/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Bailey Thompson

\* Trump Will Triump (1.3)

\* 26 September 2016

\* Info: You must play as Donald Trump to conquer the U.S.A. a state at a time by playing various mini-games. Each state has

\* Info: its own mini-game. If the mini-game is won, the state becomes Republican, and thus red. If the mini-game is lost, the

\* Info: state becomes Democrat, and thus blue. Since there are 50 states in the U.S.A., at the end of the game, if 25 or more

\* Info: states become Republican, you, Donald Trump, become president of the United States of America. However, if less than

\* Info: 25 states are won, you lose the race to presidency. Let it be noted that Triump was spelled like that on purpose.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

//declaring package

package trumpwilltriump;

//declaring imports

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.GridLayout;

import java.awt.Point;

import java.awt.Rectangle;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.InputEvent;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.event.MouseMotionListener;

import java.io.BufferedOutputStream;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.OutputStream;

import static java.lang.Integer.parseInt;

import java.nio.file.FileAlreadyExistsException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import static java.nio.file.StandardOpenOption.TRUNCATE\_EXISTING;

import static java.nio.file.StandardOpenOption.WRITE;

import java.util.ArrayList;

import java.util.List;

import java.util.Timer;

import java.util.TimerTask;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JSlider;

//declaring main class

public class TrumpWillTriump {

//declaring variables used for the gui including saving and loading

String gameTitle = "Trump Will Triump";

ImageIcon iconTrump = new ImageIcon("Assets/Trump.png");

Path file = Paths.get("TrumpWillTriump.txt");

String[] split;

String saveFile;

//declaring gui variables

private JFrame frame, mainFrame;

private JPanel panel, controlPanel1, controlPanel2, controlPanel3, controlPanel4, controlPanel5;

private JLabel lblTour, headerLabel, subheaderLabel, responseScore, responseAnswer;

private JButton btnTour, btnReset;

private List<Rectangle> cells;

//declaring various variables used for minigames

int[][] stateDisplay = new int[48][64];

int guiDisplay, touringState, wins, loses, check, likeness, scoreScramble, damage, temperature, pressure, rotations, time;

int memoryLevel, memorySequence, memoryCounter, memoryCompletion, skiingCompletion, skiingColour, iowaLocation, recallTries;

int score, click, randomJeopardy, palindromeLength, palindromeScore, palindromeTries = 3;

boolean done, endTimer, next, palindromeTechnical, palindromeActual, palindromeUsed;

String dialogText, userAnswer, dialogOptions[], palindromeWord = "", palindromeCorrect = "";

long startTime, totalTime;

Point selectedCell;

//declaring variable used for the palindrome library; 208 palindromes

String palindromeLibrary[] = {"tattarrattat", "aibohphobia", "detartrated", "kinnikinnik", "deleveled", "evitative", "malayalam",

"redivider", "releveler", "rotavator", "adinida", "deified", "hagigah", "murdrum", "nauruan", "peeweep", "racecar", "reifier",

"repaper", "reviver", "rotator", "seities", "sememes", "senones", "sixaxis", "soosoos", "tacocat", "zerorez", "degged",

"denned", "hallah", "hannah", "mallam", "marram", "pippip", "pullup", "redder", "renner", "revver", "selles", "sesses",

"succus", "terret", "tirrit", "tuttut", "alala", "alula", "arara", "civic", "debed", "deked", "deled", "dered", "dewed",

"dexed", "hamah", "igigi", "irori", "kaiak", "kanak", "kayak", "kazak", "kelek", "level", "liril", "madam", "minim", "neven",

"putup", "radar", "refer", "rotor", "sagas", "semes", "seres", "sexes", "shahs", "sinis", "siris", "solos", "stats", "stets",

"stots", "sulus", "susus", "tenet", "torot", "wakaw", "xanax", "acca", "adda", "affa", "alla", "anna", "beeb", "boob", "deed",

"esse", "goog", "immi", "keek", "kook", "maam", "naan", "noon", "oppo", "otto", "peep", "poop", "sees", "toot", "aba", "aga",

"aha", "ala", "ama", "ana", "ara", "ava", "awa", "bib", "bob", "bub", "dad", "did", "dud", "eke", "eme", "ere", "eve", "ewe",

"eye", "gag", "gig", "gog", "hah", "heh", "huh", "mem", "mim", "mom", "mum", "nan", "non", "nun", "oho", "omo", "ono", "oxo",

"pap", "pep", "pip", "pop", "pup", "sis", "sos", "tat", "tet", "tit", "tot", "tut", "vav", "waw", "wow", "yay", "zuz", "zzz",

"aa", "ee", "mm", "oo", "akasaka", "glenelg", "halalah", "hamamah", "hararah", "ogopogo", "qaanaaq", "eleele", "serres",

"aeaea", "aiaia", "anona", "ardra", "aviva", "capac", "kodok", "laval", "natan", "navan", "noyon", "oruro", "tebet", "tevet",

"tumut", "xenex", "abba", "akka", "amma", "atta", "elle", "ada", "krk", "nen", "a", "lol", "aka", "i"};

//main method

public static void main(String[] args) {

//sending to prepareGUI method

TrumpWillTriump TrumpWillTriump = new TrumpWillTriump();

TrumpWillTriump.prepareGUI();

}

//method used for setting the GUI of the map of the U.S.A.

private void prepareGUI() {

//loading from file and setting variables

load();

wins = parseInt(split[0]);

loses = parseInt(split[1]);

for (int vertical = 0; vertical < 48; vertical++) {

for (int horizontal = 0; horizontal < 64; horizontal++) {

stateDisplay[vertical][horizontal] = parseInt(split[horizontal + vertical \* 64 + 2]);

}

}

//checking the monitor dimensions

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

//setting the gui size

if (screenSize.getWidth() < screenSize.getHeight()) {

guiDisplay = (int) (screenSize.getWidth() \* 0.8);

} else {

guiDisplay = (int) (screenSize.getHeight() \* 0.8);

}

//making guiDisplay a multiple of 64

guiDisplay = (int) (guiDisplay / 64) \* 64;

//making it so you can only play it if your monitor is big enough

if (guiDisplay == 0) {

JOptionPane.showConfirmDialog(null, "Your monitor is too small to play! :(", gameTitle, JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

System.exit(0);

}

//setting the frame title

frame = new JFrame(gameTitle);

//making it so when the x button is pressed the program exits

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//making the frame non-resizable

frame.setResizable(false);

//making the GUI more user-friendly

frame.setLayout(new BorderLayout());

frame.add(new GridPane());

frame.pack();

//setting the row of buttons

panel = new JPanel();

//seting what is displayed from the label

lblTour = new JLabel(" Select a state to tour ");

//setting buttons and what is displayed on them

btnReset = new JButton("Reset");

btnTour = new JButton("Tour");

//setting row of assets to variable panel

panel.add(btnReset);

panel.add(lblTour);

panel.add(btnTour);

//setting panel to the frame

frame.add(panel, BorderLayout.SOUTH);

//setting icon of frame

frame.setIconImage(iconTrump.getImage());

//centering the frame

frame.setLocationRelativeTo(null);

//setting frame to visible

frame.setVisible(true);

//setting what is displayed if the user opens a finished game

if (wins + loses == 50) {

lblTour.setText(" Reset to play again. Republican States: " + wins + " / 50 ");

btnTour.setEnabled(false);

}

//setting what happens when user clicks on reset button

btnReset.addActionListener((ActionEvent e) -> {

frame.setVisible(false);

dialogText = "DO YOU REALLY WANT TO RESET ALL DATA?\nTHIS MEANS THAT ALL DATA WILL BE RESET TO DEFAULT";

dialogOptions = new String[]{"Reset", "Back"};

customText();

if (check == 0) {

try {

Files.delete(file);

} catch (IOException x) {

//useless but needed

}

dialogText = "Application will now close, re-launch\nthe application to play it again.";

dialogOptions = new String[]{"Continue"};

customText();

System.exit(0);

} else {

frame.setVisible(true);

}

});

//setting what happens when user clicks on tour button

btnTour.addActionListener((ActionEvent e) -> {

switch (touringState) {

case 50:

case 68:

case 84:

shooting();

break;

case 52:

case 90:

reactor();

break;

case 61:

case 77:

case 82:

case 83:

sports();

break;

case 51:

case 60:

case 66:

clicking();

break;

case 69:

case 98:

fishing();

break;

case 59:

case 76:

case 78:

case 81:

racing();

break;

case 53:

case 71:

case 96:

memory();

break;

case 89:

case 91:

case 92:

skiing();

break;

case 65:

case 72:

case 80:

recall();

break;

case 64:

sorting();

break;

case 54:

california();

break;

case 85:

newYork();

break;

case 74:

wisconsin();

break;

case 67:

oklahoma();

break;

case 57:

utah();

break;

case 73:

louisiana();

break;

case 62:

newMexico();

break;

case 75:

illinois();

break;

case 94:

massachusetts();

break;

case 58:

arizona();

break;

case 97:

newJersey();

break;

case 93:

newHampshire();

break;

case 56:

nevada();

break;

case 99:

maryland();

break;

case 55:

idaho();

break;

case 88:

virginia();

break;

case 87:

westVirginia();

break;

case 70:

iowa();

break;

case 79:

kentucky();

break;

case 63:

northDakota();

break;

case 95:

rhodeIsland();

break;

case 86:

jeopardyPrepareGUI();

break;

}

});

}

//class used for drawing for the map of the U.S.A.

public class GridPane extends JPanel {

//declaring public used for mouse events

public GridPane() {

//declaring an array list used for grid gui

cells = new ArrayList<>(48 \* 64);

//creating a mouse listener

addMouseListener(new MouseAdapter() {

@Override

//execute when mouse is clicked

public void mouseClicked(MouseEvent e) {

int horizontalClickPosition = (e.getX()) / (getWidth() / 64);

int verticalClickPosition = (e.getY()) / (getHeight() / 48);

if (horizontalClickPosition >= 0 && horizontalClickPosition < 64 && verticalClickPosition >= 0 && verticalClickPosition < 48) {

touringState = stateDisplay[verticalClickPosition][horizontalClickPosition];

switch (touringState) {

case 50:

lblTour.setText(" Tour Alaska? ");

break;

case 51:

lblTour.setText(" Tour Hawaii? ");

break;

case 52:

lblTour.setText(" Tour Washington? ");

break;

case 53:

lblTour.setText(" Tour Oregon? ");

break;

case 54:

lblTour.setText(" Tour California? ");

break;

case 55:

lblTour.setText(" Tour Idaho? ");

break;

case 56:

lblTour.setText(" Tour Nevada? ");

break;

case 57:

lblTour.setText(" Tour Utah? ");

break;

case 58:

lblTour.setText(" Tour Arizona? ");

break;

case 59:

lblTour.setText(" Tour Montana? ");

break;

case 60:

lblTour.setText(" Tour Wyoming? ");

break;

case 61:

lblTour.setText(" Tour Colorado? ");

break;

case 62:

lblTour.setText(" Tour New Mexico? ");

break;

case 63:

lblTour.setText(" Tour North Dakota? ");

break;

case 64:

lblTour.setText(" Tour South Dakota? ");

break;

case 65:

lblTour.setText(" Tour Nebraska? ");

break;

case 66:

lblTour.setText(" Tour Kansas? ");

break;

case 67:

lblTour.setText(" Tour Oklahoma? ");

break;

case 68:

lblTour.setText(" Tour Texas? ");

break;

case 69:

lblTour.setText(" Tour Minnesota? ");

break;

case 70:

lblTour.setText(" Tour Iowa? ");

break;

case 71:

lblTour.setText(" Tour Missouri? ");

break;

case 72:

lblTour.setText(" Tour Arkansas? ");

break;

case 73:

lblTour.setText(" Tour Louisiana? ");

break;

case 74:

lblTour.setText(" Tour Wisconsin? ");

break;

case 75:

lblTour.setText(" Tour Illinois? ");

break;

case 76:

lblTour.setText(" Tour Mississipi? ");

break;

case 77:

lblTour.setText(" Tour Michigan? ");

break;

case 78:

lblTour.setText(" Tour Indiana? ");

break;

case 79:

lblTour.setText(" Tour Kentuky? ");

break;

case 80:

lblTour.setText(" Tour Tennessee? ");

break;

case 81:

lblTour.setText(" Tour Alabama? ");

break;

case 82:

lblTour.setText(" Tour Ohio? ");

break;

case 83:

lblTour.setText(" Tour Georgia? ");

break;

case 84:

lblTour.setText(" Tour Florida? ");

break;

case 85:

lblTour.setText(" Tour New York? ");

break;

case 86:

lblTour.setText(" Tour Pennsylvania? ");

break;

case 87:

lblTour.setText(" Tour West Virginia? ");

break;

case 88:

lblTour.setText(" Tour Virginia? ");

break;

case 89:

lblTour.setText(" Tour North Carolina? ");

break;

case 90:

lblTour.setText(" Tour South Carolina? ");

break;

case 91:

lblTour.setText(" Tour Maine? ");

break;

case 92:

lblTour.setText(" Tour Vermont? ");

break;

case 93:

lblTour.setText(" Tour New Hampshire? ");

break;

case 94:

lblTour.setText(" Tour Massachusetts? ");

break;

case 95:

lblTour.setText(" Tour Rhode Island? ");

break;

case 96:

lblTour.setText(" Tour Connecticut? ");

break;

case 97:

lblTour.setText(" Tour New Jersey? ");

break;

case 98:

lblTour.setText(" Tour Delaware? ");

break;

case 99:

lblTour.setText(" Tour Maryland? ");

break;

}

}

}

});

}

//setting size of the grid gui

@Override

public Dimension getPreferredSize() {

return new Dimension(guiDisplay, (int) (guiDisplay \* 0.75));

}

//protected void used for setting cell colour

@Override

protected void paintComponent(Graphics g) {

//following lines used to determine x and y coordinates

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g.create();

int width = getWidth();

int height = getHeight();

int cellWidth = width / 64;

int cellHeight = height / 48;

if (cells.isEmpty()) {

for (int row = 0; row < 48; row++) {

for (int col = 0; col < 64; col++) {

Rectangle cell = new Rectangle(

+(col \* cellWidth),

+(row \* cellHeight),

cellWidth,

cellHeight);

cells.add(cell);

}

}

}

//2d array used for setting colour of cell

for (int vertical = 0; vertical < 48; vertical++) {

for (int horizontal = 0; horizontal < 64; horizontal++) {

Rectangle cell = cells.get(horizontal + vertical \* 64);

if (stateDisplay[vertical][horizontal] == 0) {

g2d.setColor(Color.WHITE);

} else if (stateDisplay[vertical][horizontal] == 1) {

g2d.setColor(Color.BLACK);

} else if (stateDisplay[vertical][horizontal] >= 150) {

g2d.setColor(Color.RED);

} else if (stateDisplay[vertical][horizontal] < 150 && stateDisplay[vertical][horizontal] >= 100) {

Color color = new Color(30, 144, 255);

g2d.setColor(color);

} else if (stateDisplay[vertical][horizontal] < 100 && stateDisplay[vertical][horizontal] >= 50) {

Color color = new Color(186, 85, 211);

g2d.setColor(color);

}

g2d.fill(cell);

repaint();

}

}

}

}

//method used for setting text when JOptionPane is being used

private void customText() {

check = JOptionPane.showOptionDialog(null, dialogText, gameTitle, JOptionPane.DEFAULT\_OPTION, JOptionPane.PLAIN\_MESSAGE, null, dialogOptions, dialogOptions[0]);

if (check == -1) {

System.exit(0);

}

}

//method used for when the state is won

private void win() {

//incrementing the wins by one

wins += 1;

//changing the state colour to red

for (int vertical = 0; vertical < 48; vertical++) {

for (int horizontal = 0; horizontal < 64; horizontal++) {

if (stateDisplay[vertical][horizontal] == touringState) {

stateDisplay[vertical][horizontal] += 100;

}

}

}

//saving the win to file

save();

//notifying the user that he/she has won

dialogText = "Mr. Trump, you won!";

dialogOptions = new String[]{"Continue"};

customText();

//resetting variables

touringState = likeness = 0;

//checking if the game is finished

if (wins + loses != 50) {

frame.setVisible(true);

} else {

checkGameWin();

}

}

//method used for when the state is lost

private void lose() {

//incrementing the loses by one

loses += 1;

//changing the state colour to blue

for (int vertical = 0; vertical < 48; vertical++) {

for (int horizontal = 0; horizontal < 64; horizontal++) {

if (stateDisplay[vertical][horizontal] == touringState) {

stateDisplay[vertical][horizontal] += 50;

}

}

}

//saving the loss to file

save();

//notifying the user that he/she has lost

dialogText = "Mr. Trump, you lost...";

dialogOptions = new String[]{"Continue"};

customText();

touringState = likeness = 0;

//checking if the game is finished

if (wins + loses != 50) {

frame.setVisible(true);

} else {

checkGameWin();

}

}

//method used for if game is finished

private void checkGameWin() {

if (wins >= 25) {

dialogText = "Mr. Trump, you became president!\nTo play again, click the reset\nbutton on the map of the U.S.A";

customText();

} else {

dialogText = "Mr. Trump, you lost the race to presidency...\nTo play again, click the reset button on the\nmap of the U.S.A";

customText();

}

lblTour.setText(" Reset to play again. Republican States: " + wins + " / 50 ");

btnTour.setEnabled(false);

frame.setVisible(true);

}

//method used for shooting minigame

private void shooting() {

//setting the map to invisible

frame.setVisible(false);

//determining which state the user is playing and setting text to that

switch (touringState) {

case 50:

dialogText = "Mr. Trump, a bear has just appeared!\nShoot it quickly in the head so that it does not run away.\nThis will surely bring us good media coverage in Alaska.";

break;

case 68:

dialogText = "Mr. Trump, a mountain lion has just appeared!\nShoot it quickly in the head so that it does not run away.\nThis will surely bring us good media coverage in Texas.";

break;

case 84:

dialogText = "Mr. Trump, an alligator has just appeared!\nShoot it quickly in the head or body so that it does not run away.\nThis will surely bring us good media coverage in Florida.";

break;

}

dialogOptions = new String[]{"Continue"};

customText();

//making the GUI more user friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

JLabel background = null;

//setting background

switch (touringState) {

case 50:

background = new JLabel(new ImageIcon("Assets/Bear.png"));

frameState.setSize(320, 240);

break;

case 68:

background = new JLabel(new ImageIcon("Assets/MountainLion.png"));

frameState.setSize(323, 265);

break;

case 84:

background = new JLabel(new ImageIcon("Assets/Alligator.png"));

frameState.setSize(356, 225);

break;

}

//more updates to GUI

frameState.add(background);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

//setting gun scope

frameState.setCursor(Toolkit.getDefaultToolkit().createCustomCursor(new ImageIcon("Assets/GunCursor.png").getImage(), new Point(0, 0), ""));

//starting timer and mouse listener

startTime = System.nanoTime();

frameState.addMouseListener(new MouseListener() {

@Override

public void mousePressed(MouseEvent e) {

//setting frame to invisible on shot

frameState.setVisible(false);

//setting hitboxes

switch (touringState) {

case 50:

if (e.getX() > 118 & e.getX() < 138 && e.getY() > 145 && e.getY() < 175) {

dialogText = "Congratulations! You killed it!";

customText();

win();

} else {

dialogText = "Mr. Trump, you missed!";

customText();

lose();

}

break;

case 68:

if (e.getX() > 185 & e.getX() < 215 && e.getY() > 75 && e.getY() < 105) {

dialogText = "Congratulations! You killed it!";

customText();

win();

} else {

dialogText = "Mr. Trump, you missed!";

customText();

lose();

}

break;

case 84:

if (e.getX() > 133 & e.getX() < 182 && e.getY() > 85 && e.getY() < 105) {

dialogText = "Congratulations! You killed it!";

customText();

win();

} else {

dialogText = "Mr. Trump, you missed!";

customText();

lose();

}

break;

}

}

@Override

public void mouseClicked(MouseEvent me) {

//useless but needed

}

@Override

public void mouseReleased(MouseEvent me) {

//useless but needed

}

@Override

public void mouseEntered(MouseEvent me) {

//useless but needed

}

@Override

public void mouseExited(MouseEvent me) {

//useless but needed

}

});

//if user is taking too long, user is notified and loses

frameState.addMouseMotionListener(new MouseMotionListener() {

@Override

public void mouseMoved(MouseEvent me) {

totalTime = (System.nanoTime() - startTime) / 1000000000;

if (totalTime > 5) {

frameState.setVisible(false);

dialogText = "Mr. Trump, you took too long, as such it got startled!";

customText();

lose();

}

}

@Override

public void mouseDragged(MouseEvent me) {

//useless but needed

}

});

}

//method used for reactor minigame

private void reactor() {

//setting variables to default

temperature = 500;

pressure = 250;

rotations = 2500;

time = 600;

//setting map to false

frame.setVisible(false);

//setting text depending on which state the user is playing

if (touringState == 52) {

dialogText = "Mr. Trump, the Grand Coulee Dam is going out of control!\nYou need to operate it while the engine room shuts down.\nThis will take one minute.";

} else {

dialogText = "Mr. Trump, a nuclear powerplant is going out of control\nand our security personnel is nowhere to be found.\nPlease keep the reactor from melting down for one minute!";

}

dialogOptions = new String[]{"Continue"};

customText();

//setting the GUI to be more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

//setting temperature GUI component

JPanel temperaturePanel = new JPanel();

JLabel temperatureLabel = new JLabel("Temperature: 500°C"); //1000°C max

JButton temperatureButton = new JButton("Cool");

JLabel temperatureWarning = new JLabel(new ImageIcon("Assets/ReactorSafe.png")); //800°C or beyond

temperaturePanel.add(temperatureLabel);

temperaturePanel.add(temperatureButton);

temperaturePanel.add(temperatureWarning);

//setting pressure GUI component

JPanel pressurePanel = new JPanel();

JLabel pressureLabel = new JLabel("Pressure: 250kPa "); //500kPa max

JButton pressureButton = new JButton("Blow");

JLabel pressureWarning = new JLabel(new ImageIcon("Assets/ReactorSafe.png")); //400kPa or beyond

pressurePanel.add(pressureLabel);

pressurePanel.add(pressureButton);

pressurePanel.add(pressureWarning);

//setting rotations GUI component

JPanel rotationsPanel = new JPanel();

JLabel rotationsLabel = new JLabel("Rotations: 2500rpm"); //5000rpm max

JButton rotationsButton = new JButton("Slow");

JLabel rotationsWarning = new JLabel(new ImageIcon("Assets/ReactorSafe.png")); //4000rpm or beyond

rotationsPanel.add(rotationsLabel);

rotationsPanel.add(rotationsButton);

rotationsPanel.add(rotationsWarning);

//setting time GUI component

JPanel timePanel = new JPanel();

JLabel timeLabel = new JLabel("Time Until System Restore: 60 seconds");

timePanel.add(timeLabel);

//adding GUI components to frame

frameState.add(temperaturePanel);

frameState.add(pressurePanel);

frameState.add(rotationsPanel);

frameState.add(timePanel);

frameState.setLayout(new GridLayout(4, 1));

//making GUI more user-friendly

frameState.pack();

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

//lower temperature if button is pressed

temperatureButton.addActionListener((ActionEvent e) -> {

if (temperature > 40) {

temperature -= 20;

} else {

temperature = 20;

}

});

//lower pressure if button is pressed

pressureButton.addActionListener((ActionEvent e) -> {

if (pressure > 10) {

pressure -= 10;

} else {

pressure = 0;

}

});

//lower rotations if button is pressed

rotationsButton.addActionListener((ActionEvent e) -> {

if (rotations > 100) {

rotations -= 100;

} else {

rotations = 0;

}

});

//creating a timer

Timer timer = new Timer();

timer.schedule(new TimerTask() {

@Override

public void run() {

//increase variables after timer executes periodically

if ((int) (Math.random() \* 8) != 0) {

temperature += 4;

}

if ((int) (Math.random() \* 8) != 0) {

pressure += 2;

}

if ((int) (Math.random() \* 8) != 0) {

rotations += 20;

}

time -= 1;

//setting text displayed on frame

temperatureLabel.setText("Temperature: " + temperature + "°C");

pressureLabel.setText("Pressure: " + pressure + "kPa ");

rotationsLabel.setText("Rotations: " + rotations + "rpm");

timeLabel.setText("Time Until System Restore: " + (int) (time / 10) + " seconds");

//if temperature gets into dangerous area, red warning is displayed

if (temperature >= 800) {

temperatureWarning.setIcon(new ImageIcon("Assets/ReactorDangerous.png"));

} else {

temperatureWarning.setIcon(new ImageIcon("Assets/ReactorSafe.png"));

}

//if pressure gets into dangerous area, red warning is displayed

if (pressure >= 400) {

pressureWarning.setIcon(new ImageIcon("Assets/ReactorDangerous.png"));

} else {

pressureWarning.setIcon(new ImageIcon("Assets/ReactorSafe.png"));

}

//if rotations gets into dangerous area, red warning is displayed

if (rotations >= 4000) {

rotationsWarning.setIcon(new ImageIcon("Assets/ReactorDangerous.png"));

} else {

rotationsWarning.setIcon(new ImageIcon("Assets/ReactorSafe.png"));

}

//for if user wins or loses game

if (temperature >= 1000 || pressure >= 500 || rotations >= 5000) {

timer.cancel();

frameState.setVisible(false);

if (touringState == 52) {

dialogText = "Evacuate! The dam is going to explode!";

} else {

dialogText = "Evacuate! A meltdown will soon occur!";

}

customText();

lose();

} else if (time <= 1) {

timer.cancel();

frameState.setVisible(false);

if (touringState == 52) {

dialogText = "Mr. Trump, the hydroelectric dam successfully shut down!";

} else {

dialogText = "Mr. Trump, the powerplant successfully shut down!";

}

customText();

win();

}

}

}, 0, 100);

}

//method used for sports minigame

private void sports() {

//setting map to invisible

frame.setVisible(false);

//which text to be displayed is dependant upon which state user is playing as

switch (touringState) {

case 61:

dialogText = "Mr. Trump, Colorado likes hockey. To gain their vote, get\na ball in net on a free shot. To do this click swipe and release.";

break;

case 77:

dialogText = "Mr. Trump, Michigan likes soccer. To gain their vote, get\na ball in net on free kick. To do this click swipe and release.";

break;

case 82:

dialogText = "Mr. Trump, Ohio likes football. To gain their vote, get\na ball in goal. To do this click swipe and release.";

break;

case 83:

dialogText = "Mr. Trump, Georgia likes basketball. To gain their vote, get\na ball in net on free throw. To do this click swipe and release.";

break;

}

dialogOptions = new String[]{"Continue"};

customText();

//making GUI more user-friendy

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

JLabel background = null;

//setting the background and the mouse cursor to be displayed

switch (touringState) {

case 61:

background = new JLabel(new ImageIcon("Assets/HockeyNet.png"));

frameState.setSize(303, 325);

frameState.setCursor(Toolkit.getDefaultToolkit().createCustomCursor(new ImageIcon("Assets/HockeyPuck.png").getImage(), new Point(0, 0), ""));

break;

case 77:

background = new JLabel(new ImageIcon("Assets/SoccerBackground.png"));

frameState.setSize(303, 325);

frameState.setCursor(Toolkit.getDefaultToolkit().createCustomCursor(new ImageIcon("Assets/SoccerBall.png").getImage(), new Point(0, 0), ""));

break;

case 82:

background = new JLabel(new ImageIcon("Assets/FootballPost.png"));

frameState.setSize(303, 325);

frameState.setCursor(Toolkit.getDefaultToolkit().createCustomCursor(new ImageIcon("Assets/Football.png").getImage(), new Point(0, 0), ""));

break;

case 83:

background = new JLabel(new ImageIcon("Assets/BasketballBackground.png"));

frameState.setSize(203, 325);

frameState.setCursor(Toolkit.getDefaultToolkit().createCustomCursor(new ImageIcon("Assets/Basketball.png").getImage(), new Point(0, 0), ""));

break;

}

//making GUI more user-friendly

frameState.add(background);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

//initiating mouse listener

frameState.addMouseListener(new MouseListener() {

int startX, startY, endX, endY;

double vectorX, vectorY;

boolean win;

@Override

public void mousePressed(MouseEvent e) {

//determining start positions

startX = e.getX();

startY = e.getY();

}

@Override

public void mouseReleased(MouseEvent me) {

//determining end positions

endX = me.getX();

endY = me.getY();

//determining the stepping amount

vectorX = (endX - startX) / 100;

vectorY = (endY - startY) / 100;

//declaring hit boxes for liniar recognition of stepping amount

if (vectorX != 0 || vectorY != 0) {

while (startX > 0 && startX < 300 && startY > 0 && startY < 300) {

startX += vectorX;

startY += vectorY;

switch (touringState) {

case 61:

if (startX > 65 && startX < 225 && startY > 20 && startY < 110) {

win = true;

}

break;

case 77:

if (startX > 0 && startX < 300 && startY > 20 && startY < 130) {

win = true;

}

break;

case 82:

if (startX > 90 && startX < 200 && startY > 10 && startY < 90) {

win = true;

}

break;

case 83:

if (startX > 65 && startX < 105 && startY > 35 && startY < 95) {

win = true;

}

break;

}

}

}

//notifying user of if he/she missed or scored

if (win == true) {

frameState.setVisible(false);

dialogText = "Mr. Trump, you scored!";

customText();

win();

} else {

frameState.setVisible(false);

dialogText = "Mr. Trump, you missed...";

customText();

lose();

}

}

@Override

public void mouseClicked(MouseEvent me) {

//useless but needed

}

@Override

public void mouseEntered(MouseEvent me) {

//useless but needed

}

@Override

public void mouseExited(MouseEvent me) {

//useless but needed

}

});

}

//method used for clicking minigame

private void clicking() {

//setting map to invisible

frame.setVisible(false);

//which text to be displayed is dependant upon which state user is playing as

switch (touringState) {

case 51:

dialogText = "Mr. Trump, Hawaii is a tropical state, so\nto win it, you must break open a coconut\nby repeatedly left clicking on it.";

break;

case 60:

dialogText = "Mr. Trump, Wyoming is a farming state, so\nto win it, you must collect a chicken's\neggs. To do so, repeatedly left click on it.";

break;

case 66:

dialogText = "Mr. Trump, Kansas is a farming state,\nso to win it, you must milk a cow.\nTo do so, repeatedly left click on it.";

break;

}

dialogOptions = new String[]{"Continue"};

customText();

//making GUI more user-friendy

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

JLabel background = null;

//setting the background to be displayed

switch (touringState) {

case 51:

background = new JLabel(new ImageIcon("Assets/Coconut.png"));

frameState.setSize(228, 250);

break;

case 60:

background = new JLabel(new ImageIcon("Assets/Chicken.png"));

frameState.setSize(185, 139);

break;

case 66:

background = new JLabel(new ImageIcon("Assets/Cow.png"));

frameState.setSize(223, 109);

break;

}

//making the GUI more user-friendly

frameState.add(background);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

startTime = System.nanoTime();

//initializing mouse listener

frameState.addMouseListener(new MouseListener() {

int clickCount;

@Override

public void mousePressed(MouseEvent e) {

if (e.getModifiers() == InputEvent.BUTTON1\_MASK) {

//user has 10 seconds to click 30 times

clickCount += 1;

if (clickCount == 30) {

totalTime = (System.nanoTime() - startTime) / 1000000000;

frameState.setVisible(false);

if (totalTime <= 10) {

win();

} else {

lose();

}

}

}

}

@Override

public void mouseClicked(MouseEvent me) {

//useless but needed

}

@Override

public void mouseReleased(MouseEvent me) {

//useless but needed

}

@Override

public void mouseEntered(MouseEvent me) {

//useless but needed

}

@Override

public void mouseExited(MouseEvent me) {

//useless but needed

}

});

}

//method used for fishing minigame

private void fishing() {

//setting map to invisible

frame.setVisible(false);

//which text to be displayed is dependant upon which state user is playing as

if (touringState == 69) {

dialogText = "Mr. Trump, ice fishing would greatly improve your success in\nMinnesota. Left click to throw your line out, right click to bring\nit back in. You only get one cast. The longer your line is\nout, the higher chance you have of catching something.";

} else {

dialogText = "Mr. Trump, fishing would greatly improve your success in\nDelaware. Left click to throw your line out, right click to bring\nit back in. You only get one cast. The longer your line is\nout, the higher chance you have of catching something.";

}

dialogOptions = new String[]{"Continue"};

customText();

//making GUI more user-friendy

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

JLabel background;

//setting background depending on which state the user is playing

if (touringState == 69) {

background = new JLabel(new ImageIcon("Assets/IceFishing.png"));

frameState.setSize(353, 258);

} else {

background = new JLabel(new ImageIcon("Assets/Fishing.png"));

frameState.setSize(353, 287);

}

//making GUI more user-friendly

frameState.add(background);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

//setting the cursor to a fishing rod

frameState.setCursor(Toolkit.getDefaultToolkit().createCustomCursor(new ImageIcon("Assets/FishingCursor.png").getImage(), new Point(0, 0), ""));

//initializing mouse listener

frameState.addMouseListener(new MouseListener() {

@Override

public void mousePressed(MouseEvent e) {

//catching a fish is randomly inversly proportional to the time the fishing rod is out

if (e.getModifiers() == InputEvent.BUTTON1\_MASK) {

startTime = System.nanoTime();

} else if (e.getModifiers() == InputEvent.BUTTON3\_MASK) {

totalTime = (System.nanoTime() - startTime) / 1000000000;

int fish = (int) (Math.random() \* 30 / totalTime);

if (fish == 0) {

frameState.setVisible(false);

dialogText = "Mr. Trump, you caught a fish!";

customText();

win();

} else {

frameState.setVisible(false);

dialogText = "Mr. Trump, you didn't catch anything...";

customText();

lose();

}

}

}

@Override

public void mouseClicked(MouseEvent me) {

//useless but needed

}

@Override

public void mouseReleased(MouseEvent me) {

//useless but needed

}

@Override

public void mouseEntered(MouseEvent me) {

//useless but needed

}

@Override

public void mouseExited(MouseEvent me) {

//useless but needed

}

});

}

//method used for racing minigame

private void racing() {

//setting map to invisible

frame.setVisible(false);

//which text to be displayed is dependant upon which state user is playing as

switch (touringState) {

case 59:

dialogText = "Mr. Trump, Montana loves horse racing, so to win their vote,\nbet on a horse. This will be during a four horse race.";

break;

case 76:

dialogText = "Mr. Trump, Mississippi loves boat racing, so to win their vote,\nbet on a boat. This will be during a four boat race.";

break;

case 78:

dialogText = "Mr. Trump, Indiana loves Formula 1, so to win their vote,\nbet on a car. This will be during a four car race.";

break;

case 81:

dialogText = "Mr. Trump, Alabama loves Nascar, so to win their vote,\nbet on a car. This will be during a four car race.";

break;

}

dialogOptions = new String[]{"Continue"};

customText();

//making GUI more user-friendy

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

//declaring GUI componenents

JLabel lblInfo = new JLabel("Select What To Bet On - Success Chance");

JButton btnHorse87 = new JButton("Horse 87 - 12%");

JButton btnHorse17 = new JButton("Horse 17 - 9%");

JButton btnHorse65 = new JButton("Horse 65 - 7%");

JButton btnHorse37 = new JButton("Horse 37 - 72%");

JButton btnBoat12 = new JButton("Boat 12 - 64%");

JButton btnBoat16 = new JButton("Boat 16 - 3%");

JButton btnBoat56 = new JButton("Boat 56 - 23%");

JButton btnBoat37 = new JButton("Boat 37 - 10%");

JButton btnCar71 = new JButton("Car 71 - 21%");

JButton btnCar87 = new JButton("Car 87 - 56%");

JButton btnCar12 = new JButton("Car 12 - 8%");

JButton btnCar36 = new JButton("Car 36 - 15%");

JButton btnCar45 = new JButton("Car 45 - 18%");

JButton btnCar11 = new JButton("Car 11 - 14%");

JButton btnCar23 = new JButton("Car 23 - 62%");

JButton btnCar73 = new JButton("Car 73 - 6%");

//setting buttons depending on which state user is playing as

JPanel buttonPanel1 = new JPanel();

JPanel buttonPanel2 = new JPanel();

switch (touringState) {

case 59:

buttonPanel1.add(btnHorse87);

buttonPanel1.add(btnHorse17);

buttonPanel2.add(btnHorse65);

buttonPanel2.add(btnHorse37);

break;

case 76:

buttonPanel1.add(btnBoat12);

buttonPanel1.add(btnBoat16);

buttonPanel2.add(btnBoat56);

buttonPanel2.add(btnBoat37);

break;

case 78:

buttonPanel1.add(btnCar45);

buttonPanel1.add(btnCar11);

buttonPanel2.add(btnCar23);

buttonPanel2.add(btnCar73);

break;

case 81:

buttonPanel1.add(btnCar71);

buttonPanel1.add(btnCar87);

buttonPanel2.add(btnCar12);

buttonPanel2.add(btnCar36);

break;

}

//making GUI more user-friendly

frameState.add(lblInfo, BorderLayout.NORTH);

frameState.add(buttonPanel1, BorderLayout.CENTER);

frameState.add(buttonPanel2, BorderLayout.SOUTH);

frameState.pack();

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

btnHorse87.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 8.33) == 0) {

win();

} else {

lose();

}

});

btnHorse17.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 11.11) == 0) {

win();

} else {

lose();

}

});

btnHorse65.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 14.29) == 0) {

win();

} else {

lose();

}

});

btnHorse37.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 1.39) == 0) {

win();

} else {

lose();

}

});

btnBoat12.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 1.56) == 0) {

win();

} else {

lose();

}

});

btnBoat16.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 33.33) == 0) {

win();

} else {

lose();

}

});

btnBoat56.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 4.35) == 0) {

win();

} else {

lose();

}

});

btnBoat37.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 10) == 0) {

win();

} else {

lose();

}

});

btnCar45.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 5.56) == 0) {

win();

} else {

lose();

}

});

btnCar11.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 7.14) == 0) {

win();

} else {

lose();

}

});

btnCar23.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 1.61) == 0) {

win();

} else {

lose();

}

});

btnCar73.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 16.67) == 0) {

win();

} else {

lose();

}

});

btnCar71.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 4.76) == 0) {

win();

} else {

lose();

}

});

btnCar87.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 1.79) == 0) {

win();

} else {

lose();

}

});

btnCar12.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 12.5) == 0) {

win();

} else {

lose();

}

});

btnCar36.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if ((int) (Math.random() \* 6.67) == 0) {

win();

} else {

lose();

}

});

}

//method used for memory minigame

private void memory() {

//setting variables to default

done = false;

memoryLevel = memorySequence = memoryCounter = memoryCompletion = 0;

int memoryArray[] = new int[5];

//setting map to invisible

frame.setVisible(false);

//notifying user about the game

dialogText = "Mr. Trump, to test your memory, you shall play a memory game!\nTo play, just hit the arrow keys on your keyboard when propted to type.";

dialogOptions = new String[]{"Continue"};

customText();

//making GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

JLabel image = new JLabel(new ImageIcon("Assets/MemoryGo.png"));

frameState.add(image);

frameState.setSize(303, 325);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

//initializing key listener

frameState.addKeyListener(new KeyListener() {

@Override

public void keyPressed(KeyEvent e) {

//if user gets it wrong he/she loses, if right, either goes to next round or wins

if (done == true) {

if (e.getKeyCode() == memoryArray[memorySequence]) {

if (memorySequence != 4) {

if (memorySequence == memoryLevel + 1) {

memorySequence = memoryCounter = 0;

memoryCompletion += 1;

} else {

memorySequence += 1;

}

} else {

endTimer = true;

frameState.setVisible(false);

win();

}

} else {

endTimer = true;

frameState.setVisible(false);

lose();

}

}

}

@Override

public void keyTyped(KeyEvent e) {

//useless but needed

}

@Override

public void keyReleased(KeyEvent e) {

//useless but needed

}

});

//creating new timer

Timer timer = new Timer();

timer.schedule(new TimerTask() {

@Override

public void run() {

//what is done to end the timer

if (endTimer == true) {

timer.cancel();

endTimer = false;

}

//displaying a round to the user

if (memoryLevel == memoryCompletion && (memoryLevel == 0 || memoryLevel == 1 || memoryLevel == 2)) {

if (memoryCounter < memoryLevel + 3) {

done = false;

if (memoryCounter != 0) {

do {

memoryArray[memoryCounter] = (int) (Math.random() \* 4) + 37;

} while (memoryArray[memoryCounter] == memoryArray[memoryCounter - 1]);

} else {

memoryArray[memoryCounter] = (int) (Math.random() \* 4) + 37;

}

switch (memoryArray[memoryCounter]) {

case 37:

image.setIcon(new ImageIcon("Assets/MemoryLeft.png"));

break;

case 38:

image.setIcon(new ImageIcon("Assets/MemoryUp.png"));

break;

case 39:

image.setIcon(new ImageIcon("Assets/MemoryRight.png"));

break;

case 40:

image.setIcon(new ImageIcon("Assets/MemoryDown.png"));

break;

}

memoryCounter += 1;

} else if (memoryCounter == memoryLevel + 3) {

image.setIcon(new ImageIcon("Assets/MemoryGo.png"));

memoryLevel += 1;

done = true;

}

}

}

}, 0, 1000);

}

//method used for skiing minigame

private void skiing() {

//setting variables to default

done = false;

next = true;

skiingCompletion = 0;

//setting map to invisible

frame.setVisible(false);

//notifying user about the game depending on which state the user is playing as

switch (touringState) {

case 89:

dialogText = "Mr. Trump, North Carolina loves water skiing; therefore,\nsuch is what you should do to gain their vote. Type the left\nkey when red is shown and right when blue is shown.\nYou will need to immediately use your keyboard!";

break;

case 91:

dialogText = "Mr. Trump, Maine loves tubing; therefore, such is what\nyou should do to gain their vote. Type the left key\nwhen red is shown and right when blue is shown.\nYou will need to immediately use your keyboard!";

break;

case 92:

dialogText = "Mr. Trump, Vermont loves downhill skiing; therefore,\nsuch is what you should do to gain their vote. Type the left\nkey when red is shown and right when blue is shown.\nYou will need to immediately use your keyboard!";

break;

}

dialogOptions = new String[]{"Continue"};

customText();

//making GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

JLabel image = new JLabel(new ImageIcon("Assets/SkiingRed.png"));

frameState.add(image);

frameState.setSize(303, 325);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

//initializing new key listener

frameState.addKeyListener(new KeyListener() {

@Override

public void keyPressed(KeyEvent e) {

//what is done when user clicks on a key

if (done == true) {

if (skiingCompletion < 10) {

if (e.getKeyCode() == skiingColour) {

skiingCompletion += 1;

done = false;

next = true;

if (Math.random() \* 3 < 1) {

image.setIcon(new ImageIcon("Assets/SkiingSuper.png"));

} else if (Math.random() \* 3 < 1) {

image.setIcon(new ImageIcon("Assets/SkiingGreat.png"));

} else {

image.setIcon(new ImageIcon("Assets/SkiingGood.png"));

}

} else {

next = endTimer = true;

frameState.setVisible(false);

lose();

}

} else {

next = endTimer = true;

frameState.setVisible(false);

win();

}

}

}

@Override

public void keyTyped(KeyEvent e) {

//useless but needed

}

@Override

public void keyReleased(KeyEvent e) {

//useless but needed

}

});

//initializing new timer

Timer timer = new Timer();

timer.schedule(new TimerTask() {

@Override

public void run() {

//how to end the timer

if (endTimer == true) {

timer.cancel();

endTimer = false;

}

//showing to the user one of the two options

if (next == true) {

done = true;

next = false;

skiingColour = ((int) (Math.random() + 0.5)) \* 2 + 37;

if (skiingColour == 37) {

image.setIcon(new ImageIcon("Assets/SkiingRed.png"));

} else {

image.setIcon(new ImageIcon("Assets/SkiingBlue.png"));

}

} else {

timer.cancel();

endTimer = false;

frameState.setVisible(false);

lose();

}

}

}, 0, 2000);

}

//method used for recall minigame

private void recall() {

//setting variables to default

int recallArray[] = new int[4];

recallTries = 10;

//setting map to invisible

frame.setVisible(false);

//telling user about the game

dialogText = "Mr. Trump, to test your skill, play a logic game. There will be\nfour numbers that are randomly generated but you are not\ntold the numbers. Use the sliders to guess the numbers, but\nto help you, you will be told how many correct and how many\nincorrect numbers you have selected.";

dialogOptions = new String[]{"Continue"};

customText();

//generating random numbers

for (int counter = 0; counter < 4; counter++) {

recallArray[counter] = (int) (Math.random() \* 4) + 1;

}

//making GUI more user-friendly

JFrame frameState = new JFrame("TWT");

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

//creating slider array

JSlider slider[] = new JSlider[4];

//creating four sliders with same properties

for (int counter = 0; counter < 4; counter++) {

slider[counter] = new JSlider(JSlider.HORIZONTAL, 1, 4, 1);

slider[counter].setPaintLabels(true);

slider[counter].setMajorTickSpacing(1);

slider[counter].setPreferredSize(new Dimension(150, 40));

}

//creating GUI components

JLabel correct = new JLabel("Correct");

JLabel incorrect = new JLabel("Incorrect");

JLabel tries = new JLabel("Tries Left: 10");

JButton button = new JButton("Check");

//creating grid layout

frameState.setLayout(new GridLayout(8, 1));

//adding GUI components to frame

for (int counter = 0; counter < 4; counter++) {

frameState.add(slider[counter]);

}

frameState.add(correct);

frameState.add(incorrect);

frameState.add(tries);

frameState.add(button);

//making GUI more user-friendly

frameState.pack();

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

//initializing what happens when button is clicked

button.addActionListener((ActionEvent e) -> {

//setting variables to default

int good = 0, bad = 0;

recallTries -= 1;

//displaying to user the amount of correct and incorrect

for (int counter = 0; counter < 4; counter++) {

if (slider[counter].getValue() == recallArray[counter]) {

good += 1;

} else {

bad += 1;

}

}

//determining if user wins loses or goes to next round

if (good != 4 && recallTries >= 0) {

correct.setText("Correct: " + good);

incorrect.setText("Incorrect: " + bad);

tries.setText("Tries Left: " + recallTries);

} else if (good == 4 && recallTries >= -1) {

frameState.setVisible(false);

win();

} else {

frameState.setVisible(false);

lose();

}

});

}

//method used for sorting minigame

private void sorting() {

//setting variables to default

String sortingLine;

int sortingArray[] = new int[5];

int temp;

//making map invisible

frame.setVisible(false);

//giving user information about the minigame

dialogText = "Mr. Trump, to test your skill, given the steps produced by\na random sorting algorithm, pick which one was used.";

dialogOptions = new String[]{"Continue"};

customText();

//randomly generating numbers to be sorted

for (int counter = 0; counter < 5; counter++) {

sortingArray[counter] = (int) (Math.random() \* 100);

}

//adding pre-sorted numbers to variable

sortingLine = sortingArray[0] + " " + sortingArray[1] + " " + sortingArray[2] + " " + sortingArray[3] + " " + sortingArray[4];

//randomly selecting which sorting type to use

int sortingType = (int) (Math.random() \* 3);

switch (sortingType) {

case 0:

for (int i = 0; i < 5; i++) {

for (int j = 1; j < (5 - i); j++) {

if (sortingArray[j - 1] > sortingArray[j]) {

temp = sortingArray[j - 1];

sortingArray[j - 1] = sortingArray[j];

sortingArray[j] = temp;

sortingLine += " → " + sortingArray[0] + " " + sortingArray[1] + " " + sortingArray[2] + " " + sortingArray[3] + " " + sortingArray[4];

}

}

}

break;

case 1:

for (int counter1 = 0; counter1 < 5 - 1; counter1++) {

int minIndex = counter1;

for (int counter2 = counter1 + 1; counter2 < 5; counter2++) {

if (sortingArray[minIndex] > sortingArray[counter2]) {

minIndex = counter2;

}

}

if (minIndex != counter1) {

temp = sortingArray[counter1];

sortingArray[counter1] = sortingArray[minIndex];

sortingArray[minIndex] = temp;

}

sortingLine += " → " + sortingArray[0] + " " + sortingArray[1] + " " + sortingArray[2] + " " + sortingArray[3] + " " + sortingArray[4];

}

break;

case 2:

int counter1,

counter2,

newValue;

for (counter1 = 1; counter1 < 5; counter1++) {

newValue = sortingArray[counter1];

counter2 = counter1;

while (counter2 > 0 && sortingArray[counter2 - 1] > newValue) {

sortingArray[counter2] = sortingArray[counter2 - 1];

counter2--;

}

sortingArray[counter2] = newValue;

sortingLine += " → " + sortingArray[0] + " " + sortingArray[1] + " " + sortingArray[2] + " " + sortingArray[3] + " " + sortingArray[4];

}

break;

}

//making GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

//creating GUI components

JLabel label = new JLabel(sortingLine);

JButton bubbleSort = new JButton("Bubble Sort");

JButton selectionSort = new JButton("Selection Sort");

JButton insertionSort = new JButton("InsertionSort");

//adding GUI components to panel

JPanel sortingPanel = new JPanel();

sortingPanel.add(bubbleSort);

sortingPanel.add(selectionSort);

sortingPanel.add(insertionSort);

//making GUI more user-friendly

frameState.setLayout(new GridLayout(2, 1));

frameState.add(label);

frameState.add(sortingPanel);

frameState.pack();

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

//determining if user's selection is correct

bubbleSort.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if (sortingType == 0) {

win();

} else {

lose();

}

});

selectionSort.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if (sortingType == 1) {

win();

} else {

lose();

}

});

insertionSort.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

if (sortingType == 2) {

win();

} else {

lose();

}

});

}

//method used for california

private void california() {

//making map invisible

frame.setVisible(false);

//telling user about the minigame

dialogText = "Mr. Trump, California loves surfing, so to win their vote, just left\nclick on the surf board as fast as you can to clean it! If you do\nit fast enough, they surely will see us eye to eye!";

dialogOptions = new String[]{"Continue"};

customText();

//making the GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

JLabel background = new JLabel(new ImageIcon("Assets/Surfboard1.png"));

frameState.add(background);

frameState.setSize(303, 114);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

startTime = System.nanoTime();

//initializing mouse listener

frameState.addMouseListener(new MouseListener() {

int clickCount;

@Override

public void mousePressed(MouseEvent e) {

if (e.getModifiers() == InputEvent.BUTTON1\_MASK) {

//setting image depending on the amount of left mouse clicks

if (clickCount >= 10 && clickCount < 20) {

background.setIcon(new ImageIcon("Assets/Surfboard2.png"));

} else if (clickCount >= 20 && clickCount < 30) {

background.setIcon(new ImageIcon("Assets/Surfboard3.png"));

} else if (clickCount >= 30 && clickCount < 40) {

background.setIcon(new ImageIcon("Assets/Surfboard4.png"));

} else if (clickCount >= 40 && clickCount < 41) {

background.setIcon(new ImageIcon("Assets/Surfboard5.png"));

} else if (clickCount == 41) {

frameState.setVisible(false);

totalTime = (System.nanoTime() - startTime) / 1000000000;

if (totalTime < 10) {

win();

} else {

lose();

}

}

if (clickCount <= 41) {

clickCount += 1;

}

}

}

@Override

public void mouseClicked(MouseEvent me) {

//useless but needed

}

@Override

public void mouseReleased(MouseEvent me) {

//useless but needed

}

@Override

public void mouseEntered(MouseEvent me) {

//useless but needed

}

@Override

public void mouseExited(MouseEvent me) {

//useless but needed

}

});

}

//method used for new york

private void newYork() {

//making map invisible

frame.setVisible(false);

//asking the user various questions and adding points to likeness variable depending on answer

dialogText = "Hello, Mr. Trump. It is a pleasure to have you on this show.";

dialogOptions = new String[]{"I know", "Thank you"};

customText();

if (check == 0) {

likeness -= 1;

}

dialogText = "Mr. Trump, if you were to become president,\nwhat is the first thing that you would do?";

dialogOptions = new String[]{"Build wall", "Export immigrants"};

customText();

if (check == 0) {

likeness -= 1;

}

dialogText = "Mr. Trump, as I'm sure you are aware of, our state is the one in which the\ntwin tower attacks occured. What do you have to say on this matter?";

dialogOptions = new String[]{"I would rebuild them!", "I was on 7/11 when it happened!"};

customText();

if (check == 0) {

likeness += 3;

} else if (check == 1) {

likeness -= 1;

}

dialogText = "For anybody that is on the fence, what would you say the biggest thing going for you is?";

dialogOptions = new String[]{"I'm really rich", "I'm almost completely self-funded"};

customText();

if (check == 0) {

likeness -= 1;

} else if (check == 1) {

likeness += 1;

}

if (likeness >= 3) {

win();

} else {

lose();

}

}

//method used for wisconsin

private void wisconsin() {

//making map invisible

frame.setVisible(false);

//telling user about the minigame

dialogText = "Mr. Trump, Wisconsin loves making cheese, to win\ntheir vote, just left click and churn the cheese!";

dialogOptions = new String[]{"Continue"};

customText();

//making the GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

JLabel background = new JLabel(new ImageIcon("Assets/Churn1.png"));

frameState.add(background);

frameState.setSize(222, 346);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

startTime = System.nanoTime();

//initializing mouse listener

frameState.addMouseListener(new MouseListener() {

int clickCount;

@Override

public void mousePressed(MouseEvent e) {

if (e.getModifiers() == InputEvent.BUTTON1\_MASK) {

//setting image depending on the amount of left mouse clicks

if (clickCount < 62) {

clickCount += 1;

}

if (clickCount > 0 && clickCount <= 30) {

background.setIcon(new ImageIcon("Assets/Churn" + clickCount + ".png"));

} else if (clickCount > 30 && clickCount <= 60) {

background.setIcon(new ImageIcon("Assets/Churn" + (clickCount - 30) + ".png"));

} else {

frameState.setVisible(false);

totalTime = (System.nanoTime() - startTime) / 1000000000;

if (totalTime < 15) {

win();

} else {

lose();

}

}

}

}

@Override

public void mouseClicked(MouseEvent me) {

//useless but needed

}

@Override

public void mouseReleased(MouseEvent me) {

//useless but needed

}

@Override

public void mouseEntered(MouseEvent me) {

//useless but needed

}

@Override

public void mouseExited(MouseEvent me) {

//useless but needed

}

});

}

//method used for oklahoma

private void oklahoma() {

//declaring variable

String word;

//making map invisble

frame.setVisible(false);

//letting user know information about the state

word = JOptionPane.showInputDialog(null, "Mr. Trump, Oklahoma is considered the least intelligent state,\nI'm sure that if we just wrote a sponsered tweet in their area,\nthey would vote for us since most of them are republican anyways.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

if (word == null) {

System.exit(0);

}

//must be between 1 and 140 characters

while (word.length() > 140 || word.length() == 0) {

if (word.length() == 0) {

word = JOptionPane.showInputDialog(null, "You must write something.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

} else if (word.length() > 140) {

word = JOptionPane.showInputDialog(null, "Length cannot exceed 140 characters.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

}

if (word == null) {

System.exit(0);

}

}

//90% chance that user wins

if ((int) (Math.random() \* 10) != 0) {

win();

} else {

lose();

}

}

//method used for utah

private void utah() {

//variable used to let user know correct answer or if answer is correct

String answer;

//making map invisible

frame.setVisible(false);

//asking the user various questions and adding points to likeness variable depending on answer

dialogText = "Mr. Trump, I am the arch-bishop of Utah, and I will ask you\na few questions to see if you are worthy voting for.";

dialogOptions = new String[]{"Ok", "God bless America"};

customText();

if (check == 1) {

likeness += 1;

}

dialogText = "In the Bible, after resting on Mount Ararat for 40 days in\nthe ark, what type of bird did Noah first send out?";

dialogOptions = new String[]{"Raven", "Dove", "Seagle", "Eagle"};

customText();

if (check == 0) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is a raven.\n";

}

dialogText = answer + "In the Bible, how long were Moses and his followers in the desert for?";

dialogOptions = new String[]{"40 years 12 days", "12 years", "40 years", "40 days"};

customText();

if (check == 2) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is 40 years.\n";

}

dialogText = answer + "In the Bible, what was written above Jesus's cross when he was nailed to it?";

dialogOptions = new String[]{"JNKJ", "JCKJ", "ICNR", "INRI"};

customText();

if (check == 3) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is INRI.\n";

}

dialogText = answer + "In the Bible, the eating of what is known as original sin?";

dialogOptions = new String[]{"Carrot", "Fruit", "Apple", "Plum"};

customText();

if (check == 1) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is a fruit. The fruit isn't actually specified.\n";

}

dialogText = answer + "Thank you Mr. Trump, it was a pleasure.";

dialogOptions = new String[]{"I know", "Thank you too"};

customText();

if (likeness >= 4) {

win();

} else {

lose();

}

}

//method used for louisiana

private void louisiana() {

//variable used to let user know correct answer or if answer is correct

String answer;

//making map invisible

frame.setVisible(false);

//asking the user various questions and adding points to likeness variable depending on answer

dialogText = "Mr. Trump, since Louisiana is a French state, we will ask you a\nfew questions to test if you are worth voting for.";

dialogOptions = new String[]{"Ok", "D'accord"};

customText();

if (check == 1) {

likeness += 1;

}

dialogText = "How do you say 'hot air balloon' in French?";

dialogOptions = new String[]{"Montgolfière", "Balon à air chaude", "Iritaboulis", "Balon a air chaud"};

customText();

if (check == 0) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is 'montgolfière'.\n";

}

dialogText = answer + "How do you say 'poppy' in French?";

dialogOptions = new String[]{"Poppile", "Coquelicot", "Popèle", "Aragondessieux"};

customText();

if (check == 1) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is 'coquelicot'.\n";

}

dialogText = answer + "What error is in the following sentence: 'La dame va à le salle de bain'?";

dialogOptions = new String[]{"Conjugation", "Spelling", "Gender", "According"};

customText();

if (check == 2) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is gender.\n";

}

dialogText = answer + "Thank you Mr. Trump, ce fut un plaisir.";

dialogOptions = new String[]{"Merci à vous aussi", "Thank you too"};

customText();

if (likeness >= 3) {

win();

} else {

lose();

}

}

//method used for new mexico

private void newMexico() {

//variable used to let user know correct answer or if answer is correct

String answer;

//making map invisible

frame.setVisible(false);

//asking the user various questions and adding points to likeness variable depending on answer

dialogText = "Mr. Trump, since New Mexico is a Spanish state, we will ask you a\nfew questions to test if you are worth voting for.";

dialogOptions = new String[]{"Ok", "De acuerdo"};

customText();

if (check == 1) {

likeness += 1;

}

dialogText = "How do you say 'ground' in Spanish?";

dialogOptions = new String[]{"Groundito", "Solito", "Soloe", "Suelo"};

customText();

if (check == 3) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is 'suelo'.\n";

}

dialogText = answer + "How do you say 'nice' in Spanish?";

dialogOptions = new String[]{"Nicito", "Bonito", "Niceito", "Bontatio"};

customText();

if (check == 1) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is 'bonito'.\n";

}

dialogText = answer + "How do you say 'alligator' in Spanish?";

dialogOptions = new String[]{"Artatilio", "Caimán", "Caimánito", "Arstitatio"};

customText();

if (check == 1) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is 'caimán'.\n";

}

dialogText = answer + "Thank you Mr. Trump, fue un honor.";

dialogOptions = new String[]{"Gracias", "Thank you too"};

customText();

if (likeness >= 3) {

win();

} else {

lose();

}

}

//method used for illinois

private void illinois() {

//making map invisible

frame.setVisible(false);

//asking the user various questions and adding points to likeness variable depending on answer

dialogText = "Hello, Mr. Trump. I will just ask you a\nfew questions for our interview.";

dialogOptions = new String[]{"Ok", "Go ahead"};

customText();

dialogText = "Mr. Trump, you have talked a lot about building a wall, realistically, how will you fund it?";

dialogOptions = new String[]{"Make Mexico pay", "Challenge Mexico to a game of Janga"};

customText();

if (check == 1) {

likeness -= 1;

}

dialogText = "How will you realistically deport all the illegal immigrants?";

dialogOptions = new String[]{"I just will", "Get good people on the job"};

customText();

if (check == 0) {

likeness -= 1;

}

dialogText = "Mr. Trump, in your opinion, for what reason do you\nbelieve that our is economy failing?";

dialogOptions = new String[]{"China", "Illegal Immigrants"};

customText();

if (check == 0) {

likeness += 2;

} else if (check == 1) {

likeness += 1;

}

if (likeness >= 1) {

win();

} else {

lose();

}

}

//method used for massachusetts

private void massachusetts() {

//making map invisible

frame.setVisible(false);

//letting user know information about the minigame

dialogText = "Mr. Trump, to win you need 200 points. If the word is a palindrome\nyou will receive the amount of letters times ten in points. The game\nends once you enter three consecutive incorrect palindromes.";

dialogOptions = new String[]{"Continue"};

customText();

//starts loop for when user has not lost, giving user ability to enter palindromes

while (palindromeTries > 0) {

//getting word that user wants to submit

palindromeWord = JOptionPane.showInputDialog(null, "Enter a palindrome.\n" + palindromeCorrect + "Score: " + palindromeScore + " points\nTries: " + palindromeTries, gameTitle, JOptionPane.PLAIN\_MESSAGE);

//checking if user pressed cancel or the x button

if (palindromeWord == null) {

System.exit(0);

}

//if user inputs nothing and hits enter, user is notified to re-write the message

while ("".equals(palindromeWord) || palindromeWord.contains(" ")) {

palindromeWord = JOptionPane.showInputDialog(null, "Put in one palindrome, not two, not zero... ONE.\n" + palindromeCorrect + "Score: " + palindromeScore + " points\nTries: " + palindromeTries, gameTitle, JOptionPane.PLAIN\_MESSAGE);

//checking if user pressed cancel or the x button

if (palindromeWord == null) {

System.exit(0);

}

}

//converting word to lower case

palindromeWord = palindromeWord.toLowerCase();

//setting technical to false

palindromeTechnical = false;

//checking length of word

palindromeLength = palindromeWord.length();

//declaring char array based on length of string

char letter[] = new char[palindromeLength];

//filling in array

for (int counter = 0; counter < palindromeLength; counter++) {

letter[counter] = (char) (palindromeWord.charAt(counter));

}

//declaring counter used for palindrome

int counterPalindrome = 0;

//checking if palindrome

for (int counter = 0; counter < (palindromeLength / 2); counter++) {

if (letter[counter] == letter[(palindromeLength - counter - 1)]) {

counterPalindrome += 1;

}

}

//sets wether the palindrome is a technical one

if (counterPalindrome >= (palindromeLength / 2)) {

palindromeTechnical = true;

}

//setting variables to false

palindromeActual = false;

palindromeUsed = false;

//creating a temporary variable that is upper case

String upper = palindromeWord.toUpperCase();

//going through all 208 palindromes and determining if it is on the list

for (int counter = 0; counter < 208; counter++) {

if (palindromeWord.equals(palindromeLibrary[counter])) {

//if the entered palindrome is on the list, and the one on the list is lowercase, the palindrome is an actual one

palindromeActual = true;

//setting palindrome on array list to upper case, so that it is not re-used

palindromeLibrary[counter] = palindromeLibrary[counter].toUpperCase();

} else if (upper.equals(palindromeLibrary[counter])) {

//if palindrome has already been used, used is set to true

palindromeUsed = true;

}

}

//executes when entered palindrome has not yet been used and is on the list

if (palindromeActual == true) {

//telling user how many points is added

palindromeCorrect = "+ " + (10 \* palindromeLength) + " points\n";

//adding points

palindromeScore = palindromeScore + (10 \* palindromeLength);

//reseting palindromeTries

palindromeTries = 3;

//executes when the palindrome is not on the list or it has already been used

} else {

//telling user that palindrome has already been used

if (palindromeUsed == true) {

palindromeCorrect = "-1 Try... Palindrome already used.\n";

//telling user the word is a palindrome but not an English word

} else if (palindromeTechnical == true && palindromeActual == false && palindromeUsed == false) {

palindromeCorrect = "-1 Try... Not an English word.\n";

//telling user that it is not a palindrome

} else {

palindromeCorrect = "-1 Try... Not a palindrome.\n";

}

//taking a try away from the user

palindromeTries -= 1;

}

}

if (palindromeScore >= 200) {

win();

} else {

lose();

}

}

//method used for arizona

private void arizona() {

//making map invisible

frame.setVisible(false);

//asking the user various questions and adding points to likeness variable depending on answer

dialogText = "Senior Trump, I am mexican...";

dialogOptions = new String[]{"(Let him speak)", "(Interrupt him)"};

customText();

if (check == 0) {

likeness += 1;

}

dialogText = "I am deeply offended by your statements...";

dialogOptions = new String[]{"(Let him speak)", "(Interrupt him)"};

customText();

if (check == 0) {

likeness -= 1;

}

dialogText = "I will ask you a question...";

dialogOptions = new String[]{"(Let him speak)", "(Interrupt him)"};

customText();

if (check == 0) {

likeness -= 1;

}

dialogText = "My question is...";

dialogOptions = new String[]{"(Let him speak)", "(Interrupt him)"};

customText();

if (check == 0) {

likeness += 1;

}

dialogText = "Would you build a wall around every state in the\nunited states to keep them safe from each other?";

dialogOptions = new String[]{"Only Mexico", "No"};

customText();

if (check == 0) {

likeness += 2;

}

if (likeness >= 3) {

win();

} else {

lose();

}

}

//method used for new jersey

private void newJersey() {

//making map invisible

frame.setVisible(false);

//making the GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

JLabel background = new JLabel(new ImageIcon("Assets/BoxingTitle.png"));

frameState.add(background);

frameState.setSize(353, 275);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

startTime = System.nanoTime();

//initializing mouse listner

frameState.addMouseListener(new MouseListener() {

int hillaryDamage, trumpDamage, tempTime, combo;

boolean matchStarted;

@Override

public void mousePressed(MouseEvent e) {

if (matchStarted == true) {

//setting cooldown and damage amount

totalTime = (System.nanoTime() - startTime) / 1000000000;

if (e.getModifiers() == InputEvent.BUTTON1\_MASK && hillaryDamage != 200) {

if (totalTime >= 1) {

startTime = System.nanoTime();

trumpDamage += totalTime;

if (combo == 0 || combo == 1) {

combo += 1;

}

hillaryDamage += 1;

}

} else if (e.getModifiers() == InputEvent.BUTTON3\_MASK && hillaryDamage != 200) {

if (totalTime >= 3) {

startTime = System.nanoTime();

trumpDamage += totalTime;

if (combo == 2) {

hillaryDamage += 10;

combo = 0;

}

hillaryDamage += 3;

}

}

//lose condition

if (trumpDamage > 50) {

frameState.setVisible(false);

lose();

}

//changing image based on hillaryDamage

if (hillaryDamage >= 5 && hillaryDamage < 10) {

background.setIcon(new ImageIcon("Assets/Hillary1.png"));

} else if (hillaryDamage >= 10 && hillaryDamage < 15) {

background.setIcon(new ImageIcon("Assets/Hillary2.png"));

} else if (hillaryDamage >= 15 && hillaryDamage < 20) {

background.setIcon(new ImageIcon("Assets/Hillary3.png"));

} else if (hillaryDamage >= 20 && hillaryDamage < 25) {

background.setIcon(new ImageIcon("Assets/Hillary4.png"));

} else if (hillaryDamage >= 25 && hillaryDamage < 30) {

background.setIcon(new ImageIcon("Assets/Hillary5.png"));

} else if (hillaryDamage >= 30 && hillaryDamage < 35) {

background.setIcon(new ImageIcon("Assets/Hillary6.png"));

} else if (hillaryDamage >= 35 && hillaryDamage < 40) {

background.setIcon(new ImageIcon("Assets/Hillary7.png"));

} else if (hillaryDamage >= 40 && hillaryDamage < 45) {

background.setIcon(new ImageIcon("Assets/Hillary8.png"));

} else if (hillaryDamage >= 45 && hillaryDamage < 50) {

background.setIcon(new ImageIcon("Assets/Hillary9.png"));

} else if (hillaryDamage >= 50 && hillaryDamage < 200) {

background.setIcon(new ImageIcon("Assets/Hillary10.png"));

frameState.setSize(218, 149);

frameState.setLocationRelativeTo(null);

hillaryDamage = 200;

} else if (hillaryDamage == 200) {

frameState.setVisible(false);

win();

}

} else {

background.setIcon(new ImageIcon("Assets/Hillary0.png"));

frameState.setSize(213, 375);

frameState.setLocationRelativeTo(null);

matchStarted = true;

}

}

@Override

public void mouseClicked(MouseEvent me) {

//useless but needed

}

@Override

public void mouseReleased(MouseEvent me) {

//useless but needed

}

@Override

public void mouseEntered(MouseEvent me) {

//useless but needed

}

@Override

public void mouseExited(MouseEvent me) {

//useless but needed

}

});

}

//method used for new hampshire

private void newHampshire() {

if (scoreScramble == 0) {

frame.setVisible(false);

dialogText = "Mr. Trump, we will test you with a word scramble\ngame. Get 10 points and you will win the state's\nvote. Get one word wrong and you will lose it.";

dialogOptions = new String[]{"Continue"};

customText();

}

String computerOvertime[] = {"rock", "puke", "love", "more", "tape", "dizzy", "quark", "waltz", "blitz", "hotel", "strife", "joyful", "jiggle", "jungle", "masque", "maximum", "minimum", "judging", "zombies", "freezing"};

String word, answer, checkScramble;

int length;

//generates random number from 0 to 19

int random = ((int) (Math.random() \* ((10 - 0) + 0))) + 0;

//sets word and answer to random string

word = answer = computerOvertime[random];

//checks length of random string

length = answer.length();

//declaring char array based on length of string

char letter[] = new char[length];

//filling in array

for (int counterScramble = 0; counterScramble < length; counterScramble++) {

letter[counterScramble] = (char) (word.charAt(counterScramble));

}

//creating another array to be randomly filled in by old array

char scrambled[] = new char[length];

//for loop to randomly fill in new array

for (int counterScramble = 0; counterScramble < length; counterScramble++) {

//random number from 0 to length number

random = ((int) (Math.random() \* ((length - 0) + 0))) + 0;

//if randomly picked number contains a space, repick a new random number

while (letter[random] == ' ') {

random = ((int) (Math.random() \* ((length - 0) + 0))) + 0;

}

//set scrambled number from random place in array of letter

scrambled[counterScramble] = letter[random];

//set place where char was taken from letter array so it does not get picked again

letter[random] = ' ';

}

//set string word from the char array scrambled

word = String.valueOf(scrambled);

checkScramble = JOptionPane.showInputDialog(null, scoreScramble + " points\nUnscramble: " + word, gameTitle, JOptionPane.PLAIN\_MESSAGE);

if (checkScramble == null) {

System.exit(0);

}

if (checkScramble.equals(answer)) {

scoreScramble += length;

if (scoreScramble >= 10) {

win();

} else {

newHampshire();

}

} else {

lose();

}

}

//method used for nevada

private void nevada() {

JLabel background[] = new JLabel[9];

//setting map to invisible

frame.setVisible(false);

//giving user information about minigame

dialogText = "Mr. Trump, Nevada is the gambling state, so\nnaturally, to win their vote, you shall gamble.\nYou will have ten tries to get a matching pair.\nTo try the slot again, click on it.";

dialogOptions = new String[]{"Continue"};

customText();

//making GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

frameState.setLayout(new GridLayout(3, 3));

for (int counter = 0; counter < 9; counter++) {

background[counter] = new JLabel(new ImageIcon("Assets/Slot" + counter + ".png"));

frameState.add(background[counter]);

}

frameState.setSize(423, 445);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

startTime = System.nanoTime();

//initializing mouse listner

frameState.addMouseListener(new MouseListener() {

int clickCount;

int val[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};

@Override

public void mousePressed(MouseEvent e) {

//declaring win and lose case

if ((val[0] == val[1] && val[1] == val[2]) || (val[3] == val[4] && val[4] == val[5]) || (val[6] == val[7] && val[7] == val[8]) || (val[0] == val[4] && val[4] == val[8]) || (val[6] == val[4] && val[4] == val[2]) || (val[0] == val[3] && val[3] == val[6]) || (val[1] == val[4] && val[4] == val[7]) || (val[2] == val[5] && val[5] == val[8])) {

frameState.setVisible(false);

win();

} else if (clickCount >= 10) {

frameState.setVisible(false);

lose();

}

//showing the pictures for gambling

for (int counter = 0; counter < 9; counter++) {

val[counter] = (int) (Math.random() \* 9);

background[counter].setIcon(new ImageIcon("Assets/Slot" + val[counter] + ".png"));

}

clickCount += 1;

}

@Override

public void mouseClicked(MouseEvent me) {

//useless but needed

}

@Override

public void mouseReleased(MouseEvent me) {

//useless but needed

}

@Override

public void mouseEntered(MouseEvent me) {

//useless but needed

}

@Override

public void mouseExited(MouseEvent me) {

//useless but needed

}

});

}

//method used for maryland

private void maryland() {

//setting map to invisible

frame.setVisible(false);

//giving user information about minigame

dialogText = "Mr. Trump, as you might know, Maryland is home of the NSA. So, let's\nspy on some people. Please pick which ethnic group to spy on.";

dialogOptions = new String[]{"Continue"};

customText();

//making GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

//declaring buttons

JLabel background = new JLabel(new ImageIcon("Assets/NSA.png"));

JButton btnMexican = new JButton("Mexicans");

JButton btnAsians = new JButton("Asians");

JButton btnAA = new JButton("African-Americans");

JButton btnCaucasians = new JButton("Caucasians");

JButton btnAfricans = new JButton("Africans");

JButton btnEuropeans = new JButton("Europeans");

//declaring panels

JPanel buttonPanel1 = new JPanel();

JPanel buttonPanel2 = new JPanel();

//adding buttons to panels

buttonPanel1.add(btnMexican);

buttonPanel1.add(btnAsians);

buttonPanel1.add(btnAA);

buttonPanel2.add(btnCaucasians);

buttonPanel2.add(btnAfricans);

buttonPanel2.add(btnEuropeans);

//making GUI more user-friendly

frameState.add(background, BorderLayout.NORTH);

frameState.add(buttonPanel1, BorderLayout.CENTER);

frameState.add(buttonPanel2, BorderLayout.SOUTH);

frameState.setSize(353, 367);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

//declaring what happens on button click

btnMexican.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

dialogText = "No progress was achieved what so ever, but for\nsome reason, people are content on your decision.";

customText();

win();

});

btnAsians.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

dialogText = "No progress was achieved what so ever, and for some reason,\nnot many people care that you are infringing on their privacy.\nAs Maryland is a blue state anyways, they have not voted for you.";

customText();

lose();

});

btnAA.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

dialogText = "No progress was achieved what so ever, and for some reason,\nnot many people care that you are infringing on their privacy.\nAs Maryland is a blue state anyways, they have not voted for you.";

customText();

lose();

});

btnCaucasians.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

dialogText = "No progress was achieved what so ever, and for some reason,\nnot many people care that you are infringing on their privacy.\nAs Maryland is a blue state anyways, they have not voted for you.";

customText();

lose();

});

btnAfricans.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

dialogText = "No progress was achieved what so ever, and for some reason,\nnot many people care that you are infringing on their privacy.\nAs Maryland is a blue state anyways, they have not voted for you.";

customText();

lose();

});

btnEuropeans.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

dialogText = "No progress was achieved what so ever, and for some reason,\nnot many people care that you are infringing on their privacy.\nAs Maryland is a blue state anyways, they have not voted for you.";

customText();

lose();

});

}

//method used for idaho

private void idaho() {

//making map invisible

frame.setVisible(false);

//giving user information of minigame

dialogText = "Mr. Trump, Idaho is the potato state, to win\ntheir vote, just left click and chop the potatos!";

dialogOptions = new String[]{"Continue"};

customText();

//making GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

JLabel background = new JLabel(new ImageIcon("Assets/Potato1.png"));

frameState.add(background);

frameState.setSize(353, 226);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

startTime = System.nanoTime();

//initializing mouse listner

frameState.addMouseListener(new MouseListener() {

int clickCount;

@Override

public void mousePressed(MouseEvent e) {

if (e.getModifiers() == InputEvent.BUTTON1\_MASK) {

if (clickCount < 49) {

clickCount += 1;

}

//setting image based on amount of mouse clicks

if (clickCount > 0 && clickCount <= 16) {

background.setIcon(new ImageIcon("Assets/Potato" + clickCount + ".png"));

} else if (clickCount > 16 && clickCount <= 32) {

background.setIcon(new ImageIcon("Assets/Potato" + (clickCount - 16) + ".png"));

} else if (clickCount > 32 && clickCount <= 48) {

background.setIcon(new ImageIcon("Assets/Potato" + (clickCount - 32) + ".png"));

} else {

frameState.setVisible(false);

totalTime = (System.nanoTime() - startTime) / 1000000000;

//win lose condition based on time taken

if (totalTime < 12) {

win();

} else {

lose();

}

}

}

}

@Override

public void mouseClicked(MouseEvent me) {

//useless but needed

}

@Override

public void mouseReleased(MouseEvent me) {

//useless but needed

}

@Override

public void mouseEntered(MouseEvent me) {

//useless but needed

}

@Override

public void mouseExited(MouseEvent me) {

//useless but needed

}

});

}

//method used for virginia

private void virginia() {

//making map invisible

frame.setVisible(false);

//giving user information of minigame

dialogText = "Mr. Trump, Virginia is the home of the CIA, so naturallly, we will torture\nsomeone. On agreement, we will rig the Virginian election in your favour.";

dialogOptions = new String[]{"Ok", "No (Opt Out)"};

customText();

if (check == 1) {

lose();

} else {

//making GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

//declaring GUI components

JLabel background = new JLabel(new ImageIcon("Assets/CIA.png"));

JLabel healthLeft = new JLabel("Health Left: 100%");

JButton btnBreakArms = new JButton("Break Arms");

JButton btnBreakLegs = new JButton("Break Legs");

JButton btnElectrocute = new JButton("Electrocute");

JButton btnPullToothOut = new JButton("Pull Tooth Out");

JButton btnWaterboard = new JButton("Waterboard");

JButton btnStop = new JButton("Stop");

//declaring panel

JPanel buttonPanel = new JPanel();

//adding buttons to panel

buttonPanel.add(btnBreakArms);

buttonPanel.add(btnBreakLegs);

buttonPanel.add(btnElectrocute);

buttonPanel.add(btnPullToothOut);

buttonPanel.add(btnWaterboard);

buttonPanel.add(btnStop);

buttonPanel.setLayout(new GridLayout(2, 3));

//making GUI more user-friendly

frameState.add(background, BorderLayout.NORTH);

frameState.add(healthLeft, BorderLayout.CENTER);

frameState.add(buttonPanel, BorderLayout.SOUTH);

frameState.pack();

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

//butons used to reduce health

btnBreakArms.addActionListener((ActionEvent e) -> {

damage += 1;

if (damage < 10) {

healthLeft.setText("Health Left: " + 10 \* (10 - damage) + "%");

} else {

healthLeft.setText("Inmate Is Dead");

}

});

btnBreakLegs.addActionListener((ActionEvent e) -> {

damage += 1;

if (damage < 10) {

healthLeft.setText("Health Left: " + 10 \* (10 - damage) + "%");

} else {

healthLeft.setText("Inmate Is Dead");

}

});

btnElectrocute.addActionListener((ActionEvent e) -> {

damage += 1;

if (damage < 10) {

healthLeft.setText("Health Left: " + 10 \* (10 - damage) + "%");

} else {

healthLeft.setText("Inmate Is Dead");

}

});

btnPullToothOut.addActionListener((ActionEvent e) -> {

damage += 1;

if (damage < 10) {

healthLeft.setText("Health Left: " + 10 \* (10 - damage) + "%");

} else {

healthLeft.setText("Inmate Is Dead");

}

});

btnWaterboard.addActionListener((ActionEvent e) -> {

damage += 1;

if (damage < 10) {

healthLeft.setText("Health Left: " + 10 \* (10 - damage) + "%");

} else {

healthLeft.setText("Inmate Is Dead");

}

});

//button used to end game

btnStop.addActionListener((ActionEvent e) -> {

frameState.setVisible(false);

dialogOptions = new String[]{"Continue"};

if (damage < 5) {

dialogText = "Mr. Trump, you didn't damage the inmate enough, the inmate won't talk.";

customText();

lose();

} else if (damage >= 10) {

dialogText = "Mr. Trump, you killed the inmate!";

customText();

lose();

} else {

dialogText = "Mr. Trump, great job, we will make sure you win Virginia!";

customText();

win();

}

});

}

}

//method used for west virginia

private void westVirginia() {

//making map invisible

frame.setVisible(false);

//asking the user various questions and adding points to likeness variable depending on answer

dialogText = "Mr. Trump, West Virginia is considered one of the least intelligent states.\nTo win their vote, just make a YouTube video to be sponsored in their state.";

dialogOptions = new String[]{"Continue"};

customText();

dialogText = "What type of introduction would you like?";

dialogOptions = new String[]{"Flashy", "Loud", "Professional"};

customText();

if (check == 0) {

likeness += 1;

} else if (check == 1) {

likeness += 2;

}

dialogText = "What slogan do you want to use?";

dialogOptions = new String[]{"Make America Great Again", "Trump Will Triump"};

customText();

if (check == 0) {

likeness += 2;

} else if (check == 1) {

likeness += 1;

}

dialogText = "What main topic would you to discuss?";

dialogOptions = new String[]{"Immigration", "China", "Wall"};

customText();

if (check == 0 || check == 1) {

likeness += 1;

}

dialogText = "What conclusion would you like to use?";

dialogOptions = new String[]{"Memorable", "Flashy", "Loud"};

customText();

if (check == 0 || check == 1) {

likeness += 1;

} else {

likeness += 2;

}

if (likeness >= 5) {

win();

} else {

lose();

}

}

//method used for iowa

private void iowa() {

//making map invisible

frame.setVisible(false);

//making GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

//declaring GUI components

JLabel background = new JLabel(new ImageIcon("Assets/IowaBackground.png"));

JButton btnOne = new JButton("How stupid are the people of Iowa");

JButton btnTwo = new JButton("I love the people of Iowa");

//making GUI more user-friendly

frameState.add(background, BorderLayout.NORTH);

frameState.add(btnOne, BorderLayout.CENTER);

frameState.add(btnTwo, BorderLayout.SOUTH);

frameState.setSize(348, 310);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

//asking the user various questions and adding points to likeness variable depending on answer

btnOne.addActionListener((ActionEvent e) -> {

switch (iowaLocation) {

case 0:

likeness -= 1;

btnOne.setText("How stupid are the American people");

btnTwo.setText("I love America");

break;

case 1:

likeness -= 1;

btnOne.setText("I will build a wall");

btnTwo.setText("I will make America great again");

break;

case 2:

likeness += 1;

btnOne.setText("Hillary Clinton was the worst secretary of state");

btnTwo.setText("Hillary Clinton is a great person");

break;

case 3:

likeness += 1;

btnOne.setText("Mexican are bringing drugs and crime");

btnTwo.setText("Some Mexicans are good people");

break;

case 4:

likeness += 1;

frameState.setVisible(false);

if (likeness >= 5) {

win();

} else {

lose();

}

break;

}

iowaLocation += 1;

});

//asking the user various questions and adding points to likeness variable depending on answer

btnTwo.addActionListener((ActionEvent e) -> {

switch (iowaLocation) {

case 0:

likeness += 1;

btnOne.setText("How stupid are the American people");

btnTwo.setText("I love America");

break;

case 1:

likeness += 1;

btnOne.setText("I will build a wall");

btnTwo.setText("I will make America great again");

break;

case 2:

likeness += 2;

btnOne.setText("Hillary Clinton was the worst secretary of state");

btnTwo.setText("Hillary Clinton is a great person");

break;

case 3:

likeness -= 2;

btnOne.setText("Mexican are bringing drugs and crime");

btnTwo.setText("Some Mexicans are good people");

break;

case 4:

likeness -= 1;

frameState.setVisible(false);

if (likeness >= 5) {

win();

} else {

lose();

}

break;

}

iowaLocation += 1;

});

}

//method used for kentucky

private void kentucky() {

//making map invisible

frame.setVisible(false);

//asking the user various questions and adding points to likeness variable depending on answer

dialogText = "Mr. Trump, Kentucky is the chicken state, to win\ntheir vote, just left click and cook the chicken!";

dialogOptions = new String[]{"Continue"};

customText();

//making GUI more user-friendly

JFrame frameState = new JFrame(gameTitle);

frameState.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frameState.setResizable(false);

JLabel background = new JLabel(new ImageIcon("Assets/Chicken1.png"));

frameState.add(background);

frameState.setSize(371, 365);

frameState.setLocationRelativeTo(null);

frameState.setIconImage(iconTrump.getImage());

frameState.setVisible(true);

startTime = System.nanoTime();

frameState.addMouseListener(new MouseListener() {

int clickCount;

@Override

public void mousePressed(MouseEvent e) {

if (e.getModifiers() == InputEvent.BUTTON1\_MASK) {

if (clickCount < 35) {

clickCount += 1;

}

//setting image based on amount of mouse clicks

if (clickCount != 35) {

background.setIcon(new ImageIcon("Assets/Chicken" + clickCount + ".png"));

} else {

frameState.setVisible(false);

totalTime = (System.nanoTime() - startTime) / 1000000000;

if (totalTime <= 10) {

win();

} else {

lose();

}

}

}

}

@Override

public void mouseClicked(MouseEvent me) {

//useless but needed

}

@Override

public void mouseReleased(MouseEvent me) {

//useless but needed

}

@Override

public void mouseEntered(MouseEvent me) {

//useless but needed

}

@Override

public void mouseExited(MouseEvent me) {

//useless but needed

}

});

}

//method used for north dakota

private void northDakota() {

//declaring variable used for telling user if he/she is correct and if not the correct answer

String answer;

//making map invisible

frame.setVisible(false);

//asking the user various questions and adding points to likeness variable depending on answer

dialogText = "Mr. Trump, since North Dakota is the most German state, we\nwill ask you a few questions to test if you are worth voting for.";

dialogOptions = new String[]{"Ok", "Fortsetzen"};

customText();

if (check == 1) {

likeness += 1;

}

dialogText = "How do you say 'book' in German?";

dialogOptions = new String[]{"buch", "Buch", "book", "bookich"};

customText();

if (check == 1) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is 'Buch'.\n";

}

dialogText = answer + "How do you say 'dark' in German?";

dialogOptions = new String[]{"dutchen", "Darken", "darken", "dunkel"};

customText();

if (check == 3) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is 'dunkel'.\n";

}

dialogText = answer + "What does 'Ich bin gut' mean?";

dialogOptions = new String[]{"I am good", "You are good", "I get it", "I understand"};

customText();

if (check == 0) {

answer = "Correct! ";

likeness += 1;

} else {

answer = "The correct answer is 'I am good'.\n";

}

dialogText = answer + "Thank you Mr. Trump, es war eine Freude.";

dialogOptions = new String[]{"Danke dir auch", "Thank you too"};

customText();

if (likeness >= 3) {

win();

} else {

lose();

}

}

//method used for rhode island

private void rhodeIsland() {

//making map invisible

frame.setVisible(false);

//giving user information of state

dialogText = "Mr. Trump, Rhode Island is so small that we\nwill just host a barbeque, and will most probably win.";

dialogOptions = new String[]{"Continue"};

customText();

//90% chance that user wins

int randomState = (int) (Math.random() \* 10);

if (randomState != 0) {

win();

} else {

lose();

}

}

//method used for starting jeopardy program

private void jeopardyPrepareGUI() {

frame.setVisible(false);

dialogText = "Mr. Trump, we have a Jeopardy game prepared for you to test you.\nYou need at least 1000 points to win the state.";

dialogOptions = new String[]{"Continue"};

customText();

//setting mainFrame

mainFrame = new JFrame(gameTitle);

mainFrame.setSize(300, 400);

mainFrame.setLayout(new GridLayout(9, 1));

mainFrame.setLocationRelativeTo(null);

mainFrame.setIconImage(iconTrump.getImage());

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

mainFrame.setResizable(false);

//setting text for labels

headerLabel = new JLabel("Jeopardy", JLabel.CENTER);

subheaderLabel = new JLabel("Science Math History Games", JLabel.CENTER);

responseScore = new JLabel("Total score: 0 points", JLabel.CENTER);

responseAnswer = new JLabel("", JLabel.CENTER);

//setting buttons for mainFrame

controlPanel1 = new JPanel();

controlPanel1.setLayout(new FlowLayout());

controlPanel2 = new JPanel();

controlPanel2.setLayout(new FlowLayout());

controlPanel3 = new JPanel();

controlPanel3.setLayout(new FlowLayout());

controlPanel4 = new JPanel();

controlPanel4.setLayout(new FlowLayout());

controlPanel5 = new JPanel();

controlPanel5.setLayout(new FlowLayout());

//setting buttons and labels to mainFrame

mainFrame.add(headerLabel);

mainFrame.add(subheaderLabel);

mainFrame.add(controlPanel1);

mainFrame.add(controlPanel2);

mainFrame.add(controlPanel3);

mainFrame.add(controlPanel4);

mainFrame.add(controlPanel5);

mainFrame.add(responseScore);

mainFrame.add(responseAnswer);

settingButtons();

}

//method used for declaring buttons for jeopardy

private void settingButtons() {

//setting text to displayed buttons

JButton science100 = new JButton("100");

JButton science200 = new JButton("200");

JButton science300 = new JButton("300");

JButton science400 = new JButton("400");

JButton science500 = new JButton("500");

JButton math100 = new JButton("100");

JButton math200 = new JButton("200");

JButton math300 = new JButton("300");

JButton math400 = new JButton("400");

JButton math500 = new JButton("500");

JButton history100 = new JButton("100");

JButton history200 = new JButton("200");

JButton history300 = new JButton("300");

JButton history400 = new JButton("400");

JButton history500 = new JButton("500");

JButton games100 = new JButton("100");

JButton games200 = new JButton("200");

JButton games300 = new JButton("300");

JButton games400 = new JButton("400");

JButton games500 = new JButton("500");

//declaring blank as an array and creating 20 blank buttons

JButton blank[];

blank = new JButton[20];

for (int blankCounter = 0; blankCounter < 20; blankCounter++) {

blank[blankCounter] = new JButton(" ");

blank[blankCounter].setVisible(false);

}

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

science100.addActionListener((ActionEvent e) -> {

science100.setVisible(false);

blank[0].setVisible(true);

science100();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

science200.addActionListener((ActionEvent e) -> {

science200.setVisible(false);

blank[1].setVisible(true);

science200();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

science300.addActionListener((ActionEvent e) -> {

science300.setVisible(false);

blank[2].setVisible(true);

science300();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

science400.addActionListener((ActionEvent e) -> {

science400.setVisible(false);

blank[3].setVisible(true);

science400();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

science500.addActionListener((ActionEvent e) -> {

science500.setVisible(false);

blank[4].setVisible(true);

science500();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

math100.addActionListener((ActionEvent e) -> {

math100.setVisible(false);

blank[5].setVisible(true);

math100();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

math200.addActionListener((ActionEvent e) -> {

math200.setVisible(false);

blank[6].setVisible(true);

math200();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

math300.addActionListener((ActionEvent e) -> {

math300.setVisible(false);

blank[7].setVisible(true);

math300();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

math400.addActionListener((ActionEvent e) -> {

math400.setVisible(false);

blank[8].setVisible(true);

math400();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

math500.addActionListener((ActionEvent e) -> {

math500.setVisible(false);

blank[9].setVisible(true);

math500();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

history100.addActionListener((ActionEvent e) -> {

history100.setVisible(false);

blank[10].setVisible(true);

history100();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

history200.addActionListener((ActionEvent e) -> {

history200.setVisible(false);

blank[11].setVisible(true);

history200();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

history300.addActionListener((ActionEvent e) -> {

history300.setVisible(false);

blank[12].setVisible(true);

history300();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

history400.addActionListener((ActionEvent e) -> {

history400.setVisible(false);

blank[13].setVisible(true);

history400();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

history500.addActionListener((ActionEvent e) -> {

history500.setVisible(false);

blank[14].setVisible(true);

history500();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

games100.addActionListener((ActionEvent e) -> {

games100.setVisible(false);

blank[15].setVisible(true);

games100();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

games200.addActionListener((ActionEvent e) -> {

games200.setVisible(false);

blank[16].setVisible(true);

games200();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

games300.addActionListener((ActionEvent e) -> {

games300.setVisible(false);

blank[17].setVisible(true);

games300();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

games400.addActionListener((ActionEvent e) -> {

games400.setVisible(false);

blank[18].setVisible(true);

games400();

caseWin();

});

//set action listener for button, when button is clicked, it becomes invisible and blank button becomes visible

games500.addActionListener((ActionEvent e) -> {

games500.setVisible(false);

blank[19].setVisible(true);

games500();

caseWin();

});

//setting visible and invisible buttons to control panels

controlPanel1.add(science100);

controlPanel1.add(blank[0]);

controlPanel1.add(math100);

controlPanel1.add(blank[5]);

controlPanel1.add(history100);

controlPanel1.add(blank[10]);

controlPanel1.add(games100);

controlPanel1.add(blank[15]);

controlPanel2.add(science200);

controlPanel2.add(blank[1]);

controlPanel2.add(math200);

controlPanel2.add(blank[6]);

controlPanel2.add(history200);

controlPanel2.add(blank[11]);

controlPanel2.add(games200);

controlPanel2.add(blank[16]);

controlPanel3.add(science300);

controlPanel3.add(blank[2]);

controlPanel3.add(math300);

controlPanel3.add(blank[7]);

controlPanel3.add(history300);

controlPanel3.add(blank[12]);

controlPanel3.add(games300);

controlPanel3.add(blank[17]);

controlPanel4.add(science400);

controlPanel4.add(blank[3]);

controlPanel4.add(math400);

controlPanel4.add(blank[8]);

controlPanel4.add(history400);

controlPanel4.add(blank[13]);

controlPanel4.add(games400);

controlPanel4.add(blank[18]);

controlPanel5.add(science500);

controlPanel5.add(blank[4]);

controlPanel5.add(math500);

controlPanel5.add(blank[9]);

controlPanel5.add(history500);

controlPanel5.add(blank[14]);

controlPanel5.add(games500);

controlPanel5.add(blank[19]);

//set buttons default to visible, except for blanks which will be set

//to false later in the program, but is up above in the displayed code

mainFrame.setVisible(true);

}

//method used for determing if user has won jeopardy program

private void caseWin() {

if (click == 20) {

mainFrame.setVisible(false);

if (score >= 1000) {

win();

} else {

lose();

}

}

}

//method used for removing certain characters and converting to lower case

private void clickCommand() {

//if amount of clicks is less than 20, add 1 to click variable

if (click < 20) {

click += 1;

}

//prevents text to be edited if cancel button or x button is clicked, thus preventing error messages

if (userAnswer != null) {

//reset the response answer text

responseAnswer.setText("");

//convert userAnswer to lower case

userAnswer = userAnswer.toLowerCase();

//remove ' ', '.', ',', ';', ':', '!', and '?' from userAnswer for easier manipulation

userAnswer = userAnswer.replaceAll("[ .,;:!?]", "");

}

}

//method used for jeopardy questions

private void science100() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "The powerhouse of the cell is?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("mitochondria".equals(userAnswer) || "mitochondrion".equals(userAnswer) || "themitochondria".equals(userAnswer) || "themitochondrion".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: The Mitochondrion");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "The controler of the cell is?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("nucleus".equals(userAnswer) || "thenucleus".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: The Nucleus");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "The outside part of the animal cell is?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("cellularmembrane".equals(userAnswer) || "thecellularmembrane".equals(userAnswer) || "membrane".equals(userAnswer) || "themembrane".equals(userAnswer) || "plasmamembrane".equals(userAnswer) || "theplasmamembrane".equals(userAnswer) || "cellmembrane".equals(userAnswer) || "thecellmembrane".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: The Cellular Membrane");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "The outside part of the plant cell is?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("cellwall".equals(userAnswer) || "thecellwall".equals(userAnswer) || "cellularwall".equals(userAnswer) || "thecellularwall".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: The Cell Wall");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "Information carrying substances with double helix structure is called?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("dna".equals(userAnswer) || "deoxyribosenucleicacid".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: Deoxyribose Nucleic Acid");

}

break;

}

}

//method used for jeopardy questions

private void science200() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "The father of the theory of evolution.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("charlesdarwin".equals(userAnswer) || "charles".equals(userAnswer) || "darwin".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Charles Darwin");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "The father of the theory of relativity.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("alberteinstein".equals(userAnswer) || "albert".equals(userAnswer) || "einstein".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Albert Einstein");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "Name of the person in which the fundemental particle associated with mass was named after.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("peterhiggs".equals(userAnswer) || "peter".equals(userAnswer) || "higgs".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Peter Higgs");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "Name of the person that hypothesized three laws of motion.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("sirisaacnewton".equals(userAnswer) || "isaacnewton".equals(userAnswer) || "isaac".equals(userAnswer) || "newton".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Sir Isaac Newton");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "Who patented the light bulb?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("thomasedison".equals(userAnswer) || "thomas".equals(userAnswer) || "edison".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Thomas Edison");

}

break;

}

}

//method used for jeopardy questions

private void science300() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "A unit of magnitude and direction is a?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("vector".equals(userAnswer) || "avector".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Vector");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "A unit of magnitude is a?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("scalar".equals(userAnswer) || "ascalar".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Scalar");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "The SI unit of force is measured in?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("newtons".equals(userAnswer) || "newton".equals(userAnswer) || "n".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Newtons");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "A newton metre is a?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("joule".equals(userAnswer) || "ajoule".equals(userAnswer) || "j".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Joule");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "the SI base unit of mass is the?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("kilogram".equals(userAnswer) || "kilograms".equals(userAnswer) || "kg".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Kilogram");

}

break;

}

}

//method used for jeopardy questions

private void science400() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "The current model of the atom is the?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("thequantummodel".equals(userAnswer) || "quantummodel".equals(userAnswer) || "quantum".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: The Quantum Model");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "What is the family name of group 18 on the periodic table?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("noblegas".equals(userAnswer) || "noblegasses".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Noble Gasses");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "What is the family name of group 17 on the periodic table?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("halogen".equals(userAnswer) || "halogens".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Halogens");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "What is the family name of group 2 on the periodic table?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("akalineearthmetals".equals(userAnswer) || "alkalineearthmetal".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Alkaline Earth Metals");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "What is the family name of group 1 on the periodic table?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("alkalimetals".equals(userAnswer) || "alkalimetal".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Alkali Metals");

}

break;

}

}

//method used for jeopardy questions

private void science500() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "How much more energy does a Calorie have compared to a calorie as a ratio?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1000".equals(userAnswer) || "thousand".equals(userAnswer) || "onethousand".equals(userAnswer) || "athousand".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1000");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "As an integer value, how many joules are there per kilocalorie?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("4184".equals(userAnswer) || "4180".equals(userAnswer) || "4190".equals(userAnswer) || "4200".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: 4184");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "What colour has the most energy on the standard visible light spectrum?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("violet".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: Violet");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "What type of wave is a sound wave?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("mechanical".equals(userAnswer) || "mechanicalwave".equals(userAnswer) || "amechanicalwave".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: A Mechanical Wave");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "To no decimal places, how many light-years are in 100 parsecs?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("326".equals(userAnswer) || "330".equals(userAnswer) || "320".equals(userAnswer) || "300".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: 326");

}

break;

}

}

//method used for jeopardy questions

private void math100() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "What is (5+12-2\*9)/(1/2)", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("-2".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: -2");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "What is (7+3-3\*4)/(1/3)", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("-6".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: -6");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "What is (9+12-2\*13)\*(4/5)", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("-4".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: -4");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "What is (9+12-2\*13)\*(8/5)", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("-8".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: -8");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "What is (9+13-2\*7)/(4/5)", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("10".equals(userAnswer) || "+10".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: 10");

}

break;

}

}

//method used for jeopardy questions

private void math200() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "A number divided by zero is known as?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("undefined".equals(userAnswer) || "undefinedform".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Undefined");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "As n approaches infinity, what does 1/n equal?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("0".equals(userAnswer) || "zero".equals(userAnswer) || "nothing".equals(userAnswer) || "null".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: 0");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "Assuming it exists, as n approaches zero, what does 1/n equal?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("infinity".equals(userAnswer) || "infinite".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Infinity");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "0.999999(infinite amount of 9's) is the same as what integer value?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1".equals(userAnswer) || "one".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "0.000001(infinite amount of 0's followed by a 1) is the same as what integer value?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("0".equals(userAnswer) || "zero".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: 0");

}

break;

}

}

//method used for jeopardy questions

private void math300() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "Division is the \_\_\_\_\_\_\_ of multiplication.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("inverse".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Inverse");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "sin^(-1)x is the \_\_\_\_\_\_\_ of sin x.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("inverse".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Inverse");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "Find x: 2x+5=x-5", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("-10".equals(userAnswer) || "x=-10".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: -10");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "Find x: 4x+20=3x-10", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("-30".equals(userAnswer) || "x=-30".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: -30");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "Find x: 6x+10=9x-5", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("5".equals(userAnswer) || "x=5".equals(userAnswer) || "+5".equals(userAnswer) || "x=+5".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: 5");

}

break;

}

}

//method used for jeopardy questions

private void math400() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "When using limits, zero divided by zero is known as?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("indeterminate".equals(userAnswer) || "indeterminateform".equals(userAnswer) || "theindeterminateform".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Indeterminate Form");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "The irrational number e is known as?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("euler'snumber".equals(userAnswer) || "eulersnumber".equals(userAnswer) || "euler's".equals(userAnswer) || "eulers".equals(userAnswer) || "euler".equals(userAnswer) || "eulernumber".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Euler's Number");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "To no decimal places, 100e is?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("272".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: 272");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "The square root of -1 is?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("i".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: i");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "The square root of -9 is?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("3i".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: 3i");

}

break;

}

}

//method used for jeopardy questions

private void math500() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "Solve for x: x^2=10x-25\*0!", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("x=5".equals(userAnswer) || "5".equals(userAnswer) || "x=+5".equals(userAnswer) || "+5".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: x=5");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "Solve for x: x^2=20x-100\*0!", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("x=10".equals(userAnswer) || "10".equals(userAnswer) || "x=+10".equals(userAnswer) || "+10".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: x=10");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "e^(i\*pi)+1=\_", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("0".equals(userAnswer) || "zero".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: 0");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "Solve for x: x^2=6x-9\*0!", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("x=3".equals(userAnswer) || "3".equals(userAnswer) || "x=+3".equals(userAnswer) || "+3".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: x=3");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "Solve for x: x^2=8x-16\*0!", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("x=4".equals(userAnswer) || "4".equals(userAnswer) || "x=+4".equals(userAnswer) || "+4".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: x=4");

}

break;

}

}

//method used for jeopardy questions

private void history100() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "The year the first black president was elected into American Parliamentary power?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("2008".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: 2008");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "The year the first black president was re-elected into American Parliamentary power?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("2012".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: 2012");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "The year the first black president was elected out of American Parliamentary power?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("2016".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: 2016");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "What year was Canada founded?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1867".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1867");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "What year did World War 1 start?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1914".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1914");

}

break;

}

}

//method used for jeopardy questions

private void history200() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "The year of the first Moon landing?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1969".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1969");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "The year water was announced to be found on Mars?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("2015".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: 2015");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "The year gravitational waves were announced to be discovered?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("2016".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: 2016");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "The year the discovery of the Higgs Boson was announced?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("2013".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: 2013");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "The year the ISS Space Station was launched?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1998".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1998");

}

break;

}

}

//method used for jeopardy questions

private void history300() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "The year Adolf Hitler was nominated man of the year.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1939".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1939");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "The year Adolf Hitler became chancellor of Germany.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1933".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1933");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "The year Adolf Hitler became head of the Nazi Party.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1921".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1921");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "The year America declared war on the Axis Party during World War 2.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1941".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1941");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "The attack on which country sparked World War 2?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("poland".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Poland");

}

break;

}

}

//method used for jeopardy questions

private void history400() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "In 1945 a nuclear bomb was detonated in Hiroshima, Japan. What was the name of the bomb?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("littleboy".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Little Boy");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "In 1945 a nuclear bomb was detonated in Nagasaki, Japan. What was the name of the bomb?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("fatman".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Fat Man");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "In which year did Einstein publish his special theory of relativity?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1905".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1905");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "In which year did Einstein publish his general theory of relativity?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1915".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1915");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "What was the name of the city in which Archduke Franz Ferdinand of Austria was assassinated in 1914?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("sarajevo".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Sarajevo");

}

break;

}

}

//method used for jeopardy questions

private void history500() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "Who assassinated Archduke Franz Ferdinand of Austria in 1914?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("gavriloprincip".equals(userAnswer) || "gavrilo".equals(userAnswer) || "princip".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: Gavrilo Princip");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "In which year was Albert Einstein born?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1879".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1879");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "In which year was Sir Isaac Newton born?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1643".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1643");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "In which year was Stephen Hawking born?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1942".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1942");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "In which year was Charles Darwin born?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1809".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1809");

}

break;

}

}

//method used for jeopardy questions

private void games100() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "The company that created the Mario Brothers is called?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("nintendo".equals(userAnswer) || "nintendor&d".equals(userAnswer) || "nintendoresearch&development".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: Nintendo");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "The company that created Nathan Drake is called?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("naughtydog".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: Naughty Dog");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "The company that published Link is called?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("nintendo".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: Nintendo");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "The company that created Ezio Auditore is called?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("ubisoft".equals(userAnswer) || "ubisoftmontreal".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: Ubisoft");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "The company that created Solid Snake is called?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("kojimaproductions".equals(userAnswer) || "kojima".equals(userAnswer)) {

score += 100;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +100 points");

} else {

responseAnswer.setText("Incorrect. Answer: Kojima Productions");

}

break;

}

}

//method used for jeopardy questions

private void games200() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "The game in which two users have paddles on seperate side of the screen,\nand each user attempts to hit a ball past the paddle of the opposing user.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("pong".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Pong");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "The game released in 1978 in which the player controls a ship,\nand must destroy opposing alien ships.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("spaceinvaders".equals(userAnswer) || "spaceinvader".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Space Invaders");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "The game released in 1981 in which the player controls a character,\nand must save a princess by jumping over barrels while climbing ladders.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("donkeykong".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Donkey Kong");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "The game in which you assemble shapes as they move down,\neach shape consisting of four squares arranged in various manners.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("tetris".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Tetris");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "The game in which you play as a big yellow dot that eats smaller yellow dots and runs away from ghosts.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("pacman".equals(userAnswer) || "pac-man".equals(userAnswer)) {

score += 200;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +200 points");

} else {

responseAnswer.setText("Incorrect. Answer: Pacman");

}

break;

}

}

//method used for jeopardy questions

private void games300() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "What was the best selling video game of 2015?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("blackops3".equals(userAnswer) || "blackopsthree".equals(userAnswer) || "bo3".equals(userAnswer) || "callofdutyblackops3".equals(userAnswer) || "callofdutyblackopsthree".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Black Ops 3");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "What was the best selling video game of 2014?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("callofdutyadvancedwarfare".equals(userAnswer) || "advancedwarfare".equals(userAnswer) || "aw".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Advanced Warfare");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "What was the best selling video game of 2013?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("grandtheftautofive".equals(userAnswer) || "grandtheftauto5".equals(userAnswer) || "gta5".equals(userAnswer) || "gtafive".equals(userAnswer) || "grandtheftautov".equals(userAnswer) || "gtav".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Grand Theft Auto V");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "What was the best selling video game of 2012?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("blackops2".equals(userAnswer) || "blackopstwo".equals(userAnswer) || "bo2".equals(userAnswer) || "callofdutyblackops2".equals(userAnswer) || "callofdutyblackopstwo".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Black Ops 2");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "What was the best selling video game in 2011?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("modernwarfare3".equals(userAnswer) || "modernwarfarethree".equals(userAnswer) || "mw3".equals(userAnswer) || "callofdutymodernwarfare3".equals(userAnswer) || "callofdutymodernwarfarethree".equals(userAnswer)) {

score += 300;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +300 points");

} else {

responseAnswer.setText("Incorrect. Answer: Modern Warfare 3");

}

break;

}

}

//method used for jeopardy questions

private void games400() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "Name of the pacman ghost with a name starting with I.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("inky".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Inky");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "Name of the pacman ghost with a name starting with B.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("blinky".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Blinky");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "Name of the pacman ghost with a name starting with P.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("pinky".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Pinky");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "Name of the pacman ghost with a name starting with C.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("clyde".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: Clyde");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "Year the game Pac-Man was released.", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1980".equals(userAnswer)) {

score += 400;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +400 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1980");

}

break;

}

}

//method used for jeopardy questions

private void games500() {

//creating random number from 1 to 5

randomJeopardy = ((int) (Math.random() \* ((5 - 1) + 1))) + 1;

//use random number to decide which question to ask.

//in each scenario, a question is asked, the user answers,

//if the answer is correct the user gains points,

//if it is incorrect, the user is told what the correct answer is, but does not gain points,

//afterwards, the switch case statement is broken.

switch (randomJeopardy) {

case 1:

userAnswer = JOptionPane.showInputDialog(null, "The name of the person that created Monopoly?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("charlesdarrow".equals(userAnswer) || "charles".equals(userAnswer) || "darrow".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: Charles Darrow");

}

break;

case 2:

userAnswer = JOptionPane.showInputDialog(null, "The year the game Monopoly was released?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1935".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1935");

}

break;

case 3:

userAnswer = JOptionPane.showInputDialog(null, "The name of the person that created the game Life?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("miltonbradley".equals(userAnswer) || "milton".equals(userAnswer) || "bradley".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: Milton Bradley");

}

break;

case 4:

userAnswer = JOptionPane.showInputDialog(null, "The year the game Life was created?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1860".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1860");

}

break;

case 5:

userAnswer = JOptionPane.showInputDialog(null, "The year the game Trivial Pursuit was conceived?", gameTitle, JOptionPane.PLAIN\_MESSAGE);

clickCommand();

if ("1979".equals(userAnswer)) {

score += 500;

responseScore.setText("Total score: " + score + " points");

responseAnswer.setText("Correct! +500 points");

} else {

responseAnswer.setText("Incorrect. Answer: 1979");

}

break;

}

}

//method used for loading from file

private void load() {

try {

//trying to create file

Files.createFile(file);

//executed if file already exists

} catch (FileAlreadyExistsException x) {

//file is read from and saved to variable saveFile is file already exists

try (InputStream in = Files.newInputStream(file);

BufferedReader reader = new BufferedReader(new InputStreamReader(in))) {

String line;

while ((line = reader.readLine()) != null) {

//content of file is saved to saveFile

saveFile = line;

}

} catch (IOException y) {

System.err.println(y);

}

} catch (IOException x) {

System.err.println(x);

}

//if the file does not contain anything since it was just created, default variables are used for save file

if (saveFile == null) {

saveFile = "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 91 1 0 0 0 "

+ "0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 91 91 1 0 0 0 "

+ "0 0 0 0 0 1 52 52 52 52 52 52 52 1 55 55 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 91 91 91 1 0 0 "

+ "0 0 0 0 0 1 52 52 52 52 52 52 52 1 55 55 1 59 59 59 59 59 59 59 59 1 63 63 63 63 1 69 69 69 1 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 91 91 91 91 1 0 "

+ "0 0 0 0 0 1 52 52 52 52 52 52 52 1 55 55 1 59 59 59 59 59 59 59 59 1 63 63 63 63 1 69 69 69 1 0 1 74 1 0 1 77 77 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 91 91 91 91 1 0 "

+ "0 0 0 0 0 1 52 52 52 52 52 52 52 1 55 55 1 59 59 59 59 59 59 59 59 1 63 63 63 63 1 69 69 69 1 1 74 74 74 1 77 77 1 0 0 0 0 0 0 0 0 0 1 92 92 92 92 1 1 91 91 91 1 0 "

+ "0 0 0 0 0 1 1 1 1 1 1 1 1 1 55 55 1 59 59 59 59 59 59 59 59 1 1 1 1 1 1 69 69 69 1 74 74 74 74 1 77 1 0 0 1 0 0 0 0 0 0 1 85 1 92 92 1 93 93 1 1 1 0 0 "

+ "0 0 0 0 0 1 53 53 53 53 53 53 53 1 55 55 1 59 59 59 59 59 59 59 59 1 64 64 64 64 1 69 69 69 1 74 74 74 74 1 1 0 0 1 77 1 0 0 0 0 1 85 85 85 1 92 1 93 93 93 1 0 0 0 "

+ "0 0 0 0 0 1 53 53 53 53 53 53 53 1 55 55 55 1 1 1 1 1 1 1 1 1 64 64 64 64 1 69 69 69 1 74 74 74 74 1 0 0 0 1 77 1 0 0 0 0 1 85 85 85 1 1 1 1 1 1 0 0 0 0 "

+ "0 0 0 0 0 1 53 53 53 53 53 53 53 1 55 55 55 55 55 1 60 60 60 60 1 64 64 64 64 64 1 69 69 69 69 1 74 74 74 1 0 0 1 77 77 1 0 0 0 1 85 85 85 85 1 94 94 94 1 0 0 0 0 0 "

+ "0 0 0 0 1 53 53 53 53 53 53 53 53 1 55 55 55 55 55 1 60 60 60 60 1 64 64 64 64 64 1 1 1 1 1 1 74 74 74 1 0 0 1 77 77 1 0 0 1 85 85 85 85 85 1 1 1 1 1 0 0 0 0 0 "

+ "0 0 0 0 1 1 1 1 1 1 1 1 1 1 55 55 55 55 55 1 60 60 60 60 1 1 1 1 1 1 1 70 70 70 70 1 1 1 1 1 0 1 77 77 1 0 0 1 1 1 1 1 1 1 1 96 1 95 1 0 0 0 0 0 "

+ "0 0 0 0 1 54 54 54 1 56 56 56 56 56 1 1 1 1 1 1 60 60 60 60 1 65 65 65 65 65 65 1 70 70 70 1 75 75 75 75 1 1 1 1 1 1 1 1 86 86 86 86 86 86 1 1 1 1 0 0 0 0 0 0 "

+ "0 0 0 0 1 54 54 54 1 56 56 56 56 56 56 1 57 57 57 1 60 60 60 60 1 65 65 65 65 65 65 1 70 70 70 1 75 75 75 75 1 78 78 1 82 82 82 1 86 86 86 86 86 1 97 97 1 0 0 0 0 0 0 0 "

+ "0 0 0 0 1 54 54 54 1 56 56 56 56 56 56 1 57 57 57 1 1 1 1 1 1 1 1 1 65 65 65 1 1 1 1 1 75 75 75 75 1 78 78 1 82 82 82 1 86 86 1 1 1 1 97 1 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 1 54 54 54 1 56 56 56 56 56 56 1 57 57 57 57 57 1 61 61 61 61 61 1 65 65 65 1 71 71 71 71 1 75 75 75 1 78 78 1 82 82 82 1 1 1 99 99 99 1 1 1 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 1 54 54 54 54 1 56 56 56 56 56 1 57 57 57 57 57 1 61 61 61 61 61 1 1 1 1 1 1 71 71 71 1 75 75 1 78 78 78 1 1 82 1 87 87 87 1 99 99 1 98 1 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 1 54 54 54 54 54 1 56 56 56 56 1 57 57 57 57 57 1 61 61 61 61 61 1 66 66 66 66 1 71 71 71 71 1 75 1 78 78 78 1 79 1 87 87 87 1 88 1 1 1 1 1 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 1 54 54 54 54 54 1 56 56 56 1 57 57 57 57 57 1 61 61 61 61 61 1 66 66 66 66 1 71 71 71 71 1 75 1 1 1 1 79 79 79 1 87 87 1 88 88 88 88 88 1 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 1 54 54 54 54 54 1 56 56 56 1 57 57 57 57 57 1 61 61 61 61 61 1 66 66 66 66 66 1 71 71 71 1 1 79 79 79 79 79 79 79 1 1 1 88 88 88 88 88 88 1 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 1 54 54 54 54 54 1 56 56 56 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 71 71 71 1 79 79 79 79 79 79 79 1 88 88 88 88 88 88 88 88 1 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 1 54 54 54 54 54 1 56 56 1 58 58 58 1 62 62 62 62 62 1 67 67 67 67 67 67 67 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 1 54 54 54 54 54 54 1 56 1 58 58 58 1 62 62 62 62 62 1 1 1 1 67 67 67 67 1 72 72 72 1 80 80 80 80 80 80 80 80 1 89 89 89 89 89 89 1 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 1 54 54 54 54 54 54 1 1 58 58 58 1 62 62 62 62 62 1 68 68 1 67 67 67 67 1 72 72 72 1 80 80 80 80 80 80 80 1 89 89 89 89 89 89 1 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 1 54 54 54 54 54 54 1 58 58 58 1 62 62 62 62 62 1 68 68 1 67 67 67 67 1 72 72 72 1 1 1 1 1 1 1 1 1 1 1 1 1 89 89 1 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 0 1 1 1 54 54 54 1 58 58 58 1 62 62 62 62 62 1 68 68 68 1 1 1 1 1 1 1 72 1 76 76 76 1 81 81 1 83 83 1 90 90 1 89 1 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 0 0 0 0 1 54 54 1 58 58 58 1 62 62 1 1 1 1 68 68 68 68 68 68 68 68 68 1 1 1 1 76 76 1 81 81 81 1 83 83 1 90 90 1 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 0 0 0 0 0 1 54 1 58 58 1 1 1 1 68 68 68 68 68 68 68 68 68 68 68 68 68 1 73 73 1 76 76 1 81 81 81 1 83 83 83 1 90 1 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 68 68 68 68 68 68 68 68 68 68 68 68 68 1 73 73 73 1 1 1 81 81 81 1 83 83 83 83 1 0 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 68 68 68 68 68 68 68 68 68 68 68 68 68 1 73 73 73 73 73 1 81 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 1 50 50 50 50 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 68 68 68 68 68 68 68 68 68 68 68 68 1 73 1 1 1 73 73 1 0 0 0 1 84 84 84 84 1 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 1 50 50 50 50 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 68 68 68 68 68 1 1 68 68 68 68 68 68 1 0 0 0 1 1 0 0 0 0 0 1 84 84 84 1 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 1 50 50 50 50 50 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 68 68 68 1 0 0 1 68 68 68 68 1 0 0 0 0 0 0 0 0 0 0 0 1 84 84 84 1 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 1 50 50 50 50 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 68 68 1 0 0 0 1 68 68 68 68 1 0 0 0 0 0 0 0 0 0 0 0 1 84 84 84 1 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 1 50 50 50 50 50 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 68 68 68 1 0 0 0 0 0 0 0 0 0 0 0 1 84 84 84 84 1 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 1 50 50 50 50 50 1 0 0 0 0 0 0 1 51 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 68 68 1 0 0 0 0 0 0 0 0 0 0 0 1 84 84 84 84 1 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 1 50 50 50 50 1 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 68 1 0 0 0 0 0 0 0 0 0 0 0 0 1 84 84 84 1 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 1 50 1 1 50 1 0 0 0 0 0 0 0 0 1 51 51 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 84 84 84 1 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 1 50 1 0 0 1 50 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 84 84 84 1 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 1 50 1 0 0 0 0 1 50 1 0 0 0 0 0 0 0 0 0 0 1 51 51 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 84 84 1 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 51 51 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 51 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "

+ "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ";

}

//a String array is created and each part of the array is saved to from saveFile seperated by spaces

split = saveFile.split("\\s+");

}

//method used for saving to file

private void save() {

//creating a way for data to be saved

saveFile = wins + " " + loses + " ";

for (int vertical = 0; vertical < 48; vertical++) {

for (int horizontal = 0; horizontal < 64; horizontal++) {

saveFile += stateDisplay[vertical][horizontal] + " ";

}

}

//saveFile is converted to byte data

byte data[] = saveFile.getBytes();

//byte data is saved to file using file io

try (OutputStream out = new BufferedOutputStream(

Files.newOutputStream(file, WRITE, TRUNCATE\_EXISTING))) {

out.write(data, 0, data.length);

} catch (IOException x) {

System.err.println(x);

}

}

}