

**pat project**

Technical document



Bryce grahn 12 key 1

Table of Contents

[References: 2](#_Toc519944947)

[Critical algorithms/FlowCharts: 3](#_Toc519944948)

[Sophisticated techniques: 6](#_Toc519944949)

[Hard copy of the program source code: 8](#_Toc519944950)

# References:

Concepts:

* **Beta snake game :** Jan Bodnar. 2018. Java Snake. [ONLINE] Available at: http://zetcode.com/tutorials/javagamestutorial/snake/. [Accessed 15 July 2018].
* **Database linking:** Mathew Broderick and Brandon Goldhawk
* **Resize jFrame:** Mathew Broderick
* **Linking JFrames:** Mathew Broderick
* **jTable:** Mathew Broderick
* **countDownTimer:** Nir Graydi. 2016. JAVA Countdown Timer + Source code!! Simple example (NetBeans 8.1). [ONLINE] Available at: https://www.youtube.com/watch?v=Euexl32lB8w. [Accessed 15 July 2018].

Full Use:

* **Blank cursor:** Jessy. 2017. How to hide cursor in a Swing application?. [ONLINE] Available at: https://stackoverflow.com/questions/1984071/how-to-hide-cursor-in-a-swing-application. [Accessed 15 July 2018].
* **Apple image:** Bortania, (2009), Cute Bright Red Apple Icon Isolated on White [ONLINE]. Available at: https://www.istockphoto.com/za/vector/cute-bright-red-apple-icon-isolated-on-white-gm90692172-617189 [Accessed 15 July 2018].
* **Double points head:** Tynker, (2016), gold block 2.0 [ONLINE]. Available at: https://www.google.co.za/search?q=minecraft+golden+block&rlz=1C1CAFA\_enZA786ZA790&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjV6tOvq5XcAhVGYlAKHUSXCgcQ\_AUICigB&biw=1280&bih=615#imgrc=IMwnHrV1NGSHwM: [Accessed 15 July 2018].
* **Double points body:** SpoonFlower, (2016), 8 bit gold ore [ONLINE]. Available at: https://www.google.co.za/search?rlz=1C1CAFA\_enZA786ZA790&biw=1280&bih=615&tbm=isch&sa=1&ei=ixJFW4S3AoKEgAbvyJPAAw&q=minecraft+golden+ore&oq=minecraft+golden+ore&gs\_l=img.3..0i10k1l2j0i30k1.62597.64931.0.66344.8.8.0.0.0.0.217.840.2-4.4.0....0...1c.1.64.img..4.4.838...0j0i8i30k1j0i24k1.0.jO1vcJWy6mc#imgdii=JQKB\_tXOiN747M:&imgrc=CA1iAWWU5NvqQM: [Accessed 15 July 2018].
* **Invincibility head:** nova skin, (2014), Block Diamond [ONLINE]. Available at: https://www.google.co.za/search?rlz=1C1CAFA\_enZA786ZA790&biw=1280&bih=615&tbm=isch&sa=1&ei=ixJFW4S3AoKEgAbvyJPAAw&q=minecraft+diamond+block&oq=minecraft+diamond+block&gs\_l=img.3..0l6j0i7i30k1j0i30k1l3.7245.12520.0.13999.13.13.0.0.0.0.280.2000.2-9.9.0....0...1c.1.64.img..5.8.1719....0.kKZxRscMi3A#imgdii=Z3\_P7hfA6OFWsM:&imgrc=NtHIMUAb5nJZiM: [Accessed 15 July 2018].
* **Invincibility body:** Roblox, (2017), minecraft diamond ore [ONLINE]. Available at: https://www.google.co.za/search?rlz=1C1CAFA\_enZA786ZA790&biw=1280&bih=615&tbm=isch&sa=1&ei=DRNFW7KGGJLvgAaxuoOIDg&q=minecraft+diamond+ore&oq=minecraft+diamond+ore&gs\_l=img.3..0l8j0i30k1l2.146960.149517.0.150451.8.6.0.2.2.0.305.599.2-1j1.2.0....0...1c.1.64.img..4.4.603...0i67k1.0.i4Glc0T9jyw#imgrc=LQwgVr1iFYiksM: [Accessed 15 July 2018].

# Critical algorithms/FlowCharts:

* **keyListener:** The key listener checks if a key was pressed and implements the KeyPressed methodwhich executes the necessary procedures according to the key pressed, such as change in direction.

When a key is pressed (

integer key 🡨 e’s keyCode

if reverse is false (

if key pressed is the left arrow key and rightDirec is not true (

leftDirec 🡨 true

upDirec 🡨false

downDirec 🡨 false

)

if key pressed is the right arrow key and leftDirec is not true

(

rightDirec 🡨 true

upDirec 🡨 false

downDirec 🡨 false

)

if key pressed is the up arrow key and downDirec is not true

(

upDirec 🡨 true

rightDirec 🡨false

leftDirec 🡨 false

)

if key pressed is the down arrow key and upDirec is not true

(

downDirec 🡨 true

rightDirec 🡨false

leftDirec 🡨false

)

)

)

* **How the snake moves:** Each snake block/unit copies the co-ordinates of the previous block starting from the black end to the front end of the snake. The snakes head will then move one unit according to the last know direction and update its co-ordinates. The graphics are refreshed and the snake appears to move one unit in a specific direction. This is looped over and over.

Integer I 🡨 number of snake units

if move is true

(

Loop for the number of body units (

X Array at position i 🡨 X Array at position (i – 1)

Y Array at position i 🡨 Y Array at position (i – 1)

I 🡨 I - 1

)

if leftDirec is true (

X Array at position 0 🡨 X Array at position 0 - unitSize

)

if rightDirec is true (

X Array at position 0 🡨 X Array at position 0 + unitSize

)

if upDirec is true (

Y Array at position 0 🡨 Y Array at position 0 - unitSize

)

if downDirec is true (

Y Array at position 0 🡨 Y Array at position 0 + unitSize

)

)

* **Check hit:** Checks if the co-ordinates of the snake’s head is equal to the co-ordinates of either itself, lilys, pyramids or beyond the edge of the board. The necessary procedures take place depending on the collision.

if invincibility is false and move is true (

integer I 🡨 number of snake units

loop for the number of snake units less 5(

if i is bigger than 4 and x Array at position 0 is equal to x array at position i and y Array at position 0 is equal to y array at position i (

if lives is equal to 1 (

inGame 🡨 false

) otherwise (

Lives 🡨 lives - 1

move 🡨 false

)

)

I 🡨 I - 1

)

If difficulty is easy //Check hit Lily or pyramid

(

Integer I 🡨 0

Loop for number of objects (

if x Array at position 0 is equal to object X Array at position i and y Array at 0 is equal to object Y Array at position i (

if lives is 1 (

inGame 🡨 false

) otherwise (

Lives 🡨 lives - 1

move 🡨 false

)

)

I 🡨 I + 1

)

)

)

if y Array at position 0 is bigger than or equal to the game Height or y Array at position 0 is smaller than 0 or x Array at position 0 is bigger than or equal to game Width or x Array at position 0 is smaller than 0(

if lives is 1 (

inGame 🡨 false

)otherwise (

reverse 🡨 true

if invincibility is false (

lives 🡨 lives - 1

)

Change Timer to 250;

)

)

if in Game is false) (

stop game timer

)

# Sophisticated techniques:

* **Overrides:** Using the @Override annotation acts as a compile-time safeguard against a common programming mistake. It will throw a compilation error if you have the annotation on a method you're not actually overriding the superclass method.

**Graphics:** With the use of the paintComponent () method and g.drawString/Image () the various graphics can be displayed on a jPanel.

EG: public void paintComponent(Graphics g) { //draws snake, score, objects etc... on the playfield

super.paintComponent(g);

draw(g, ("LIVES : " + lives), 1450, 35);

draw(g, ("SCORE : " + score), 10, 35);

draw(g, powerMsg, 665, 35);

if (countDown == true) //Only starts 20 second countdown for specific powers

{

drawCountDown(g);

}

initiateDrawing(g);

}

* **keyListeners:** A private class TAdapter extends KeyAdapter was created with the method Keypressed which checks if a key was pressed and which specific key it was and makes the necessary actions according to the key pressed, such as change in direction.

EG: if ((key == KeyEvent.VK\_LEFT) && (!rightDirec)) { // if left key is pressed

leftDirec = true;

upDirec = false;

downDirec = false;

}

if ((key == KeyEvent.VK\_RIGHT) && (!leftDirec)) { //if right key is pressed

rightDirec = true;

upDirec = false;

downDirec = false;

}

* **clear mouse:** Sets the cursor image to a transparent image

Eg: BufferedImage cursorImg = new BufferedImage(16, 16, BufferedImage.TYPE\_INT\_ARGB);

Cursor blankCursor = Toolkit.getDefaultToolkit().createCustomCursor(

cursorImg, new Point(0, 0), "blank cursor");

setCursor(blankCursor);

* **Adding images:** fetching images from outside the internal code such as the snake images

Eg: ImageIcon iI = new ImageIcon(image);

object= iI.getImage();

* **linking database:** First a link must be created between the interface and the database. The link (url), ResultSet method and update (change method) is created in the Database class. Whereas the specific queries are created in the UseDatabase class. And the reformatting and displaying of the information in the tables is done within the GUI.

Eg: public ResultSet query(String stmt) throws SQLException

{

statement = connection.prepareStatement(stmt);

resultSet = statement.executeQuery();

System.out.println(resultSet);

return resultSet;

}

public void UpdateTable(String name) {

int score = fetchScore();

String what = "Difficulty";

String difficulty = fetchWhat(what);

System.out.println(difficulty);

String tbl = "";

try {

database.change("INSERT INTO tblHighScore (FirstName,Score,Difficulty)" + "VALUES ('" + name + "','" + score + "','" + difficulty + "')"); // Inserts a new players username and score into a specific database table indicated

} catch (SQLException ex) {

System.out.println(ex);

}

}

* **Timers:** the timer specifies the rate at which the game runs by firing action events (action performed method) at the set intervals specified by the delay.

Eg: gameTimer = new Timer(delay, this); //starts game timer (the delay specifies the rate at which everyhting is processed and thus the speed of the snake)

gameTimer.start();

**CountDownTimer:** creates a 20 second countdown

Eg: public void startTimer() // creates a 20 second count down timer for certain power ups

{

Timer timer = new Timer();

counter = 20;

TimerTask task;

task = new TimerTask() {

public void run() {

System.out.println(counter);

counter--;

if (counter <= -1) { // Cancels the timer after 20 seconds

timer.cancel();

}

}

};

timer.scheduleAtFixedRate(task, 1000, 1000); //1000ms = 1sec

}

# Hard copy of the program source code:

**Main face:**

package grahn;

import java.io.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

public class mainFace extends javax.swing.JFrame {

private useDatabase ud = new useDatabase();

private String[] args;

private String what = "Status";

public mainFace() {

initComponents(); // Thanks to Mathew broderick

dispose(); // Resizes and centres JFrame

setUndecorated(true);

pack();

setLocationRelativeTo(null);

setVisible(true);

String stat = ""; // This whole process ensures the loading face doesn't duplicate each time you go back to the main face. sends across colomn heading

String option = "Status";

stat = ud.fetchWhat(option);

if (stat.equalsIgnoreCase("true"))

{

displayLoadingFace();

stat = "false";

ud.updateWhat(what,stat);

}

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

playBut = new javax.swing.JButton();

helpBut = new javax.swing.JButton();

difficultBut = new javax.swing.JButton();

exitBut = new javax.swing.JButton();

lblHead = new javax.swing.JLabel();

lblBack = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setBounds(new java.awt.Rectangle(0, 0, 0, 0));

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

playBut.setBackground(new java.awt.Color(255, 255, 255));

playBut.setFont(new java.awt.Font("Tahoma", 1, 40)); // NOI18N

playBut.setText("PLAY");

playBut.setBorderPainted(false);

playBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

playButActionPerformed(evt);

}

});

getContentPane().add(playBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(650, 300, 310, 130));

helpBut.setBackground(new java.awt.Color(255, 255, 255));

helpBut.setFont(new java.awt.Font("Tahoma", 1, 40)); // NOI18N

helpBut.setText("HELP");

helpBut.setBorder(null);

helpBut.setBorderPainted(false);

helpBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

helpButActionPerformed(evt);

}

});

getContentPane().add(helpBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(260, 300, 310, 130));

difficultBut.setBackground(new java.awt.Color(255, 255, 255));

difficultBut.setFont(new java.awt.Font("Tahoma", 1, 40)); // NOI18N

difficultBut.setText("DIFFICULTY");

difficultBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

difficultButActionPerformed(evt);

}

});

getContentPane().add(difficultBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(1060, 300, 310, 130));

exitBut.setBackground(new java.awt.Color(255, 255, 255));

exitBut.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N

exitBut.setText("EXIT");

exitBut.setOpaque(false);

exitBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

exitButActionPerformed(evt);

}

});

getContentPane().add(exitBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(700, 660, 220, 100));

lblHead.setBackground(new java.awt.Color(0, 102, 102));

lblHead.setFont(new java.awt.Font("Engravers MT", 1, 80)); // NOI18N

lblHead.setForeground(new java.awt.Color(255, 255, 255));

lblHead.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

lblHead.setText("EAT IT");

lblHead.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

lblHead.setOpaque(true);

getContentPane().add(lblHead, new org.netbeans.lib.awtextra.AbsoluteConstraints(540, 20, 520, 120));

lblBack.setForeground(new java.awt.Color(0, 102, 255));

lblBack.setIcon(new javax.swing.ImageIcon(getClass().getResource("/grahn/main backround1.png"))); // NOI18N

lblBack.setOpaque(true);

getContentPane().add(lblBack, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1600, 910));

setSize(new java.awt.Dimension(1616, 952));

setLocationRelativeTo(null);

}// </editor-fold>

private void playButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

SnakeMain.main(args); //Starts game

}

private void difficultButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

difficultFace dFace = new difficultFace();

dFace.setVisible(true);

}

private void displayLoadingFace()

{

loadingFace lFace = new loadingFace ();

lFace.setVisible(true);

}

private void exitButActionPerformed(java.awt.event.ActionEvent evt) {

int score = 0;

String stat = "true";

String close = "false";

int id = 1;

String difficulty = "normal";

ud.deleteData();

ud.InsertData(id, score, difficulty, stat, close); //Saves stats and additional information to the database before closing. sends across, id, score, difficulty, status, close

System.exit(0); //Ends and closes Game (thanks to matthew broderick)

}

private void helpButActionPerformed(java.awt.event.ActionEvent evt) {

try {

this.dispose();

helpFace hFace = new helpFace();

hFace.setVisible(true);

} catch (IOException ex) {

Logger.getLogger(mainFace.class.getName()).log(Level.SEVERE, null, ex); //needed for IOException error

System.out.println("Help Button did not execute");

}

}

**Difficulty face:**

package grahn;

public class difficultFace extends javax.swing.JFrame {

private useDatabase ud = new useDatabase();

private String difficulty;

private String what = "Difficulty";

public difficultFace() {

initComponents(); //Thanks to Mathew Broderick

dispose(); // Resizes and centres JFrame

setUndecorated(true);

pack();

setLocationRelativeTo(null);

setVisible(true);

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

hardBut = new javax.swing.JButton();

normBut = new javax.swing.JButton();

easyBut = new javax.swing.JButton();

backBut = new javax.swing.JButton();

lblBack = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

hardBut.setBackground(new java.awt.Color(255, 255, 255));

hardBut.setFont(new java.awt.Font("Tahoma", 1, 40)); // NOI18N

hardBut.setText("HARD");

hardBut.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, new java.awt.Color(204, 255, 51), new java.awt.Color(204, 255, 51), null, null));

hardBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

hardButActionPerformed(evt);

}

});

getContentPane().add(hardBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(1060, 670, 230, 120));

normBut.setBackground(new java.awt.Color(255, 255, 255));

normBut.setFont(new java.awt.Font("Tahoma", 1, 40)); // NOI18N

normBut.setText("NORMAL");

normBut.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, new java.awt.Color(204, 255, 51), new java.awt.Color(204, 255, 51), null, null));

normBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

normButActionPerformed(evt);

}

});

getContentPane().add(normBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(720, 670, 230, 120));

easyBut.setBackground(new java.awt.Color(255, 255, 255));

easyBut.setFont(new java.awt.Font("Tahoma", 1, 40)); // NOI18N

easyBut.setText("EASY");

easyBut.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, new java.awt.Color(255, 255, 51), new java.awt.Color(255, 255, 51), null, null));

easyBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

easyButActionPerformed(evt);

}

});

getContentPane().add(easyBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(390, 670, 230, 120));

backBut.setBackground(new java.awt.Color(255, 255, 255));

backBut.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

backBut.setText("BACK");

backBut.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, new java.awt.Color(204, 255, 51), new java.awt.Color(204, 255, 51), null, null));

backBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

backButActionPerformed(evt);

}

});

getContentPane().add(backBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 11, 130, 50));

lblBack.setIcon(new javax.swing.ImageIcon(getClass().getResource("/grahn/difficulty backround1.png"))); // NOI18N

getContentPane().add(lblBack, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1600, 910));

setSize(new java.awt.Dimension(1616, 946));

setLocationRelativeTo(null);

}// </editor-fold>

private void easyButActionPerformed(java.awt.event.ActionEvent evt) {

difficulty = "Easy";

displayDiff();

this.dispose();

ud.updateWhat(what,difficulty); //Updates difficulty status to the database so that it can be fetched for later use when playing the game. sends across difficulty and column heading

mainFace mFace = new mainFace();

mFace.setVisible(true);

}

private void backButActionPerformed(java.awt.event.ActionEvent evt) {

difficulty = "normal";

this.dispose();

mainFace mFace = new mainFace();

ud.updateWhat(what,difficulty); //Updates difficulty status to the database so that it can be fetched for later use when playing the game. sends across difficulty and column heading

mFace.setVisible(true);

}

private void hardButActionPerformed(java.awt.event.ActionEvent evt) {

difficulty = "Hard";

displayDiff();

this.dispose();

ud.updateWhat(what,difficulty); //Updates difficulty status to the database so that it can be fetched for later use when playing the game. sends across difficulty and column heading

mainFace mFace = new mainFace();

mFace.setVisible(true);

}

private void normButActionPerformed(java.awt.event.ActionEvent evt) {

difficulty = "Normal";

displayDiff();

this.dispose();

ud.updateWhat(what,difficulty); //Updates difficulty status to the database so that it can be fetched for later use when playing the game. sends across difficulty and column heading

mainFace mFace = new mainFace();

mFace.setVisible(true);

}

public String getDifficulty()

{

return difficulty; //sends difficulty across

}

private void displayDiff()

{

System.out.println(difficulty);

}

**Help face:**

package grahn;

import java.io.\*;

import java.io.File;

import java.util.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

public class helpFace extends javax.swing.JFrame {

public helpFace() throws IOException {

initComponents();

dispose(); // Resizes and centres JFrame

setUndecorated(true); // Mathew Broderick

pack();

setLocationRelativeTo(null);

setVisible(true);

String out = "\n";

String line = "";

Scanner scFile = new Scanner (new File("help.txt")); //extracts and displays help information.

while (scFile.hasNext())

{

line = scFile.nextLine();

out = out + " " + line + "\n";

}

scFile.close();

txtHelp.setText(out);

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

scoreBut = new javax.swing.JButton();

backBut = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

txtHelp = new javax.swing.JTextArea();

jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

scoreBut.setBackground(new java.awt.Color(255, 255, 255));

scoreBut.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N

scoreBut.setText("HIGHSCORE");

scoreBut.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, new java.awt.Color(204, 255, 51), new java.awt.Color(204, 255, 51), null, null));

scoreBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

scoreButActionPerformed(evt);

}

});

getContentPane().add(scoreBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(1300, 380, 270, 150));

backBut.setBackground(new java.awt.Color(255, 255, 255));

backBut.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

backBut.setText("BACK");

backBut.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, new java.awt.Color(204, 255, 51), new java.awt.Color(204, 255, 51), null, null));

backBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

backButActionPerformed(evt);

}

});

getContentPane().add(backBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, 120, 50));

txtHelp.setColumns(20);

txtHelp.setFont(new java.awt.Font("Monospaced", 0, 18)); // NOI18N

txtHelp.setRows(5);

txtHelp.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

jScrollPane1.setViewportView(txtHelp);

getContentPane().add(jScrollPane1, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 100, 1240, 710));

jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/grahn/help backround.png"))); // NOI18N

jLabel1.setMaximumSize(new java.awt.Dimension(2000, 2000));

jLabel1.setPreferredSize(new java.awt.Dimension(1234, 1234));

getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1600, 900));

setSize(new java.awt.Dimension(1616, 939));

setLocationRelativeTo(null);

}// </editor-fold>

private void backButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

mainFace mFace = new mainFace();

mFace.setVisible(true);

}

private void scoreButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

pickHighScoreTable pt = new pickHighScoreTable();

pt.setVisible(true);

}

**Pick high score table:**

package grahn;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

public class pickHighScoreTable extends javax.swing.JFrame {

public pickHighScoreTable() {

initComponents(); //Thanks to Mathew Broderick

dispose(); // Resizes and centres JFrame

setUndecorated(true);

pack();

setLocationRelativeTo(null);

setVisible(true);

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

easytblBut = new javax.swing.JButton();

normaltblBut = new javax.swing.JButton();

hardtblBut = new javax.swing.JButton();

backBut = new javax.swing.JButton();

lblBack = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

easytblBut.setBackground(new java.awt.Color(255, 255, 255));

easytblBut.setFont(new java.awt.Font("Tahoma", 1, 40)); // NOI18N

easytblBut.setText("Easy High Score");

easytblBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

easytblButActionPerformed(evt);

}

});

getContentPane().add(easytblBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(330, 170, 460, 120));

normaltblBut.setBackground(new java.awt.Color(255, 255, 255));

normaltblBut.setFont(new java.awt.Font("Tahoma", 1, 40)); // NOI18N

normaltblBut.setText("Normal High Score");

normaltblBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

normaltblButActionPerformed(evt);

}

});

getContentPane().add(normaltblBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(330, 420, 460, 120));

hardtblBut.setBackground(new java.awt.Color(255, 255, 255));

hardtblBut.setFont(new java.awt.Font("Tahoma", 1, 40)); // NOI18N

hardtblBut.setText("Hard High Score");

hardtblBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

hardtblButActionPerformed(evt);

}

});

getContentPane().add(hardtblBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(330, 670, 470, 120));

backBut.setBackground(new java.awt.Color(255, 255, 255));

backBut.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

backBut.setText("BACK");

backBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

backButActionPerformed(evt);

}

});

getContentPane().add(backBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 10, 140, 50));

lblBack.setIcon(new javax.swing.ImageIcon(getClass().getResource("/grahn/pickHighScoreBackround.png"))); // NOI18N

getContentPane().add(lblBack, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1600, 910));

setSize(new java.awt.Dimension(1616, 947));

setLocationRelativeTo(null);

}// </editor-fold>

private void easytblButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

EasyScoreFace ef = new EasyScoreFace();

ef.setVisible(true);

}

private void normaltblButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

NormalScoreFace sf = new NormalScoreFace();

sf.setVisible(true);

}

private void hardtblButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

hardScoreFace hf = new hardScoreFace();

hf.setVisible(true);

}

private void backButActionPerformed(java.awt.event.ActionEvent evt) {

try {

this.dispose();

helpFace hFace = new helpFace();

hFace.setVisible(true);

} catch (IOException ex) {

Logger.getLogger(pickHighScoreTable.class.getName()).log(Level.SEVERE, null, ex); // needed for IOException error

System.out.println("Back Button did not execute");

}

}

**Easy score face:**

package grahn;

//import java.awt.Frame;

import java.util.Scanner;

/\*\*

\*

\* @author egrahn

\*/

public class EasyScoreFace extends javax.swing.JFrame {

/\*\*

\* Creates new form EasyScoreFace

\*/

public EasyScoreFace() {

initComponents();

dispose(); // Resizes and centres JFrame

setUndecorated(true); // Mathew Broderick

pack();

setLocationRelativeTo(null);

setVisible(true);

useDatabase highscores = new useDatabase();

String data = highscores.displaytblHighScore("easy"); //fetches easy players and their scores. sends across difficulty

System.out.println(data);

Scanner scData = new Scanner(data);

int row = 0;

int col = 0;

while (scData.hasNextLine()) //Changes and displays data in jTable format

{

Scanner scLine = new Scanner(scData.nextLine()).useDelimiter("#");

col = 0;

while (scLine.hasNext()) {

tblEasy.setValueAt(scLine.next(), row, col);

col++;

}

row++;

scLine.close();

}

scData.close();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

backBut = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

tblEasy = new org.jdesktop.swingx.JXTable();

lblHeading = new javax.swing.JLabel();

homeBut = new javax.swing.JButton();

lblBack = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

backBut.setBackground(new java.awt.Color(255, 255, 255));

backBut.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

backBut.setText("BACK");

backBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

backButActionPerformed(evt);

}

});

getContentPane().add(backBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, 130, 60));

tblEasy.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null}

},

new String [] {

"Position", "Name", "Score"

}

) {

Class[] types = new Class [] {

java.lang.String.class, java.lang.String.class, java.lang.String.class

};

public Class getColumnClass(int columnIndex) {

return types [columnIndex];

}

});

tblEasy.setFont(new java.awt.Font("Tahoma", 0, 24)); // NOI18N

tblEasy.setRowHeight(35);

jScrollPane1.setViewportView(tblEasy);

getContentPane().add(jScrollPane1, new org.netbeans.lib.awtextra.AbsoluteConstraints(980, 120, 600, 730));

lblHeading.setBackground(new java.awt.Color(153, 255, 204));

lblHeading.setFont(new java.awt.Font("Tahoma", 1, 45)); // NOI18N

lblHeading.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

lblHeading.setText("Easy Leaderboard");

lblHeading.setOpaque(true);

getContentPane().add(lblHeading, new org.netbeans.lib.awtextra.AbsoluteConstraints(1040, 30, 480, 60));

homeBut.setBackground(new java.awt.Color(255, 255, 255));

homeBut.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

homeBut.setText("HOME");

homeBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

homeButActionPerformed(evt);

}

});

getContentPane().add(homeBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 80, 130, 60));

lblBack.setIcon(new javax.swing.ImageIcon(getClass().getResource("/grahn/score backround.png"))); // NOI18N

getContentPane().add(lblBack, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1600, 900));

pack();

}// </editor-fold>

private void backButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

pickHighScoreTable pt = new pickHighScoreTable();

pt.setVisible(true);

}

private void homeButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

mainFace mf = new mainFace();

mf.setVisible(true);

}

**Normal score face:**

package grahn;

import java.util.Scanner;

public class NormalScoreFace extends javax.swing.JFrame {

public NormalScoreFace() {

initComponents(); // Mathew Broderick

dispose(); // Resizes and centres JFrame

setUndecorated(true);

pack();

setLocationRelativeTo(null);

setVisible(true);

useDatabase highscores = new useDatabase();

String data = highscores.displaytblHighScore("normal"); //fetches normal players and their scores. sends across difficulty

Scanner scData = new Scanner(data);

int row = 0;

int col = 0;

while (scData.hasNextLine()) //Changes and displays data in table format

{

Scanner scLine = new Scanner(scData.nextLine()).useDelimiter("#");

col = 0;

while (scLine.hasNext()) {

tblNormal.setValueAt(scLine.next(), row, col);

col++;

}

row++;

scLine.close();

}

scData.close();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

backBut = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

tblNormal = new org.jdesktop.swingx.JXTable();

lblHeading = new javax.swing.JLabel();

homeBut = new javax.swing.JButton();

lblBack = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

backBut.setBackground(new java.awt.Color(255, 255, 255));

backBut.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

backBut.setText("BACK");

backBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

backButActionPerformed(evt);

}

});

getContentPane().add(backBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, 140, 50));

tblNormal.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null}

},

new String [] {

"Position", "Name", "Score"

}

) {

Class[] types = new Class [] {

java.lang.String.class, java.lang.String.class, java.lang.String.class

};

public Class getColumnClass(int columnIndex) {

return types [columnIndex];

}

});

tblNormal.setFont(new java.awt.Font("Tahoma", 0, 24)); // NOI18N

tblNormal.setRowHeight(50);

tblNormal.setRowHeight(35);

jScrollPane1.setViewportView(tblNormal);

getContentPane().add(jScrollPane1, new org.netbeans.lib.awtextra.AbsoluteConstraints(970, 120, 610, 730));

lblHeading.setBackground(new java.awt.Color(153, 255, 204));

lblHeading.setFont(new java.awt.Font("Tahoma", 1, 45)); // NOI18N

lblHeading.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

lblHeading.setText("Normal Leaderboard");

lblHeading.setOpaque(true);

getContentPane().add(lblHeading, new org.netbeans.lib.awtextra.AbsoluteConstraints(1020, 20, 520, 80));

homeBut.setBackground(new java.awt.Color(255, 255, 255));

homeBut.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

homeBut.setText("HOME");

homeBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

homeButActionPerformed(evt);

}

});

getContentPane().add(homeBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 70, 140, 50));

lblBack.setIcon(new javax.swing.ImageIcon(getClass().getResource("/grahn/score backround.png"))); // NOI18N

getContentPane().add(lblBack, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1600, 900));

pack();

}// </editor-fold>

private void backButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

pickHighScoreTable pt = new pickHighScoreTable();

pt.setVisible(true);

}

private void homeButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

mainFace mf = new mainFace();

mf.setVisible(true);

**Hard score face:**

package grahn;

import java.util.Scanner;

public class hardScoreFace extends javax.swing.JFrame {

public hardScoreFace() {

initComponents();

dispose(); // Resizes and centres JFrame

setUndecorated(true); // Mathew Broderick

pack();

setLocationRelativeTo(null);

setVisible(true);

useDatabase highscores = new useDatabase();

String data = highscores.displaytblHighScore("hard"); //fetches hard players and their scores. sends across hard

Scanner scData = new Scanner (data);

int row = 0;

int col = 0;

while (scData.hasNextLine()) //Changes and displays data in table format

{

Scanner scLine = new Scanner (scData.nextLine()).useDelimiter("#");

col = 0;

while (scLine.hasNext())

{

tblHard.setValueAt(scLine.next(), row, col);

col++;

}

row++;

scLine.close();

}

scData.close();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

backBut = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

tblHard = new org.jdesktop.swingx.JXTable();

lblHeading = new javax.swing.JLabel();

homeBut = new javax.swing.JButton();

lblBack = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

backBut.setBackground(new java.awt.Color(255, 255, 255));

backBut.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

backBut.setText("BACK");

backBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

backButActionPerformed(evt);

}

});

getContentPane().add(backBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, 140, 50));

tblHard.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null}

},

new String [] {

"Position", "Name", "Score"

}

) {

Class[] types = new Class [] {

java.lang.String.class, java.lang.String.class, java.lang.String.class

};

public Class getColumnClass(int columnIndex) {

return types [columnIndex];

}

});

tblHard.setFont(new java.awt.Font("Tahoma", 0, 24)); // NOI18N

tblHard.setRowHeight(35);

jScrollPane1.setViewportView(tblHard);

getContentPane().add(jScrollPane1, new org.netbeans.lib.awtextra.AbsoluteConstraints(980, 100, 610, 730));

lblHeading.setBackground(new java.awt.Color(153, 255, 204));

lblHeading.setFont(new java.awt.Font("Tahoma", 1, 45)); // NOI18N

lblHeading.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

lblHeading.setText("Hard Leaderboard");

lblHeading.setOpaque(true);

getContentPane().add(lblHeading, new org.netbeans.lib.awtextra.AbsoluteConstraints(1070, 20, 440, 60));

homeBut.setBackground(new java.awt.Color(255, 255, 255));

homeBut.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

homeBut.setText("HOME");

homeBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

homeButActionPerformed(evt);

}

});

getContentPane().add(homeBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 80, 140, 60));

lblBack.setIcon(new javax.swing.ImageIcon(getClass().getResource("/grahn/score backround.png"))); // NOI18N

getContentPane().add(lblBack, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1600, 900));

pack();

}// </editor-fold>

private void backButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

pickHighScoreTable pt = new pickHighScoreTable();

pt.setVisible(true);

}

private void homeButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

mainFace mf = new mainFace();

mf.setVisible(true);

}

**Database:**

package grahn;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

/\*\*

\*

\* @author Bryce Tristan

\*/

public class Database {

public static Connection connection;

private PreparedStatement statement;

private ResultSet resultSet;

static public Connection conn;

public Database(String databaseName) //Forms the link between the interface and the database. receives the database name

{

String directory = System.getProperty("user.dir");

String url = directory + "\\" + databaseName;

try {

Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");

connection = DriverManager.getConnection("jdbc:ucanaccess://" + url + ".accdb");

System.out.println("CONNECTED");

} catch (ClassNotFoundException | SQLException ex) {

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

} catch (ExceptionInInitializerError ex) {

System.out.println("ERROR");

ex.getCause();

}

}

public ResultSet query(String stmt) throws SQLException //Used for fetchig data from the database. recieves query // Mathew Broderick

{

statement = connection.prepareStatement(stmt);

resultSet = statement.executeQuery();

System.out.println(resultSet);

return resultSet;

}

public void change(String changeIn) throws SQLException //used for removing or inserting data from or to the database. revieves query

{

statement = connection.prepareStatement(changeIn);

statement.executeUpdate();

statement.close();

}

}

**Usedatabase:**

package grahn;

import java.sql.ResultSet;

import java.sql.SQLException;

/\*\*

\*

\* @author Bryce Tristan

\*/

public class useDatabase {

private Database database = new Database("HighScoreDatabase"); //creates object of class database

public String displaytblHighScore(String difficulty) //Used for fetching and sending across a string with the players and their scores according to the difficulty specified

{

String temp = "";

int pos = 0;

try {

ResultSet rs = database.query("SELECT TOP 20\* FROM tblHighScore WHERE Difficulty = '" + difficulty + "'" + "ORDER BY Score DESC, FirstName"); //fetches the top 20 players occording to their score and difficuly level

while (rs.next()) {

pos++;

String username = rs.getString("FirstName");

int score = rs.getInt("Score");

temp = temp + pos + "#" + username + "#" + score + "\n";

}

System.out.println("DATA EXTRACTED");

} catch (SQLException ex) {

System.out.println(ex);

}

return temp;

}

public void UpdateTable(String name) { //Uploads a new player, their score and difficulty to the table

int score = fetchScore();

String what = "Difficulty";

String difficulty = fetchWhat(what); //fetches difficulty uploaded to the database

System.out.println(difficulty);

String tbl = "";

try {

database.change("INSERT INTO tblHighScore (FirstName,Score,Difficulty)" + "VALUES ('" + name + "','" + score + "','" + difficulty + "')"); // Inserts a new players username and score into a specific database table indicated

} catch (SQLException ex) {

System.out.println(ex);

}

}

public void InsertData(int id, int score1, String difficulty, String stat, String closeGame) //inserts vital game status information into tblData

{

try {

database.change("INSERT INTO tblData(Id,Score,Difficulty,Status,Close)" + "VALUES ('" + id + "','" + score1 + "','" + difficulty + "','" + stat + "','"

+ closeGame + "')");

} catch (SQLException ex) {

System.out.println(ex);

}

}

public void UpdateScore(int score) //used to update score to the database each time an apple is eaten

{

int id = 1;

try {

database.change("UPDATE tblData SET Score = '" + score + "'" + "WHERE ID = '" + id + "'");

System.out.println("Score updated");

} catch (SQLException ex) {

System.out.println(ex);

}

}

public void updateWhat(String what, String value) //update a value in a certain column (what) into tblData when required

{

int id = 1;

try {

database.change("UPDATE tblData SET " + what + " = '" + value + "'" + "WHERE ID = '" + id + "'");

System.out.println("Status updated");

} catch (SQLException ex) {

System.out.println(ex);

}

}

public void deleteData() //delete tblData's information at the end of each game to ensure each new game doesn't mix data

{

int id = 1;

try {

database.change("DELETE FROM tblData WHERE ID = '" + id + "'");

} catch (SQLException ex) {

System.out.println(ex);

}

}

public int fetchScore() { //fetches the score from tblData so that it can be saved into the highcore tables with the username

int id = 1;

int score = 0;

try {

ResultSet rs = database.query("SELECT Score FROM tblData WHERE ID = '" + id + "'");

rs.next();

score = rs.getInt("Score");

System.out.println("DATA EXTRACTED");

} catch (SQLException ex) {

System.out.println(ex);

}

return score;

}

public String fetchWhat(String what) // fetches a value from a certain column in tblData when required

{

String item = "";

int id = 1;

try {

ResultSet rs = database.query("SELECT " + what + " FROM tblData WHERE ID = '" + id + "'");

rs.next();

item = rs.getString(what);

System.out.println("Fetch Item : " + item);

System.out.println("DATA EXTRACTED");

} catch (SQLException ex) {

System.out.println(ex);

}

return item;

}

}

**Save Score:**

package grahn;

public class saveScore extends javax.swing.JFrame {

public saveScore() {

initComponents(); //Thanks to Mathew Broderick

dispose(); // Resizes and centres JFrame

setUndecorated(true);

pack();

setLocationRelativeTo(null);

setVisible(true);

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

yesBut = new javax.swing.JButton();

noBut = new javax.swing.JButton();

lblHeading = new javax.swing.JLabel();

lblBackround = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setBackground(new java.awt.Color(204, 255, 204));

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

yesBut.setFont(new java.awt.Font("Tahoma", 1, 48)); // NOI18N

yesBut.setText("YES");

yesBut.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, new java.awt.Color(204, 255, 51), new java.awt.Color(204, 255, 51), null, null));

yesBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

yesButActionPerformed(evt);

}

});

getContentPane().add(yesBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(410, 410, 270, 140));

noBut.setFont(new java.awt.Font("Tahoma", 1, 48)); // NOI18N

noBut.setText("NO");

noBut.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, new java.awt.Color(204, 255, 51), new java.awt.Color(204, 255, 51), null, null));

noBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

noButActionPerformed(evt);

}

});

getContentPane().add(noBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(880, 410, 270, 140));

lblHeading.setBackground(new java.awt.Color(0, 153, 153));

lblHeading.setFont(new java.awt.Font("Engravers MT", 1, 50)); // NOI18N

lblHeading.setForeground(new java.awt.Color(255, 255, 255));

lblHeading.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

lblHeading.setText("Would you like to save your score?");

lblHeading.setOpaque(true);

getContentPane().add(lblHeading, new org.netbeans.lib.awtextra.AbsoluteConstraints(50, 20, 1480, 110));

lblBackround.setIcon(new javax.swing.ImageIcon(getClass().getResource("/grahn/saveScoreBackround.png"))); // NOI18N

getContentPane().add(lblBackround, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1600, 910));

setSize(new java.awt.Dimension(1616, 953));

setLocationRelativeTo(null);

}// </editor-fold>

private void yesButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

scoreInput sInput = new scoreInput();

sInput.setVisible(true);

}

private void noButActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

mainFace mFace = new mainFace();

mFace.setVisible(true);

}

**Score Input:**

package grahn;

import java.awt.Color;

public class scoreInput extends javax.swing.JFrame {

public scoreInput() {

initComponents(); //Thanks to Mathew Broderick

dispose(); // Resizes and centres JFrame

setUndecorated(true);

pack();

setLocationRelativeTo(null);

setVisible(true);

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

lblHeading = new javax.swing.JLabel();

txtNameInput = new javax.swing.JTextField();

enterBut = new javax.swing.JButton();

lblStat = new javax.swing.JLabel();

lblBackround = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setBackground(new java.awt.Color(204, 255, 204));

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

lblHeading.setBackground(new java.awt.Color(0, 153, 153));

lblHeading.setFont(new java.awt.Font("Engravers MT", 1, 45)); // NOI18N

lblHeading.setForeground(new java.awt.Color(255, 255, 255));

lblHeading.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

lblHeading.setText("Please type in your name and clicK Enter");

lblHeading.setOpaque(true);

getContentPane().add(lblHeading, new org.netbeans.lib.awtextra.AbsoluteConstraints(40, 20, 1510, 110));

txtNameInput.setFont(new java.awt.Font("Tahoma", 0, 48)); // NOI18N

txtNameInput.setHorizontalAlignment(javax.swing.JTextField.CENTER);

txtNameInput.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

txtNameInput.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

txtNameInputActionPerformed(evt);

}

});

getContentPane().add(txtNameInput, new org.netbeans.lib.awtextra.AbsoluteConstraints(540, 220, 480, 150));

enterBut.setFont(new java.awt.Font("Tahoma", 1, 40)); // NOI18N

enterBut.setText("ENTER");

enterBut.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, new java.awt.Color(204, 255, 51), new java.awt.Color(204, 255, 51), null, null));

enterBut.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

enterButActionPerformed(evt);

}

});

getContentPane().add(enterBut, new org.netbeans.lib.awtextra.AbsoluteConstraints(630, 710, 290, 100));

lblStat.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N

lblStat.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

getContentPane().add(lblStat, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 440, 1600, 90));

lblBackround.setIcon(new javax.swing.ImageIcon(getClass().getResource("/grahn/scoreInputBackround.png"))); // NOI18N

getContentPane().add(lblBackround, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1600, 910));

setSize(new java.awt.Dimension(1616, 953));

setLocationRelativeTo(null);

}// </editor-fold>

private void txtNameInputActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void enterButActionPerformed(java.awt.event.ActionEvent evt) {

String name = txtNameInput.getText();

boolean stat = false;

String nameOut = "";

String text = "abc";

int len = name.length();

if(!name.equals("")) //Validates username

{

if(len <= 15)

{

if(name.matches("[a-zA-Z]\*")) //Checks for non-alphabetic characters

{

nameOut = name;

lblStat.setText("CORRECT");

lblStat.setForeground(Color.green);

stat = true;

this.dispose();

mainFace mFace = new mainFace();

mFace.setVisible(true);

}

else

{

lblStat.setText("Invalid format (contains non aphabetical values)");

lblStat.setForeground(Color.red);

}

}

else

{

lblStat.setText("Invalid length (15 characters or less)");

lblStat.setForeground(Color.red);

}

}

else

{

lblStat.setText("Please enter your name");

lblStat.setForeground(Color.red);

}

String difficulty = "";

if (stat == true)

{

useDatabase ud = new useDatabase();

System.out.println(difficulty);

ud.UpdateTable(name); //Updates database with relevant username

}

}

**CountDown Timer:**

package grahn;

import java.util.Timer;

import java.util.TimerTask;

/\*\*

\*

\* @author Bryce Tristan

\*/

public class CountDownTimer {

private int counter;

public CountDownTimer() {

}

public void startTimer() // creates a 20 second count down timer for certain power ups

{

Timer timer = new Timer();

counter = 20;

TimerTask task;

task = new TimerTask() {

public void run() {

System.out.println(counter);

counter--;

if (counter <= -1) { // Cancels the timer after 20 seconds

timer.cancel();

}

}

};

timer.scheduleAtFixedRate(task, 1000, 1000); //1000ms = 1sec

}

public int getTime() { // returns time in seconds

return counter;

}

}

**SnakeMain:**

package grahn;

import javax.swing.JFrame;

import java.awt.EventQueue;

public class SnakeMain extends javax.swing.JFrame {

public SnakeMain() {

add(new GameBoard()); //Initializes starts the game board (aka the game)

dispose(); // resizes and centres the JFrame

setUndecorated(true); //Thanks to Mathew Broderick

pack();

setLocationRelativeTo(null);

setVisible(true);

setResizable(false);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

JFrame ex = new SnakeMain();

ex.setVisible(true);

}

});

}

}

**GameBoard:**

package grahn;

import javax.swing.ImageIcon;

import java.awt.FontMetrics;

import java.awt.Graphics;

import java.awt.Color;

import java.awt.Cursor;

import java.awt.Dimension;

import javax.swing.JPanel;

import javax.swing.Timer;

import java.awt.Font;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.awt.Image;

import java.awt.Point;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.image.BufferedImage;

public class GameBoard extends JPanel implements ActionListener {

private final int gridSize = 900;

private final int randAppleXValue = 63;

private final int randAppleYValue = 32;

private final int numOfLily = 45;

private final int numOfPyramids = 35;

private final int numOfWall = 140;

private final int xArr[] = new int[gridSize];

private final int yArr[] = new int[gridSize];

private final int objectXArr[] = new int[numOfWall];

private final int objectYArr[] = new int[numOfWall];

private final int gameWidth = 1600;

private final int gameHeight = 900;

private final int unitSize = 25;

private final int powerUpProbability = 8;

private final int powerProbability = 4;

private Timer gameTimer;

private Image unitImage;

private Image appleImage;

private Image headImage;

private Image goldenAppleImage;

private Image snakeHead;

private Image snakeBod;

private Image object;

private boolean upDirec = false;

private boolean downDirec = false;

private boolean leftDirec = false;

private boolean rightDirec = true;

private boolean inGame = true;

private boolean goldenStat = false;

private boolean powerStat = false;

private boolean redo = false;

private boolean redoObject = false;

private boolean invincibility = false;

private boolean doublePoints = false;

private boolean reverse = false;

private boolean countDown = false;

private boolean move = true;

private int appleYPos;

private int appleXPos;

private int numSnakeUnits;

private int goldenYPos;

private int goldenXPos;

private int numObject = 0;

private int rNum = 0;

private int score = 0;

private int lives = 3;

private int timer = 20;

private int delay = 140;

private int speedDelay;

private int usualDelay;

private int tempDelay;

private int flashCount = 2;

private int objectXPos;

private int objectYPos;

private int xPos;

private int yPos;

private String difficulty;

private String what;

private String appleLocation;

private String bodLocation;

private String headLocation;

private String powerMsg = "";

private String appleType = "A";

private useDatabase ud = new useDatabase();

private CountDownTimer t = new CountDownTimer ();

public GameBoard() {

addKeyListener(new TAdapter());

setBackground(Color.BLUE);

setFocusable(true);

setPreferredSize(new Dimension(gameWidth, gameHeight));

BufferedImage cursorImg = new BufferedImage(16, 16, BufferedImage.TYPE\_INT\_ARGB); // creates a blank cursor

Cursor blankCursor = Toolkit.getDefaultToolkit().createCustomCursor(

cursorImg, new Point(0, 0), "blank cursor");

setCursor(blankCursor);

addNewGameLayout();

loadSnakeImages();

loadAppleImages();

beginGame();

}

private void addNewGameLayout() //Loads either the normal, hard, or easy game layout

{

what = "Difficulty";

difficulty = ud.fetchWhat(what); // fetches difficulty and sends arross column heading

if (difficulty.equalsIgnoreCase("Hard")) {

delay = 70;

usualDelay = 70;

speedDelay = 140;

appleLocation = "hardApple.png";

bodLocation = "hardBod.png";

headLocation = "hardHead.png";

setBackground(Color.orange);

addObjects(numOfPyramids, "wall.png"); // sends across number of pyramids and image name

} else if (difficulty.equalsIgnoreCase("Easy")) {

usualDelay = 140;

speedDelay = 160;

appleLocation = "easyApple.png";

bodLocation = "easyBod.png";

headLocation = "easyHead.png";

setBackground(Color.black);

} else {

delay = 110;

usualDelay = 110;

speedDelay = 140;

appleLocation = "normalApple.png";

bodLocation = "normalBod.png";

headLocation = "normalHead.png";

setBackground(Color.cyan);

addObjects(numOfLily, "lily.png"); // sends across number of lilys and image name

}

}

private void addObjects(int num, String image) //Adds either lilys or pyramids. recives number of obejcts and image name

{

ImageIcon iI = new ImageIcon(image);

object= iI.getImage();

for (int i = 0; i < num; i++) {

locateObject();

}

}

private void locateObject() //randomly generates the positions of the lilys or pyramids

{

redoObject = false;

int num = (int) (Math.random() \* randAppleXValue - 3) + 3;

objectXPos = ((num \* unitSize));

num = (int) (Math.random() \* randAppleYValue - 3) + 6;

objectYPos = ((num \* unitSize));

checkObjectTrue();

}

private void checkObjectTrue() //checks if the objects co-ordinates aren't of any previously generated ones

{

for (int i = 0; i < numObject; i++) {

if ((objectXPos == objectXArr[i]) && (objectYPos == objectYArr[i])) { //check for lily

redoObject = true;

}

if(difficulty.equalsIgnoreCase("Hard")) //check for pyramid

{

if ((objectXPos == objectXArr[i]) && ((objectYPos - (1 \* unitSize)) == objectYArr[i])) {

redoObject = true;

}

if (((objectXPos + (1 \* unitSize)) == objectXArr[i]) && ((objectYPos - (1 \* unitSize)) == objectYArr[i])) {

redoObject = true;

}

if (((objectXPos + (1 \* unitSize)) == objectXArr[i]) && (objectYPos == objectYArr[i])) {

redoObject = true;

}

}

}

if (redoObject == true) {

locateObject();

} else {

if (difficulty.equalsIgnoreCase("Normal")) //if generating lilys

{

objectXArr[numObject] = objectXPos;

objectYArr[numObject] = objectYPos;

numObject++;

}

else //if generating pyramids

{

objectXArr[numObject] = objectXPos;

objectYArr[numObject] = objectYPos;

objectXArr[(numObject + 1)] = objectXPos;

objectYArr[(numObject + 1)] = objectYPos - (1 \* unitSize);

objectXArr[(numObject + 2)] = objectXPos + (1 \* unitSize);

objectYArr[(numObject + 2)] = objectYPos - (1 \* unitSize);

objectXArr[(numObject + 3)] = objectXPos + (1 \* unitSize);

objectYArr[(numObject + 3)] = objectYPos;

numObject = numObject + 4;

}

}

}

private void loadSnakeImages() { //Loads snake images relating to the difficulty level chosen

ImageIcon iB = new ImageIcon(bodLocation); //uses image name

unitImage = iB.getImage();

ImageIcon iH = new ImageIcon(headLocation);

headImage = iH.getImage();

}

private void loadAppleImages()

{

ImageIcon iA = new ImageIcon(appleLocation);

appleImage = iA.getImage();

ImageIcon iG = new ImageIcon("goldenApple.png"); // uses image name

goldenAppleImage = iG.getImage();

}

private void beginGame() { //initialise snake and generates apples

numSnakeUnits = 8;

for (int i = 0; i < numSnakeUnits; i++) { //generates snake with 3 units and 5 virtual units (used when reversing) in the top left hand corner

xArr[i] = 100 - i \* 10;

yArr[i] = 50;

}

locateTheApple();

chooseGoldenApple();

gameTimer = new Timer(delay, this); //starts game timer (the delay specifies the rate at which everyhting is processed and thus the speed of the snake)

gameTimer.start();

}

private void locateTheApple() { //randomly generates apple co-ordinates

redo = false;

int num = (int) (Math.random() \* randAppleXValue);

xPos = ((num \* unitSize));

num = (int) (Math.random() \* randAppleYValue) + 4;

yPos = ((num \* unitSize));

if(appleType.equalsIgnoreCase("A")) //if normal apple

{

appleXPos = xPos;

appleYPos = yPos;

}

if(appleType.equalsIgnoreCase("G")) //if golden apple

{

goldenXPos = xPos;

goldenYPos = yPos;

}

checkAppleTrue();

}

private void checkAppleTrue() { //checks the apples co-ordinates to see if it clashes with either the snakes, lilys or pyramids co-ordinates

for (int i = numSnakeUnits; i > 0; i--) {

if ((xPos == xArr[i]) && (yPos == yArr[i])) {

redo = true;

}

}

for (int i = 0; i < numObject; i++) { //checks if the apples co-ordinates are the same as the lilys or pyramids

if ((xPos == objectXArr[i]) && (yPos == objectYArr[i])) {

redo = true;

}

}

if (redo == true) {

locateTheApple(); //if co-ordinates clash the apples co-ordinates are regenerated

}

}

private void chooseGoldenApple() { //determines the probability of generating a power up (12.5% chance of generating a power up each time a normal apple is generated)

if (goldenStat == false) {

int val = (int) (Math.random() \* powerUpProbability);

if (val == 1) {

goldenStat = true;

appleType = "G";

locateTheApple();

}

}

}

@Override

public void paintComponent(Graphics g) { //draws snake, score, objects etc... on the playfield

super.paintComponent(g);

draw(g, ("LIVES : " + lives), 1450, 35); // sends across the message heading and co-ordinates

draw(g, ("SCORE : " + score), 10, 35);

draw(g, powerMsg, 665, 35);

if (countDown == true) //Only starts 20 second coutdown for specific powers

{

drawCountDown(g);

}

initiateDrawing(g);

}

private void draw(Graphics g, String text, int x, int y) //draws either the score, power or number of lives. receives the heading and co-ordinates

{

Font size = new Font("Helvetica", Font.BOLD, 30);

String message = text;

FontMetrics metric = getFontMetrics(size);

if (difficulty.equalsIgnoreCase("easy")) {

g.setColor(Color.white);

} else {

g.setColor(Color.black);

}

g.setFont(size);

g.drawString(message, (x), y);

}

private void drawCountDown(Graphics g) { //fetches and displays the countdown next to the power message on the playing field

timer = t.getTime();

draw(g, (timer + ""), 965, 35);

if (timer == 0)

{

invincibility = false;

doublePoints = false;

countDown = false;

powerMsg = "";

changeSpeed(usualDelay); // sends across the usual delay depending on the difficulty

}

}

private void initiateDrawing(Graphics g) { //Draws Snake and Apples

if (inGame == true) {

g.drawImage(appleImage, appleXPos, appleYPos, this); //Displays apple

if (goldenStat == true) {

g.drawImage(goldenAppleImage, goldenXPos, goldenYPos, this);

}

if(!difficulty.equalsIgnoreCase("Easy"))

{

for (int i = 0; i < numObject; i++) { //Displays the lilys or pyramids

g.drawImage(object, objectXArr[i], objectYArr[i], this);

}

}

if (invincibility == true) //if invincibility power is activated these are the snake images to be used

{

ImageIcon inH = new ImageIcon("invincibleHead.png");

snakeHead = inH.getImage();

ImageIcon inB = new ImageIcon("invincibleBod.png");

snakeBod = inB.getImage();

}

else

{

if (doublePoints == true) //if doube points is activated then these are the snake images to be used

{

ImageIcon dH = new ImageIcon("doubleHead.png");

snakeHead = dH.getImage();

ImageIcon dB = new ImageIcon("doubleBod.png");

snakeBod = dB.getImage();

flash(snakeHead, snakeBod); // sends across image names

}

else // used to flash the snake if the snake hits a pyramid or lily

{

flash(headImage, unitImage); //

}

}

for (int i = numSnakeUnits - 6; i >= 0; i--) { //Displays all the snake units except the head and last 5 virtual units used for reversing

if (i == 0) {

g.drawImage(snakeHead, xArr[i], yArr[i], this); //Displays the head

} else {

g.drawImage(snakeBod, xArr[i], yArr[i], this); //Displays the body units

}

}

Toolkit.getDefaultToolkit().sync();

} else {

endGame(g); // ends the game if inGame is equal to false

}

}

private void flash(Image h, Image b) //simulatanously switches between normal images and clear images to provide the appearance of flashing when needed. receives image names

{

if (move == false && flashCount < 10)

{

ImageIcon iI = new ImageIcon("invisiSnake.png");

if(flashCount % 2 == 0)

{

snakeHead = iI.getImage();

snakeBod = iI.getImage();

}

else

{

snakeHead = h;

snakeBod = b;

}

flashCount++;

}

else

{

snakeHead = h;

snakeBod = b;

move = true;

if (flashCount >= 10)

{

moveSnake();

}

flashCount = 2;

}

}

private void endGame(Graphics g) { //when the player runs out of lives the game will end and the option of saving your score will be brought up

Font size = new Font("Helvetica", Font.BOLD, 14);

String message = "Game Over";

ud.UpdateScore(score);

FontMetrics metric = getFontMetrics(size);

g.setFont(size);

g.setColor(Color.white);

g.drawString(message, (gameWidth - metric.stringWidth(message)) / 2, gameHeight / 2);

setVisible(false);

saveScore ss = new saveScore();

ss.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) { //This method constantly loops with regard to the gameTimer

if (inGame) {

checkHitApple();

if (goldenStat == true)

{

checkHitGoldenApple();

}

checkHit();

if (reverse == false) {

moveSnake();

} else {

reverseSnake();

rNum++;

}

if (rNum > 5) {

reverse = false;

rNum = 0;

changeTimer(tempDelay);

}

}

repaint();

}

private void checkHitApple() { //checks if snake's head co-ordinates match that of an apple

if ((xArr[0] == appleXPos) && (yArr[0] == appleYPos)) {

numSnakeUnits++;

if (doublePoints == true) {

score = score + 2;

} else {

score++;

}

ud.UpdateScore(score); // sends across the score

if(t.getTime() <= 0) //Only starts regenerating golden apples if the coundown of the previous power has come to 0

{

chooseGoldenApple();

}

appleType = "A";

locateTheApple(); //locates new apple

}

}

private void checkHitGoldenApple() { //checks if snake's head co-ordinates match that of a golden apple

if ((xArr[0] == goldenXPos) && (yArr[0] == goldenYPos)) {

goldenStat = false;

powerType(); //If yes; the program randomly chooses a power type

}

}

private void powerType() { //choose a power ocordingly to the number generated(Invincibility, Extra life, Speed change or Double points

powerStat = true;

int prob = (int) (Math.random() \* powerProbability);

if (prob == 1) {

powerMsg = "INVINCIBILITY";

countDown = true;

t.startTimer();

invincibility = true;

doublePoints = false;

} else if (prob == 2) {

powerMsg = "EXTRA LIFE";

lives++;

invincibility = false;

doublePoints = false;

} else if (prob == 3) {

powerMsg = "SPEED CHANGE";

countDown = true;

t.startTimer();

invincibility = false;

doublePoints = false;

changeSpeed(speedDelay);

} else {

powerMsg = "DOUBLE POINTS";

countDown = true;

t.startTimer();

invincibility = false;

doublePoints = true;

}

}

private void changeSpeed(int tempDelay) //Changes game timers delay in order to change the speed of the snake. receives delay factor

{

gameTimer.stop();

gameTimer = new Timer(tempDelay, this);

gameTimer.start();

}

private void checkHit() { //Checks if the snake hits either itself, a lily, a pyramid or the edge of the board

if (invincibility == false && move == true) {

for (int i = numSnakeUnits - 5; i > 0; i--) {

if ((i > 4) && (xArr[0] == xArr[i]) && (yArr[0] == yArr[i])) {

if (lives == 1) {

inGame = false;

} else {

lives--;

move = false;

}

}

}

if(!difficulty.equalsIgnoreCase("Easy")) //Check hit Lily or pyramid

{

for (int i = 0; i < numObject; i++) {

if ((xArr[0] == objectXArr[i]) && (yArr[0] == objectYArr[i])) {

if (lives == 1) {

inGame = false;

} else {

lives--;

move = false;

}

}

}

}

}

if (yArr[0] >= gameHeight || yArr[0] < 0 || xArr[0] >= gameWidth || xArr[0] < 0) { //Check if the snake hit the edge of the board

if (lives == 1) {

inGame = false;

} else {

reverse = true;

if (invincibility == false) {

lives--;

}

changeTimer(250);

}

}

if (inGame == false) {

gameTimer.stop();

}

}

private void changeTimer(int newDelay) { //Changes timer speed by changing the delay factor. recieves delay factor

tempDelay = delay;

gameTimer.stop();

delay = newDelay;

gameTimer = new Timer(delay, this);

gameTimer.start();

}

private void moveSnake() { //moves the snake one unit at a time in a specific direction

if (move == true)

{

for (int i = numSnakeUnits; i > 0; i--) {

xArr[i] = xArr[(i - 1)];

yArr[i] = yArr[(i - 1)];

}

if (leftDirec == true) {

xArr[0] = xArr[0] - unitSize;

}

if (rightDirec == true) {

xArr[0] = xArr[0] + unitSize;

}

if (upDirec == true) {

yArr[0] = yArr[0] - unitSize;

}

if (downDirec == true) {

yArr[0] = yArr[0] + unitSize;

}

}

}

private void reverseSnake() { //reverses the snake using the virtual units when the snake hits the edge of the board

for (int i = 0; i < numSnakeUnits; i++) {

xArr[i] = xArr[(i + 1)];

yArr[i] = yArr[(i + 1)];

}

if (leftDirec) {

xArr[numSnakeUnits] = xArr[numSnakeUnits] + unitSize;

}

if (rightDirec) {

xArr[numSnakeUnits] = xArr[numSnakeUnits] - unitSize;

}

if (upDirec) {

yArr[numSnakeUnits] = yArr[numSnakeUnits] + unitSize;

}

if (downDirec) {

yArr[numSnakeUnits] = yArr[numSnakeUnits] - unitSize;

}

}

private class TAdapter extends KeyAdapter { //extends the class and creates a key listener

@Override

public void keyPressed(KeyEvent e) {

int key = e.getKeyCode();

if (reverse == false) {

if ((key == KeyEvent.VK\_LEFT) && (!rightDirec)) { // if left key is pressed

leftDirec = true;

upDirec = false;

downDirec = false;

}

if ((key == KeyEvent.VK\_RIGHT) && (!leftDirec)) { //if right key is pressed

rightDirec = true;

upDirec = false;

downDirec = false;

}

if ((key == KeyEvent.VK\_UP) && (!downDirec)) { //if up key is pressed

upDirec = true;

rightDirec = false;

leftDirec = false;

}

if ((key == KeyEvent.VK\_DOWN) && (!upDirec)) { //if down key is pressed

downDirec = true;

rightDirec = false;

leftDirec = false;

}

}

}

}

}