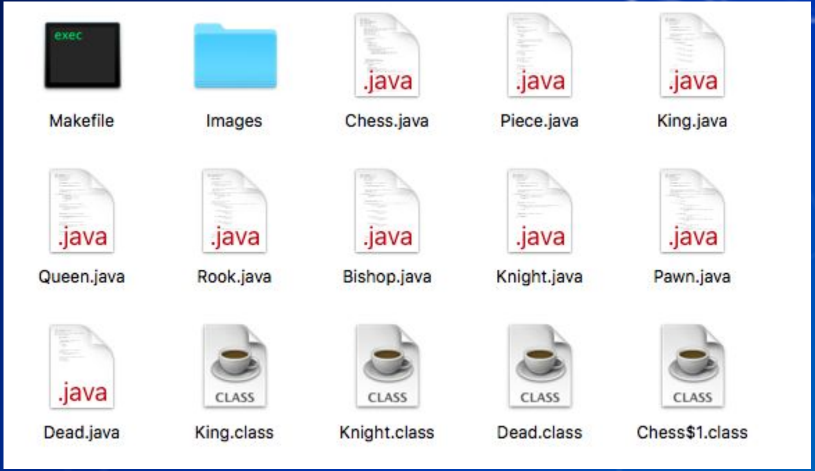
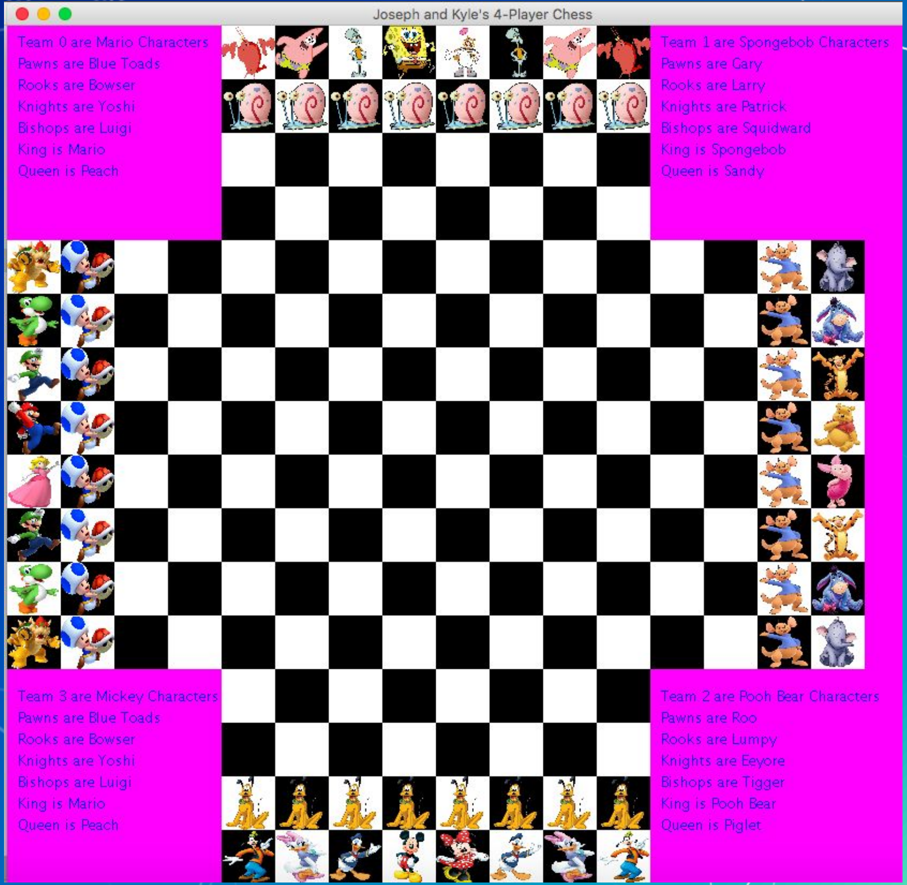
Computer Science Project 2nd Nine Weeks

Joseph Kuang and Kyle Cohen
8th Period
Rosier

Our Work this Quarter

- 4 Player Chess




Joseph and Kyle's 4-Player Chess

Team 0 are Mario Characters
Pawns are Blue Toads
Rooks are Bowser
Knights are Yoshi
Bishops are Luigi
King is Mario
Queen is Peach

Team 1 are Spongebob Characters
Pawns are Gary
Rooks are Larry
Knights are Patrick
Bishops are Squidward
King is Spongebob
Queen is Sandy

Team 3 are Mickey Characters
Pawns are Blue Toads
Rooks are Bowser
Knights are Yoshi
Bishops are Luigi
King is Mario
Queen is Peach

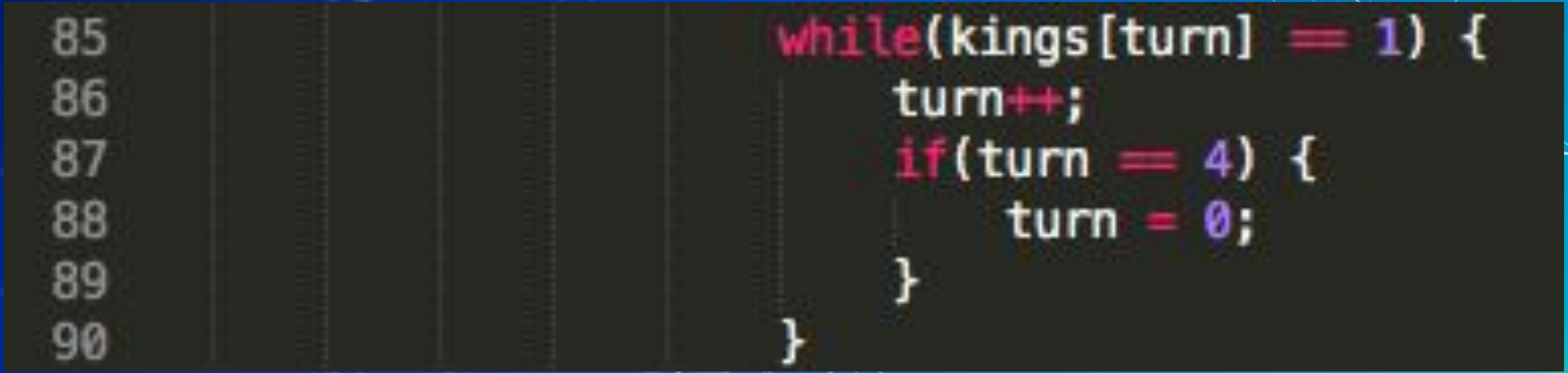
Team 2 are Pooh Bear Characters
Pawns are Roo
Rooks are Lumpy
Knights are Eeyore
Bishops are Tigger
King is Pooh Bear
Queen is Piglet

Why 4 Player Chess

- We decided to make 4 Player Chess because we thought that it would be more fun to make a game this quarter
- We chose 4 Player Chess, opposed to regular two person chess, so that we change the rules of the game to how we felt fit
- We also wanted to practice with graphics and inheritance in Java.

TEK #1: Looping - Single Loop

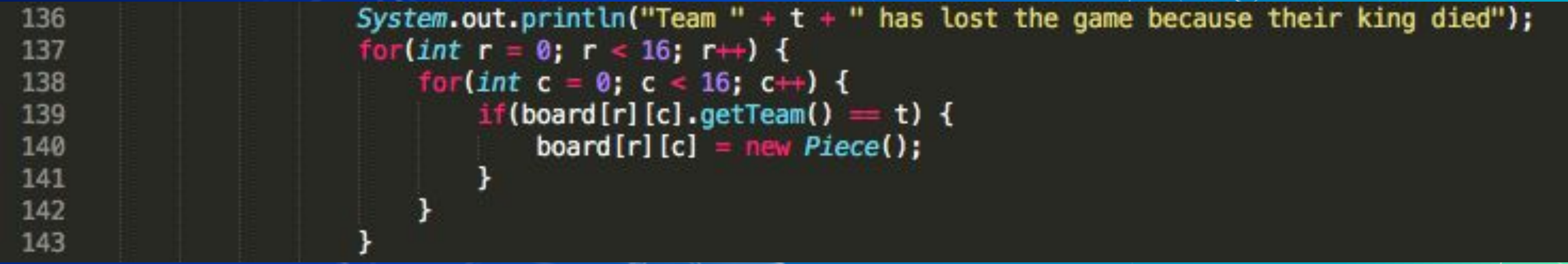
In the chess class of our lab, we used a while loop to loop through the kings array (an array that has 0 if the king is still alive and 1 if the king is dead). The while loop will continue to add 1 to the turn variable, until it finds a team with a king that is not dead to figure out whose turn it is.



```
while(kings[turn] == 1) {
    turn++;
    if(turn == 4) {
        turn = 0;
    }
}
```

TEK #1: Looping - Nested Loop

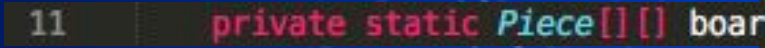
In the chess class of our lab, we use a nested for loop to go through the entire chess board. We needed to go through every spot on the board in order to find the pieces that needed to be removed after their king has died.



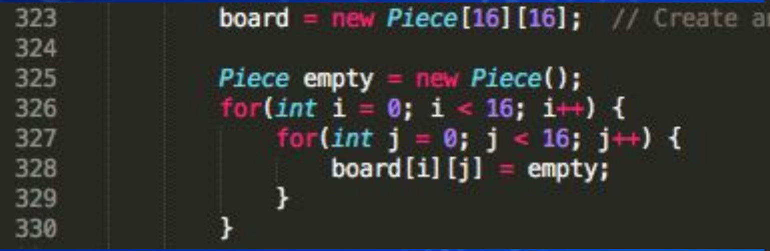
```
            System.out.println("Team " + t + " has lost the game because their king died");
            for(int r = 0; r < 16; r++) {
                for(int c = 0; c < 16; c++) {
                    if(board[r][c].getTeam() == t) {
                        board[r][c] = new Piece();
                    }
                }
            }
```

TEK #2: Data Structures

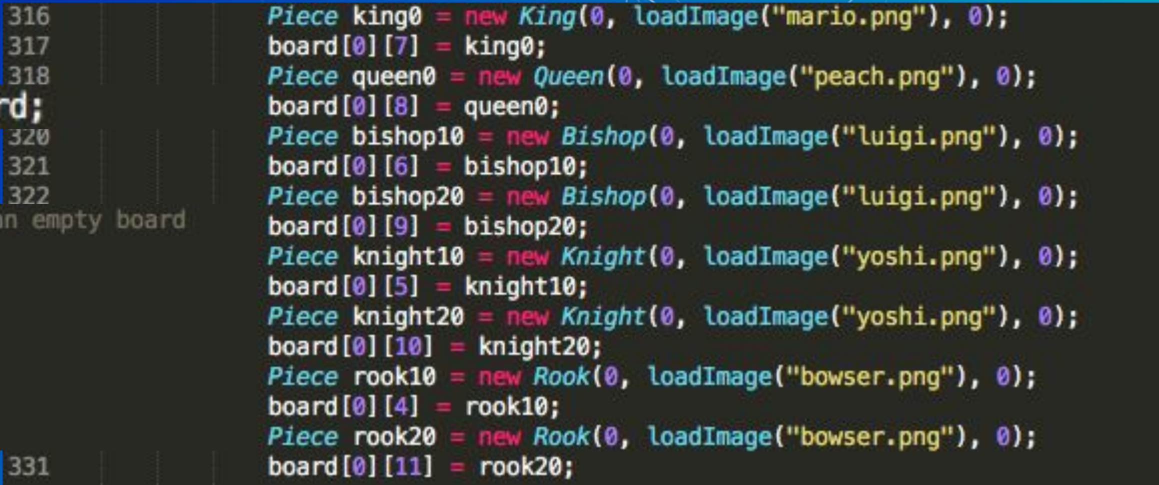
In the chess class of our lab, we created a matrix of Piece (Kings, Queens, Pawns, etc.) objects. Then later in the lab in the createNewGame() method, we assign pieces to some of the spots on the matrix.



```java
private static Piece[][] board;
```

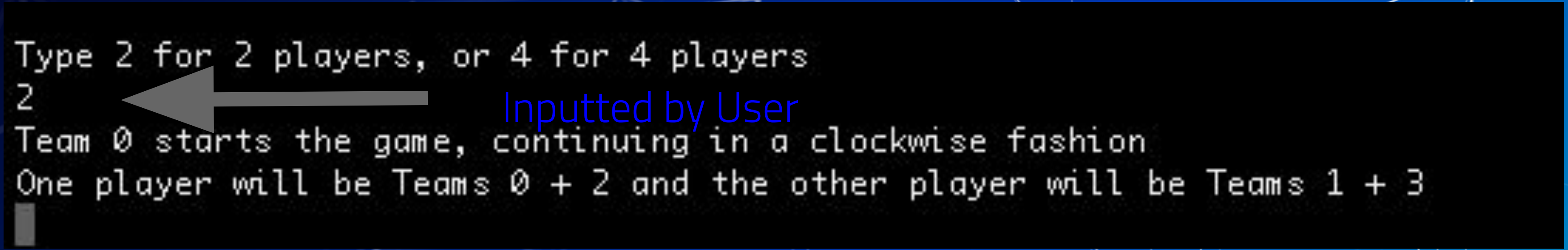


```java
board = new Piece[16][16];  // Create an empty board

Piece empty = new Piece();
for(int i = 0; i < 16; i++) {
    for(int j = 0; j < 16; j++) {
        board[i][j] = empty;
    }
}
```

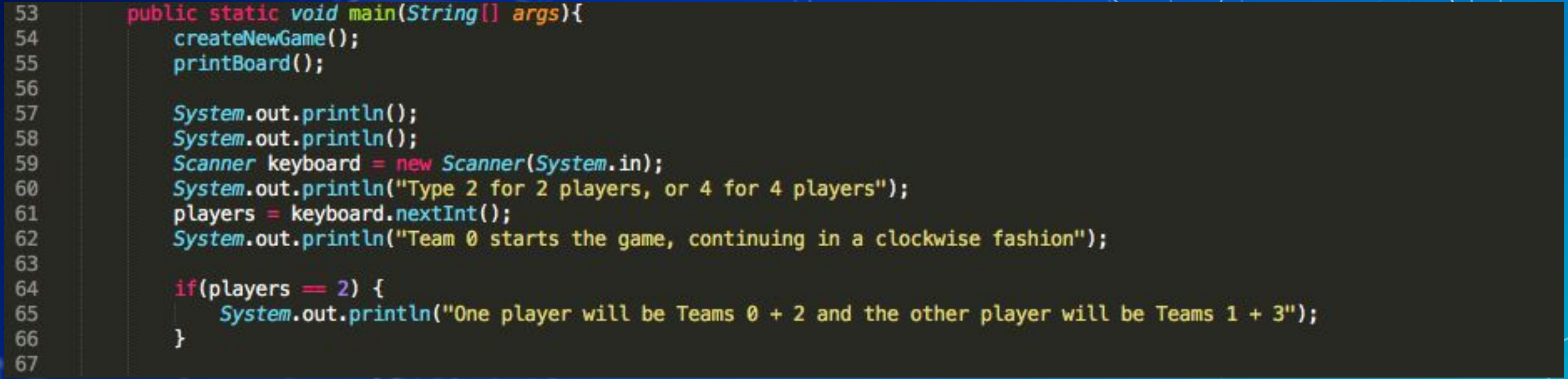


```java
Piece king0 = new King(0, loadImage("mario.png"), 0);
board[0][7] = king0;
Piece queen0 = new Queen(0, loadImage("peach.png"), 0);
board[0][8] = queen0;
Piece bishop10 = new Bishop(0, loadImage("luigi.png"), 0);
board[0][6] = bishop10;
Piece bishop20 = new Bishop(0, loadImage("luigi.png"), 0);
board[0][9] = bishop20;
Piece knight10 = new Knight(0, loadImage("yoshi.png"), 0);
board[0][5] = knight10;
Piece knight20 = new Knight(0, loadImage("yoshi.png"), 0);
board[0][10] = knight20;
Piece rook10 = new Rook(0, loadImage("bowser.png"), 0);
board[0][4] = rook10;
Piece rook20 = new Rook(0, loadImage("bowser.png"), 0);
board[0][11] = rook20;
```

TEK #3: Input and Output



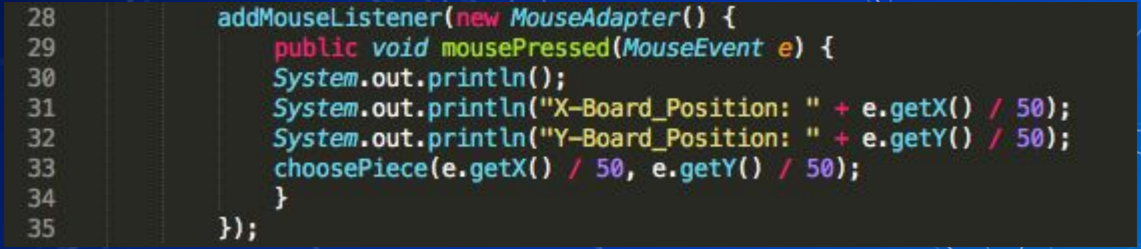
```
Type 2 for 2 players, or 4 for 4 players
2
Team 0 starts the game, continuing in a clockwise fashion
One player will be Teams 0 + 2 and the other player will be Teams 1 + 3
```



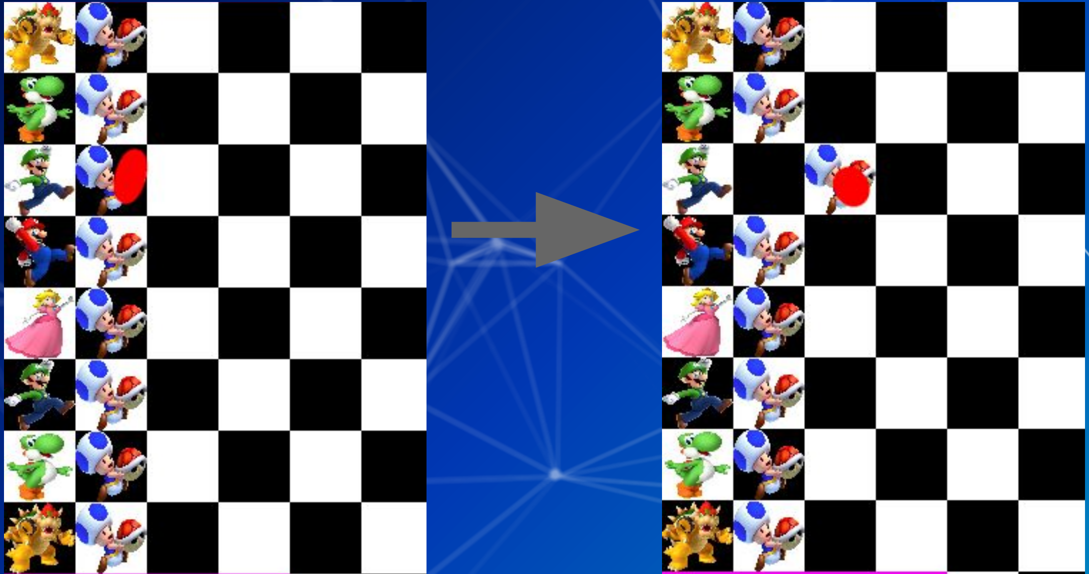
```
    public static void main(String[] args){
        createNewGame();
        printBoard();

        System.out.println();
        System.out.println();
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Type 2 for 2 players, or 4 for 4 players");
        players = keyboard.nextInt();
        System.out.println("Team 0 starts the game, continuing in a clockwise fashion");

        if(players == 2) {
            System.out.println("One player will be Teams 0 + 2 and the other player will be Teams 1 + 3");
        }

```

In the main method of the chess class, the user is prompted to enter the number of players participating. Then the program outputs the details of the game for the number of players.

TEK #4: Input and Output Continued

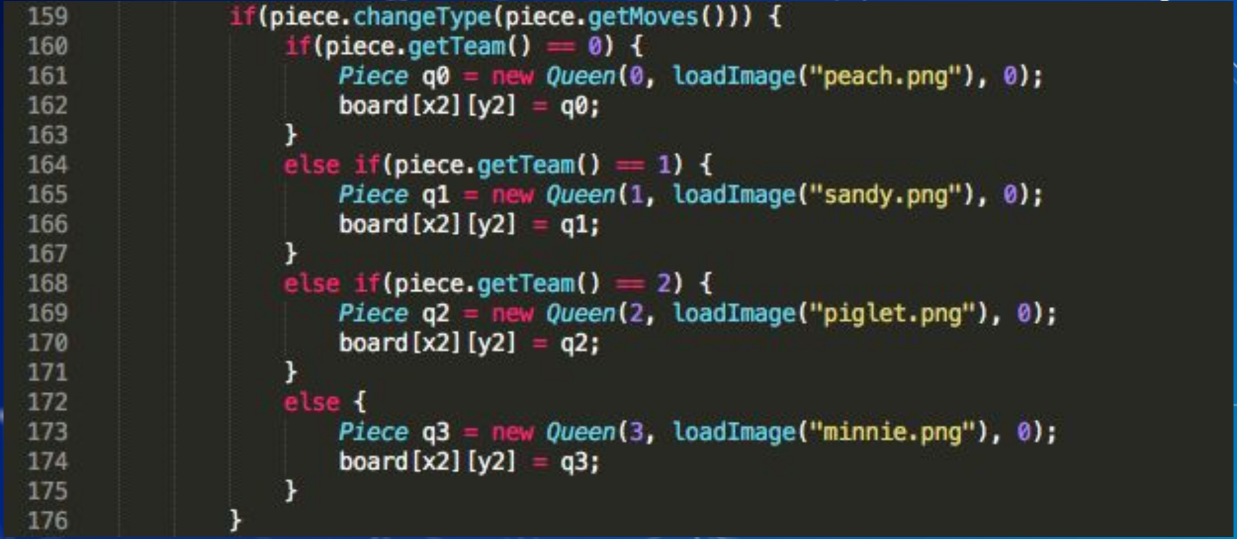


```
addMouseListener(new MouseAdapter() {
    public void mousePressed(MouseEvent e) {
    System.out.println();
    System.out.println("X-Board_Position: " + e.getX() / 50);
    System.out.println("Y-Board_Position: " + e.getY() / 50);
    choosePiece(e.getX() / 50, e.getY() / 50);
    }
});
```



In the Main method of the chess class, we used a mouse listener to record the mouse actions of the user and then it outputs a new screen with the moved piece.

TEK #4: If Else If

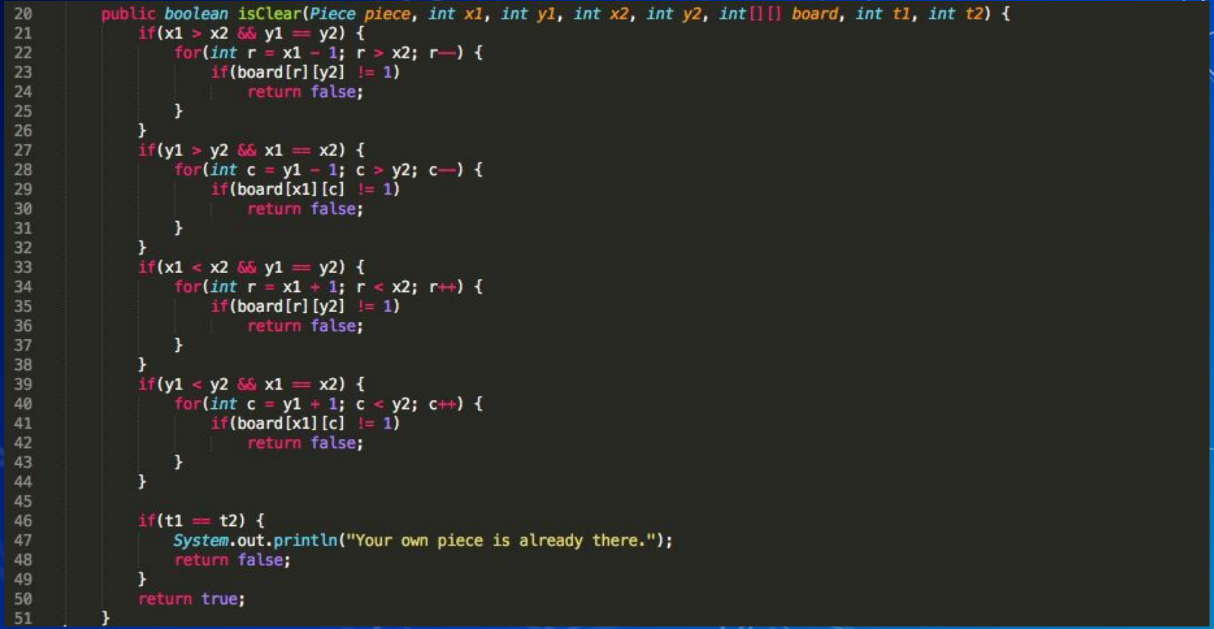


```
if(piece.changeType(piece.getMoves())) {
    if(piece.getTeam() == 0) {
        Piece q0 = new Queen(0, loadImage("peach.png"), 0);
        board[x2][y2] = q0;
    }
    else if(piece.getTeam() == 1) {
        Piece q1 = new Queen(1, loadImage("sandy.png"), 0);
        board[x2][y2] = q1;
    }
    else if(piece.getTeam() == 2) {
        Piece q2 = new Queen(2, loadImage("piglet.png"), 0);
        board[x2][y2] = q2;
    }
    else {
        Piece q3 = new Queen(3, loadImage("minnie.png"), 0);
        board[x2][y2] = q3;
    }
}
```

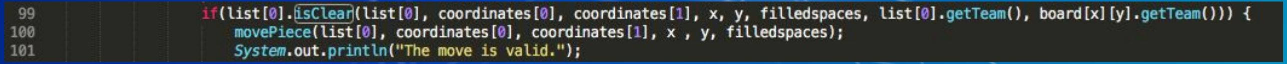
In the chess class, when trying to change a pawn into a queen, we used an if, else if, else statement in order to determine which type of queen object would be set, depending on their respective teams.

TEK #5: Methods

In the isClear Method for the Rook Class, we am trying to determine if there are any pieces in the way of the Rooks path. So we had to check the four different directions that the rook could've traveled in, along with checking the final place to see if its teammate was already there. The method is then called in the Chess class.



```java
    public boolean isClear(Piece piece, int x1, int y1, int x2, int y2, int[][] board, int t1, int t2) {
        if(x1 > x2 && y1 == y2) {
            for(int r = x1 - 1; r > x2; r--) {
                if(board[r][y2] != 1)
                    return false;
            }
        }
        if(y1 > y2 && x1 == x2) {
            for(int c = y1 - 1; c > y2; c--) {
                if(board[x1][c] != 1)
                    return false;
            }
        }
        if(x1 < x2 && y1 == y2) {
            for(int r = x1 + 1; r < x2; r++) {
                if(board[r][y2] != 1)
                    return false;
            }
        }
        if(y1 < y2 && x1 == x2) {
            for(int c = y1 + 1; c < y2; c++) {
                if(board[x1][c] != 1)
                    return false;
            }
        }

        if(t1 == t2) {
            System.out.println("Your own piece is already there.");
            return false;
        }
        return true;
    }
```

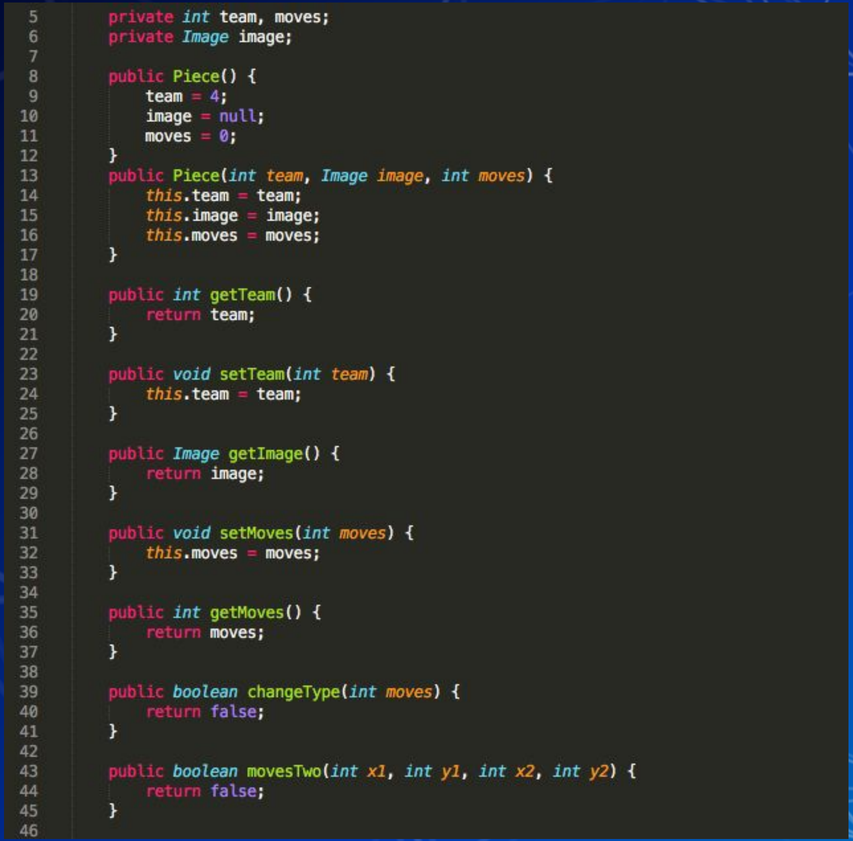


```java
            if(list[0].isClear(list[0], coordinates[0], coordinates[1], x, y, filledspaces, list[0].getTeam(), board[x][y].getTeam())) {
                movePiece(list[0], coordinates[0], coordinates[1], x , y, filledspaces);
                System.out.println("The move is valid.");
```

TEK #6: Classes

Our Chess game has nine different classes. The majority of the code being in the Chess Class. Additionally there are the King, Queen, Pawn, Bishop, Rook, Knight, and Dead Class that all extends the Piece class.

TEK #6: Classes Continued



```
    private int team, moves;
    private Image image;

    public Piece() {
        team = 4;
        image = null;
        moves = 0;
    }
    public Piece(int team, Image image, int moves) {
        this.team = team;
        this.image = image;
        this.moves = moves;
    }

    public int getTeam() {
        return team;
    }

    public void setTeam(int team) {
        this.team = team;
    }

    public Image getImage() {
        return image;
    }

    public void setMoves(int moves) {
        this.moves = moves;
    }

    public int getMoves() {
        return moves;
    }

    public boolean changeType(int moves) {
        return false;
    }

    public boolean movesTwo(int x1, int y1, int x2, int y2) {
        return false;
    }

```

The Piece Class has three instance variables team, moves, and image, which are initialized in the initialization constructor, public Piece(int team, Image image, int moves).

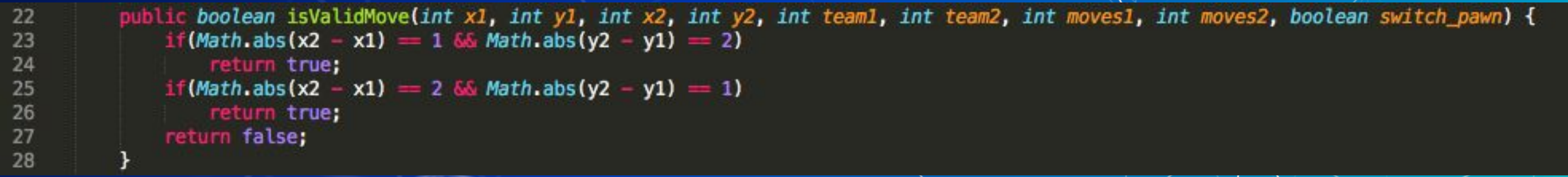
Additionally it has getTeam(), getImage(), and getMoves() methods.

Along with also having a setTeam(int Team) and a setMoves(int moves) method.

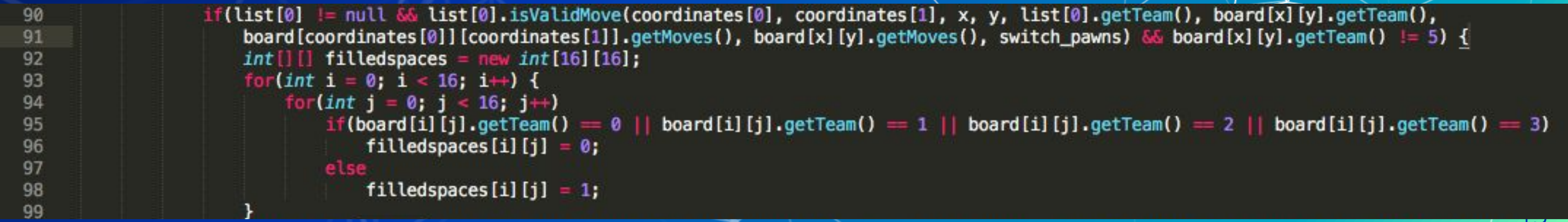
TEK #6: Classes Continued

The Knight Class, which extends the Piece class, has an isValidMove method, which determines if the knight is capable of moving to a certain spot on the board, when given its initial and final positions.

The method is then called in the Chess class to see if the piece is able to move to the desired spot.

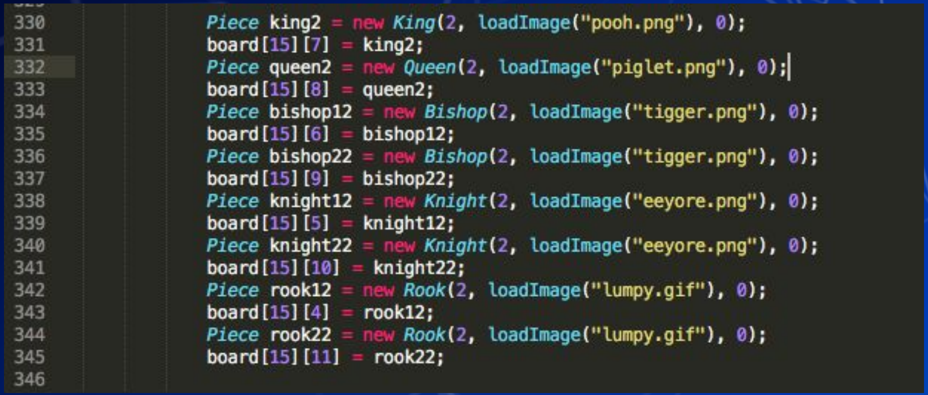


```
    public boolean isValidMove(int x1, int y1, int x2, int y2, int team1, int team2, int moves1, int moves2, boolean switch_pawn) {
        if(Math.abs(x2 - x1) == 1 && Math.abs(y2 - y1) == 2)
            return true;
        if(Math.abs(x2 - x1) == 2 && Math.abs(y2 - y1) == 1)
            return true;
        return false;
    }
```



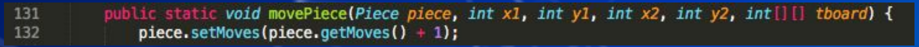
```
            if(list[0] != null && list[0].isValidMove(coordinates[0], coordinates[1], x, y, list[0].getTeam(), board[x][y].getTeam(),
                board[coordinates[0]][coordinates[1]].getMoves(), board[x][y].getMoves(), switch_pawns) && board[x][y].getTeam() != 5) {
                int[][] filledspaces = new int[16][16];
                for(int i = 0; i < 16; i++) {
                    for(int j = 0; j < 16; j++)
                        if(board[i][j].getTeam() == 0 || board[i][j].getTeam() == 1 || board[i][j].getTeam() == 2 || board[i][j].getTeam() == 3)
                            filledspaces[i][j] = 0;
                        else
                            filledspaces[i][j] = 1;
                }
```

TEK #6: Classes Continued



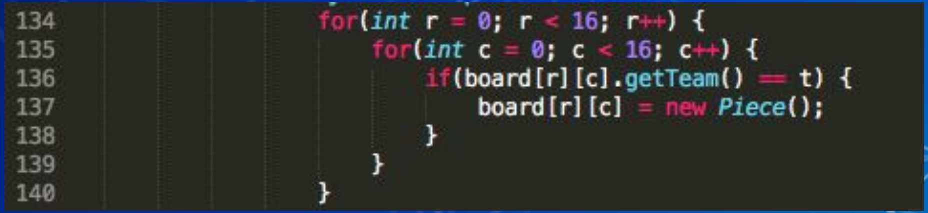
```java
Piece king2 = new King(2, loadImage("pooh.png"), 0);
board[15][7] = king2;
Piece queen2 = new Queen(2, loadImage("piglet.png"), 0);
board[15][8] = queen2;
Piece bishop12 = new Bishop(2, loadImage("tigger.png"), 0);
board[15][6] = bishop12;
Piece bishop22 = new Bishop(2, loadImage("tigger.png"), 0);
board[15][9] = bishop22;
Piece knight12 = new Knight(2, loadImage("eeyore.png"), 0);
board[15][5] = knight12;
Piece knight22 = new Knight(2, loadImage("eeyore.png"), 0);
board[15][10] = knight22;
Piece rook12 = new Rook(2, loadImage("lumpy.gif"), 0);
board[15][4] = rook12;
Piece rook22 = new Rook(2, loadImage("lumpy.gif"), 0);
board[15][11] = rook22;
```

There are many reference variables that refer to the different Piece objects.



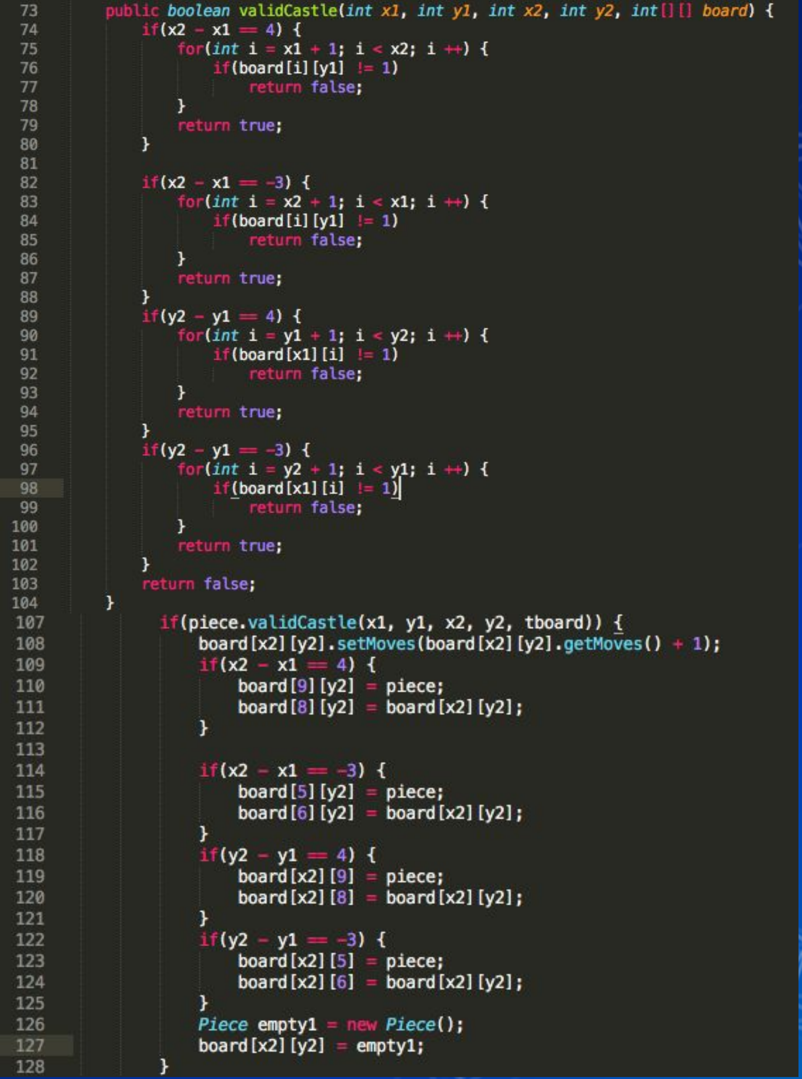
```java
public static void movePiece(Piece piece, int x1, int y1, int x2, int y2, int[][] tboard) {
    piece.setMoves(piece.getMoves() + 1);
```

Additionally in the movePiece method of the Chess class, we use the setMoves method to add one to the number of moves made by that after it has been moved.



```java
for(int r = 0; r < 16; r++) {
    for(int c = 0; c < 16; c++) {
        if(board[r][c].getTeam() == t) {
            board[r][c] = new Piece();
        }
    }
}
```

Also we use the getTeam method in the Chess class to find the team number of the object at a certain point on the board.



```
public boolean validCastle(int x1, int y1, int x2, int y2, int[][] board) {
    if(x2 - x1 == 4) {
        for(int i = x1 + 1; i < x2; i ++) {
            if(board[i][y1] != 1)
                return false;
        }
        return true;
    }

    if(x2 - x1 == -3) {
        for(int i = x2 + 1; i < x1; i ++) {
            if(board[i][y1] != 1)
                return false;
        }
        return true;
    }
    if(y2 - y1 == 4) {
        for(int i = y1 + 1; i < y2; i ++) {
            if(board[x1][i] != 1)
                return false;
        }
        return true;
    }
    if(y2 - y1 == -3) {
        for(int i = y2 + 1; i < y1; i ++) {
            if(board[x1][i] != 1)
                return false;
        }
        return true;
    }
    return false;
}
```

```
        if(piece.validCastle(x1, y1, x2, y2, tboard)) {
            board[x2][y2].setMoves(board[x2][y2].getMoves() + 1);
            if(x2 - x1 == 4) {
                board[9][y2] = piece;
                board[8][y2] = board[x2][y2];
            }

            if(x2 - x1 == -3) {
                board[5][y2] = piece;
                board[6][y2] = board[x2][y2];
            }
            if(y2 - y1 == 4) {
                board[x2][9] = piece;
                board[x2][8] = board[x2][y2];
            }
            if(y2 - y1 == -3) {
                board[x2][5] = piece;
                board[x2][6] = board[x2][y2];
            }
            Piece empty1 = new Piece();
            board[x2][y2] = empty1;
        }
```

TEK #6: Classes Continued

The King Class has the method validCastle, which determines if it is a valid castle between the chosen king and rook

The method is then called in the Chess class to determine if the castle was a valid move for the user so that the board can change.