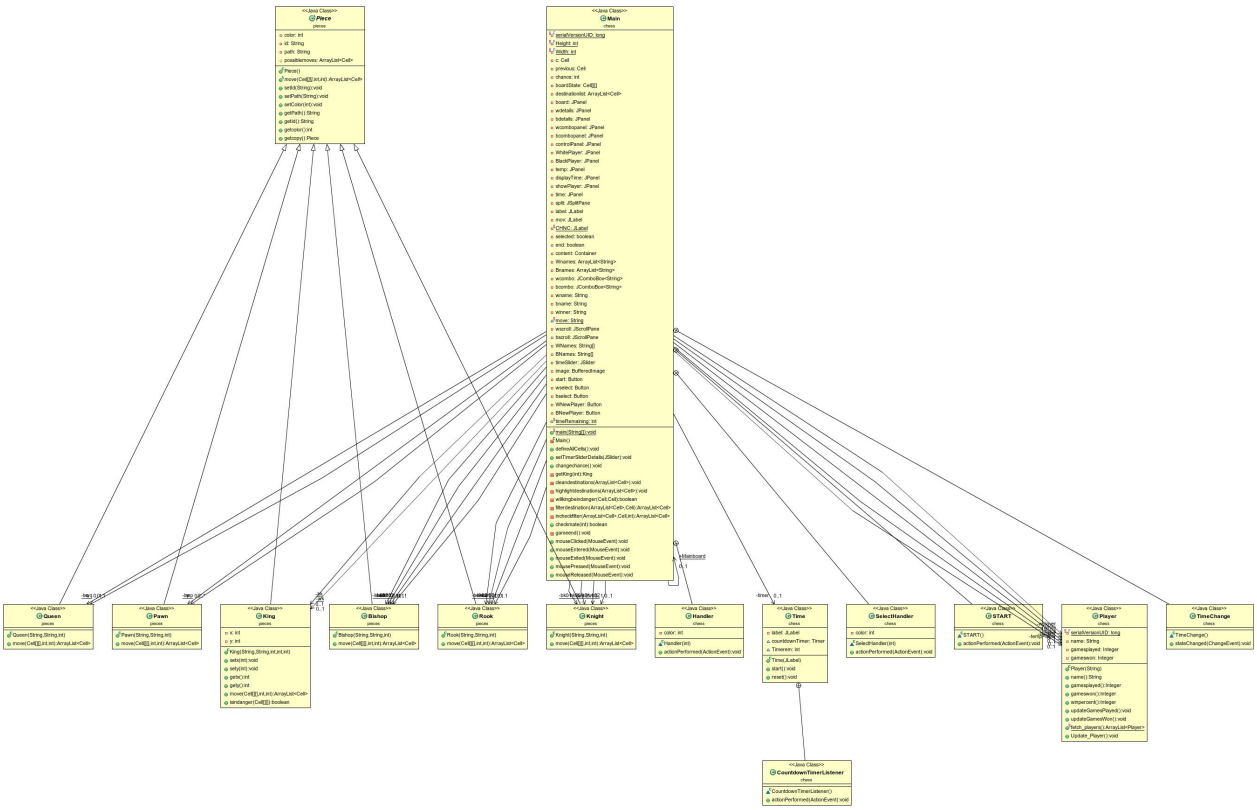
Design Patterns and Refactoring

Phase-1

Group Members:

Muhammad Umar Hayat	16030011
Muhammad Feroze Waris	16030056
Muhammad Qasim Hunain	16030012
Hassaan Fayyaz	16030009

Class Diagram



Code Smells

1.
Code Smell: Long Methods

Class: Main.java
Method: defineAllCells()
Re-factoring: Extract Method

```
//A function used to define all cells
public void defineAllCells(){
pieces.Piece P;
Cell cell;
boardState=new Cell[8][8];
for(int i=0;i<8;i++)
    for(int j=0;j<8;j++)
    {
        P=null;
        if(i==0&&j==0)
            P=br01;
        else if(i==0&&j==7)
            P=br02;
        else if(i==7&&j==0)
            P=wr01;
        else if(i==7&&j==7)
            P=wr02;
        else if(i==0&&j==1)
            P=bk01;
        else if (i==0&&j==6)
            P=bk02;
        else if(i==7&&j==1)
            P=wk01;
        else if (i==7&&j==6)
            P=wk02;
        else if(i==0&&i==2)
```

2.
Code Smell: Long Methods
Class: Main.java
Re-factoring: Extract Method

```
//variable initialization
wr01=new Rook("WR01","White_Rook.png",0);
wr02=new Rook("WR02","White_Rook.png",0);
br01=new Rook("BR01","Black_Rook.png",1);
br02=new Rook("BR02","Black_Rook.png",1);
wk01=new Knight("WK01","White_Knight.png",0);
wk02=new Knight("WK02","White_Knight.png",0);
bk01=new Knight("BK01","Black_Knight.png",1);
bk02=new Knight("BK02","Black_Knight.png",1);
wb01=new Bishop("WB01","White_Bishop.png",0);
wb02=new Bishop("WB02","White_Bishop.png",0);
bb01=new Bishop("BB01","Black_Bishop.png",1);
bb02=new Bishop("BB02","Black_Bishop.png",1);
wq=new Queen("WQ","White_Queen.png",0);
bq=new Queen("BQ","Black_Queen.png",1);
wk=new King("WK","White_King.png",0,7,3);
bk=new King("BK","Black_King.png",1,0,3);
wp=new Pawn[8];
bp=new Pawn[8];
for(int i=0;i<8;i++)
{
    wp[i]=new Pawn("WP0"+(i+1),"White_Pawn.png",0);
    bp[i]=new Pawn("BP0"+(i+1),"Black_Pawn.png",1);
}
```

3.
Code smell: Long Method
Class: Main.java
Re-factoring: Extract Method

```
for(int i=0;i<8;i++)
{
    wp[i]=new Pawn("WP0"+(i+1),"White_Pawn.png",0);
    bp[i]=new Pawn("BP0"+(i+1),"Black_Pawn.png",1);
}

//Setting up the board
Mainboard = new Main();
Mainboard.setVisible(true);
Mainboard.setResizable(false);
}
```

4.
Code smell: Long Method
Class: Player.java
Method: update_player
Re-factoring: Extract Method and Renaming

```
ObjectInputStream input = null;
ObjectOutputStream output = null;
Player temp_player;
File inputfile=null;
File outputfile=null;
try
{
    inputfile = new File(System.getProperty("user.dir")+ File.separator + "chessgamedata.dat");
    outputfile = new File(System.getProperty("user.dir")+ File.separator + "tempfile.dat");
} catch (SecurityException e)
{
    JOptionPane.showMessageDialog(null, "Read-Write Permission Denied !! Program Cannot Start");
    System.exit(0);
}
boolean playerdonotexist;
try
{
    if(outputfile.exists()==false)
        outputfile.createNewFile();
    if(inputfile.exists()==false)
    {
            output = new ObjectOutputStream(new java.io.FileOutputStream(outputfile,true));
            output.writeObject(this);
    }
    else
    {
        input = new ObjectInputStream(new FileInputStream(inputfile));
        output = new ObjectOutputStream(new FileOutputStream(outputfile));
```

5.
Code Smell: Type Checking
Class: Main.java
Method: mouseClicked
Re-factoring: Replace Conditional with Polymorphism

```
                destinationlist.clear();
                destinationlist=c.getpiece().move(boardState, c.x, c.y);
                if(c.getpiece() instanceof King)
                    destinationlist=filterdestination(destinationlist,c);
                else
                {
                    if(boardState[getKing(chance).getx()][getKing(chance).gety()].ischeck())
                        destinationlist = new ArrayList<Cell>(filterdestination(destinationlist,c));
                    else if(destinationlist.isEmpty()==false && willkingbeindanger(c,destinationlist.get(0)))
                        destinationlist.clear();
                }
                highlightdestinations(destinationlist);
            }
        }
```

6.
Code Smell: Type Checking
Class: Cell.java
Method: removePiece
Re-factoring: Replace Conditional with polymorphism

```
public void removePiece()      //Function to remove a piece from the cell
{
    if (piece instanceof King)
    {
        piece=null;
        this.remove(content);
    }
    else
    {
        piece=null;
        this.remove(content);
    }
}
```

7.
Code Smell: God Class
Class: Main.java
Re-factoring: Extract Class

```
private boolean willkingbeindanger(Cell fromcell,Cell tocell)
{
    Cell newboardstate[][] = new Cell[8][8];
    for(int i=0;i<8;i++)
        for(int j=0;j<8;j++)
        {   try { newboardstate[i][j] = new Cell(boardState[i][j]);} catch (CloneNotSupportedException e){e.printStackTrace(); System.out.println

    if(newboardstate[tocell.x][tocell.y].getpiece()!=null)
        newboardstate[tocell.x][tocell.y].removePiece();

    newboardstate[tocell.x][tocell.y].setPiece(newboardstate[fromcell.x][fromcell.y].getpiece());
    if(newboardstate[tocell.x][tocell.y].getpiece() instanceof King)
    {
        ((King)(newboardstate[tocell.x][tocell.y].getpiece())).setx(tocell.x);
        ((King)(newboardstate[tocell.x][tocell.y].getpiece())).sety(tocell.y);
    }
    newboardstate[fromcell.x][fromcell.y].removePiece();
    if (((King)(newboardstate[getKing(chance).getx()][getKing(chance).gety()].getpiece())).isindanger(newboardstate)==true)
        return true;
    else
        return false;
}

//A function to eliminate the possible moves that will put the King in danger
private ArrayList<Cell> filterdestination (ArrayList<Cell> destlist, Cell fromcell)
{
    ArrayList<Cell> newlist = new ArrayList<Cell>();
    Cell newboardstate[][] = new Cell[8][8];
```

8.
Code Smell: Java naming convention violated
Class: Main.java
Re-factoring: Use meaningful names

```
private static Rook wr01,wr02,br01,br02;
private static Knight wk01,wk02,bk01,bk02;
private static Bishop wb01,wb02,bb01,bb02;
private static Pawn wp[],bp[];
private static Queen wq,bq;
```

9.
Code Smell: Avoid using implementation types like 'ArrayList'
Class: Main.java
Re-factoring: Use the interface instead. i.e, List interface

```
private ArrayList<Cell> destinationlist = new ArrayList<Cell>();
```

10.
Code Smell: Some variables in code start with uppercase character.
Class: Main.java
Re-factoring: Variables should start with a lowercase character

```
private ArrayList<String> Wnames=new ArrayList<String>();
private ArrayList<String> Bnames=new ArrayList<String>();
```

11.
Code Smell: Action level scope is defined for the variables which are used locally in one function only.
Class: Main.java

Re-factoring: Perhaps 'wscroll' and 'bscroll' could be replaced by a local variable.

```
    private JScrollPane wscroll,bscroll;
```

12.
Code Smell: Bad Programming Practice.
Class: Main.java
Re-factoring: Use a logger instead.

```
                    System.out.println("not found");
```

13.
Code Smell: Method Chaining.
Class: Main.java
Re-factoring:

```
        newboardstate[fromcell.x][fromcell.y].removePiece();
        if (((King)(newboardstate[getKing(chance).getx()][getKing(chance).gety()].getpiece())).isindanger(newboardstate)==true)
            return true;
        else
            return false;
    }
```

14.
Code Smell: Short variable names used
Class: King.java
Re-factoring: Use meaningful names

```
    //King Constructor
    public King(String i,String p,int c,int x,int y)
    {
        setx(x);
        sety(y);
        setId(i);
        setPath(p);
        setColor(c);
    }
```

15.
Code Smell: Useless parentheses.
Class: King.java
Re-factoring: Do not use extra parenthesis.

```
int posx[]={x,x,x+1,x+1,x+1,x-1,x-1,x-1};
int posy[]={y-1,y+1,y-1,y,y+1,y-1,y,y+1};
for(int i=0;i<8;i++)
    if((posx[i]>=0&&posx[i]<8&&posy[i]>=0&&posy[i]<8))
```

16.

Code Smell: Unnecessary long comment.

Class: Time.java

Re-factoring: Do not use comments. Instead function or class name should be descriptive.

```
}

//A function that is called after every second. It updates the timer and takes other necessary actions
class CountdownTimerListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
     int min,sec;
     if (Timerem > 0)
     {
        min=Timerem/60;
        sec=Timerem%60;
```

17.

Code Smell: if and for statements used without curly braces.

Class: Knight.java

Re-factoring: Avoid using if and for statements without curly braces

```
    for(int i=0;i<8;i++)
        if((posx[i]>=0&&posx[i]<8&&posy[i]>=0&&posy[i]<8))
            if((state[posx[i]][posy[i]].getpiece()==null||state[posx[i]][posy[i]].getpiece().getcolor()!=this.getcolor()))
            {
                possiblemoves.add(state[posx[i]][posy[i]]);
            }
    return possiblemoves;
}
```