Project 4 NQueen Problem

Team members:

Ramzy ashraf ramzy 20210333

Ahmed abdelrashed 20210071

Abdallah Mahmoud 20210558

Abdallah ali Mohamed 20210552

Abdelrahman anwar 20210494

Gamal elden ayman 20210250

1)Project description

The N Queen is the problem of placing N chess queens on an $N \times N$ chessboard so that no two queens attack
each other by being in the same row, column or diagonal.

We use multi threading to solve this

2)Team member role

Ramzy ashraf and ahmed abdelrashed made gui runtime and threads functions and link project

Files (Nqueenproject and Threads)

Gamal elden and Abdallah Mahmoud and Abdallah ali and abdelrhman anwar made and help in algorithm codes to solve n queen using multithreads

And the algorithm is used is backtracking algorithm

Files (Statutory and Algorithm)

3)what we actually did

We made nqueen solving problem as we have n queens in n*n grid the project can solve any n queen

By clicking on buttons in gui to solve problem and to change the nqueen and grid and we use coding in gui not drag and drop and use backtracking algorithm and multithreading to solve this type of problem

4) code

Nqueenproject.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Nqueenproject extends JPanel {
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    static int activeQueenBoard[][]=new int[Algorithm.n][Algorithm.n];
    static int currentHKey=-1;
    static Thread SearchThread;
    static boolean firstTime=true;
    static int border=20;
    static double squareSize;
    static JFrame javaF=new JFrame("NQueens by ramzy ashraf");
    static Nqueenproject javaUI=new Nqueenproject();
    public static void main(String[] args) {
        javaF.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        javaF.add(javaUI);
        javaF.setSize(750, 512);
        javaF.setLocation((Toolkit.getDefaultToolkit().getScreenSize().wid
th-javaF.getWidth())/2,
                (Toolkit.getDefaultToolkit().getScreenSize().height-
javaF.getHeight())/2);
        javaF.setVisible(true);
        squareSize=(double)(Math.min(javaUI.getHeight(),
javaUI.getWidth()-250-border)-2*border)/Algorithm.n;
        Thread TimerThread=new Thread(new Threads());
        TimerThread.start();
    }
    @Override
    public void paintComponent(Graphics g) {
        if (firstTime) {firstTime=false; computerThink();}
        super.paintComponent(g);
        this.setBackground(new Color(255, 255, 255));
        KeyboardFocusManager.getCurrentKeyboardFocusManager()
                .addKeyEventDispatcher(new KeyEventDispatcher() {
                    @Override
                    public boolean dispatchKeyEvent(KeyEvent e) {
                        if (e.getID()==KeyEvent.KEY_PRESSED) {
                            if (!Algorithm.HMapSolutions.isEmpty()) {
                                if (e.getKeyCode()==KeyEvent.VK_DOWN) {
                                    int
length=Algorithm.HMapSolutions.size();
                                    if (length-1==currentHKey) {
                                        currentHKey=0;
                                        getSolution();
                                    } else {
                                        currentHKey++;
                                        getSolution();
                                    }
                                }
                                if (e.getKeyCode()==KeyEvent.VK_UP) {
                                    int
length=Algorithm.HMapSolutions.size();
                                    if (currentHKey==0) {
                                        currentHKey=length-1;
                                        getSolution();
                                    } else {
                                        currentHKey--;
                                        getSolution();
                                    }
                                }
                            }
                            if (e.getKeyCode()==KeyEvent.VK_RIGHT) {
                                Algorithm.n++;
                                newDimension();
                            }
                            if (e.getKeyCode()==KeyEvent.VK_LEFT) {
                                if (Algorithm.n>4) {Algorithm.n--;
newDimension();}
                            }
                        }
                        return true;
                    }
                });
        this.addComponentListener(new ComponentAdapter() {
            @Override
            public void componentResized(ComponentEvent e) {
                squareSize=(double)(Math.min(getHeight(), getWidth()-250-
border)-2*border)/Algorithm.n;
            }
        });
        if (Algorithm.n%2==0) {
            for (int i=0;i<Algorithm.n*Algorithm.n;i+=2) {
                g.setColor(new Color(0, 0, 0));
                g.fillRect((int)((i%Algorithm.n+(i/Algorithm.n)%2)*squareS
ize)+border, (int)((i/Algorithm.n)*squareSize)+border, (int)squareSize,
(int)squareSize);
                g.setColor(new Color(255, 255, 255));
                g.fillRect((int)(((i+1)%Algorithm.n-
((i+1)/Algorithm.n)%2)*squareSize)+border,
(int)(((i+1)/Algorithm.n)*squareSize)+border, (int)squareSize,
(int)squareSize);
            }
        } else {
            for (int i=0;i<Algorithm.n*Algorithm.n-1;i+=2) {
                g.setColor(new Color(0, 0, 0));
                g.fillRect((int)((i%Algorithm.n)*squareSize)+border,
(int)((i/Algorithm.n)*squareSize)+border, (int)squareSize,
(int)squareSize);
                g.setColor(new Color(255, 255, 255));
                g.fillRect((int)(((i+1)%Algorithm.n)*squareSize)+border,
(int)(((i+1)/Algorithm.n)*squareSize)+border, (int)squareSize,
(int)squareSize);
            }
            int i=Algorithm.n*Algorithm.n-1;
            g.setColor(new Color(0, 0, 0));
            g.fillRect((int)((i%Algorithm.n)*squareSize)+border,
(int)((i/Algorithm.n)*squareSize)+border, (int)squareSize,
(int)squareSize);
        }
        g.setColor(new Color(0, 0, 0));
        g.fill3DRect(0, border, border, (int)(Algorithm.n*squareSize),
true);
        g.fill3DRect((int)(Algorithm.n*squareSize)+border, border, border,
(int)(Algorithm.n*squareSize), true);
        g.fill3DRect(border, 0, (int)(Algorithm.n*squareSize), border,
true);
        g.fill3DRect(border, (int)(Algorithm.n*squareSize)+border,
(int)(Algorithm.n*squareSize), border, true);
        g.setColor(Color.WHITE);
        g.fill3DRect(0, 0, border, border, true);
        g.fill3DRect((int)(Algorithm.n*squareSize)+border, 0, border,
border, true);
        g.fill3DRect(0, (int)(Algorithm.n*squareSize)+border, border,
border, true);
        g.fill3DRect((int)(Algorithm.n*squareSize)+border,
(int)(Algorithm.n*squareSize)+border, border, border, true);

        Image chessPieceImage;
```

```java
        chessPieceImage=new
ImageIcon(System.getProperty("user.dir")+"\\ChessPieces.png").getImage();
        for (int i=0;i<Algorithm.n*Algorithm.n;i++) {
            int j=-1,k=-1;
            if (activeQueenBoard[i/Algorithm.n][i%Algorithm.n]==1)
{j=1;k=0;}
            if (j!=-1 && k!=-1) {
                g.drawImage(chessPieceImage,
(int)((i%Algorithm.n)*squareSize)+border,
(int)((i/Algorithm.n)*squareSize)+border,
(int)((i%Algorithm.n+1)*squareSize)+border,
(int)((i/Algorithm.n+1)*squareSize)+border, j*64, k*64, (j+1)*64,
(k+1)*64, this);
            }
        }
        g.setColor(Color.BLACK);
        Font fontDepth=new Font("plIN", Font.PLAIN, 20);
        g.setFont(fontDepth);
        int x=(int)(Algorithm.n*squareSize)+2*border+10;
        int y=border+10;
        String intType;
        switch (currentHKey+1) {
            case 1: intType="st";
                break;
            case 2: intType="nd";
                break;
            case 3: intType="rd";
                break;
            default: intType="th";
                break;
        }
        g.drawString("CASES",x+80,y);
        g.drawString("1) Grid size:
"+Algorithm.n+"x"+Algorithm.n,x,y+2*g.getFont().getSize());
        if (Algorithm.doneSearch) {
            g.drawString("2) There are "+Algorithm.progress+"
solutions.",x,y+3*g.getFont().getSize());
            g.drawString("3) That took "+((Algorithm.endTime-
Algorithm.startTime)/1000)+" seconds.",x,y+5*g.getFont().getSize());
        } else {
            g.drawString("2) Currently, "+Algorithm.progress+"
solutions",x,y+3*g.getFont().getSize());
            g.drawString("have been found.",x,y+4*g.getFont().getSize());
            g.drawString("3) "+((System.currentTimeMillis()-
Algorithm.startTime)/1000)+" seconds",x,y+5*g.getFont().getSize());
```

```
            g.drawString("have elapsed so 
far.",x,y+6*g.getFont().getSize());
        }
        g.drawString("4) You are currently looking 
at",x,y+7*g.getFont().getSize());
        g.drawString("the "+(currentHKey+1)+intType+" 
solution.",x,y+8*g.getFont().getSize());

        g.setColor(Color.black);
        g.drawString("UP/DOWN arrow keys",x,y+10*g.getFont().getSize());
        g.drawString("navigate through 
solutions",x,y+11*g.getFont().getSize());
        g.drawString("RIGHT/LEFT arrow 
keys",x,y+12*g.getFont().getSize());
        g.drawString("adjust grid size",x,y+13*g.getFont().getSize());
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jButton5 = new javax.swing.JButton();

        jButton2.setText("UP");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }

            private void jButton2ActionPerformed(ActionEvent evt) {
                int length=Algorithm.HMapSolutions.size();
                                if (currentHKey==0) {
                                    currentHKey=length-1;
                                    getSolution();
                                } else {
                                    currentHKey--;
                                    getSolution();
                                }
            }
        });

        jButton3.setText("RIGHT");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }

            private void jButton3ActionPerformed(ActionEvent evt) {
                Algorithm.n++;
                newDimension();
            }
        });

        jButton4.setText("LEFT");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }

            private void jButton4ActionPerformed(ActionEvent evt) {
                 if (Algorithm.n>4) {Algorithm.n--; newDimension();}
            }
        });

        jButton5.setText("DOWN");
        jButton5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }

            private void jButton5ActionPerformed(ActionEvent evt) {
                int length=Algorithm.HMapSolutions.size();
                                    if (length-1==currentHKey) {
                                        currentHKey=0;
                                        getSolution();
                                    } else {
                                        currentHKey++;
                                        getSolution();
                                    }
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(504, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.LEADING)
```

```
                    .addComponent(jButton4,
javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jButton2,
javax.swing.GroupLayout.Alignment.TRAILING))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacemen
t.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.LEADING)
                    .addComponent(jButton3)
                    .addComponent(jButton5))
                .addGap(46, 46, 46))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(242, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.BASELINE)
                    .addComponent(jButton5)
                    .addComponent(jButton2))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacemen
t.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.BASELINE)
                    .addComponent(jButton3)
                    .addComponent(jButton4))
                .addContainerGap())
        );
    }
    public static void computerThink() {
        activeQueenBoard=new int[Algorithm.n][Algorithm.n];
        SearchThread=new Thread(new Algorithm());
        SearchThread.start();
        javaF.repaint();
    }
    public static void newDimension() {
        computerThink();
        squareSize=(double)(Math.min(javaUI.getHeight(),
javaUI.getWidth()-250-border)-2*border)/Algorithm.n;
    }
    public static void getSolution() {
        activeQueenBoard=(int[][])Algorithm.HMapSolutions.get(currentHKey)
;
```

```java
        javaF.repaint();
    }
}
```

Threads.java

```java
public class Threads implements Runnable{
    @Override
    public void run() {
        while(true)
        {
            Nqueenproject.javaF.repaint();
            try{
                Thread.sleep(1000);
            } catch (Exception e) {}
        }
    }
}
```

Statutory.java

```java
public class Statutory {
    public static boolean posibleQ(int i, int[][] queenBoard) {
        int r=i/Algorithm.n, c=i%Algorithm.n;
        int temp=1;
        try {
            while (0==queenBoard[r-temp][c-temp])
            {temp++;}
            if (1==queenBoard[r-temp][c-temp]) {
                return false;
            }
        } catch (Exception e) {}
        temp=1;
        try {
            while (0==queenBoard[r-temp][c])
            {temp++;}
            if (1==queenBoard[r-temp][c]) {
                return false;
            }
        } catch (Exception e) {}
        temp=1;
        try {//up, right
            while (0==queenBoard[r-temp][c+temp])
            {temp++;}
            if (1==queenBoard[r-temp][c+temp]) {
                return false;
            }
        } catch (Exception e) {}
        return true;
    }
}
```

Algorithm.java

```java
import java.awt.*;
import java.util.*;
public class Algorithm implements Runnable{
    static HashMap HMapSolutions=new HashMap();
    static HashMap HMapUnique=new HashMap();
    static int progress;
    static long startTime, endTime;
    static boolean doneSearch;
    static int totalNodes;
    static int solutions;
    static int n=4;
    public static void masterController() {
        doneSearch=false;
        totalNodes=0;
        HMapSolutions.clear();
        Nqueenproject.currentHKey=-1;
        solutions=0;
        progress=0;
        guess(0,0,new int[n][n]);
        doneSearch=true;
    }
    public static void guess(int x, int y, int[][] queenBoard) {
        while (y<n)
        {
            queenBoard[x][y]=1;
            totalNodes++;
            if (Statutory.posibleQ(x*n+y, queenBoard)) {
                if (x+1==n) {
                    int queenBoard2[][]=new int[n][n];
                    for (int i=0;i<n;i++) {
                        System.arraycopy(queenBoard[i],0,queenBoard2[i],0,
n);
                    }
                    HMapSolutions.put(progress, queenBoard2);
                    queenBoard[x][y]=0;
                    progress++;
                    if (Nqueenproject.currentHKey==-1) {
                        Nqueenproject.currentHKey=0;
                        Nqueenproject.getSolution();
                    }
                    return;
                }
                guess(x+1, 0, queenBoard);
            }
            queenBoard[x][y]=0;
            y++;
        }
    }
    public static void unique() {
        HMapUnique.clear();
        int counter=0;
        int originalBoard[][];
        int rotateBoard1[][]=new int[n][n];
        int rotateBoard2[][]=new int[n][n];
        int rotateBoard3[][]=new int[n][n];
        for (int i=0;i<HMapSolutions.size();i++) {
            originalBoard=(int[][])HMapSolutions.get(i);
            for(int x=0;x<n;x++){
                for(int y=0;y<n;y++){
                    rotateBoard1[x][y]=originalBoard[n-1-y][x];
                }
            }
            for(int x=0;x<n;x++){
                for(int y=0;y<n;y++){
                    rotateBoard2[x][y]=rotateBoard1[n-1-y][x];
                }
            }
            for(int x=0;x<n;x++){
                for(int y=0;y<n;y++){
                    rotateBoard3[x][y]=rotateBoard2[n-1-y][x];
                }
            }
            int j=0;
            boolean uniqueBoard=true;
            while (j<HMapUnique.size() && uniqueBoard)
            {
                if (Arrays.deepEquals((int[][])HMapUnique.get(j),
rotateBoard1) ||
                        Arrays.deepEquals((int[][])HMapUnique.get(j),
rotateBoard2) ||
                        Arrays.deepEquals((int[][])HMapUnique.get(j),
rotateBoard3)) {
                    uniqueBoard=false;
                }
                j++;
            }
            if (uniqueBoard) {
                int originalBoard2[][]=new int[n][n];
                for (int k=0;k<n;k++) {
                    System.arraycopy(originalBoard[k],0,originalBoard2[k],
0,n);
                }
                HMapUnique.put(counter, originalBoard2);
                counter++;
            }
        }
        System.out.println(HMapUnique.size());
    }
    public static void drawToArray(int[][] queenBoard) {
        for (int i=0;i<n;i++) {
            System.out.println(Arrays.toString(queenBoard[i]));
        }
        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");
    }
    @Override
    public void run() {
        Nqueenproject.javaF.setCursor(new Cursor(Cursor.WAIT_CURSOR));
        startTime=System.currentTimeMillis();
        masterController();
        endTime=System.currentTimeMillis();
        Toolkit.getDefaultToolkit().beep();
        Nqueenproject.javaF.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
        Nqueenproject.javaF.repaint();
    }
}
```