JAVA\_Assignment05

## Calling the user defined class

2 user defined classes to be created and called from the main class.

class SquareSeries

{

    public static void main(String args[]) throws IOException

    {

        int n;

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        System.out.println(\"Enter no.of elements in series: \");

        n = Integer.parseInt(br.readLine());

        if(n>0)

        {

            System.out.println(\"Series:\");

            for(int i=0; i<(n-1); i++)

            {

                System.out.println(\"\\t\"+(int)(Math.pow(i,2)));

            }

        }

        else

            System.out.println(\"You Entered the Wrong no.\");

    }

}

import java.io.\*;

public class Assignment3 //class name here, same as file name

{

public Assignment3() throws IOException{ //constructor, place class name here

// use BufferedReader class to input from the keyboard

// declare a variable of type BufferedReader

BufferedReader input = new BufferedReader(new InputStreamReader(System.in));

//declare variable for input

String inputString;

// houseKeeping()

String yourNumber;

int number;

int totalSquare = 0;

int totalCube = 0;

int count;

int badNumber=0;

String squareCube = " Your number squared is" +square +"your number cubed is"+cube;

System.out.print("Enter a number: ");

inputString = input.readLine();

yourNumber = inputString;

}//end constructor

}

public static void main(String [] args) throws IOException

{

new Assignment3(); //class constructor name

}

## Number of Years of Living

**Create an input box which accepts Date as input**

* + The output should be an excel file with (if date is valid)
  + Number of Years you have lived
  + Number of Months
  + Number of Days
  + Number of Seconds

[GUI message from Java Program]

* + If you have lived more than 18 years then you are eligible for voting
  + If you have lived more than 60 Years then you are a senior citizen
  + If you have lived less than 18 years then you are not eligible to vote
  + If the date is greater than the current system date then 'Best Wishes when you are born'
  + If the date component is same then display 'Happy birthday message'

public static void doPrintMonth( String pattern,String input ) {

try{

SimpleDateFormat sdf=new SimpleDateFormat(pattern);

Date output=sdf.parse(input);

String mon[]={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};

int m=output.getMonth();

System.out.println("\n\t" + mon[m] );

System.exit(0);

}

catch(Exception e){}

}

}

public class VariableDateParser {

private static final Pattern DATE\_PATTERN = Pattern

.compile("((?:(?:\\d+(?:[./]\\s\*)?)+)?)\\s\*((?:(?:\\d+[:]?)+)?)");

public Date getDate(final String dateString) {

final Calendar calendar = Calendar.getInstance();

final Matcher matcher = DATE\_PATTERN.matcher(dateString);

if (matcher.matches()) {

final String dateGroup = matcher.group(1).trim();

if (!"".equals(dateGroup)) {

final Iterator<Integer> fields = Arrays.asList(

Calendar.DATE, Calendar.MONTH, Calendar.YEAR).iterator();

final String[] items = dateGroup.split("\\D+");

for (final String item : items) {

if ("".equals(item))

break;

else if (fields.hasNext()) {

final Integer field = fields.next();

calendar.set(field, Integer.parseInt(item) -

// months are 0-based, grrrr!!!

(field.equals(Calendar.MONTH) ? 1 : 0));

} else {

throw new IllegalArgumentException(

"Bad date part: " + dateGroup);

}

}

}

final String timeGroup = matcher.group(2).trim();

if (!"".equals(timeGroup)) {

final Iterator<Integer> fields = Arrays.asList(

Calendar.HOUR, Calendar.MINUTE, Calendar.SECOND,

Calendar.MILLISECOND).iterator();

final String[] items = timeGroup.split("\\D+");

for (final String item : items) {

if ("".equals(item))

break;

else if (fields.hasNext()) {

final Integer field = fields.next();

calendar.set(field, Integer.parseInt(item));

} else {

throw new IllegalArgumentException(

"Bad time part: " + timeGroup);

}

}

}

} else

throw new IllegalArgumentException(

"Bad date string: " + dateString);

return calendar.getTime();

}

}

public static void main(final String[] args) {

VariableDateParser parser = new VariableDateParser();

DateFormat df = DateFormat.getDateTimeInstance(

DateFormat.MEDIUM, DateFormat.LONG, Locale.GERMAN);

System.out.println(df.format(parser.getDate("11")));

System.out.println(df.format(parser.getDate("11. 10.")));

System.out.println(df.format(parser.getDate("11. 10. 4")));

System.out.println(df.format(parser.getDate("11. 10. 2004")));

System.out.println(df.format(parser.getDate("11. 10. 2004 11")));

System.out.println(df.format(parser.getDate("11. 10. 2004 11:35")));

System.out.println(df.format(parser.getDate("11. 10. 2004 11:35:18")));

System.out.println(df.format(parser.getDate("11. 10. 2004 11:35:18:123")));

System.out.println(df.format(parser.getDate("11:35")));

System.out.println(df.format(parser.getDate("11:35:18")));

System.out.println(df.format(parser.getDate("11:35:18:123")));

}

3. Bank dynamics implementation

Problem statement:

The following are the requirements for which the object design and implementation should happen

o Should start with the bank balance 2500;

o User can make a deposit any money, which gets added to the bank balance

o User can make an withdraw any money, which gets credited from the bank balance

Create 2 bank objects bankTom and bankJack

o Both starts with 2500 in their bank balance

o Tom makes 4 deposit sequentially (of Rs.150 each) and 1 withdrawl (of Rs.1000)

o Jack makes 2 withdrawls sequentially (of Rs. 100 each) and 1 deposit of 500 Rs.

Conditions

1. All the inputs should be sent in the console and there should not be any hard-coding (other than the starting bank balance)

2. There should not be unnecessary variables declared

import java.util.Scanner;

class Bank

{

private String accno;

private String name;

private long balance;

Scanner KB=new Scanner(System.in);

//method to open an account

void openAccount()

{

System.out.print("Enter Account No: ");

accno=KB.next();

System.out.print("Enter Name: ");

name=KB.next();

System.out.print("Enter Balance: ");

balance=KB.nextLong();

}

//method to display account details

void showAccount()

{

System.out.println(accno+","+name+","+balance);

}

//method to deposit money

void deposit()

{

long amt;

System.out.println("Enter Amount U Want to Deposit : ");

amt=KB.nextLong();

balance=balance+amt;

}

//method to withdraw money

void withdrawal()

{

long amt;

System.out.println("Enter Amount U Want to withdraw : ");

amt=KB.nextLong();

if(balance>=amt)

{

balance=balance-amt;

}

else

{

System.out.println("Less Balance..Transaction Failed..");

}

}

//method to search an account number

boolean search(String acn)

{

if(accno.equals(acn))

{

showAccount();

return(true);

}

return(false);

}

}

class ExBank

{

public static void main(String arg[])

{

Scanner KB=new Scanner(System.in);

//create initial accounts

System.out.print("How Many Customer U Want to Input : ");

int n=KB.nextInt();

Bank C[]=new Bank[n];

for(int i=0;i<C.length;i++)

{

C[i]=new Bank();

C[i].openAccount();

}

//run loop until menu 5 is not pressed

int ch;

do

{

System.out.println("Main Menu\n

1.Display All\n

2.Search By Account\n

3.Deposit\n

4.Withdrawal\n

5.Exit");

System.out.println("Ur Choice :");

ch=KB.nextInt();

switch(ch)

{

case 1:

for(int i=0;i<C.length;i++)

{

C[i].showAccount();

}

break;

case 2:

System.out.print("Enter Account No U Want to Search...: ");

String acn=KB.next();

boolean found=false;

for(int i=0;i<C.length;i++)

{

found=C[i].search(acn);

if(found)

{

break;

}

}

if(!found)

{

System.out.println("Search Failed..Account Not Exist..");

}

break;

case 3:

System.out.print("Enter Account No : ");

acn=KB.next();

found=false;

for(int i=0;i<C.length;i++)

{

found=C[i].search(acn);

if(found)

{

C[i].deposit();

break;

}

}

if(!found)

{

System.out.println("Search Failed..Account Not Exist..");

}

break;

case 4:

System.out.print("Enter Account No : ");

acn=KB.next();

found=false;

for(int i=0;i<C.length;i++)

{

found=C[i].search(acn);

if(found)

{

C[i].withdrawal();

break;

}

}

if(!found)

{

System.out.println("Search Failed..Account Not Exist..");

}

break;

case 5:

System.out.println("Good Bye..");

break;

}

}

while(ch!=5);

}

}

Output

How Many Customer U Want to Input : 2

Enter Account No: 101

Enter Name: Chintu

Enter Balance: 25000

Enter Account No: 102

Enter Name: Alexander

Enter Balance: 30000

Main Menu

1.Display All

2.Search By Account

3.Deposit

4.Withdrawal

5.Exit

Ur Choice :

1

101,Chintu,25000

102,Alexander,30000

Main Menu

1.Display All

2.Search By Account

3.Deposit

4.Withdrawal

5.Exit

Ur Choice :

2

Enter Account No U Want to Search...: 102

102,Alexander,30000

Main Menu

1.Display All

2.Search By Account

3.Deposit

4.Withdrawal

5.Exit

Ur Choice :

2

Enter Account No U Want to Search...: 105

Search Failed..Account Not Exist..

Main Menu

1.Display All

2.Search By Account

3.Deposit

4.Withdrawal

5.Exit

Ur Choice :

3

Enter Account No : 102

102,Alexander,30000

Enter Amount U Want to Deposit :

25000

Main Menu

1.Display All

2.Search By Account

3.Deposit

4.Withdrawal

5.Exit

Ur Choice :

3

Enter Account No : 105

Search Failed..Account Not Exist..

Main Menu

1.Display All

2.Search By Account

3.Deposit

4.Withdrawal

5.Exit

Ur Choice :

4

Enter Account No : 102

102,Alexander,55000

Enter Amount U Want to withdraw :

15000

Main Menu

1.Display All

2.Search By Account

3.Deposit

4.Withdrawal

5.Exit

Ur Choice :

4

Enter Account No : 105

Search Failed..Account Not Exist..

Main Menu

1.Display All

2.Search By Account

3.Deposit

4.Withdrawal

5.Exit

Ur Choice :

5

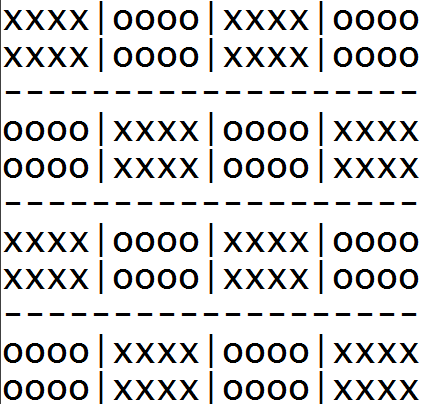
Good Bye..

## Chessboard

Write a program to display a chess board as below

1. Consider x as black O as white
2. If the input is 1x1 matrix the black and white pattern should be X and O
3. If the input is 2x1 matrix the black and white pattern shoud be XX and OO
4. Likewise the 'X' and 'O' pattern should be designed using runtime value
5. Use '|' as vertical separator.
6. Use '-' as horizontal separator.
7. The size of the chess board and size of Pattern will be retrieved from the user.

**Sample output:**



public class MainPanel extends JPanel {

private player1 P1=new player1();

private player2 P2=new player2();

private final int Divide=600/8;

private int move =0;

private Rectangle2D rec;

private short players\_turn=1;

public final ToolPanel myTool;

private final StatusPanel myStatus;

private boolean GameOver=false;

private boolean Iam\_Server=false;

private boolean Iam\_Client=false;

private ServerSocket ServerSock;

private Socket Sock;

private BufferedReader in;

private PrintWriter out;

private String Box;

private boolean local=true;

private JButton startServer;

private JButton startClient;

private String MyIp\_Address;

private String MyPort\_number;

private boolean Game\_started=true;

private Recv\_Thread Recv\_from;

private ChatPanel Refe\_Chat;

public void start\_As\_Server(String Ip,String Port,ChatPanel newChat) {

Recv\_from=new Recv\_Thread();

Refe\_Chat=newChat;

Game\_started=false;

MyIp\_Address=Ip;

MyPort\_number=Port;

start\_Again();

startServer=new JButton(" Start server");

startServer.setSize(150,25);

startServer.setLocation(200,300);

startServer.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

ServerSock=new ServerSocket(Integer.parseInt(MyPort\_number));

Thread Server=new Thread(new Runnable() {

public synchronized void run() {

try {

Sock=ServerSock.accept();

Refe\_Chat.listen\_chat();

in=new BufferedReader(new InputStreamReader(Sock.getInputStream()));

out=new PrintWriter(Sock.getOutputStream());

startServer.setVisible(false);

startServer=null;

Recv\_from.start();

Game\_started=true;

} catch (IOException ex) {

ex.printStackTrace();

}

}

});

Server.start();

/\*in=new BufferedReader(new InputStreamReader(Sock.getInputStream()));

out=new PrintWriter(Sock.getOutputStream());\*/

// Sock.setSoTimeout(999999);

// Refe\_Chat.listen\_chat();

} catch (IOException ex) {

ex.printStackTrace();

JOptionPane.showConfirmDialog(null,"Server error","Error",JOptionPane.ERROR\_MESSAGE);

}

startServer.setText("Waiting...");

}

});

local=false;

add(startServer);

Iam\_Server=true;

repaint();

}

public void start\_As\_Client(String Ip,String Port,ChatPanel newChat) {

Recv\_from=new Recv\_Thread();

Refe\_Chat=newChat;

Game\_started=false;

start\_Again();

MyIp\_Address=Ip;

MyPort\_number=Port;

local=false;

startClient=new JButton("Start Client");

startClient.setSize(150,25);

startClient.setLocation(200,300);

startClient.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

Sock=new Socket(MyIp\_Address,Integer.parseInt(MyPort\_number));

in=new BufferedReader(new InputStreamReader(Sock.getInputStream()));

out=new PrintWriter(Sock.getOutputStream());

Recv\_from.start();

Game\_started=true;

Refe\_Chat.start\_chat();

} catch (UnknownHostException ex) {

ex.printStackTrace();

} catch (IOException ex) {

ex.printStackTrace();

JOptionPane.showConfirmDialog(null,"Client error","Error",JOptionPane.ERROR\_MESSAGE);

}

startClient.setVisible(false);

startClient=null;

}

});

Iam\_Client=true;

add(startClient);

}

public void start\_Again() {

P1=new player1();

P2=new player2();

move =0;

players\_turn=1;

GameOver=false;

local=true;

myTool.start\_Again();

myStatus.start\_Again();

Iam\_Server=false;

Iam\_Client=false;

repaint();

}

public MainPanel(ToolPanel myToolPanel,StatusPanel myStatusPanel) {

setBackground(Color.WHITE);

setSize(600,600);

setLocation(3,10);

MousewhenMove mouseDragAndDrop=new MousewhenMove();

Mousehere mouseHereEvent=new Mousehere();

addMouseMotionListener(mouseDragAndDrop);

addMouseListener(mouseHereEvent);

myTool=myToolPanel;

myStatus=myStatusPanel;

setLayout(null);

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2 = (Graphics2D)g;

int iWidth = 600;

int iHeight = 600;

// Drawing the board

for (int i=0; i<8; i=i+2) {

for (int j=0; j<8; j=j+2) {

g2.setColor(Color.BLUE);

rec=new Rectangle2D.Double(j\*iWidth/8,(1+i)\*iWidth/8,Divide,Divide);

g2.fill(rec);

rec=new Rectangle2D.Double((1+j)\*iWidth/8,i\*iWidth/8,Divide,Divide);

g2.fill(rec);

}

}

/// Puting the pieces

Point postionPoint;

int postX;

int postY;

Image img;

for (int i = 1; i <= 32; i++) {

if(i<17) {

if(i==P2.GetInhand()) {

postionPoint=P2.getPixelPoint(i);

} else {

postionPoint=P2.returnPostion(i); }

img=P2.returnIconImage(i);

}

else {

if(i==P1.GetInhand()) {

postionPoint=P1.getPixelPoint(i);

} else {

postionPoint=P1.returnPostion(i); }

img=P1.returnIconImage(i);

}

if(i==P1.GetInhand())

g2.drawImage(img,postionPoint.x-25,postionPoint.y-25,Divide-40,Divide-12 ,this);

else if(i==P2.GetInhand())

g2.drawImage(img,postionPoint.x-25,postionPoint.y-25,Divide-40,Divide-12 ,this);

else {

postX=rowToX(postionPoint.x);

postY=colToY(postionPoint.y);

g2.drawImage(img,postX+20,postY+4,Divide-40,Divide-12 ,this);

}

}

}

/// You can inherit from Adapter and avoid meaningless

private class Mousehere implements MouseListener {

public void mouseClicked(MouseEvent e) {

}

public void mousePressed(MouseEvent e) {

}

/\*

--<<" In the name of \*\*GOD\*\* ">>--

\*

Hena el engain bat3 7arakaat kollaha. it is long bas ma3lsh . hwa unf3 ut3adel.

Ana henah bakshef al3 "Solider eaten move" and "Check King"

and "Other pices and Illagel and eaten moves"

\*

\*I Hope U under stand it ("Remember i didn't make this in one hour")

\*<<<<!!!!So don't expect you understand it in one Minute!!!!>>>>>

\*

\*

\* If Can improve it (It would be a Good Thing)

\*

\*/

public void mouseReleased(MouseEvent e) {

boolean can\_Send=false;

if(!GameOver) {

Point newP;

Point samePostion;

if(P1.GetInhand()!=-1) {

newP=P1.getPixelPoint(P1.GetInhand());

newP.x/=Divide;

newP.y/=Divide;

newP.x++;

newP.y++;

int otherindex;

Point old=P1.returnOldPostion(P1.GetInhand());

int x=old.x;

int y=old.y;

Point present=P1.returnPostion(P1.GetInhand());

///////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////

if(Iam\_Server||local) {

// set the seen of the solider -white

if(P1.GetInhand()<33&&P1.GetInhand()>24) {

for(int i=1;i<17;i++) {

samePostion=P2.returnPostion(i);

if(samePostion.x==newP.x&&samePostion.y==newP.y) {

if(P1.setSeentoSiliders(P1.GetInhand(),samePostion))

break;

}

}

}

///////////////////////////////////////////////////////////////////////////////////

if(!(newP.x== present.x&&newP.y== present.y)/\*&&!P1.returncheckKing()\*/)

if(P1.checkthemove(newP,P1.GetInhand() )) // if the move is illegal

{

boolean flag=false;

for(int i=1;i<=32;i++) {

if(P1.GetInhand()!=i)// check if there is peices in the WAY

{

if(i<17)

flag=P1.checktheWay(newP,P2.returnPostion(i),P1.GetInhand());//Means there is somting in the Way so can't move

else {

flag=P1.checktheWay(newP,P1.returnPostion(i),P1.GetInhand());

}

if(flag==true)break;//Means there is a Pice in the Way

}

//

}

if(!flag&&P1.Pice\_already\_there(newP))

//(if flag =false this means "The pice able to MOVE as logic""

{

// So We Check If the New Place Make a Check To Black King !!!

boolean kin2=true;

Point myold=new Point();

Point o=P1.returnPostion(P1.GetInhand());

myold.x=o.x;

myold.y=o.y;

Point other=new Point();

Point f=new Point();

boolean kill=false;

int killed=-1;

boolean end\_move=true;

////\*\*\* Start Here to Check the King

for(int k=1;k<17;k++) {

// I have to Check the Place

other=P2.returnPostion(k);

if(newP.x==other.x&&newP.y==other.y) {

int inHand=P1.GetInhand();

if(inHand>24&&P1.returnsoliderSeen(inHand)) {

kill=true;

f.x=other.x;

f.y=other.y;

P2.Killedpiec(k);

} else if(inHand<=24) {

kill=true;

f.x=other.x;

f.y=other.y;

P2.Killedpiec(k);

} else {

P1.changePostion(myold,inHand);

end\_move=false;

break;

}

killed=k;//!!!

break;

}

}

if(end\_move)

P1.changePostion(newP,P1.GetInhand());// Here is the mOve ended

P1.checkKing(false);

if(P1.see\_king\_Check(P2))

// if my king will be in check if i move

//so i can't move and i will return back to old postion'

{

P1.changePostion(myold,P1.GetInhand());

P1.checkKing(true);

end\_move=false;

}

if(kill&&P1.returncheckKing()) {

P2.changePostion(f,killed);

}

if(!P1.returncheckKing()) {

if(P2.see\_king\_Check(P1))

// if my king will be in check if i move

//so i can't move and i will return back to old postion'

{

P2.checkKing(true);

end\_move=false;

if(P2.Check\_Mate\_GameOver(P1)) {

GameOver();

Box=Integer.toString(P2.GetInhand())+Integer.toString(newP.x)+Integer.toString(newP.y);

can\_Send=true;

}

else {

Box=Integer.toString(P1.GetInhand())+Integer.toString(newP.x)+Integer.toString(newP.y);

CheckStatus();

can\_Send=true;

}

}

if(end\_move) {

Box=Integer.toString(P1.GetInhand())+Integer.toString(newP.x)+Integer.toString(newP.y);

ChangeTurn();

can\_Send=true;

}

}

}

}

P1.SetInhand(-1);

repaint();

if(can\_Send&&((Iam\_Server||Iam\_Client))) {

Send\_move();

//Send\_to.resume();

// Recv\_from.resume();

}

if(GameOver)

JOptionPane.showConfirmDialog(null,"Check Mate\n White won the game","Game Over",JOptionPane.PLAIN\_MESSAGE);

}

}

///////////////////////////////Black/////////////////////////////////////////

//////////////////////////////Black///////////////////////////////////////////

//////////////////////////////Black//////////////////////////////////////////////

//////////////////////////////Black//////////////////////////////////////////////

else if(P2.GetInhand()!=-1)//white

{

if(Iam\_Client||local) {

newP=P2.getPixelPoint(P2.GetInhand());

newP.x/=Divide;

newP.y/=Divide;

newP.x++;

newP.y++;

boolean Kingch=false;

Point old=P2.returnOldPostion(P2.GetInhand());

Point present=P2.returnPostion(P2.GetInhand());

// set the seen of the solider -black

// set the seen of the solider -black

// set the seen of the solider -black

if(P2.GetInhand()<17&&P2.GetInhand()>8) {

for(int i=17;i<33;i++) {

samePostion=P1.returnPostion(i);

if(samePostion.x==newP.x&&samePostion.y==newP.y) {

if(P2.setSeentoSiliders(P2.GetInhand(),samePostion)) {

break;

}

}

}

}

if(!(newP.x== present.x&&newP.y== present.y)/\*&&!P2.returncheckKing()\*/)

if(P2.checkthemove(newP,P2.GetInhand())) {

boolean flag=false;

for(int i=1;i<=32;i++) {

if(P2.GetInhand()!=i) {

if(i<17)

flag=P2.checktheWay(newP,P2.returnPostion(i),P2.GetInhand());

else

flag=P2.checktheWay(newP,P1.returnPostion(i),P2.GetInhand());

if(flag)break;

}

}

for(int i=1;i<=16&&!flag;i++) {

if(P2.GetInhand()!=i) {

if(flag==false) {

samePostion=P2.returnPostion(i);

if(newP.x==samePostion.x&&newP.y==samePostion.y) {

flag =true;

break;

}

}

}

if(flag)break;

}

if(!flag) {

Point kingPostion2=P2.returnPostion(8);

Point myold=new Point();

Point o=P2.returnPostion(P2.GetInhand());

myold.x=o.x;

myold.y=o.y;

Point other=new Point();

Point f=new Point();

boolean kill=false;

boolean end\_move=true;

int killed=-1;

for(int k=17;k<33;k++) {

other=P1.returnPostion(k);

if(newP.x==other.x&&newP.y==other.y) {

int inHand=P2.GetInhand();

if(inHand>8&&P2.returnsoliderSeen(inHand)) {

kill=true;

other=P1.returnPostion(k);

f.x=other.x;

f.y=other.y;

P1.Killedpiec(k);

} else if(inHand<=8) {

kill=true;

other=P1.returnPostion(k);

f.x=other.x;

f.y=other.y;

P1.Killedpiec(k);

} else {

end\_move=false;

P2.changePostion(myold,inHand);

}

killed=k;

break;

}

}

//boolean kin2=true;

if(end\_move)

P2.changePostion(newP,P2.GetInhand());

P2.checkKing(false);

if(P2.see\_king\_Check(P1))

// if my king will be in check if i move

//so i can't move and i will return back to old postion'

{

P2.changePostion(myold,P2.GetInhand());

P2.checkKing(true);

end\_move=false;

}

if(kill&&P2.returncheckKing()) {

P1.changePostion(f,killed);

}

if(P2.returncheckKing()) {

P2.changePostion(myold,P2.GetInhand());

}

if(!P2.returncheckKing()) {

if(P1.see\_king\_Check(P2))

// if my king will be in check if i move

//so i can't move and i will return back to old postion'

{

P1.checkKing(true);

end\_move=false;

if(P1.Check\_Mate\_GameOver(P2)) {

Box=Integer.toString(P2.GetInhand())+Integer.toString(newP.x)+Integer.toString(newP.y);

GameOver();

can\_Send=true;

}

else {

Box=Integer.toString(P2.GetInhand())+Integer.toString(newP.x)+Integer.toString(newP.y);

CheckStatus();

can\_Send=true;

}

}

if(end\_move) {

Box=Integer.toString(P2.GetInhand())+Integer.toString(newP.x)+Integer.toString(newP.y);

ChangeTurn();

can\_Send=true;

}

}

}

}

P2.SetInhand(-1);

repaint();

if(can\_Send&&((Iam\_Server||Iam\_Client))) {

//Send\_to.resume();

Send\_move();

/// Recv\_from.resume();

}

if(GameOver)

JOptionPane.showConfirmDialog(null,"Check Mate\n Black won the game","Game Over",JOptionPane.DEFAULT\_OPTION);

}

}

}

}

public void mouseEntered(MouseEvent e) {

}

public void mouseExited(MouseEvent e) {

}

}

////////\*---------------Mohamed Sami ------------------\*//////////////////

public boolean BoardgetPostion(int x, int y)

{

if(!GameOver&&Game\_started) {

if((Iam\_Server&&players\_turn==1)||(local)||(Iam\_Client&&players\_turn==2)) {

int newX=x/Divide;

int newY=y/Divide;

newX++;

newY++;

if(newX>8||newY>8||newX<1||newY<1) {

repaint();

return false;

}

if(players\_turn==1&&P1.GetInhand()==-1)//Player 1

{

for(int i=17;i<=32;i++) {

Point p=P1.returnPostion(i);

if(p.x==newX&&p.y==newY) {

P1.SetInhand(i); whenHandleAndPice(x,y);return true;}

}

} else if(players\_turn==2&&P2.GetInhand()==-1)//Player 2

{

for(int i=1;i<=16;i++) {

Point p=P2.returnPostion(i);

if(p.x==newX&&p.y==newY) {

P2.SetInhand(i); whenHandleAndPice(x,y);return true;}

}

}

else if(players\_turn==1&&P1.GetInhand()!=-1)//Player 1

{

whenHandleAndPice(x,y);

return true;

} else if(players\_turn==2&&P2.GetInhand()!=-1)//Player 2

{

whenHandleAndPice(x,y);

return true;

}

P1.SetInhand(-1);

move=0;

return false;

}

}

return false;

}

public boolean whenHandleAndPice(int x,int y) {

if(players\_turn==1&&P1.GetInhand()!=-1) {

P1.changePixel(x,y,P1.GetInhand());return true;

} else if(players\_turn==2&&P2.GetInhand()!=-1) {

P2.changePixel(x,y,P2.GetInhand());return true;

}

return false;

}

private int rowToX(int r) {

int myx;

int iHeight = this.getHeight();

myx=(r\*iHeight/8) - Divide;

return myx;

}

private int colToY(int c) {

int myy;

int iWidth = getWidth();

myy = (c\*iWidth/8) - Divide;

return myy;

}

private class MousewhenMove implements MouseMotionListener {

public void mouseDragged(MouseEvent e) {

int x = e.getX();

int y = e.getY();

if(controll\_game\_type(x,y)) {

repaint();

}

}

public void mouseMoved(MouseEvent e) {

}

}

public boolean controll\_game\_type(int x,int y) {

if(Iam\_Server==true||Iam\_Client==true&&Game\_started) {

if(Iam\_Server&&players\_turn==1) {

return BoardgetPostion(x,y);

} else if(Iam\_Client&&players\_turn==2) {

return BoardgetPostion(x,y);

} else

return false;

} else {

return BoardgetPostion(x,y);

}

// return false;

}

private void ChangeTurn() {

if(players\_turn==1) {

players\_turn=2;

myTool.add\_to\_History("White : "+P1.Tell\_me\_About\_last\_move());

myStatus.changeStatus(" Black player turn");

myTool.change\_to\_Timer2();

}

else if(players\_turn==2) {

players\_turn=1;

myTool.add\_to\_History("Black : "+P2.Tell\_me\_About\_last\_move());

myTool.change\_to\_Timer1();

myStatus.changeStatus(" White player turn");

}

}

private void NetChangeTurn() {

if(players\_turn==2) {

myTool.add\_to\_History("White : "+P1.Tell\_me\_About\_last\_move());

myStatus.changeStatus(" Black player turn");

myTool.change\_to\_Timer2();

}

else if(players\_turn==1) {

myTool.add\_to\_History("Black : "+P2.Tell\_me\_About\_last\_move());

myTool.change\_to\_Timer1();

myStatus.changeStatus(" White player turn");

}

}

private void NeTGameCheckStatus() {

if(players\_turn==1) {

myTool.add\_to\_History("White : "+P1.Tell\_me\_About\_last\_move());

myTool.change\_to\_Timer2();

} else if(players\_turn==2) {

myTool.add\_to\_History("Black : "+P2.Tell\_me\_About\_last\_move());

myTool.change\_to\_Timer1();

}

myStatus.changeStatus(" Check! ");

}

private void CheckStatus() {

if(players\_turn==1) {

players\_turn=2;

myTool.add\_to\_History("White : "+P1.Tell\_me\_About\_last\_move());

myTool.change\_to\_Timer2();

} else if(players\_turn==2) {

players\_turn=1;

myTool.add\_to\_History("Black : "+P2.Tell\_me\_About\_last\_move());

myTool.change\_to\_Timer1();

}

myStatus.changeStatus(" Check! ");

}

private void GameOver() {

myStatus.changeStatus(" Check Mate! ");

GameOver=true;

}

public void Send\_move() {

out.print(Box);

out.print("\r\n");

out.flush();

}

class Recv\_Thread extends Thread {

public synchronized void run () {

while(true) {

try {

Box=in.readLine();

} catch (IOException ex) {

ex.printStackTrace();

System.out.println("ERROR");

}

if(Box!=null) {

int newInHand=Integer.parseInt(Box);

int newX=Integer.parseInt(Box);

int newY=Integer.parseInt(Box);

/\*\*\*

\* Operation to Get

\*1- The # of Pice

\*2- The Location X

\*3- The Location Y

\*

\*\*/

newInHand/=100;

newX-=(newInHand\*100);

newX/=10;

newY-=(newInHand\*100)+(newX\*10);

if(players\_turn==1) {

P1.SetInhand(newInHand);

players\_turn=2;

P1.changePostion(new Point(newX,newY),newInHand);

P2.Killedpiec(P1.Get\_Pice\_already\_there\_from\_enemy(new Point(newX,newY),P2));

P2.checkKing(false);

if(P2.see\_king\_Check(P1))

// if my king will be in check if i move

//so i can't move and i will return back to old postion'

{

P2.checkKing(true);

if(P2.Check\_Mate\_GameOver(P1)) {

System.out.println("mate");

GameOver();

}

else {

NeTGameCheckStatus();

}

} else NetChangeTurn();

P1.SetInhand(-1);

} else {

P2.SetInhand(newInHand);

P2.changePostion(new Point(newX,newY),newInHand);

P1.Killedpiec(P2.Get\_Pice\_already\_there\_from\_enemy(new Point(newX,newY),P1));

players\_turn=1;

P1.checkKing(false);

if(P1.see\_king\_Check(P2))

// if my king will be in check if i move

//so i can't move and i will return back to old postion'

{

P1.checkKing(true);

if(P1.Check\_Mate\_GameOver(P2)) {

System.out.println("mate");

GameOver();

}

else {

NeTGameCheckStatus();

}

} else NetChangeTurn();

P2.SetInhand(-1);

}

// CheckStatus();

repaint();

}

}

}

}

}