Final Project Report

Name	: Nicholaus Santo Agnus Dei
NIM / ID	: 2602174415
Course	: OOP - COMP6699001
Lecturer	: Jude Joseph Lamug Martinez MCS - D4107

A. Project Specification

a. Basic Concept

A chess board simulator where whoever uses it is allowed to move the pieces freely and coordinate the pieces according to the player's command and wishes. The board contains all the basic pieces such as, 8 white pawns, 8 black pawns, 2 of each colored knights, rooks, and bishops, white and black kings, white and black queens, alongside 4 extra pieces for each color containing the rook, the queen, the bishop, and the knight.

b. Prerequisites

i. Computation Device (Windows, Mac, Linux, etc)
ii. IDE (preferably Visual Studio Code or IntelliJ)
iii. Java Development Kit version 8 or above

c. Main Problem

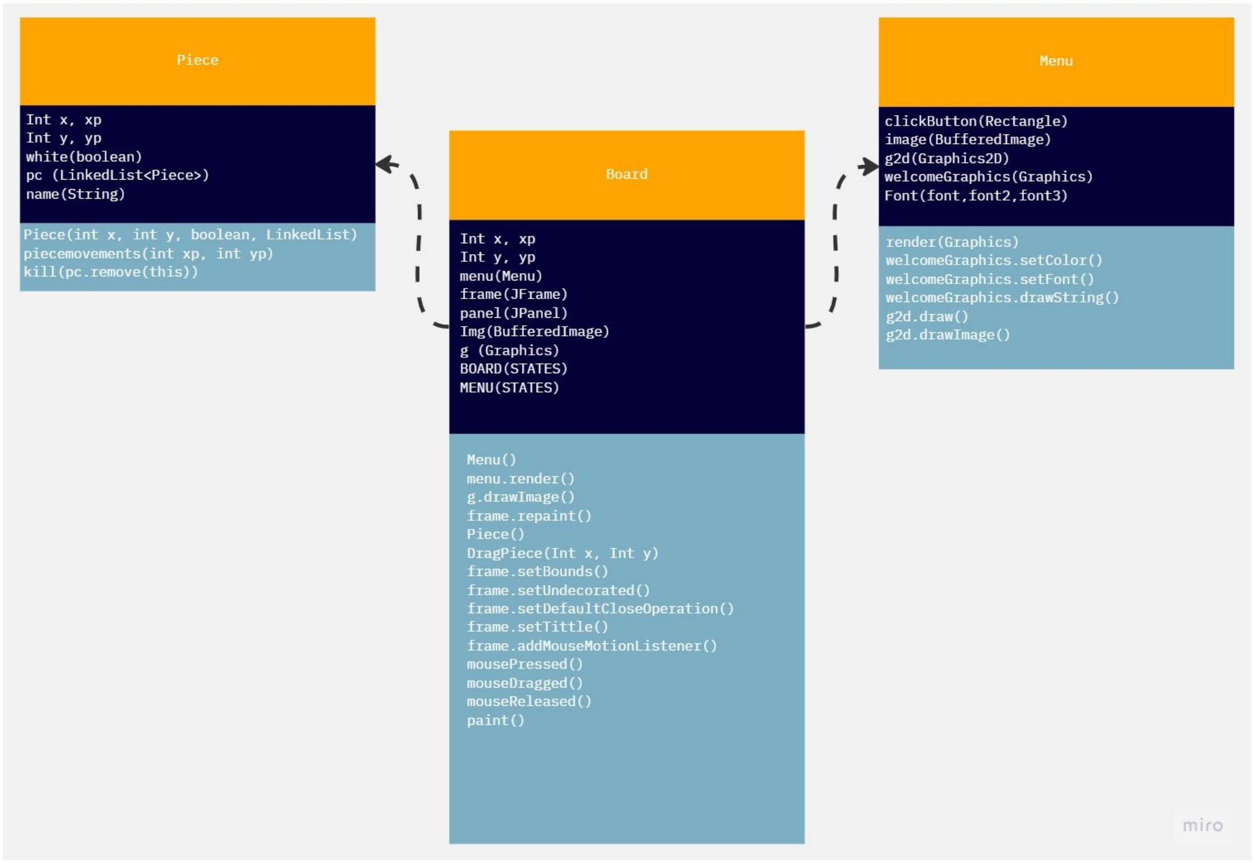
i. Drawing the Chess Board
ii. Drawing each and every piece
iii. Controlling each and every piece
iv. Special moves for killing and incorrection

B. Solution Design

a. Main Solution

i. To draw the Chess Board itself, a panel and a frame was created using the functions of Swing in Java.
ii. To draw each and every piece for the Chess Board, a LinkedList was made to store all the starting pieces alongside the extra pieces and indexes were used to crop and label every piece from the reference image of the chess pieces. A boolean value was also used to determine the color of the piece.
iii. To move the pieces on the chess board, a mouse listener was used to create a dragging action for moving the pieces, this would also mean that there are other functions that are used to help the dragging animation of the pieces.
iv. To be able to do the killing move or an action that is also considered as taking an opposite colored piece, a specific function was used for if the boolean value of the piece that are in contact with each other were to be different, the moved piece should take the piece that is in position of where the moved piece shall be.

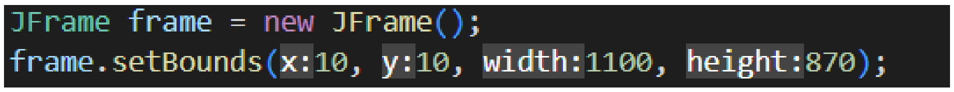
b. Class Diagram



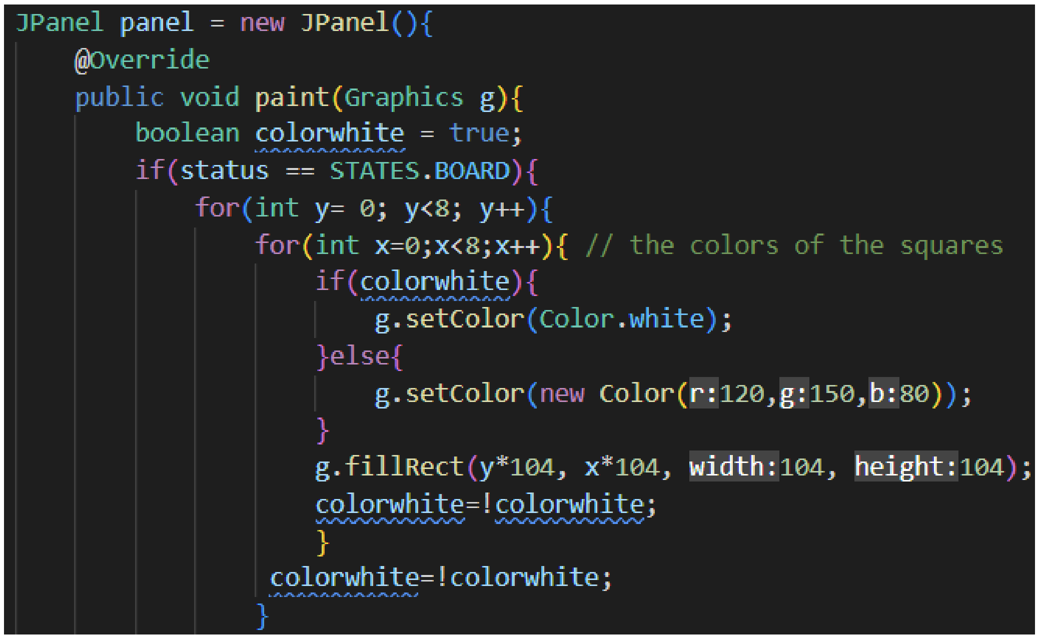
C. Code Functions and Implementations

a. The Board

To draw the actual chess-like board or the Chess Board that is filled with 8 x 8 dark and light squares, a frame was created and inside the frame, panels are used to draw each square. Every light and dark square is basically a panel.


```
JFrame frame = new JFrame();
frame.setBounds(x:10, y:10, width:1100, height:870);
```

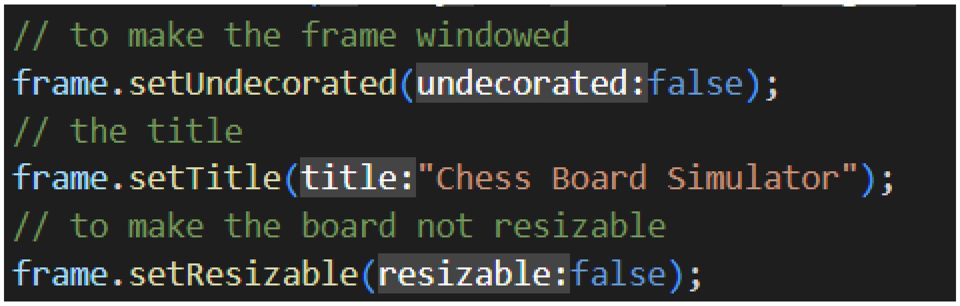
Using the above code, we can set the size of the frame, which is basically the size of the blank canvas for us to draw the board later on, to the size that we want it to be in.



```
JPanel panel = new JPanel(){
    @Override
    public void paint(Graphics g){
        boolean colorwhite = true;
        if(status == STATES.BOARD){
            for(int y= 0; y<8; y++){
                for(int x=0;x<8;x++){ // the colors of the squares
                    if(colorwhite){
                        g.setColor(Color.white);
                    }else{
                        g.setColor(new Color(r:120,g:150,b:80));
                    }
                   g.fillRect(y*104, x*104, width:104, height:104);
                    colorwhite=!colorwhite;
                    }
                 colorwhite=!colorwhite;
                }
```

The codes above are the algorithms that are used to draw each square that would make up the chess board. A new panel variable was added and the paint method was used to paint the square using a loop. The color of the squares are separated by the boolean, white or light squared.

There are also some other functions that helps decorating the board such as:

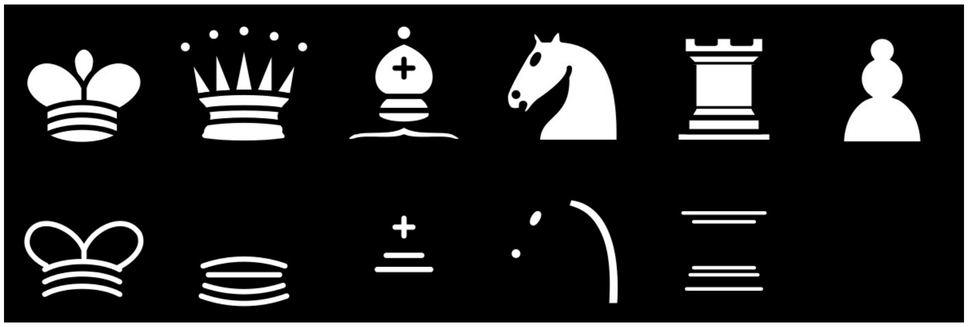


```
// to make the frame windowed
frame.setUndecorated(undecorated:false);
// the title
frame.setTitle(title:"Chess Board Simulator");
// to make the board not resizable
frame.setResizable(resizable:false);
```

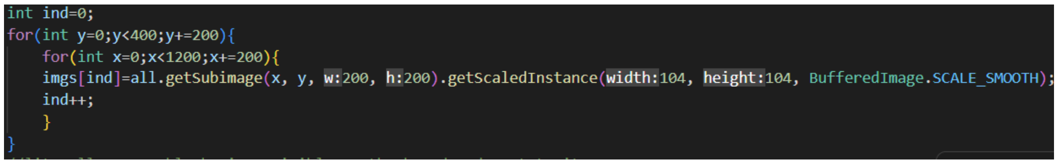
Those functions aren't really necessary but it helps with the aesthetics.

b. The Pieces

The pieces are taken from an image reference, here is the image:

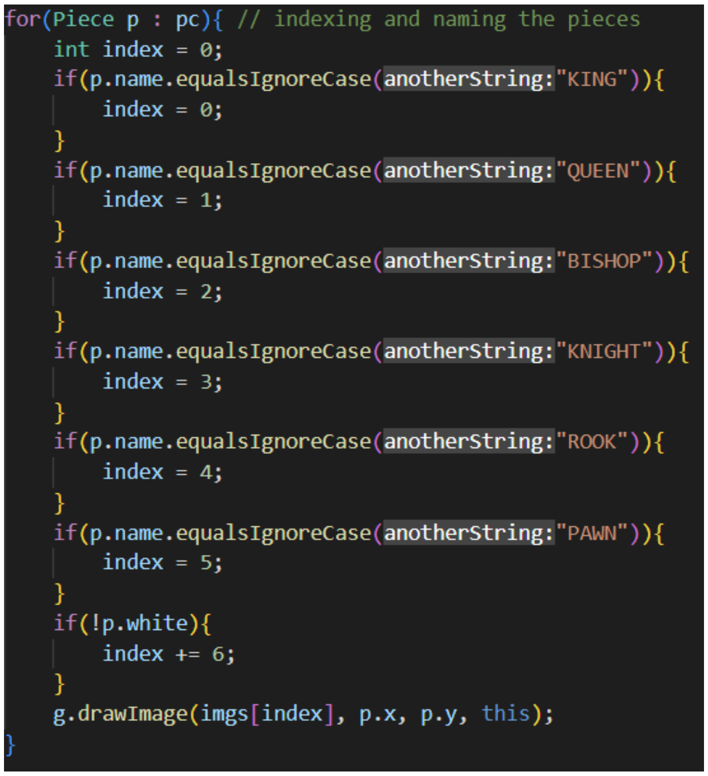


The image would then be cropped to fit singular pieces using the algorithm found below:



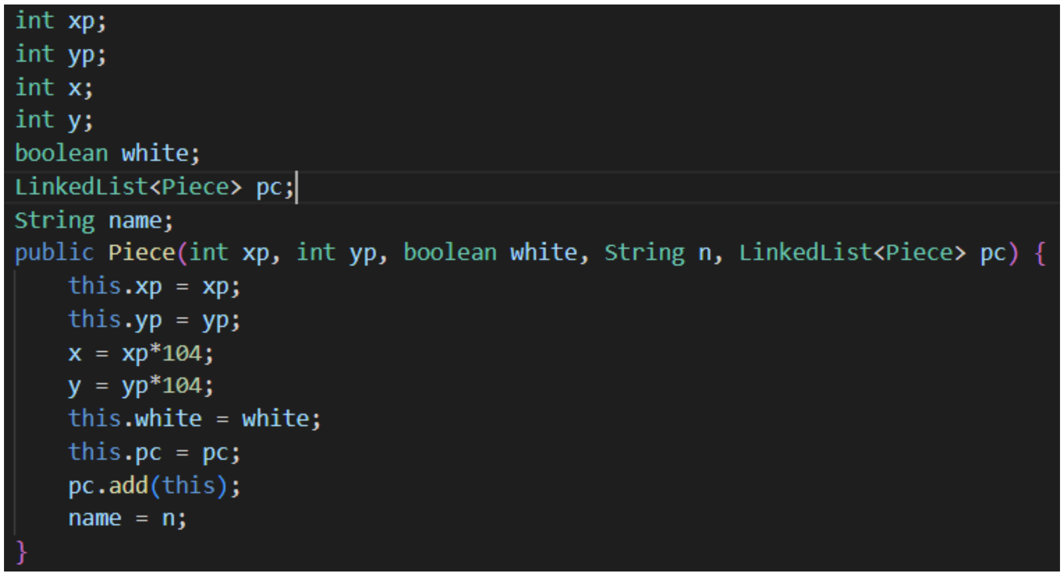
```java
int ind=0;
for(int y=0;y<400;y+=200){
    for(int x=0;x<1200;x+=200){
    imgs[ind]=all.getSubimage(x, y, w:200, h:200).getScaledInstance(width:104, height:104, BufferedImage.SCALE_SMOOTH);
    ind++;
    }
}
```

After the cropping of the image, another algorithm is used to name and index the pieces.



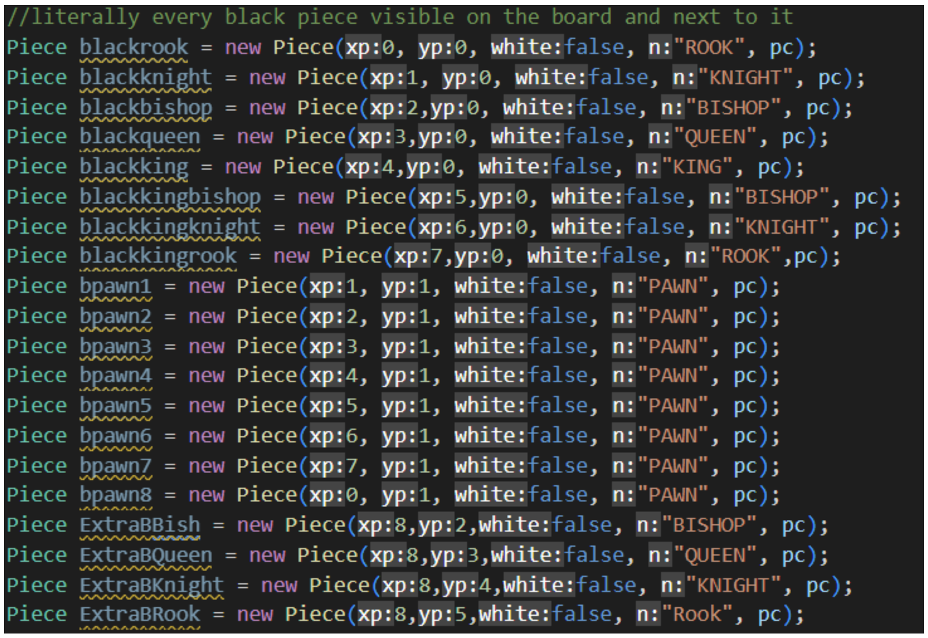
```java
for(Piece p : pc){ // indexing and naming the pieces
    int index = 0;
    if(p.name.equalsIgnoreCase(anotherString:"KING")){
        index = 0;
    }
    if(p.name.equalsIgnoreCase(anotherString:"QUEEN")){
        index = 1;
    }
    if(p.name.equalsIgnoreCase(anotherString:"BISHOP")){
        index = 2;
    }
    if(p.name.equalsIgnoreCase(anotherString:"KNIGHT")){
        index = 3;
    }
    if(p.name.equalsIgnoreCase(anotherString:"ROOK")){
        index = 4;
    }
    if(p.name.equalsIgnoreCase(anotherString:"PAWN")){
        index = 5;
    }
    if(!p.white){
        index += 6;
    }
    g.drawImage(imgs[index], p.x, p.y, this);
}
```

After all of the pieces are named and indexed, a LinkedList was created to store all the pieces.



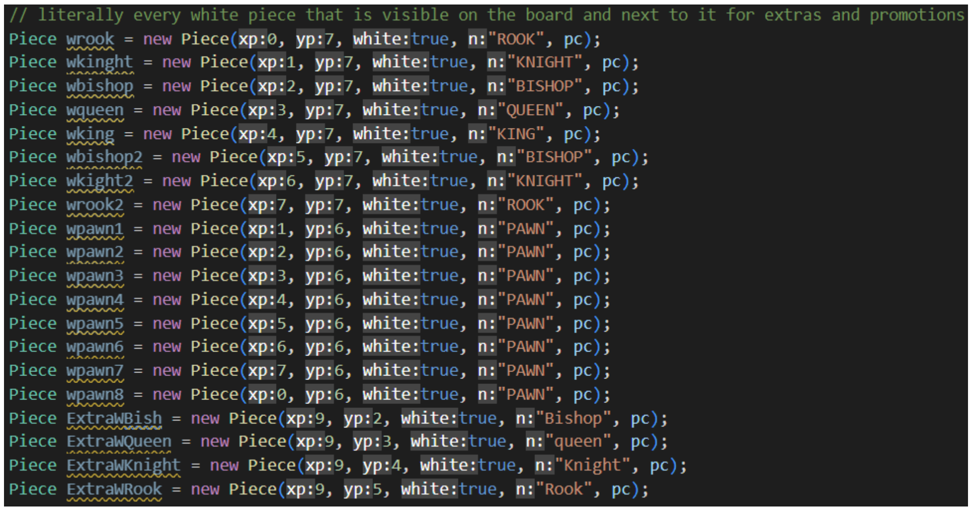
```java
int xp;
int yp;
int x;
int y;
boolean white;
LinkedList<Piece> pc;
String name;
public Piece(int xp, int yp, boolean white, String n, LinkedList<Piece> pc) {
    this.xp = xp;
    this.yp = yp;
    x = xp*104;
    y = yp*104;
    this.white = white;
    this.pc = pc;
    pc.add(this);
    name = n;
}
```

After the LinkedLIst was created the pieces would then be added to the list and then they would be added onto the board by creating them and adding them into the list manually using these lines of code.



```java
//literally every black piece visible on the board and next to it
Piece blackrook = new Piece(xp:0, yp:0, white:false, n:"ROOK", pc);
Piece blackknight = new Piece(xp:1, yp:0, white:false, n:"KNIGHT", pc);
Piece blackbishop = new Piece(xp:2,yp:0, white:false, n:"BISHOP", pc);
Piece blackqueen = new Piece(xp:3,yp:0, white:false, n:"QUEEN", pc);
Piece blackking = new Piece(xp:4,yp:0, white:false, n:"KING", pc);
Piece blackkingbishop = new Piece(xp:5,yp:0, white:false, n:"BISHOP", pc);
Piece blackkingknight = new Piece(xp:6,yp:0, white:false, n:"KNIGHT", pc);
Piece blackkingrook = new Piece(xp:7,yp:0, white:false, n:"ROOK",pc);
Piece bpawn1 = new Piece(xp:1, yp:1, white:false, n:"PAWN", pc);
Piece bpawn2 = new Piece(xp:2, yp:1, white:false, n:"PAWN", pc);
Piece bpawn3 = new Piece(xp:3, yp:1, white:false, n:"PAWN", pc);
Piece bpawn4 = new Piece(xp:4, yp:1, white:false, n:"PAWN", pc);
Piece bpawn5 = new Piece(xp:5, yp:1, white:false, n:"PAWN", pc);
Piece bpawn6 = new Piece(xp:6, yp:1, white:false, n:"PAWN", pc);
Piece bpawn7 = new Piece(xp:7, yp:1, white:false, n:"PAWN", pc);
Piece bpawn8 = new Piece(xp:0, yp:1, white:false, n:"PAWN", pc);
Piece ExtraBBish = new Piece(xp:8,yp:2,white:false, n:"BISHOP", pc);
Piece ExtraBQueen = new Piece(xp:8,yp:3,white:false, n:"QUEEN", pc);
Piece ExtraBKnight = new Piece(xp:8,yp:4,white:false, n:"KNIGHT", pc);
Piece ExtraBRook = new Piece(xp:8,yp:5,white:false, n:"Rook", pc);
```

Those lines of code are for the black pieces.



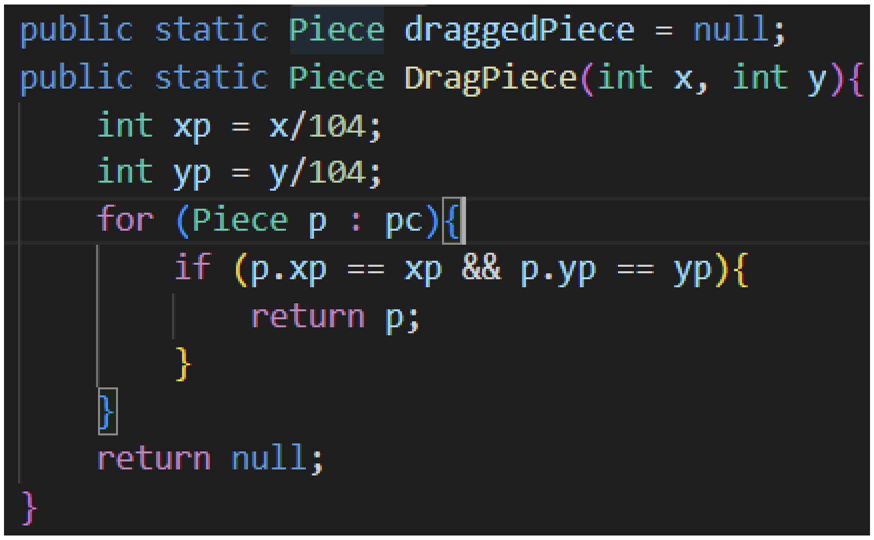
```java
// literally every white piece that is visible on the board and next to it for extras and promotions
Piece wrook = new Piece(xp:0, yp:7, white:true, n:"ROOK", pc);
Piece wkinght = new Piece(xp:1, yp:7, white:true, n:"KNIGHT", pc);
Piece wbishop = new Piece(xp:2, yp:7, white:true, n:"BISHOP", pc);
Piece wqueen = new Piece(xp:3, yp:7, white:true, n:"QUEEN", pc);
Piece wking = new Piece(xp:4, yp:7, white:true, n:"KING", pc);
Piece wbishop2 = new Piece(xp:5, yp:7, white:true, n:"BISHOP", pc);
Piece wkight2 = new Piece(xp:6, yp:7, white:true, n:"KNIGHT", pc);
Piece wrook2 = new Piece(xp:7, yp:7, white:true, n:"ROOK", pc);
Piece wpawn1 = new Piece(xp:1, yp:6, white:true, n:"PAWN", pc);
Piece wpawn2 = new Piece(xp:2, yp:6, white:true, n:"PAWN", pc);
Piece wpawn3 = new Piece(xp:3, yp:6, white:true, n:"PAWN", pc);
Piece wpawn4 = new Piece(xp:4, yp:6, white:true, n:"PAWN", pc);
Piece wpawn5 = new Piece(xp:5, yp:6, white:true, n:"PAWN", pc);
Piece wpawn6 = new Piece(xp:6, yp:6, white:true, n:"PAWN", pc);
Piece wpawn7 = new Piece(xp:7, yp:6, white:true, n:"PAWN", pc);
Piece wpawn8 = new Piece(xp:0, yp:6, white:true, n:"PAWN", pc);
Piece ExtraWBish = new Piece(xp:9, yp:2, white:true, n:"Bishop", pc);
Piece ExtraWQueen = new Piece(xp:9, yp:3, white:true, n:"queen", pc);
Piece ExtraWKnight = new Piece(xp:9, yp:4, white:true, n:"Knight", pc);
Piece ExtraWRook = new Piece(xp:9, yp:5, white:true, n:"Rook", pc);
```

Those lines of code are for the white pieces.

c. The Piece Movements and Kill Action

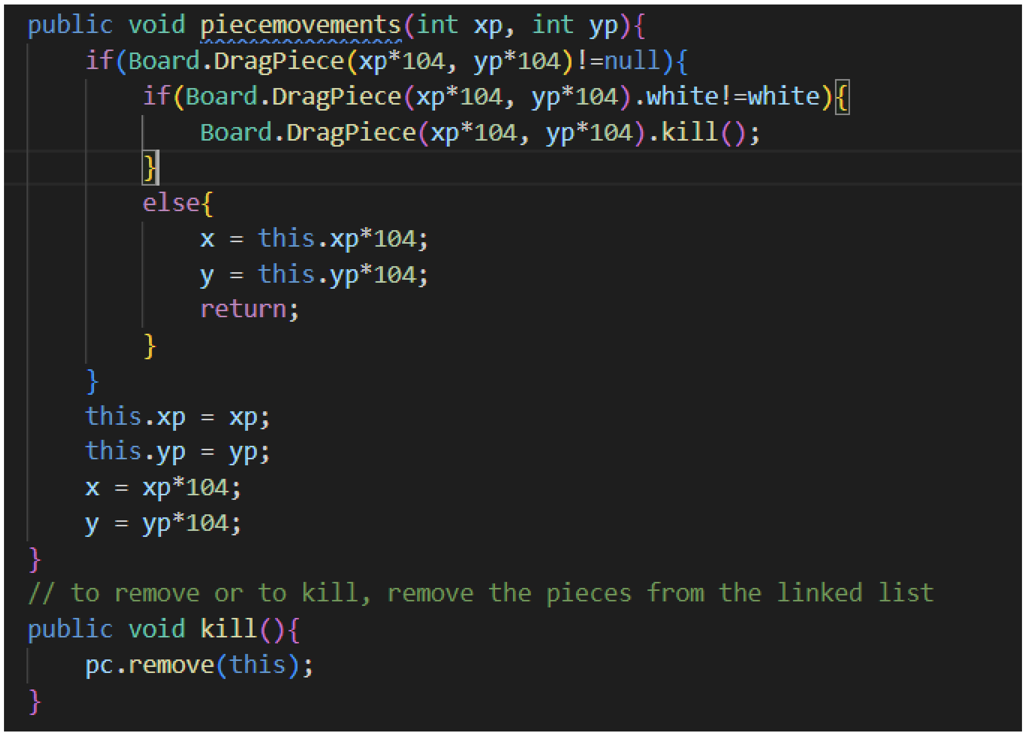
After the pieces are added onto the board, other functions are also required to make the pieces drag-able. Therefore, a mouse listener also needs to be added to record or listen to the action of the mouse and drag the pieces using the mouse cursor.

First, there needs to be a function that can be used to get the position of the pieces that are chosen.



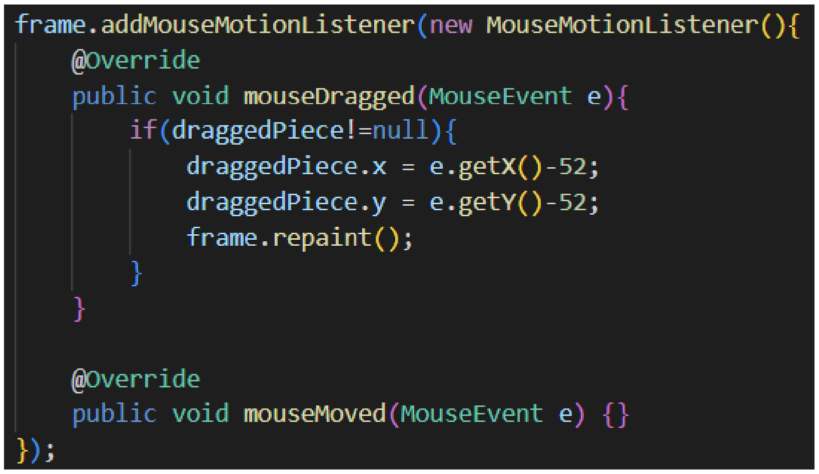
```java
public static Piece draggedPiece = null;
public static Piece DragPiece(int x, int y){
    int xp = x/104;
    int yp = y/104;
    for (Piece p : pc){
        if (p.xp == xp && p.yp == yp){
            return p;
        }
    }
    return null;
}
```

Using those lines of codes, we can implement another function inside the piece class to animate the placement of the pieces that are dragged by the cursor and to add a special function to kill as well if the pieces are to be dragged on top of another piece that has the opposite color.



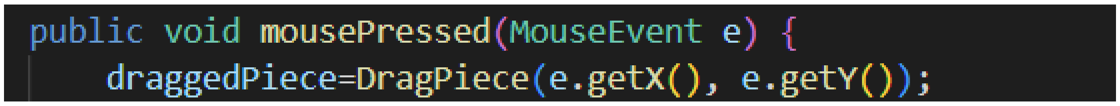
```
public void piecemovements(int xp, int yp){
    if(Board.DragPiece(xp*104, yp*104)!=null){
        if(Board.DragPiece(xp*104, yp*104).white!=white){
            Board.DragPiece(xp*104, yp*104).kill();
        }
        else{
            x = this.xp*104;
            y = this.yp*104;
            return;
        }
    }
    this.xp = xp;
    this.yp = yp;
    x = xp*104;
    y = yp*104;
}
// to remove or to kill, remove the pieces from the linked list
public void kill(){
    pc.remove(this);
}
```

To do the killing, we simply just have to remove the piece from the created linked list. After the sitting place of the pieces are created, the mouse listener would then be added alongside the animation of the pieces when they are dragged by the mouse.



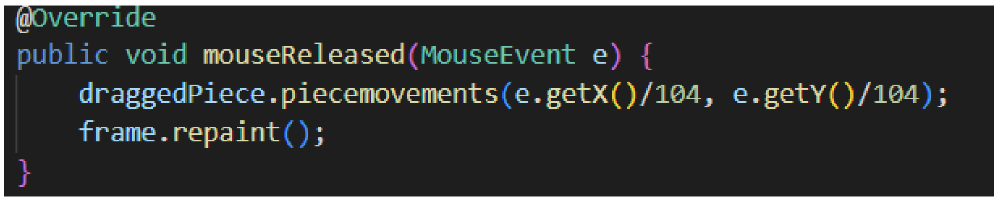
```
frame.addMouseMotionListener(new MouseMotionListener(){
    @Override
    public void mouseDragged(MouseEvent e){
        if(draggedPiece!=null){
            draggedPiece.x = e.getX()-52;
            draggedPiece.y = e.getY()-52;
            frame.repaint();
        }
    }

    @Override
    public void mouseMoved(MouseEvent e) {}
});
```

In the function above, the static field of the piece called *draggedPiece* is used to tell whether the cursor of the mouse is dragging a piece or not. If the cursor of the mouse is touching or dragging a piece, the image of the piece would then be animated to stay in the middle of the cursor.



```
public void mousePressed(MouseEvent e) {
    draggedPiece=DragPiece(e.getX(), e.getY());
```

This function is also used to get the position of the piece whilst it's being dragged.

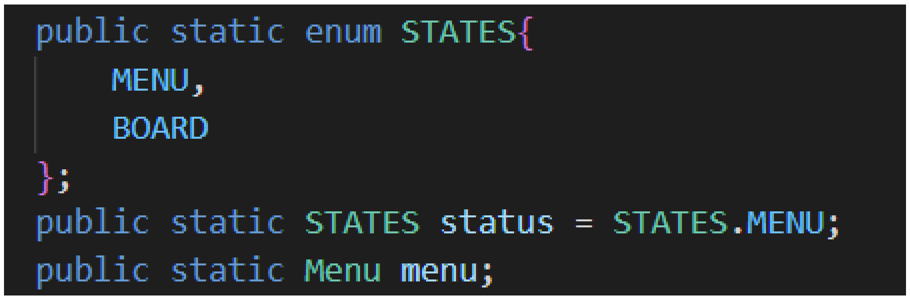


```
@Override
public void mouseReleased(MouseEvent e) {
    draggedPiece.piecemovements(e.getX()/104, e.getY()/104);
    frame.repaint();
}
```

This function is used to fix the position after the dragged piece is dropped from the cursor.

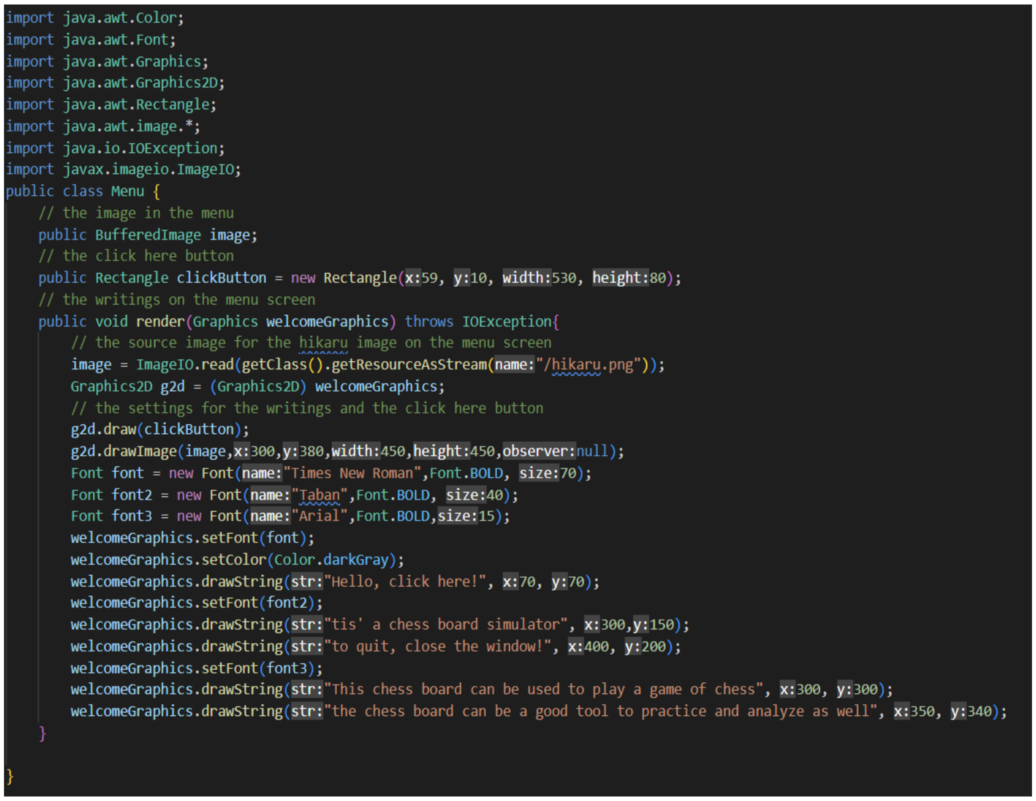
d. The Menu

Other than pieces, an enum was also created to draw the menu screen. Hence, these lines of code.



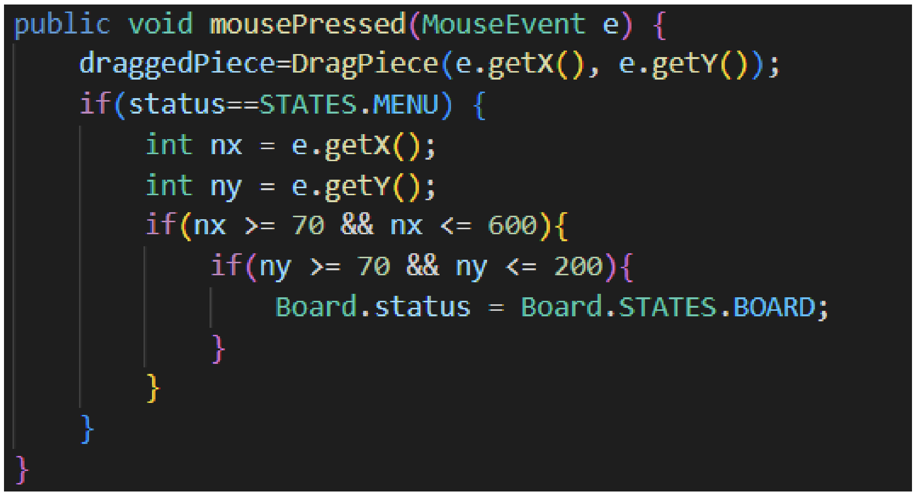
```
public static enum STATES{
    MENU,
    BOARD
};
public static STATES status = STATES.MENU;
public static Menu menu;
```

First, set the enum to the menu enum so that when the game is started, it starts on the menu screen. A menu class was also created to store all the graphics and text that is written on the menu.



```java
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Rectangle;
import java.awt.image.*;
import java.io.IOException;
import javax.imageio.ImageIO;
public class Menu {
    // the image in the menu
    public BufferedImage image;
    // the click here button
    public Rectangle clickButton = new Rectangle(x:59, y:10, width:530, height:80);
    // the writings on the menu screen
    public void render(Graphics welcomeGraphics) throws IOException{
        // the source image for the hikaru image on the menu screen
        image = ImageIO.read(getClass().getResourceAsStream(name:"/hikaru.png"));
        Graphics2D g2d = (Graphics2D) welcomeGraphics;
        // the settings for the writings and the click here button
        g2d.draw(clickButton);
        g2d.drawImage(image,x:300,y:380,width:450,height:450,observer:null);
        Font font = new Font(name:"Times New Roman",Font.BOLD, size:70);
        Font font2 = new Font(name:"Taban",Font.BOLD, size:40);
        Font font3 = new Font(name:"Arial",Font.BOLD,size:15);
        welcomeGraphics.setFont(font);
        welcomeGraphics.setColor(Color.darkGray);
        welcomeGraphics.drawString(str:"Hello, click here!", x:70, y:70);
        welcomeGraphics.setFont(font2);
        welcomeGraphics.drawString(str:"tis' a chess board simulator", x:300,y:150);
        welcomeGraphics.drawString(str:"to quit, close the window!", x:400, y:200);
        welcomeGraphics.setFont(font3);
        welcomeGraphics.drawString(str:"This chess board can be used to play a game of chess", x:300, y:300);
        welcomeGraphics.drawString(str:"the chess board can be a good tool to practice and analyze as well", x:350, y:340);
    }

}
```

Those are the codes that are perceived inside the menu class. To draw the image, the same algorithm is used to draw each of the pieces with the exception of the indexing and the loop that crops the picture from the *chess.png*. Functions such as the *render* function and the *rectangle clickButton* are used to draw the text and picture that is present in the menu screen and also as the start button to enter the game.



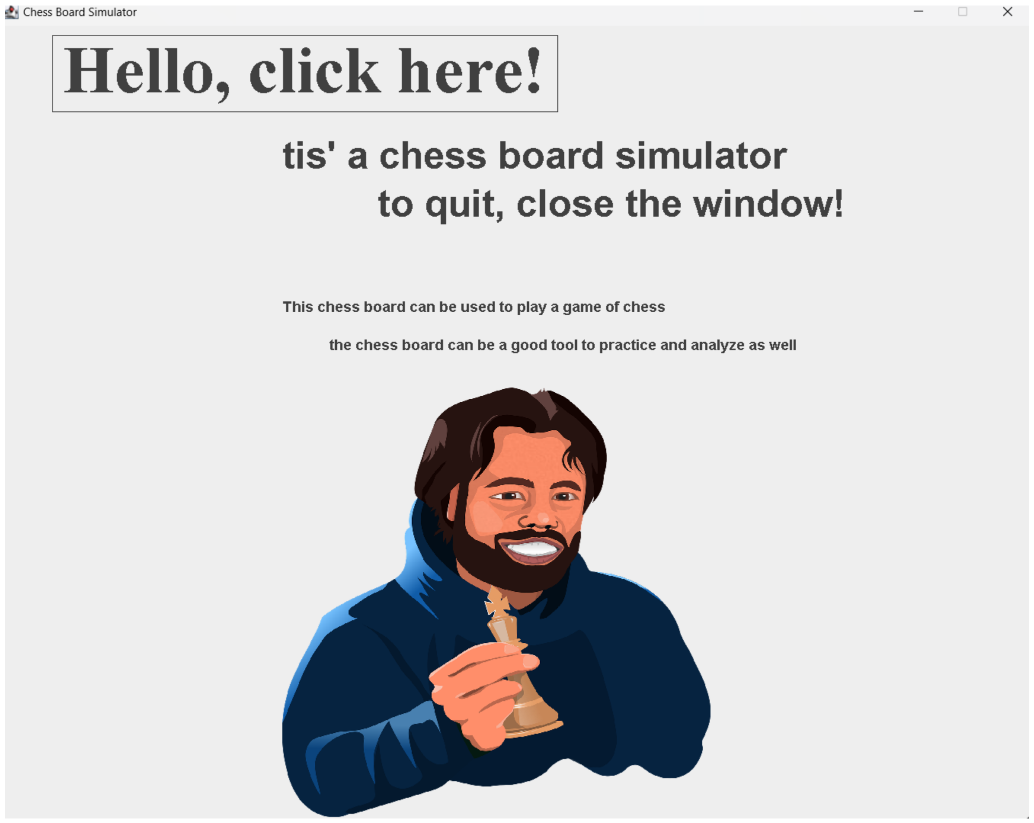
```java
public void mousePressed(MouseEvent e) {
    draggedPiece=DragPiece(e.getX(), e.getY());
    if(status==STATES.MENU) {
        int nx = e.getX();
        int ny = e.getY();
        if(nx >= 70 && nx <= 600){
            if(ny >= 70 && ny <= 200){
                Board.status = Board.STATES.BOARD;
            }
        }
    }
}
```

With the code above, after a click on the rectangle, the enum will go onto the board enum which is what was used for the main board that has the pieces and the tiles.

Those are all the important parts of the program.

D. Result and Source Code

a. Menu Screen


Chess Board Simulator

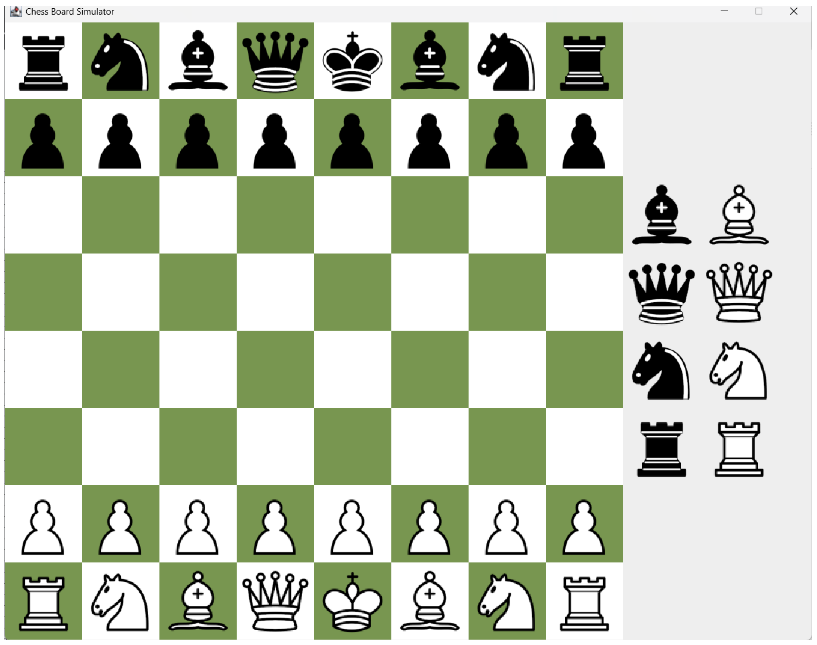
Hello, click here!

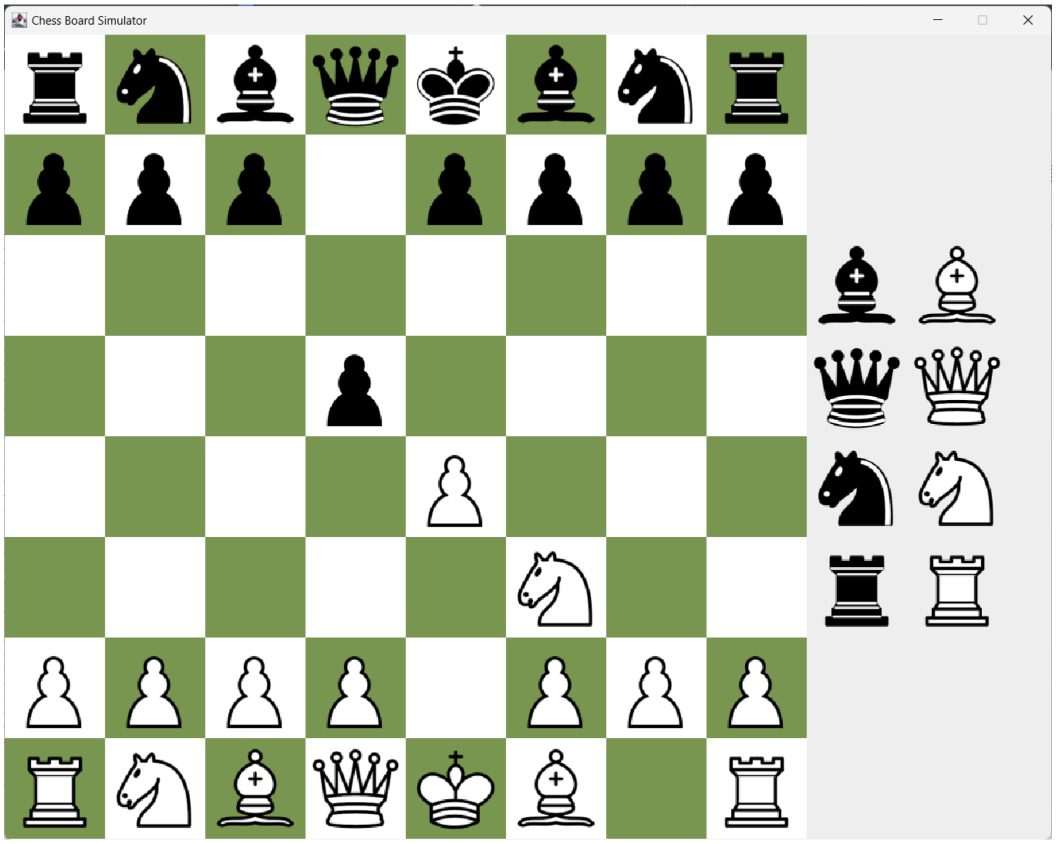
tis' a chess board simulator
to quit, close the window!

This chess board can be used to play a game of chess

the chess board can be a good tool to practice and analyze as well

b. Game Screen





c. Source Code

The program is composed using three classes, Board.java, Piece.java, and Menu.java. All of those codes can be found in this GitHub by following the link below:

https://github.com/SAD-Nich/OOPrelated/tree/0cde7ee66ddc976c57de572fa3976c51e32c782f/Final%20Project

d. Video Demonstration

The video demonstration of the running program can be found by following this link:

https://drive.google.com/drive/folders/1xmd3al9fs1ltIg_DyZr8FYY8tArgF-by?usp=sharing