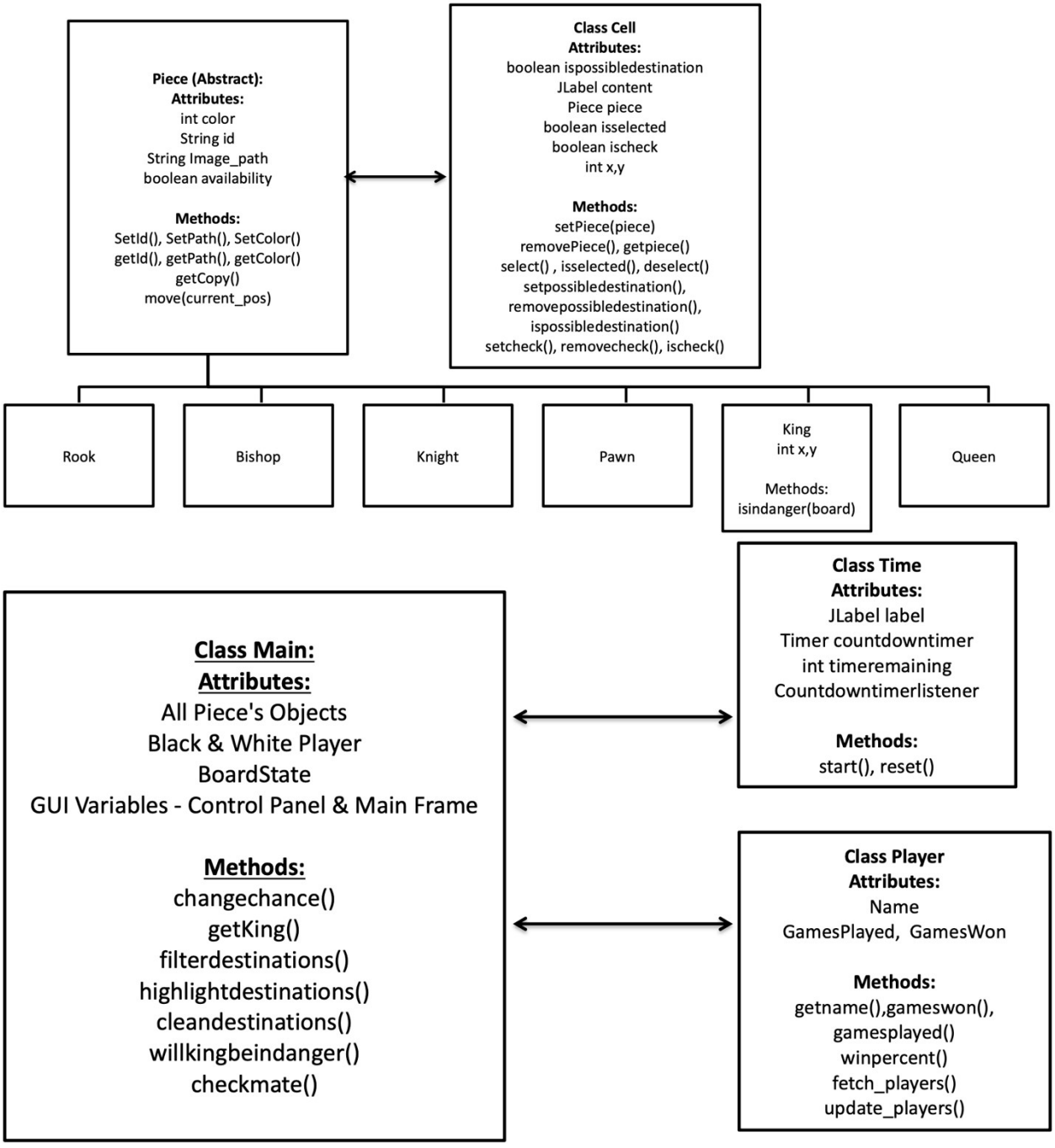
The Chess Game

Introduction

Chess is a game in which two players compete to bring their opponents' king into a checkmate position, in which no further movements on the board are possible. It implements all the basic Chess rules. Players can be chosen from a list of current players or generated from scratch. Statistics on users are updated as needed. Each pick prompts the program to display all feasible movements. Back and forward navigation is possible, and players may make their own moves at any moment, which cannot be undone and redone. It imposes restrictions if the King is under suspicion. It is a timed game, and if a player runs out of time, he forfeits his ability to move. My main goal was to get the project up and running with excellent OOP concepts and a solid framework that would allow for future customization and optimization. Coming from an engineering background, I frequently create a design that follows the core concepts and gets it operational before going in and optimizing it. In completing this assignment, I strictly adhered to this process. The game has a rudimentary statistics system (wins, losses, and draws) with the opportunity to reset them, as well as the ability to track progression and a simple computer opponent with the ability to change its "advancement" level and show movements in chess notation.

Class Diagram

1. Piece: This is an abstract class that will be extended to all the real parts. It can keep track of all components' color, id, and Image Path (for GUI). It lays forth the fundamental laws for all chess pieces' movement. The King Piece checks whether the king is in danger. It implements the Cloneable interface to allow cloning.
2. Main: This is the main class that holds the game. It holds the current state of the game and creates the GUI and controls the function of all the other classes. It also handles click events during the game and provides functionality to move a piece and change the current board state. It provides all the special "King-Related" functions like check, checkmate, restricting moves if the king is check, and so on.
3. Cell: The chessboard's token class is called Class Cell. The Chess Board is made up of 64 items from this class. It contains the item as well as its associated data. It has the capability of highlighting or cleaning all feasible motions. It can signify that the monarch is being watched. It can display the current selection as well as change the current selection.
4. Time: This class handles the function of the countdown timer. The countdown timer can be started or reset using this method. It has the capability of updating the timer every second. It can adjust the chance if the time limit is reached.
5. Player: This class keeps track of all a Player's records. File is used to make this class's objects permanent. The player information may be retrieved from the game's data file using this method. Provides the ability to change a player's statistics in the game's data file. Provides the ability to compute a player's victory percentage throughout the course of his career.


Piece (Abstract):
Attributes:
int color
String id
String Image_path
boolean availability
Methods:
SetId(), SetPath(), SetColor()
getId(), getPath(), getColor()
getCopy()
move(current_pos)
Class Cell
Attributes:
boolean ispossibledestination
JLabel content
Piece piece
boolean isselected
boolean ischeck
int x,y
Methods:
setPiece(piece)
removePiece(), getpiece()
select() , isselected(), deselect()
setpossibledestination(),
removepossibledestination(),
ispossibledestination()
setcheck(), removecheck(), ischeck()
Rook
Bishop
Knight
Pawn
King
int x,y
Methods:
isindanger(board)
Queen
Class Main:
Attributes:
All Piece's Objects
Black & White Player
BoardState
GUI Variables - Control Panel & Main Frame
Methods:
changechance()
getKing()
filterdestinations()
highlightdestinations()
cleandestinations()
willkingbeindanger()
checkmate()
Class Time
Attributes:
JLabel label
Timer countdowntimer
int timeremaining
Countdowntimerlistener
Methods:
start(), reset()
Class Player
Attributes:
Name
GamesPlayed, GamesWon
Methods:
getname(),gameswon(),
gamesplayed()
winpercent()
fetch_players()
update_players()

Requirements (Cloning/Building/Running)

The development environment used to create the chess game is IntelliJ IDE plugin and the version of Java JDK used is Java SE 15.0.2.

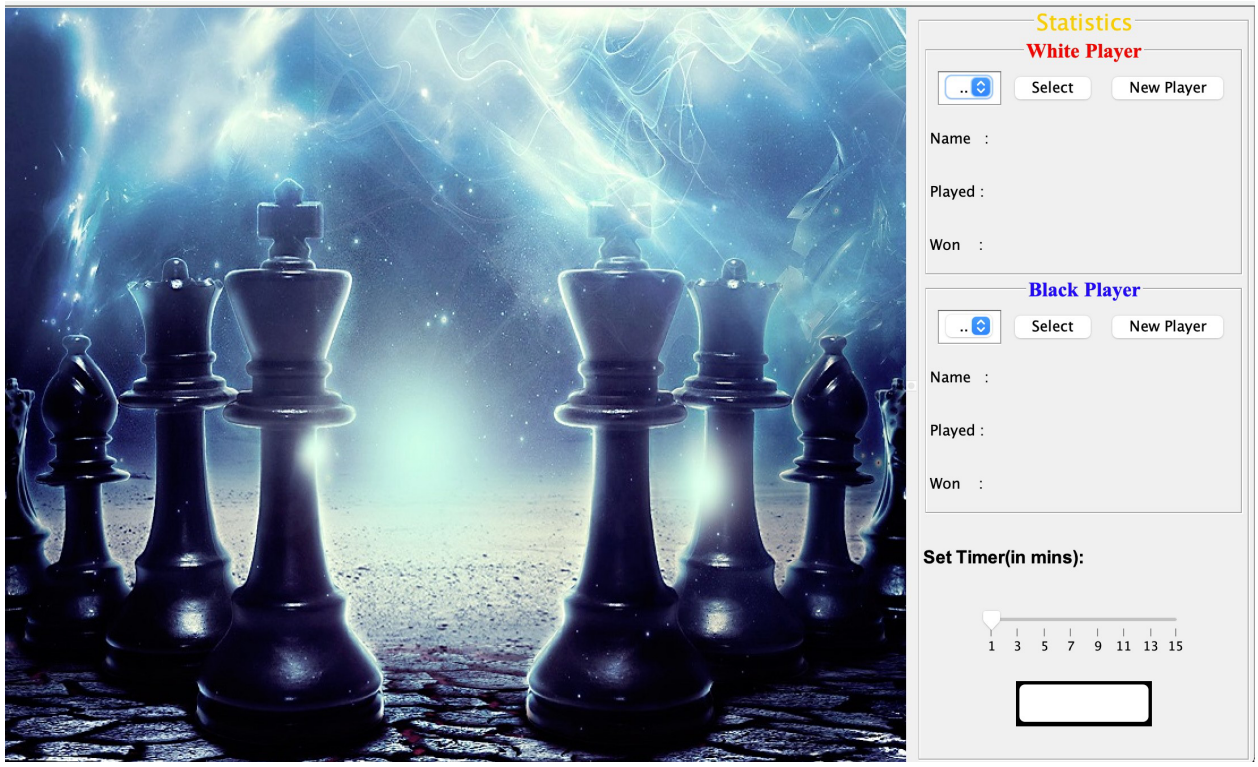
To clone the project, choose the folder through the terminal where you want to download it. Go ahead and type *git clone* following the game link on GitHub *https://github.com/csc413-SFSU-Souza/csc413secondgame-pragati-e .*

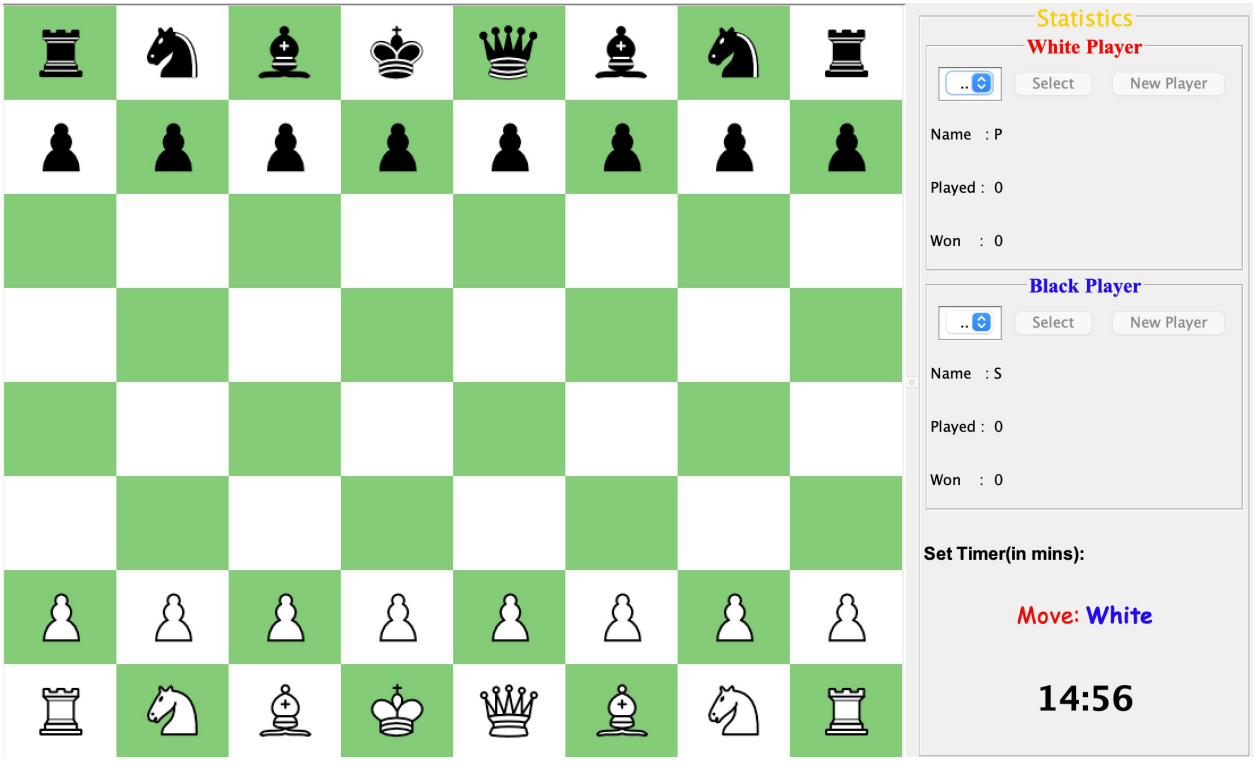
To build the project, open the IDE (IntelliJ) and navigate to the location where the cloned project is located. Open the folder by selecting Launch. The src folder contains all the source code and the all the features like graphics, screenshots of background images, and the font styles.

To run the game, select the Run option from the top right menu bar.

To run the game through the jar folder. Open the terminal and navigate to the jar folder by using cd command. After entering the jar executable file of the tank game, enter the following to run the game on your desktop, *java -jar Chess.jar*

```
Last login: Mon May 16 13:06:25 on ttys000
pragatimakani@Pragatis-MacBook-Pro ~ % cd Desktop
pragatimakani@Pragatis-MacBook-Pro Desktop % cd csc413-secondgame-pragati-e
pragatimakani@Pragatis-MacBook-Pro csc413-secondgame-pragati-e % cd jar
pragatimakani@Pragatis-MacBook-Pro jar % java -jar Chess.jar
```





were written in Java, with special care devoted to following excellent OOP concepts, particularly the Single Responsibility Principle. I am happy for the opportunity to accomplish these tasks in this manner since I believe it has significantly improved my knowledge of software development. But I must admit that throughout these projects, I have learnt a lot and gone a long way in terms of being a better programmer and building better software. The game creation process was a wonderful method for me to discover Java's hidden treasure of rich language capabilities, which I thoroughly liked while also learning a lot.