**Cover to submit the Project - Winter 2016**

**Team members (Last name First name ID Number)**

1. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (coordinator) (coordinator) (coordinator)**
2. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
3. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |
| --- | --- | --- |
| **Features** | **Check on () if this feature is used else leave it blank** | |
| Java | Interface? Yes □ Non □ Abstract class? Yes □ Non □  Packages? Yes □ Non □ | |
| JavaFX | Bean? Yes □ Non □ Property? Yes □ Non □ | |
| OOP classes. How many? | 1 to 3? Yes □ Non □ 4 to 6? Yes □ Non □  More than 6? Yes □ Non □ | |
| Other software | Yes □ Non □ Specify: \_\_\_\_\_\_\_\_ | |
| Using any existing project/code | Yes □ Non □ If yes, amount estimate (in %): \_\_\_\_\_\_\_\_  Copyright reference (if any): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| Estimate amount of individual contribution (in %) and team work done | | | Last name First name Contribution in %  (Out of 100%)  1. \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_  2. \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_  3. \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_  Total amount of code submitted by the team: \_\_\_\_\_\_\_\_\_\_  (This total may be less than 100% but greater than 95%) |
| Amount of code working  (out of 100 %) | | |  |

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package InterfacePackage;

import javafx.scene.paint.Color;

public interface InterfConst {

final String MAIN\_TITLE = "Physics Simulator";

final String COURSE\_STRING = "Courses";

final String VIEW\_STRING = "View";

final String ANIMATION\_1 = "Animation 1";

final String ANIMATION\_2 = "Animation 2";

final String IN\_CONSTRUCTION = "In Construction";

final String IN\_CONSTRUCTION\_MESSAGE = "\t\tThis page is temporarily under construction\nPlease try a different animation using the courses menu";

final String EXIT\_STRING = "Exit";

final String SPACE\_STRING = " ";

final String FULLSCREEN\_STRING = "Toggle Fullscreen";

final String WELCOME\_MESSAGE = "\tWelcome to the Physics Simulator\n Click courses to begin your experience";

final String RETURN\_STRING = "Return";

final String START\_STRING = "Start";

final String STOP\_STRING = "Stop";

final String RESET\_STRING = "Reset";

final String PAUSE\_STRING = "Pause";

final String HELP\_STRING = "Help";

final float GRAV\_ACC = 9.8f;

final String NYA\_TITLE = "Mechanic 203-NYA";

final String NYB\_TITLE = "Electricity and Magnetism 203-NYB";

final String NYC\_TITLE = "Waves, Optics & Modern Physics 203-NYC";

final String NYA\_1 = "Inclined Planes";

final String NYA\_1\_FORMULA = "F = ma";

final String NYA\_2 = "Projectile Motion";

final String NYA\_2\_FORMULA = "Y = Vyo(t) – ½(g)t^2\nX = Vxo(t) – ½(g)t^2";

final String NYB\_1 = "Ohm's law";

final String NYB\_1\_FORMULA = "V = RI";

final String NYB\_2 = "Coulomb's law";

final String NYB\_2\_FORMULA = "F = (k |Q1Q2|) / r^2";

final String NYC\_1 = "Pendulum system";

final String NYC\_1\_FORMULA = "? = ?max cos(?(t) + f)\n? = sqrt(g/L)\nEtotal = m(g)L(1-cos(?max))";

final String NYC\_2 = "Nuclear Decay";

final String NYC\_2\_FORMULA = "N(t) = N(0)\*e^(-?\*t)\nT\_(1/2) = (ln 2) / ?";

final String TABLE\_TAB\_STRING = "Table Tab";

final String GRAPH\_TAB\_STRING = "Graph Tab";

//walter

final String DONE\_STRING = "Done";

final String NYB\_1\_HELP\_MESSAGE = "An object placed on a tilted surface will often slide down the surface."

+ " The rate at which the object slides down the surface is dependent upon how tilted the surface is"

+ " and how big is the gravitational force; the greater the tilt of the surface and/or the mass of"

+ " the object, the faster the rate at which the object will slide down it."

+ " In physics, a tilted surface is called an inclined plane.";

final String NYC\_1\_HELP\_MESSAGE = "A simple pendulum consists of a relatively massive object "

+ "hung by a string from a fixed support. It typically hangs vertically"

+ " in its equilibrium position. ";

final String INFORMATION\_STRING\_SHORT = "Information";

final String LINE\_BREAK = "\n";

final String YES\_STRING = "Yes";

final String NO\_STRING = "No";

final String CANCEL\_STRING = "Cancel";

final String OK\_STRING = "Ok";

final String LENGTH\_STRING = "Length L (in m)";

final String ANGULAR\_VELOCITY\_STRING = "Angular Velocity (in rad/sec)";

final String RESET\_TITLE = "Reset Values?";

final String ANGLE\_STRING = "Angle\t";

final String MASS\_STRING = "Mass\t";

final String QUESTION\_STRING = "Would you like to change the values of the variables?\n\n(Note: If you don't want to reset the animation, press Cancel)";

final String INSTRUCTION\_STRING = "Please adjust the following fields with the desired values and press Ok.\n\n";

// NYB\_2

final String HELP\_MESSAGE\_NYB\_2 = "Two charges initially at opposite charges attract eachother, as shown from the force arrows in black."

+ " As the distance between particles increase, the force will decrease."

+ " As the charges of the particles increase so will the forces."

+ " To begin, press start, and to change any the charges of the particles press the reset button.";

final int DISTANCE = 5;

final String DISTANCE\_LABEL = "Distance in meters";

final String FORCE\_LABEL = "Force in Netwons";

final String POS\_CHARGE = "+";

final String NEG\_CHARGE = "-";

final String PARTICLE\_ONE = "The charge of the first particle is: ";

final String PARTICLE\_TWO = "The charge of the second particle is: ";

final String UNITS\_CHARGE = "µC";

final int DEFAULT\_CHARGE\_ONE = 10;

final int DEFAULT\_CHARGE\_TWO = 10;

final String DEFAULT\_CHARGE\_SIGN\_ONE = POS\_CHARGE;

final String DEFAULT\_CHARGE\_SIGN\_TWO = NEG\_CHARGE;

final Color NEG\_CHARGE\_COLOR = Color.BLUE;

final Color POS\_CHARGE\_COLOR = Color.RED;

final double K\_CONSTANT = 8.99;

// NYC\_2

final String HELP\_MESSAGE\_NYC\_2 = "The decay of the an isotope is determine by the formula given."

+ " It requires the isotopes half life and the a specific time in years."

+ " To begin press start, and to see your chosen isotopes decay from 1000 - 10 000 years, press the reset button.";

final String HALF\_LIFE\_STRING = "The half life of";

final String NIOBIUM\_ISOTOPE = "Niobium-94";

final String CARBON\_ISOTOPE = "Carbon-14";

final String THORIUM\_ISOTOPE = "Thorium-229";

final String MOLYBDENUM\_ISOTOPE = "Molybdenum-93";

final String CURIUM\_ISOTOPE = "Curium-245";

final String YEARS\_UNIT = "Years";

final String INPUT\_DESCRIPTION = "Input years here";

final String DEFAULT\_USER\_INPUT = "DEFAULT YEARS OF PROJECTION:";

final String USER\_YEARS = "User chosen project is:";

final String G\_UNITS = "grams";

final int DEFAULT\_YEARS = 6000;

final int INITIAL\_SAMPLE = 100;

final double E\_CONSTANT = 2.71828;

final int NIOBIUM\_HALF\_LIFE = 20300;

final int CARBON\_HALF\_LIFE = 5730;

final int THORIUM\_HALF\_LIFE = 7340;

final int MOLYBDENUM\_HALF\_LIFE = 4000;

final int CURIUM\_HALF\_LIFE = 8500;

final String ERROR\_MESSAGE = "Please input a value from 0-6000 years.";

final String LABEL\_FINAL\_YEARS\_INFO = "Please pick a year to see its decay: ";

final String INFO\_LABEL = "Each gram is equivalent to approximately 1 particles.";

final int HALF\_LIFE\_GRAMS = 50;

//add this

final String HELP\_MESSAGE = "This animation is about Ohm's Law.\nChoose values for resistance and voltage " +

"and then click start to view the animation.\nClick pause to pause the animation at any time " +

"during the animation and click start to resume it.\nClick reset to be able to select " +

"new values for resistance and voltage and do the same process again.\n" +

"Click done to return to the main menu";

final String RESISTANCE\_1 = "10";

final String RESISTANCE\_2 = "20";

final String RESISTANCE\_3 = "50";

final String VOLTAGE\_1 = "4";

final String VOLTAGE\_2 = "6";

final String VOLTAGE\_3 = "10";

final int SPACE\_BETWEEN\_RADIO\_BUTTONS = 10;

final String CHOOSE\_RESISTANCE = "Please choose a resistance";

final String CHOOSE\_VOLTAGE = "Please choose a voltage";

final int NUMBER\_OF\_ELECTRONS = 73;

final int RADIUS = 6;

final int WIDTH\_POSITION = -22;

final int HEIGHT\_POSITION = 0;

final int WIDTH = 985;

final int HEIGHT = 200;

final int ELECTRONS\_ANIM\_DURATION = 5500;

final int ANIMATION\_DELAY = 75;

final String YOUR\_CHOICES = "Intensity from your choices: ";

final String CURRENT\_IN\_AMPS = "CURRENT IN AMPERES";

final String VOLTAGE = "VOLTAGE";

final String CURRENT = "CURRENT";

// 2nd anim

double speed = 95;

final String NYA\_2\_HELP\_MESSAGE = "This is an animation about projectile motion\n"

+ "Click start to play the animation.\nClick reset to change the angle of the animation"

+ "and click start again to view the animation with the new angle.";

final String INFORMATION\_STRING = "INFO";

final String CHANGING\_ANGLES = "Changing angles";

final String CHANGE\_ANGLE = "Change the angle of your choice: ";

final String CHANGE\_ANGLE\_QUESTION = "Would you like to change the angle?";

final String ANGLE\_RANGE = "Choose your angle from 5-90 here: ";

final String DEGREES = " Degrees";

final String X\_TIME = "Time (s)";

final String Y\_HEIGHT = "Height(m)";

final String ZERO = "0";

}

package InterfacePackage;

import javafx.scene.layout.BorderPane;

public abstract class ComLayout implements InterfConst{

public abstract BorderPane getPane();

public abstract void tableGraphTab();

public abstract void showResetPopup();

}

package MainPackage;

import BeanPackage.\*;

import InterfacePackage.InterfConst;

import java.time.Duration;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.Background;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.FontPosture;

import javafx.stage.Stage;

import javafx.scene.layout.StackPane;

import javafx.scene.media.AudioClip;

import javafx.stage.WindowEvent;

public class MainClass extends Application implements InterfConst {

static BorderPane root = new BorderPane();

static StackPane welcomePane = new StackPane();

//http://stackoverflow.com/questions/6845231/how-to-correctly-get-image-from-resources-folder-in-netbeans

AudioClip exitSound = new AudioClip(getClass().getResource("/exitSound.mp3").toString());

@Override

public void start(Stage stage) {

//Start of Menu Bar

MenuBar menuBar = new MenuBar();

Menu coursesMenu = new Menu(COURSE\_STRING);

Menu mechanicsItems = new Menu(NYA\_TITLE);

MenuItem mechAnimation1 = new MenuItem(ANIMATION\_1);

mechAnimation1.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

root.setCenter(new Bean\_NYA1(500, 30).getPane());

}

});

MenuItem mechAnimation2 = new MenuItem(ANIMATION\_2);

mechAnimation2.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

root.setCenter(new Bean\_NYA2().getPane());

}

});

MenuItem mechInConstruction = new MenuItem(IN\_CONSTRUCTION);

mechInConstruction.setOnAction(getInConstructionEvent());

mechanicsItems.getItems().addAll(mechAnimation1, mechAnimation2, mechInConstruction);

Menu electricityItems = new Menu(NYB\_TITLE);

MenuItem electAnimation1 = new MenuItem(ANIMATION\_1);

electAnimation1.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

root.setCenter(new Bean\_NYB1().getPane());

}

});

MenuItem electAnimation2 = new MenuItem(ANIMATION\_2);

electAnimation2.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

root.setCenter(new Bean\_NYB2().getPane());

}

});

MenuItem electInConstruction = new MenuItem(IN\_CONSTRUCTION);

electInConstruction.setOnAction(getInConstructionEvent());

electricityItems.getItems().addAll(electAnimation1, electAnimation2, electInConstruction);

Menu wavesItems = new Menu(NYC\_TITLE);

MenuItem wavesAnimation1 = new MenuItem(ANIMATION\_1);

wavesAnimation1.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

root.setCenter(new Bean\_NYC1(2, 120).getPane());

}

});

MenuItem wavesAnimation2 = new MenuItem(ANIMATION\_2);

wavesAnimation2.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

root.setCenter(new Bean\_NYC2().getPane());

}

});

MenuItem wavesInConstruction = new MenuItem(IN\_CONSTRUCTION);

wavesInConstruction.setOnAction(getInConstructionEvent());

wavesItems.getItems().addAll(wavesAnimation1, wavesAnimation2, wavesInConstruction);

MenuItem exit = new MenuItem(EXIT\_STRING);

exit.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent Event) {

exit();

}

});

coursesMenu.getItems().addAll(mechanicsItems, electricityItems, wavesItems, exit);

menuBar.getMenus().add(coursesMenu);

//End of Menu Bar

Label welcome = new Label(WELCOME\_MESSAGE);

welcome.setFont(Font.font("IMPACT", FontPosture.REGULAR, 35));

welcome.setTextFill(Color.BLUE);

BorderPane.setAlignment(welcome, Pos.CENTER);

welcomePane.getChildren().add(welcome);

welcomePane.setBackground(Background.EMPTY);

VBox topVBox = new VBox();

topVBox.setSpacing(10);

topVBox.getChildren().add(menuBar);

root.setTop(topVBox);

root.setCenter(welcomePane);

stage.setOnCloseRequest(new EventHandler<WindowEvent>() {

@Override

public void handle(WindowEvent event) {

exit();

}

});

stage.setScene(new Scene(root, 1000, 900));

stage.setResizable(false);

stage.centerOnScreen();

stage.setTitle(MAIN\_TITLE);

stage.show();

}

public static BorderPane getInConstructionPane() {

BorderPane bp = new BorderPane();

Label inConstruction = new Label(IN\_CONSTRUCTION\_MESSAGE);

inConstruction.setFont(Font.font("IMPACT", FontPosture.REGULAR, 35));

bp.setCenter(inConstruction);

return bp;

}

public static void returnToMain() {

root.setCenter(welcomePane);

}

public static void resetNYA1(int mass, double angle) {

root.setCenter(new Bean\_NYA1(mass, angle).getPane());

}

public static void resetNYC1(int mass, double angle) {

root.setCenter(new Bean\_NYC1(mass, angle).getPane());

}

public static EventHandler<ActionEvent> getInConstructionEvent() {

EventHandler<ActionEvent> event = new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

root.setCenter(getInConstructionPane());

}

};

return event;

}

public void exit() {

exitSound.play();

try {

Thread.sleep(500);

} catch (Exception ex) {

}

System.exit(0);

}

}

package BeanPackage;

import InterfacePackage.ComLayout;

import static InterfacePackage.InterfConst.GRAV\_ACC;

import MainPackage.MainClass;

import javafx.animation.Animation;

import javafx.animation.FadeTransition;

import javafx.animation.ParallelTransition;

import javafx.animation.TranslateTransition;

import javafx.beans.property.DoubleProperty;

import javafx.beans.property.IntegerProperty;

import javafx.beans.property.SimpleDoubleProperty;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.chart.NumberAxis;

import javafx.scene.chart.ScatterChart;

import javafx.scene.chart.XYChart;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.ButtonType;

import javafx.scene.control.Label;

import javafx.scene.control.Spinner;

import javafx.scene.control.SplitPane;

import javafx.scene.control.Tab;

import javafx.scene.control.TabPane;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.Pane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.shape.Line;

import javafx.scene.shape.Rectangle;

import javafx.scene.transform.Rotate;

import javafx.stage.Modality;

import javafx.stage.Stage;

import javafx.util.Duration;

public class Bean\_NYA1 extends ComLayout {

int rectSize;

int x1;

int x2;

int y1;

IntegerProperty mass = new SimpleIntegerProperty();

DoubleProperty angle = new SimpleDoubleProperty();

double radAngle;

double y2;

double rectShiftX;

double rectShiftY;

double rectInitX;

double rectInitY;

double duration;

Label titleLabel1;

Label titleLabel2;

Line groundLine;

Line inclinedLine;

Line verticalLine;

Rectangle square;

Label fgLabel;

Label nLabel;

Label ffLabel;

ParallelTransition mainAnimation;

Tab graphTab;

Tab tableTab;

public Bean\_NYA1(int mass, double angle) {

this.mass.set(mass);

this.angle.set(angle);

this.radAngle = this.angle.get() \* Math.PI / 180;

this.rectSize = 45;

this.x1 = 250;

this.x2 = 750;

this.y1 = 500;

this.y2 = y1 - (Math.tan(radAngle) \* (x2 - x1));

this.rectShiftX = rectSize \* Math.cos(radAngle);

this.rectShiftY = rectSize \* Math.sin(radAngle);

this.rectInitX = x2 - rectSize;

this.rectInitY = y2 - rectSize;

this.duration = ((1/radAngle) + (1/getMass())) \* 1.5;

}

public void showResetPopup() {

Button yesButton = new Button(YES\_STRING);

Button noButton = new Button(NO\_STRING);

Button cancelButton1 = new Button(CANCEL\_STRING);

Button cancelButton2 = new Button(CANCEL\_STRING);

Button okButton = new Button(OK\_STRING);

Label questionLabel = new Label(QUESTION\_STRING);

Label instructionsLabel = new Label(INSTRUCTION\_STRING);

Spinner massField = new Spinner(5, 1000, mass.get(), 5);

massField.setEditable(true);

Spinner angleField = new Spinner(5.0, 45.0, angle.get(), 1.0);

angleField.setEditable(true);

Label massLabel = new Label(MASS\_STRING);

Label angleLabel = new Label(ANGLE\_STRING);

//Start First Scene

HBox questionHBox = new HBox();

questionHBox.getChildren().addAll(yesButton, noButton, cancelButton1);

questionHBox.setAlignment(Pos.CENTER\_RIGHT);

questionHBox.setSpacing(8);

questionHBox.setPadding(new Insets(10));

BorderPane questionRoot = new BorderPane();

questionRoot.setCenter(questionLabel);

questionRoot.setBottom(questionHBox);

Scene questionScene = new Scene(questionRoot, 400, 100);

//End First Scene

//Start Second Scene

HBox variablesHBox = new HBox();

variablesHBox.getChildren().addAll(okButton, cancelButton2);

variablesHBox.setAlignment(Pos.CENTER);

variablesHBox.setSpacing(8);

variablesHBox.setPadding(new Insets(10));

HBox massHBox = new HBox();

massHBox.getChildren().addAll(massLabel, massField);

HBox angleHBox = new HBox();

angleHBox.getChildren().addAll(angleLabel, angleField);

VBox fieldsBox = new VBox();

fieldsBox.getChildren().addAll(massHBox, angleHBox);

fieldsBox.setSpacing(8);

BorderPane variablesRoot = new BorderPane();

variablesRoot.setTop(instructionsLabel);

variablesRoot.setCenter(fieldsBox);

variablesRoot.setBottom(variablesHBox);

Scene variablesScene = new Scene(variablesRoot, 500, 150);

//End Second Scene

Stage stage = new Stage();

yesButton.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

stage.setScene(variablesScene);

}

});

noButton.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

mainAnimation.playFromStart();

mainAnimation.pause();

stage.close();

}

});

cancelButton1.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

stage.close();

}

});

cancelButton2.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

stage.close();

}

});

okButton.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

mass.set((int) massField.getValue());

angle.set((double) angleField.getValue());

stage.close();

MainClass.resetNYA1(mass.get(), angle.get());

}

});

stage.setScene(questionScene);

stage.setResizable(false);

stage.centerOnScreen();

stage.setTitle(RESET\_TITLE);

stage.initModality(Modality.APPLICATION\_MODAL);

stage.show();

}

@Override

public BorderPane getPane() {

titleLabel1 = new Label(NYA\_TITLE + LINE\_BREAK + NYA\_1);

titleLabel2 = new Label(NYA\_1\_FORMULA);

groundLine = new Line(x1, y1, x2, y1);

inclinedLine = new Line(x1, y1, x2, y2);

verticalLine = new Line(x2, y1, x2, y2);

square = new Rectangle(rectInitX, rectInitY, rectSize, rectSize);

square.getTransforms().add(new Rotate(-angle.get(), x2, y2));

square.setFill(Color.LIGHTSTEELBLUE);

square.setStrokeWidth(5);

Label massLabel = new Label(Integer.toString(mass.get()) + "g");

massLabel.setTranslateX(square.getX());

massLabel.setTranslateY(square.getY() + rectSize/2);

TranslateTransition squareAnimation = new TranslateTransition(Duration.seconds(duration), square);

squareAnimation.setFromX(0);

squareAnimation.setFromY(0);

squareAnimation.setToX((-x2 + x1) + rectShiftX);

squareAnimation.setToY((y1 - y2) - rectShiftY);

FadeTransition massFade = new FadeTransition(Duration.seconds(1), massLabel);

massFade.setFromValue(1);

massFade.setToValue(0);

mainAnimation = new ParallelTransition();

mainAnimation.getChildren().addAll(squareAnimation, massFade);

Button start = new Button(START\_STRING);

Button done = new Button(DONE\_STRING);

Button pause = new Button(PAUSE\_STRING);

Button reset = new Button(RESET\_STRING);

Button help = new Button(HELP\_STRING);

start.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

mainAnimation.play();

start.setDisable(true);

pause.setDisable(false);

}

});

done.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

MainClass.returnToMain();

}

});

pause.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

if (mainAnimation.getStatus().equals(Animation.Status.RUNNING)) {

mainAnimation.pause();

}

pause.setDisable(true);

start.setDisable(false);

}

});

reset.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

pause.fire();

showResetPopup();

}

});

Alert helpMsg = new Alert(Alert.AlertType.INFORMATION, NYB\_1\_HELP\_MESSAGE, ButtonType.OK);

helpMsg.setTitle(HELP\_STRING);

helpMsg.setHeaderText(INFORMATION\_STRING\_SHORT);

help.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

helpMsg.show();

}

});

VBox topPane = new VBox();

topPane.getChildren().addAll(titleLabel1, titleLabel2);

Pane centerPane = new Pane();

centerPane.getChildren().addAll(groundLine, inclinedLine, verticalLine, square, massLabel);

HBox buttonPane = new HBox();

buttonPane.getChildren().addAll(start, done, pause, reset, help);

buttonPane.setAlignment(Pos.CENTER);

buttonPane.setSpacing(10);

TabPane graphPane = new TabPane();

graphTab = new Tab(GRAPH\_TAB\_STRING);

tableTab = new Tab(TABLE\_TAB\_STRING);

graphTab.setClosable(false);

tableTab.setClosable(false);

graphPane.getTabs().addAll(graphTab, tableTab);

tableGraphTab();

SplitPane bottomPane = new SplitPane();

bottomPane.getItems().addAll(buttonPane, graphPane);

bottomPane.setDividerPositions(0.5f);

bottomPane.setPrefHeight(160);

bottomPane.setPrefWidth(200);

bottomPane.setMinHeight(250);

BorderPane root = new BorderPane();

root.setBottom(bottomPane);

root.setCenter(centerPane);

root.setTop(topPane);

return root;

}

public int getMass(){

return mass.get();

}

public double getAngle(){

return angle.get();

}

public void setMass(int mass){

this.mass.set(mass);

}

public void setAngle(double angle){

this.angle.set(angle);

}

public void tableGraphTab() {

int[] xGraphPoints = new int[10];

int temp = 1;

VBox xTabPoints = new VBox();

xTabPoints.setAlignment(Pos.CENTER);

Label xLabel = new Label("Mass (in kg)");

xTabPoints.getChildren().add(xLabel);

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

xGraphPoints[i] = temp;

xTabPoints.getChildren().add(new Label(Integer.toString(xGraphPoints[i])));

temp++;

}

double[] yGraphPoints = new double[10];

VBox yTabPoints = new VBox();

yTabPoints.setAlignment(Pos.CENTER);

Label yLabel = new Label("Force (in N)");

yTabPoints.getChildren().add(yLabel);

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

yGraphPoints[i] = GRAV\_ACC \* xGraphPoints[i];

yTabPoints.getChildren().add(new Label(String.format("%.2f", yGraphPoints[i])));

}

HBox xyTable = new HBox();

xyTable.setSpacing(5);

xyTable.setAlignment(Pos.CENTER);

xyTable.getChildren().addAll(xTabPoints, yTabPoints);

tableTab.setContent(xyTable);

//Graphing

NumberAxis xAxis = new NumberAxis(0, 10, 1);

NumberAxis yAxis = new NumberAxis(0, 98, 5);

ScatterChart<Number, Number> scatterGraph = new ScatterChart<>(xAxis, yAxis);

xAxis.setLabel("Mass (in g)");

yAxis.setLabel("Force (in N)");

scatterGraph.setTitle(NYA\_1);

scatterGraph.setLegendVisible(false);

XYChart.Series points = new XYChart.Series();

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

points.getData().add(new XYChart.Data(xGraphPoints[i], yGraphPoints[i]));

}

scatterGraph.getData().addAll(points);

graphTab.setContent(scatterGraph);

}

}

package BeanPackage;

import InterfacePackage.ComLayout;

import MainPackage.MainClass;

import java.io.InputStream;

import static java.lang.Math.\*;

import java.util.Optional;

import javafx.animation.\*;

import javafx.beans.property.\*;

import javafx.event.\*;

import javafx.geometry.Pos;

import javafx.scene.chart.\*;

import javafx.scene.control.\*;

import javafx.scene.image.\*;

import javafx.scene.layout.\*;

import javafx.scene.paint.Color;

import javafx.scene.shape.\*;

import javafx.scene.transform.Rotate;

import javafx.util.Duration;

public class Bean\_NYA2 extends ComLayout {

Button start;

Button done;

Button pause;

Button reset;

Button help;

Image grass;

ImageView grassImageView;

Rectangle cannon;

Circle cannonPiece;

Circle wheel;

Circle bullet;

Path animationPath;

PathTransition pathTransition;

TextInputDialog chooseAngle;

Optional<String> result;

IntegerProperty angleValue;

double speedX;

double speedY;

double horizontalDistance;

double verticalDistance;

double time;

ArcTo finalPosition;

NumberAxis xAxis;

NumberAxis yAxis;

LineChart<Number, Number> lineChart;

XYChart.Series series;

Tab graphTab;

Tab tableTab;

Spinner varInput;

Label formulaLabel;

InputStream GRASS\_URL = getClass().getResourceAsStream("/Grass.png");

@Override

public BorderPane getPane() {

angleValue = new SimpleIntegerProperty(45);

start = new Button(START\_STRING);

done = new Button(DONE\_STRING);

pause = new Button(PAUSE\_STRING);

reset = new Button(RESET\_STRING);

help = new Button(HELP\_STRING);

BorderPane root = new BorderPane();

Alert helpMsg = new Alert(Alert.AlertType.INFORMATION, null, ButtonType.OK);

helpMsg.setTitle(HELP\_STRING);

helpMsg.setContentText(NYA\_2\_HELP\_MESSAGE);

help.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

helpMsg.show();

}

});

HBox buttonPane = new HBox();

buttonPane.getChildren().addAll(start, done, pause, reset, help);

buttonPane.setAlignment(Pos.CENTER);

buttonPane.setSpacing(10);

TabPane graphPane = new TabPane();

graphTab = new Tab(GRAPH\_TAB\_STRING);

tableTab = new Tab(TABLE\_TAB\_STRING);

graphTab.setClosable(false);

tableTab.setClosable(false);

graphPane.getTabs().addAll(graphTab, tableTab);

SplitPane bottomPane = new SplitPane();

bottomPane.getItems().addAll(buttonPane, graphPane);

bottomPane.setDividerPositions(0.5f);

bottomPane.setPrefHeight(160);

bottomPane.setPrefWidth(200);

root.setBottom(bottomPane);

Pane animation = new Pane();

createBackground();

buildCannon();

createAnimation();

animation.getChildren().addAll(grassImageView, bullet, cannon, cannonPiece, wheel, formulaLabel);

root.setCenter(animation);

start.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent Event) {

start.setDisable(true);

pause.setDisable(false);

bullet.setVisible(true);

getChosenAngle();

tableGraphTab();

pathTransition.play();

}

});

done.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent Event) {

MainClass.returnToMain();

start.setDisable(false);

}

});

pause.setDisable(true);

pause.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent Event) {

start.setDisable(false);

bullet.setVisible(true);

pathTransition.pause();

}

});

Alert resetPopUp = new Alert(Alert.AlertType.CONFIRMATION, "Choose your option: ", ButtonType.YES, ButtonType.NO, ButtonType.CANCEL);

resetPopUp.setTitle(CHANGING\_ANGLES);

resetPopUp.setHeaderText(CHANGE\_ANGLE\_QUESTION);

reset.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent Event) {

pathTransition.jumpTo(Duration.ZERO);

pathTransition.stop();

Optional<ButtonType> result = resetPopUp.showAndWait();

if (result.isPresent() && result.get() == ButtonType.YES) {

pause.fire();

showResetPopup();

} else if (result.isPresent() && result.get() == ButtonType.NO) {

pause.fire();

} else {

}

}

});

cannon.getTransforms().clear();

cannon.getTransforms().add(new Rotate(-1 \* getChosenAngle(), 0, 0));

start.setDisable(false);

return root;

}

public void createBackground() {

formulaLabel = new Label(NYA\_2\_FORMULA);

formulaLabel.setLayoutX(0);

formulaLabel.setLayoutY(0);

//File file = new File(GRASS\_URL);

//grass = new Image(file.toURI().toString());

grass = new Image(GRASS\_URL);

grassImageView = new ImageView();

grassImageView.setFitHeight(60);

grassImageView.setFitWidth(1100);

grassImageView.setImage(grass);

grassImageView.setLayoutX(0);

grassImageView.setLayoutY(665);

}

public void createAnimation() {

animationPath = new Path();

MoveTo moveTo = new MoveTo();

moveTo.setX(0);

moveTo.setY(0);

finalPosition = new ArcTo();

finalPosition.setX(getHorizontalDistance());

finalPosition.setY(0);

finalPosition.setRadiusX(getHorizontalDistance() / 20);

finalPosition.setRadiusY(getVerticalDistance() / 20);

finalPosition.setLargeArcFlag(false);

finalPosition.setSweepFlag(true);

animationPath.getElements().add(moveTo);

animationPath.getElements().add(finalPosition);

pathTransition = new PathTransition(Duration.millis(getTime() \* 300), animationPath, bullet);

pathTransition.setCycleCount(Timeline.INDEFINITE);

pathTransition.setAutoReverse(false);

}

public void buildCannon() {

cannon = new Rectangle(0, 0, 100, 40);

cannon.setFill(Color.DIMGRAY);

cannon.setLayoutX(20);

cannon.setLayoutY(677);

cannon.setStroke(Color.BLACK);

cannon.setStrokeWidth(3);

cannonPiece = new Circle();

cannonPiece.setFill(Color.DIMGRAY);

cannonPiece.setRadius(30);

cannonPiece.setLayoutX(36);

cannonPiece.setLayoutY(693);

cannonPiece.setStroke(Color.BLACK);

cannonPiece.setStrokeWidth(3);

wheel = new Circle();

wheel.setRadius(25);

wheel.setFill(Color.PERU);

wheel.setLayoutX(50);

wheel.setLayoutY(705);

wheel.setStroke(Color.BLACK);

wheel.setStrokeWidth(3);

bullet = new Circle();

bullet.setLayoutX(38);

bullet.setLayoutY(685);

bullet.setRadius(10);

bullet.setFill(Color.BLACK);

bullet.setVisible(false);

}

public void showResetPopup() {

Dialog changeValues = new Dialog();

changeValues.setContentText(CHANGING\_ANGLES);

changeValues.setHeaderText(CHANGE\_ANGLE);

changeValues.getDialogPane().getButtonTypes().addAll(ButtonType.OK, ButtonType.CANCEL);

HBox hbLabels = new HBox();

VBox vbPane = new VBox();

Label angleLabel = new Label(ANGLE\_RANGE);

varInput = new Spinner(5, 90, getChosenAngle(), 5);

Label angleUnit = new Label(DEGREES);

hbLabels.getChildren().addAll(angleLabel, varInput, angleUnit);

hbLabels.setSpacing(5);

vbPane.getChildren().add(hbLabels);

changeValues.getDialogPane().setContent(vbPane);

Optional<ButtonType> result = changeValues.showAndWait();

if (result.isPresent() && result.get() == ButtonType.OK) {

setChosenAngle(Integer.parseInt(varInput.getValue().toString()));

finalPosition.setX(getHorizontalDistance());

finalPosition.setY(0);

finalPosition.setRadiusX(getHorizontalDistance() / 20);

finalPosition.setRadiusY(getVerticalDistance() / 20);

cannon.getTransforms().clear();

tableGraphTab();

cannon.getTransforms().add(new Rotate(-1 \* getChosenAngle(), 0, 0));

} else {

changeValues.close();

}

}

private int getChosenAngle() {

return angleValue.getValue();

}

private void setChosenAngle(int userInput) {

angleValue.setValue(userInput);

}

public double getHorizontalDistance() {

speedX = speed \* cos(getChosenAngle() \* (PI / 180));

horizontalDistance = speedX \* getTime();

return horizontalDistance;

}

public double getVerticalDistance() {

speedY = speed \* sin(getChosenAngle() \* (PI / 180));

verticalDistance = Math.pow(speedY, 2) / (2 \* GRAV\_ACC);

return verticalDistance;

}

public double getTime() {

time = (2 \* speed \* sin(getChosenAngle() \* (PI / 180))) / GRAV\_ACC;

return time;

}

public void tableGraphTab() {

xAxis = new NumberAxis();

yAxis = new NumberAxis();

xAxis.setLabel(X\_TIME);

yAxis.setLabel(Y\_HEIGHT);

lineChart = new LineChart<>(xAxis, yAxis);

series = new XYChart.Series();

series.getData().add(new XYChart.Data(0, 0));

series.getData().add(new XYChart.Data((double) getHorizontalDistance() / 2, (double) getVerticalDistance()));

series.getData().add(new XYChart.Data((double) getHorizontalDistance(), 0));

lineChart.setCreateSymbols(false);

lineChart.getData().add(series);

graphTab.setContent(lineChart);

VBox xTabPoints = new VBox();

xTabPoints.setAlignment(Pos.CENTER);

xTabPoints.setSpacing(5);

xTabPoints.getChildren().addAll(new Label(X\_TIME), new Label(ZERO), new Label(String.format("%.4f", (double) getHorizontalDistance() / 2)), new Label(String.format("%.4f", (double) getHorizontalDistance())));

VBox yTabPoints = new VBox();

yTabPoints.setAlignment(Pos.CENTER);

yTabPoints.setSpacing(5);

yTabPoints.getChildren().addAll(new Label(Y\_HEIGHT), new Label(ZERO), new Label(String.format("%.4f", (double) getVerticalDistance())), new Label(ZERO));

HBox table = new HBox();

table.setSpacing(5);

table.getChildren().addAll(xTabPoints, yTabPoints);

table.setAlignment(Pos.CENTER);

tableTab.setContent(table);

}

}

package BeanPackage;

import InterfacePackage.ComLayout;

import MainPackage.MainClass;

import java.io.InputStream;

import javafx.animation.\*;

import javafx.beans.property.\*;

import javafx.beans.value.\*;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Pos;

import javafx.scene.chart.\*;

import javafx.scene.control.\*;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.Pane;

import javafx.scene.layout.StackPane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.shape.\*;

import javafx.scene.text.\*;

import javafx.util.Duration;

public class Bean\_NYB1 extends ComLayout {

DoubleProperty resistance = new SimpleDoubleProperty(10);

DoubleProperty voltage = new SimpleDoubleProperty(4);

Button start;

Button done;

Button pause;

Button reset;

Button help;

Label chooseResistance;

Label chooseVoltage;

ToggleGroup resistanceGroup;

RadioButton resistance1;

RadioButton resistance2;

RadioButton resistance3;

ToggleGroup voltageGroup;

RadioButton voltage1;

RadioButton voltage2;

RadioButton voltage3;

Label formulaInfo;

Rectangle circuit;

Rectangle circuitDesign1;

Rectangle circuitDesign2;

Circle[] electrons;

Polygon animationPath;

PathTransition[] pathTransition;

LineChart<Number, Number> lineChart;

XYChart.Series series;

Image lightBulb;

ImageView lightBulbImageView;

Image noLightBulb;

ImageView noLightBulbImageView;

Image battery;

ImageView batteryImageView;

ScatterChart<Number, Number> scatterGraph;

Tab graphTab;

Tab tableTab;

boolean isAnimationReady;

InputStream BATTERY\_URL = getClass().getResourceAsStream("/battery.png");

InputStream LIGHTED\_LIGHT\_BULB\_URL = getClass().getResourceAsStream("/lightBulb\_Light.png");

InputStream UNLIGHTED\_LIGHT\_BULB\_URL = getClass().getResourceAsStream("/lightBulb\_NoLight.jpg");

@Override

public BorderPane getPane() {

start = new Button(START\_STRING);

done = new Button(DONE\_STRING);

pause = new Button(PAUSE\_STRING);

reset = new Button(RESET\_STRING);

help = new Button(HELP\_STRING);

BorderPane root = new BorderPane();

Alert helpMsg = new Alert(Alert.AlertType.INFORMATION, null, ButtonType.OK);

helpMsg.setTitle(HELP\_STRING);

helpMsg.setContentText(HELP\_MESSAGE);

help.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

helpMsg.show();

}

});

HBox buttonPane = new HBox();

buttonPane.getChildren().addAll(start, done, pause, reset, help);

buttonPane.setAlignment(Pos.CENTER);

buttonPane.setSpacing(10);

TabPane graphPane = new TabPane();

graphTab = new Tab(GRAPH\_TAB\_STRING);

tableTab = new Tab(TABLE\_TAB\_STRING);

graphTab.setClosable(false);

tableTab.setClosable(false);

graphPane.getTabs().addAll(graphTab, tableTab);

SplitPane bottomPane = new SplitPane();

bottomPane.getItems().addAll(buttonPane, graphPane);

bottomPane.setDividerPositions(0.5f);

bottomPane.setPrefHeight(160);

bottomPane.setPrefWidth(200);

root.setBottom(bottomPane);

GridPane radioButtonsGroup = new GridPane();

chooseResistance = new Label(CHOOSE\_RESISTANCE);

chooseResistance.setFont(Font.font("IMPACT", FontPosture.REGULAR, 15));

chooseVoltage = new Label(CHOOSE\_VOLTAGE);

chooseVoltage.setFont(Font.font("IMPACT", FontPosture.REGULAR, 15));

resistanceGroup = new ToggleGroup();

resistance1 = new RadioButton(RESISTANCE\_1);

resistance1.setToggleGroup(resistanceGroup);

resistance2 = new RadioButton(RESISTANCE\_2);

resistance2.setToggleGroup(resistanceGroup);

resistance3 = new RadioButton(RESISTANCE\_3);

resistance3.setToggleGroup(resistanceGroup);

voltageGroup = new ToggleGroup();

voltage1 = new RadioButton(VOLTAGE\_1);

voltage1.setToggleGroup(voltageGroup);

voltage2 = new RadioButton(VOLTAGE\_2);

voltage2.setToggleGroup(voltageGroup);

voltage3 = new RadioButton(VOLTAGE\_3);

voltage3.setToggleGroup(voltageGroup);

formulaInfo = new Label(NYB\_1 + NYB\_1\_FORMULA);

formulaInfo.setFont(Font.font("IMPACT", FontPosture.REGULAR, 15));

radioButtonsGroup.addColumn(0, formulaInfo);

radioButtonsGroup.addColumn(1, chooseResistance, resistance1, resistance2, resistance3);

radioButtonsGroup.addColumn(2, chooseVoltage, voltage1, voltage2, voltage3);

radioButtonsGroup.hgapProperty().setValue(50);

radioButtonsGroup.vgapProperty().setValue(10);

root.setTop(radioButtonsGroup);

StackPane animation = new StackPane();

showingImages();

createCircuit();

initiateAnimation();

StackPane.setAlignment(circuitDesign2, Pos.TOP\_CENTER);

electrons = new Circle[NUMBER\_OF\_ELECTRONS];

createAnimationPath();

Pane electronsPane = new Pane();

pathTransition = new PathTransition[NUMBER\_OF\_ELECTRONS];

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

electrons[i] = new Circle();

electrons[i].setRadius(RADIUS);

electrons[i].setFill(Color.YELLOW);

electrons[i].setLayoutX(492.5);

electrons[i].setLayoutY(30.0);

electrons[i].setVisible(false);

pathTransition[i] = new PathTransition(Duration.millis(ELECTRONS\_ANIM\_DURATION), animationPath, electrons[i]);

pathTransition[i].setInterpolator(Interpolator.LINEAR);

pathTransition[i].setCycleCount(Timeline.INDEFINITE);

pathTransition[i].setAutoReverse(false);

pathTransition[i].setDelay(Duration.millis(ANIMATION\_DELAY \* i));

electronsPane.getChildren().addAll(electrons[i]);

}

animation.getChildren().addAll(circuit, circuitDesign1, circuitDesign2, lightBulbImageView, noLightBulbImageView, batteryImageView, electronsPane);

start.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent Event) {

pause.setDisable(false);

isAnimationReady = false;

tableGraphTab();

startAnimation();

}

});

done.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent Event) {

MainClass.returnToMain();

}

});

pause.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent Event) {

pauseAnimation();

}

});

reset.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent Event) {

resetAnimation();

initiateAnimation();

}

});

root.setLeft(animation);

return root;

}

public void showingImages() {

//File file = new File(BATTERY\_URL);

//battery = new Image(file.toURI().toString());

battery = new Image(BATTERY\_URL);

batteryImageView = new ImageView();

batteryImageView.setFitHeight(75);

batteryImageView.setFitWidth(75);

batteryImageView.setImage(battery);

StackPane.setAlignment(batteryImageView, Pos.TOP\_CENTER);

//File file1 = new File(LIGHTED\_LIGHT\_BULB\_URL);

//lightBulb = new Image(file1.toURI().toString());

lightBulb = new Image(LIGHTED\_LIGHT\_BULB\_URL);

lightBulbImageView = new ImageView();

lightBulbImageView.setFitHeight(80);

lightBulbImageView.setFitWidth(50);

lightBulbImageView.setImage(lightBulb);

StackPane.setAlignment(lightBulbImageView, Pos.BOTTOM\_CENTER);

lightBulbImageView.setVisible(false);

//File file2 = new File(UNLIGHTED\_LIGHT\_BULB\_URL);

//noLightBulb = new Image(file2.toURI().toString());

noLightBulb = new Image(UNLIGHTED\_LIGHT\_BULB\_URL);

noLightBulbImageView = new ImageView();

noLightBulbImageView.setFitHeight(75);

noLightBulbImageView.setFitWidth(75);

noLightBulbImageView.setImage(noLightBulb);

StackPane.setAlignment(noLightBulbImageView, Pos.BOTTOM\_CENTER);

noLightBulbImageView.setVisible(true);

}

public void createCircuit() {

circuit = new Rectangle(0, 0, 1000, 625);

circuit.setFill(Color.DIMGRAY);

circuitDesign1 = new Rectangle(0, 0, 970, 595);

circuitDesign1.setFill(Color.WHITESMOKE);

circuitDesign2 = new Rectangle(0, 0, 35, 20);

circuitDesign2.setFill(Color.WHITESMOKE);

}

public void createAnimationPath() {

animationPath = new Polygon();

animationPath.getPoints().addAll(new Double[]{

6.0, 10.0,

-16.0, 10.0,

-16.0, -20.0,

-486.5, -20.0,

-486.5, 590.0,

0.0, 590.0,

0.0, 540.0,

16.0, 540.0,

16.0, 590.0,

498.5, 590.0,

498.5, -20.0,

31.0, -20.0,

31.0, 10.0

});

}

public void initiateAnimation() {

start.setDisable(true);

pause.setDisable(true);

reset.setDisable(true);

resistanceGroup.selectedToggleProperty().addListener(new ChangeListener<Toggle>() {

@Override

public void changed(ObservableValue<? extends Toggle> ov, Toggle old\_toggle, Toggle new\_toggle) {

RadioButton resistanceRadioButton = (RadioButton) new\_toggle.getToggleGroup().getSelectedToggle();

String choosenResistance = resistanceRadioButton.getText();

double resistance = Double.parseDouble(choosenResistance);

setResistance(resistance);

if (resistanceGroup.getSelectedToggle().isSelected() == true) {

resistance1.setDisable(true);

resistance2.setDisable(true);

resistance3.setDisable(true);

}

voltageGroup.selectedToggleProperty().addListener(new ChangeListener<Toggle>() {

@Override

public void changed(ObservableValue<? extends Toggle> ov, Toggle old\_toggle, Toggle new\_toggle) {

RadioButton voltageRadioButton = (RadioButton) new\_toggle.getToggleGroup().getSelectedToggle();

String choosenVoltage = voltageRadioButton.getText();

double voltage = Double.parseDouble(choosenVoltage);

setVoltage(voltage);

if (voltageGroup.getSelectedToggle().isSelected() == true) {

voltage1.setDisable(true);

voltage2.setDisable(true);

voltage3.setDisable(true);

}

if ((voltageGroup.getSelectedToggle().isSelected() == true) && (resistanceGroup.getSelectedToggle().isSelected() == true)) {

isAnimationReady = true;

start.setDisable(false);

}

}

});

}

});

}

public void tableGraphTab() {

double[] xGraphPoints = new double[10];

double temp = 2;

VBox xTabPoints = new VBox();

xTabPoints.setAlignment(Pos.CENTER);

Label xLabel = new Label(VOLTAGE);

xTabPoints.getChildren().add(xLabel);

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

xGraphPoints[i] = temp;

xTabPoints.getChildren().add(new Label(Double.toString(xGraphPoints[i])));

temp += 2;

}

double[] yGraphPoints = new double[10];

VBox yTabPoints = new VBox();

yTabPoints.setAlignment(Pos.CENTER);

Label yLabel = new Label(CURRENT\_IN\_AMPS);

yTabPoints.getChildren().add(yLabel);

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

yGraphPoints[i] = (double) (xGraphPoints[i] / getResistance());

yTabPoints.getChildren().add(new Label(Double.toString(yGraphPoints[i])));

}

HBox xyTable = new HBox();

xyTable.setSpacing(5);

Label result = new Label(YOUR\_CHOICES + getVoltage() / getResistance());

xyTable.setAlignment(Pos.CENTER);

xyTable.getChildren().addAll(xTabPoints, yTabPoints, result);

tableTab.setContent(xyTable);

//Graphing

NumberAxis xAxis = new NumberAxis(0, 20, 1);

NumberAxis yAxis = new NumberAxis(0, 1, 0.1);

scatterGraph = new ScatterChart<>(xAxis, yAxis);

xAxis.setLabel(VOLTAGE);

yAxis.setLabel(CURRENT);

scatterGraph.setTitle(NYB\_1);

scatterGraph.setLegendVisible(false);

XYChart.Series points = new XYChart.Series();

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

points.getData().add(new XYChart.Data(xGraphPoints[i], yGraphPoints[i]));

}

scatterGraph.getData().addAll(points);

graphTab.setContent(scatterGraph);

}

private double getVoltage() {

return voltage.getValue();

}

private void setVoltage(double voltageValue) {

voltage.setValue(voltageValue);

}

private double getResistance() {

return resistance.getValue();

}

private void setResistance(double voltageValue) {

resistance.setValue(voltageValue);

}

public void startAnimation() {

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

pathTransition[i].play();

electrons[i].setVisible(true);

}

noLightBulbImageView.setVisible(false);

lightBulbImageView.setVisible(true);

start.setDisable(true);

reset.setDisable(false);

}

public void pauseAnimation() {

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

pathTransition[i].pause();

start.setDisable(false);

}

}

public void resetAnimation() {

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

pathTransition[i].jumpTo(Duration.ZERO);

pathTransition[i].stop();

electrons[i].setVisible(false);

}

noLightBulbImageView.setVisible(true);

lightBulbImageView.setVisible(false);

resistance1.setDisable(false);

resistance2.setDisable(false);

resistance3.setDisable(false);

resistance1.setSelected(false);

resistance2.setSelected(false);

resistance3.setSelected(false);

voltage1.setDisable(false);

voltage2.setDisable(false);

voltage3.setDisable(false);

voltage1.setSelected(false);

voltage2.setSelected(false);

voltage3.setSelected(false);

reset.setDisable(true);

start.setDisable(false);

scatterGraph.getData().clear();

if (isAnimationReady == true) {

tableGraphTab();

}

}

public void showResetPopup() {

}

}

package BeanPackage;

import InterfacePackage.ComLayout;

import MainPackage.MainClass;

import java.util.Optional;

import javafx.animation.\*;

import javafx.beans.property.\*;

import javafx.event.\*;

import javafx.geometry.Pos;

import javafx.scene.Node;

import javafx.scene.chart.\*;

import javafx.scene.control.\*;

import javafx.scene.control.Alert.\*;

import javafx.scene.layout.\*;

import javafx.scene.paint.Color;

import javafx.scene.shape.\*;

import javafx.util.Duration;

public class Bean\_NYB2 extends ComLayout {

Color partOneColor = POS\_CHARGE\_COLOR; //change color depending on charge sign

Color partTwoColor = NEG\_CHARGE\_COLOR; //change color depending on charge sign

Label particleOne = new Label(PARTICLE\_ONE + SPACE\_STRING + DEFAULT\_CHARGE\_SIGN\_ONE + DEFAULT\_CHARGE\_ONE + SPACE\_STRING + UNITS\_CHARGE);

Label particleTwo = new Label(PARTICLE\_TWO + SPACE\_STRING + DEFAULT\_CHARGE\_SIGN\_TWO + DEFAULT\_CHARGE\_TWO + SPACE\_STRING + UNITS\_CHARGE);

Label mainTitle = new Label(NYB\_TITLE + LINE\_BREAK + NYB\_2);

Label formulaLabel = new Label(NYB\_2\_FORMULA);

Pane animationPane = new Pane();

Node secondArrow;

Node firstArrow;

Polygon animationArrowOne;

Polygon animationArrowTwo;

Circle Particle\_1;

Circle Particle\_2;

ScaleTransition firstArrowScaleAnimation;

ScaleTransition secondArrowScaleAnimation;

TranslateTransition transFirstArrow;

TranslateTransition transSecondArrow;

ComboBox chargeSignsOne;

ComboBox chargeSignsTwo;

int circleRadius;

int finalDest;

IntegerProperty userInputVarOne = new SimpleIntegerProperty();

IntegerProperty userInputVarTwo = new SimpleIntegerProperty();

StringProperty signPropertyParticleTwo = new SimpleStringProperty();

StringProperty signPropertyParticleOne = new SimpleStringProperty();

TextField variableTwoInput;

Spinner varOneInput;

Spinner varTwoInput;

TextField variableOneInput;

Tab tableTab;

Tab graphTab;

@Override

public BorderPane getPane() {

setUserInputVarOne(DEFAULT\_CHARGE\_ONE);

setUserInputVarTwo(DEFAULT\_CHARGE\_TWO);

circleRadius = 50;

finalDest = 350;

Particle\_1 = new Circle(100, 300, circleRadius, partOneColor);

Particle\_2 = new Circle(900, 300, circleRadius, partTwoColor);

animationArrowOne = getAttractionRightArrow();

animationArrowTwo = getAttractionLeftArrow();

//Adding to animation pane

animationPane.getChildren().addAll(Particle\_1, Particle\_2, animationArrowOne, animationArrowTwo);

firstArrow = animationPane.getChildren().get(2);

secondArrow = animationPane.getChildren().get(3);

//Title pane

VBox VBoxTopPane = new VBox();

VBoxTopPane.getChildren().addAll(mainTitle, formulaLabel, particleOne, particleTwo);

//Translate anim for particles

TranslateTransition transFirstParticle = new TranslateTransition(Duration.seconds(2), Particle\_1);

transFirstParticle.setByX(finalDest);

transFirstParticle.setAutoReverse(true);

transFirstParticle.setCycleCount(Timeline.INDEFINITE);

TranslateTransition transSecondParticle = new TranslateTransition(Duration.seconds(2), Particle\_2);

transSecondParticle.setByX(-finalDest);

transSecondParticle.setAutoReverse(true);

transSecondParticle.setCycleCount(Timeline.INDEFINITE);

//Scale anim for arrows

firstArrowScaleAnimation = new ScaleTransition(Duration.seconds(2), firstArrow);

firstArrowScaleAnimation.setByX(1.5f);

firstArrowScaleAnimation.setAutoReverse(true);

firstArrowScaleAnimation.setCycleCount(Timeline.INDEFINITE);

secondArrowScaleAnimation = new ScaleTransition(Duration.seconds(2), secondArrow);

secondArrowScaleAnimation.setByX(1.5f);

secondArrowScaleAnimation.setAutoReverse(true);

secondArrowScaleAnimation.setCycleCount(Timeline.INDEFINITE);

//Tranlate anim for arrows

transFirstArrow = new TranslateTransition(Duration.seconds(2), firstArrow);

transFirstArrow.setByX(-finalDest);

transFirstArrow.setAutoReverse(true);

transFirstArrow.setCycleCount(Timeline.INDEFINITE);

transSecondArrow = new TranslateTransition(Duration.seconds(2), secondArrow);

transSecondArrow.setByX(finalDest);

transSecondArrow.setAutoReverse(true);

transSecondArrow.setCycleCount(Timeline.INDEFINITE);

//Adding all tranisitions

ParallelTransition arrowFirstAnim = new ParallelTransition(firstArrowScaleAnimation, transFirstArrow);

ParallelTransition arrowSecondAnim = new ParallelTransition(secondArrowScaleAnimation, transSecondArrow);

ParallelTransition NYB\_animation = new ParallelTransition(transFirstParticle, transSecondParticle, arrowFirstAnim, arrowSecondAnim);

Button start = new Button(START\_STRING);

Button done = new Button(DONE\_STRING);

Button pause = new Button(PAUSE\_STRING);

pause.setDisable(true);

Button reset = new Button(RESET\_STRING);

Alert resetPopUp = new Alert(AlertType.CONFIRMATION, "Choose your option: ", ButtonType.YES, ButtonType.NO, ButtonType.CANCEL);

resetPopUp.setTitle("Change Values?");

resetPopUp.setHeaderText("Would you like to change the charges of the two particle?");

//Defining buttona actiong

done.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

NYB\_animation.stop();

MainClass.returnToMain();

start.setDisable(false);

}

});

start.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

NYB\_animation.play();

start.setDisable(true);

pause.setDisable(false);

tableGraphTab();

}

});

pause.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

start.setDisable(false);

pause.setDisable(true);

NYB\_animation.pause();

}

});

reset.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

NYB\_animation.jumpTo(Duration.ZERO);

NYB\_animation.stop();

tableTab.setContent(null);

graphTab.setContent(null);

Optional<ButtonType> result = resetPopUp.showAndWait();

if (result.isPresent() && result.get() == ButtonType.YES) {

pause.fire();

resetPopUp.close();

showResetPopup();

} else if (result.isPresent() && result.get() == ButtonType.NO) {

NYB\_animation.jumpTo(Duration.ZERO);

} else {

}

}

});

Button help = new Button(HELP\_STRING);

Alert helpMsg = new Alert(Alert.AlertType.INFORMATION, null, ButtonType.OK);

helpMsg.setTitle(HELP\_STRING);

helpMsg.setContentText(HELP\_MESSAGE\_NYB\_2); //add message in InterfConst

help.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

helpMsg.show();

}

});

//Button pane

HBox buttonPane = new HBox();

buttonPane.getChildren().addAll(start, done, pause, reset, help);

buttonPane.setAlignment(Pos.CENTER);

buttonPane.setSpacing(10);

//Graph Pane

TabPane graphPane = new TabPane();

graphTab = new Tab(GRAPH\_TAB\_STRING);

tableTab = new Tab(TABLE\_TAB\_STRING);

graphTab.setClosable(false);

tableTab.setClosable(false);

graphPane.getTabs().addAll(graphTab, tableTab);

//Bottom Pane

SplitPane bottomPane = new SplitPane();

bottomPane.getItems().addAll(buttonPane, graphPane);

bottomPane.setDividerPositions(0.5f);

bottomPane.setMinHeight(250);

bottomPane.setPrefHeight(160);

bottomPane.setPrefWidth(200);

//Set pane positions

BorderPane root = new BorderPane();

root.setCenter(animationPane);

root.setBottom(bottomPane);

root.setTop(VBoxTopPane);

return root;

}

public Polygon getAttractionRightArrow() {

Double[] arrowPoints = {

700.0, 300.0,

740.0, 335.0,

740.0, 310.0,

850.0, 310.0,

850.0, 290.0,

740.0, 290.0,

740.0, 265.0,

700.0, 300.0};

Polygon arrow = new Polygon();

arrow.setFill(Color.BLACK);

arrow.getPoints().addAll(arrowPoints);

return arrow;

}

public Polygon getAttractionLeftArrow() {

Double[] arrowPoints = {

300.0, 300.0,

260.0, 335.0,

260.0, 310.0,

150.0, 310.0,

150.0, 290.0,

260.0, 290.0,

260.0, 265.0,

300.0, 300.0};

Polygon arrow = new Polygon();

arrow.setFill(Color.BLACK);

arrow.getPoints().addAll(arrowPoints);

return arrow;

}

public Polygon getRepelLeftArrow() {

Double[] arrowPoints = {

-100.0, 300.0,

-60.0, 335.0,

-60.0, 310.0,

50.0, 310.0,

50.0, 290.0,

-60.0, 290.0,

-60.0, 265.0,

-100.0, 300.0};

Polygon arrow = new Polygon();

arrow.setFill(Color.BLACK);

arrow.getPoints().addAll(arrowPoints);

return arrow;

}

public Polygon getRepelRightArrow() {

Double[] arrowPoints = {

1100.0, 300.0,

1060.0, 335.0,

1060.0, 310.0,

950.0, 310.0,

950.0, 290.0,

1060.0, 290.0,

1060.0, 265.0,

1100.0, 300.0};

Polygon arrow = new Polygon();

arrow.setFill(Color.BLACK);

arrow.getPoints().addAll(arrowPoints);

return arrow;

}

public void showResetPopup() {

Dialog changeValues = new Dialog();

changeValues.setContentText("Reset Values");

changeValues.setHeaderText("Reset charge and sign for Particle One an Particle Two");

changeValues.getDialogPane().getButtonTypes().addAll(ButtonType.OK, ButtonType.CANCEL);

HBox FirstVariableInputValues = new HBox();

HBox SecondVariableInputValues = new HBox();

VBox alignmentPane = new VBox();

Label variableOne = new Label("Charge of Particle One: ");

Label unitsOne = new Label(SPACE\_STRING + UNITS\_CHARGE);

varOneInput = new Spinner(1,100, getUserInputVarOne(), 5);

Label variableTwo = new Label("Change of Particle Two: ");

Label unitsTwo = new Label(SPACE\_STRING + UNITS\_CHARGE);

varTwoInput = new Spinner(1,100, getUserInputVarTwo(), 5);

chargeSignsOne = new ComboBox();

chargeSignsOne.getItems().addAll(POS\_CHARGE, NEG\_CHARGE);

chargeSignsOne.setValue(POS\_CHARGE);

chargeSignsTwo = new ComboBox();

chargeSignsTwo.getItems().addAll(POS\_CHARGE, NEG\_CHARGE);

chargeSignsTwo.setValue(NEG\_CHARGE);

//Add all

FirstVariableInputValues.getChildren().addAll(variableOne, varOneInput, unitsOne, chargeSignsOne);

SecondVariableInputValues.getChildren().addAll(variableTwo, varTwoInput, unitsTwo, chargeSignsTwo);

alignmentPane.getChildren().addAll(FirstVariableInputValues, SecondVariableInputValues);

changeValues.getDialogPane().setContent(alignmentPane);

Optional<ButtonType> result = changeValues.showAndWait();

if (result.isPresent() && result.get() == ButtonType.OK) {

changeValues.close();

makeAnimation();

setUserInputVarOne(Integer.parseInt(varOneInput.getValue().toString()));

setUserInputVarTwo(Integer.parseInt(varTwoInput.getValue().toString()));

particleOne.setText(PARTICLE\_ONE + SPACE\_STRING + getSignPropertyParticleOne() + getUserInputVarOne() + SPACE\_STRING + UNITS\_CHARGE);

particleTwo.setText(PARTICLE\_TWO + SPACE\_STRING + getSignPropertyParticleTwo() + getUserInputVarTwo() + SPACE\_STRING + UNITS\_CHARGE);

} else {

changeValues.close();

}

}

public void makeAnimation() {

if (chargeSignsOne.getValue() == "+") {

setSignPropertyParticleOne(POS\_CHARGE);

Particle\_1.setFill(POS\_CHARGE\_COLOR);

if (chargeSignsTwo.getValue() == "-") {

Particle\_2.setFill(NEG\_CHARGE\_COLOR);

setSignPropertyParticleTwo(NEG\_CHARGE);

animationPane.getChildren().set(2, getAttractionRightArrow());

animationPane.getChildren().set(3, getAttractionLeftArrow());

transFirstArrow.setNode(animationPane.getChildren().get(2));

firstArrowScaleAnimation.setNode(animationPane.getChildren().get(2));

transSecondArrow.setNode(animationPane.getChildren().get(3));

secondArrowScaleAnimation.setNode(animationPane.getChildren().get(3));

} else {

Particle\_2.setFill(POS\_CHARGE\_COLOR);

setSignPropertyParticleTwo(POS\_CHARGE);

animationPane.getChildren().set(2, getRepelRightArrow());

animationPane.getChildren().set(3, getRepelLeftArrow());

transFirstArrow.setNode(animationPane.getChildren().get(2));

firstArrowScaleAnimation.setNode(animationPane.getChildren().get(2));

transSecondArrow.setNode(animationPane.getChildren().get(3));

secondArrowScaleAnimation.setNode(animationPane.getChildren().get(3));

}

} else {

Particle\_1.setFill(NEG\_CHARGE\_COLOR);

setSignPropertyParticleOne(NEG\_CHARGE);

if (chargeSignsTwo.getValue() == "+") {

Particle\_2.setFill(POS\_CHARGE\_COLOR);

setSignPropertyParticleTwo(POS\_CHARGE);

animationPane.getChildren().set(2, getAttractionRightArrow());

animationPane.getChildren().set(3, getAttractionLeftArrow());

transFirstArrow.setNode(animationPane.getChildren().get(2));

firstArrowScaleAnimation.setNode(animationPane.getChildren().get(2));

transSecondArrow.setNode(animationPane.getChildren().get(3));

secondArrowScaleAnimation.setNode(animationPane.getChildren().get(3));

} else {

Particle\_2.setFill(NEG\_CHARGE\_COLOR);

setSignPropertyParticleTwo(NEG\_CHARGE);

animationPane.getChildren().set(2, getRepelRightArrow());

animationPane.getChildren().set(3, getRepelLeftArrow());

transFirstArrow.setNode(animationPane.getChildren().get(2));

transSecondArrow.setNode(animationPane.getChildren().get(3));

firstArrowScaleAnimation.setNode(animationPane.getChildren().get(2));

secondArrowScaleAnimation.setNode(animationPane.getChildren().get(3));

}

}

}

public void tableGraphTab() {

//Table of value

int[] xGraphPoints = new int[10];

int temp = 1;

VBox xTabPoints = new VBox();

xTabPoints.setAlignment(Pos.CENTER);

Label xTablePoints = new Label(DISTANCE\_LABEL.toUpperCase());

xTabPoints.getChildren().add(xTablePoints);

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

xGraphPoints[i] = temp;

xTabPoints.getChildren().add(new Label(Integer.toString(xGraphPoints[i])));

temp ++;

}

double[] yGraphPoints = new double[10];

VBox yTabPoints = new VBox();

yTabPoints.setAlignment(Pos.CENTER);

Label yTableLabel = new Label(FORCE\_LABEL.toUpperCase());

yTabPoints.getChildren().add(yTableLabel);

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

yGraphPoints[i] = ((K\_CONSTANT\*Math.pow(10,9)) \* (getUserInputVarOne() \* Math.pow(10, -6))

\* (getUserInputVarTwo() \* Math.pow(10, -6))) / Math.pow(xGraphPoints[i], 2);

yTabPoints.getChildren().add(new Label(String.format("%.4f",yGraphPoints[i])));

}

HBox xyTable = new HBox();

xyTable.setSpacing(5);

xyTable.setAlignment(Pos.CENTER);

xyTable.getChildren().addAll(xTabPoints, yTabPoints);

tableTab.setContent(xyTable);

//Graphing

NumberAxis xAxis = new NumberAxis(0, 10, 1);

NumberAxis yAxis = new NumberAxis(0, 25, 0.5);

ScatterChart<Number, Number> scatterGraph = new ScatterChart<>(xAxis, yAxis);

xAxis.setLabel(DISTANCE\_LABEL);

yAxis.setLabel(FORCE\_LABEL);

scatterGraph.setTitle(NYB\_2);

scatterGraph.setLegendVisible(false);

XYChart.Series points = new XYChart.Series();

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

points.getData().add(new XYChart.Data(xGraphPoints[i], yGraphPoints[i]));

}

scatterGraph.getData().addAll(points);

graphTab.setContent(scatterGraph);

}

private String getSignPropertyParticleOne(){

return signPropertyParticleOne.getValue();

}

private void setSignPropertyParticleOne(String selection){

signPropertyParticleOne.setValue(selection);

}

private String getSignPropertyParticleTwo(){

return signPropertyParticleTwo.getValue();

}

private void setSignPropertyParticleTwo(String selection){

signPropertyParticleTwo.setValue(selection);

}

private int getUserInputVarOne() {

return userInputVarOne.getValue();

}

private void setUserInputVarOne(int varOne) {

userInputVarOne.setValue(varOne);

}

private int getUserInputVarTwo() {

return userInputVarTwo.getValue();

}

private void setUserInputVarTwo(int varTwo) {

userInputVarTwo.setValue(varTwo);

}

}

package BeanPackage;

import InterfacePackage.ComLayout;

import static InterfacePackage.InterfConst.CANCEL\_STRING;

import static InterfacePackage.InterfConst.NO\_STRING;

import static InterfacePackage.InterfConst.OK\_STRING;

import static InterfacePackage.InterfConst.YES\_STRING;

import MainPackage.MainClass;

import javafx.animation.Animation;

import javafx.animation.Interpolator;

import javafx.animation.KeyFrame;

import javafx.animation.KeyValue;

import javafx.animation.ParallelTransition;

import javafx.animation.PathTransition;

import javafx.animation.Timeline;

import javafx.beans.property.DoubleProperty;

import javafx.beans.property.IntegerProperty;

import javafx.beans.property.SimpleDoubleProperty;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.chart.NumberAxis;

import javafx.scene.chart.ScatterChart;

import javafx.scene.chart.XYChart;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.ButtonType;

import javafx.scene.control.Label;

import javafx.scene.control.Spinner;

import javafx.scene.control.SplitPane;

import javafx.scene.control.Tab;

import javafx.scene.control.TabPane;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.Pane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.shape.Arc;

import javafx.scene.shape.Circle;

import javafx.scene.shape.Line;

import javafx.scene.transform.Rotate;

import javafx.stage.Modality;

import javafx.stage.Stage;

import javafx.util.Duration;

public class Bean\_NYC1 extends ComLayout {

IntegerProperty mass = new SimpleIntegerProperty();

DoubleProperty angle = new SimpleDoubleProperty();

ParallelTransition mainAnimation;

Tab graphTab;

Tab tableTab;

public Bean\_NYC1(int mass, double angle) {

this.mass.set(mass);

this.angle.set(angle);

}

public void showResetPopup() {

Button yesButton = new Button(YES\_STRING);

Button noButton = new Button(NO\_STRING);

Button cancelButton1 = new Button(CANCEL\_STRING);

Button cancelButton2 = new Button(CANCEL\_STRING);

Button okButton = new Button(OK\_STRING);

Label questionLabel = new Label(QUESTION\_STRING);

Label instructionsLabel = new Label(INSTRUCTION\_STRING);

Spinner massField = new Spinner(1, 5, mass.get(), 1);

massField.setEditable(true);

Spinner angleField = new Spinner(90.0, 120.0, angle.get(), 2.0);

angleField.setEditable(true);

Label massLabel = new Label(MASS\_STRING);

Label angleLabel = new Label(ANGLE\_STRING);

//Start First Scene

HBox questionHBox = new HBox();

questionHBox.getChildren().addAll(yesButton, noButton, cancelButton1);

questionHBox.setAlignment(Pos.CENTER\_RIGHT);

questionHBox.setSpacing(8);

questionHBox.setPadding(new Insets(10));

BorderPane questionRoot = new BorderPane();

questionRoot.setCenter(questionLabel);

questionRoot.setBottom(questionHBox);

Scene questionScene = new Scene(questionRoot, 400, 100);

//End First Scene

//Start Second Scene

HBox variablesHBox = new HBox();

variablesHBox.getChildren().addAll(okButton, cancelButton2);

variablesHBox.setAlignment(Pos.CENTER);

variablesHBox.setSpacing(8);

variablesHBox.setPadding(new Insets(10));

HBox massHBox = new HBox();

massHBox.getChildren().addAll(massLabel, massField);

HBox angleHBox = new HBox();

angleHBox.getChildren().addAll(angleLabel, angleField);

VBox fieldsBox = new VBox();

fieldsBox.getChildren().addAll(massHBox, angleHBox);

fieldsBox.setSpacing(8);

BorderPane variablesRoot = new BorderPane();

variablesRoot.setTop(instructionsLabel);

variablesRoot.setCenter(fieldsBox);

variablesRoot.setBottom(variablesHBox);

Scene variablesScene = new Scene(variablesRoot, 500, 150);

//End Second Scene

Stage stage = new Stage();

yesButton.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

stage.setScene(variablesScene);

}

});

noButton.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

mainAnimation.playFromStart();

mainAnimation.pause();

stage.close();

}

});

cancelButton1.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

stage.close();

}

});

cancelButton2.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

stage.close();

}

});

okButton.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

mass.set((int) massField.getValue());

angle.set((double) angleField.getValue());

stage.close();

MainClass.resetNYC1(mass.get(), angle.get());

}

});

stage.setScene(questionScene);

stage.setResizable(false);

stage.centerOnScreen();

stage.setTitle(RESET\_TITLE);

stage.initModality(Modality.APPLICATION\_MODAL);

stage.show();

}

@Override

public BorderPane getPane() {

double endX = 500 - (400 \* Math.abs(Math.sin((180 - getAngle()) \* Math.PI / 180)));

double endY = 100 + (400 \* Math.abs(Math.cos((180 - getAngle()) \* Math.PI / 180)));

double duration = 5 / getMass();

Label titleLabel = new Label(NYC\_TITLE + LINE\_BREAK + NYC\_1 + LINE\_BREAK + NYC\_1\_FORMULA);

Line middleLine = new Line(500, 100, 500, 500);

middleLine.getStrokeDashArray().addAll(2d, 7d);

Line topLine = new Line(300, 100, 700, 100);

Line[] lines = new Line[5];

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

int temp = (i \* 100);

lines[i] = new Line((300 + temp), 100, (340 + temp), 70);

}

Line stringLine = new Line(500, 100, endX, endY);

Rotate stringRotate = new Rotate();

stringRotate.setPivotX(500);

stringRotate.setPivotY(100);

stringLine.getTransforms().add(stringRotate);

KeyValue kv1 = new KeyValue(stringRotate.angleProperty(), -getAngle(), Interpolator.EASE\_BOTH);

KeyFrame kf = new KeyFrame(Duration.seconds(duration), kv1);

Timeline stringAnimation = new Timeline(kf);

stringAnimation.setCycleCount(Animation.INDEFINITE);

stringAnimation.setAutoReverse(true);

Circle bob = new Circle(endX, endY, mass.get() \* 8);

Arc arcPath = new Arc(500, 100, 400, 400, 270 - (getAngle() / 2), getAngle());

arcPath.setFill(null);

arcPath.setStroke(Color.BLACK);

arcPath.setStrokeWidth(2);

PathTransition bobTransition = new PathTransition(Duration.seconds(duration), arcPath, bob);

bobTransition.setCycleCount(Animation.INDEFINITE);

bobTransition.setAutoReverse(true);

mainAnimation = new ParallelTransition(stringAnimation, bobTransition);

Button start = new Button(START\_STRING);

Button done = new Button(DONE\_STRING);

Button pause = new Button(PAUSE\_STRING);

Button reset = new Button(RESET\_STRING);

Button help = new Button(HELP\_STRING);

start.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

mainAnimation.play();

start.setDisable(true);

pause.setDisable(false);

}

});

done.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

MainClass.returnToMain();

}

});

pause.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

mainAnimation.pause();

pause.setDisable(true);

start.setDisable(false);

}

});

reset.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

pause.fire();

showResetPopup();

}

});

Alert helpMsg = new Alert(Alert.AlertType.INFORMATION, NYC\_1\_HELP\_MESSAGE, ButtonType.OK);

helpMsg.setTitle(HELP\_STRING);

helpMsg.setHeaderText(INFORMATION\_STRING\_SHORT);

help.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

helpMsg.show();

}

});

VBox topPane = new VBox();

topPane.getChildren().add(titleLabel);

Pane centerPane = new Pane();

centerPane.getChildren().addAll(middleLine, topLine, stringLine, bob);

centerPane.getChildren().addAll(lines);

HBox buttonPane = new HBox();

buttonPane.getChildren().addAll(start, done, pause, reset, help);

buttonPane.setAlignment(Pos.CENTER);

buttonPane.setSpacing(10);

TabPane graphPane = new TabPane();

graphTab = new Tab(GRAPH\_TAB\_STRING);

tableTab = new Tab(TABLE\_TAB\_STRING);

graphTab.setClosable(false);

tableTab.setClosable(false);

graphPane.getTabs().addAll(graphTab, tableTab);

tableGraphTab();

SplitPane bottomPane = new SplitPane();

bottomPane.getItems().addAll(buttonPane, graphPane);

bottomPane.setDividerPositions(0.5f);

bottomPane.setPrefHeight(160);

bottomPane.setPrefWidth(200);

bottomPane.setMinHeight(250);

BorderPane root = new BorderPane();

root.setTop(topPane);

root.setBottom(bottomPane);

root.setCenter(centerPane);

return root;

}

public int getMass() {

return mass.get();

}

public double getAngle() {

return angle.get();

}

public void setMass(int mass) {

this.mass.set(mass);

}

public void setAngle(double angle) {

this.angle.set(angle);

}

public void tableGraphTab() {

int[] xGraphPoints = new int[10];

int temp = 1;

VBox xTabPoints = new VBox();

xTabPoints.setAlignment(Pos.CENTER);

Label xLabel = new Label(LENGTH\_STRING);

xTabPoints.getChildren().add(xLabel);

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

xGraphPoints[i] = temp;

xTabPoints.getChildren().add(new Label(Integer.toString(xGraphPoints[i])));

temp++;

}

double[] yGraphPoints = new double[10];

VBox yTabPoints = new VBox();

yTabPoints.setAlignment(Pos.CENTER);

Label yLabel = new Label(ANGULAR\_VELOCITY\_STRING);

yTabPoints.getChildren().add(yLabel);

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

yGraphPoints[i] = Math.sqrt(GRAV\_ACC / xGraphPoints[i]);

yTabPoints.getChildren().add(new Label(String.format("%.4f", yGraphPoints[i])));

}

HBox xyTable = new HBox();

xyTable.setSpacing(5);

xyTable.setAlignment(Pos.CENTER);

xyTable.getChildren().addAll(xTabPoints, yTabPoints);

tableTab.setContent(xyTable);

//Graphing

NumberAxis xAxis = new NumberAxis(0, 10, 1);

NumberAxis yAxis = new NumberAxis(0, 3.5, 0.5);

ScatterChart<Number, Number> scatterGraph = new ScatterChart<>(xAxis, yAxis);

xAxis.setLabel(LENGTH\_STRING);

yAxis.setLabel(ANGULAR\_VELOCITY\_STRING);

scatterGraph.setTitle(NYC\_1);

scatterGraph.setLegendVisible(false);

XYChart.Series points = new XYChart.Series();

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

points.getData().add(new XYChart.Data(xGraphPoints[i], yGraphPoints[i]));

}

scatterGraph.getData().addAll(points);

graphTab.setContent(scatterGraph);

}

}

package BeanPackage;

import InterfacePackage.ComLayout;

import static InterfacePackage.InterfConst.\*;

import MainPackage.MainClass;

import java.util.Optional;

import javafx.animation.\*;

import javafx.beans.property.\*;

import javafx.event.\*;

import javafx.geometry.Pos;

import javafx.scene.Group;

import javafx.scene.chart.\*;

import javafx.scene.control.\*;

import javafx.scene.layout.\*;

import javafx.scene.paint.\*;

import javafx.scene.shape.\*;

import javafx.util.Duration;

public class Bean\_NYC2 extends ComLayout {

IntegerProperty userInputYears = new SimpleIntegerProperty(DEFAULT\_YEARS);

StringProperty isotopeProperty = new SimpleStringProperty();

Label mainTitle = new Label(NYC\_TITLE + LINE\_BREAK + NYC\_2 + SPACE\_STRING + "with" + SPACE\_STRING + INITIAL\_SAMPLE + SPACE\_STRING + G\_UNITS);

Label formulaLabel = new Label(NYC\_2\_FORMULA);

Label infoLabel = new Label(INFO\_LABEL);

Label initialDecayLabel = new Label(CARBON\_ISOTOPE + SPACE\_STRING + "initially at" + SPACE\_STRING + INITIAL\_SAMPLE + SPACE\_STRING + G\_UNITS);

Label halfLifeLabel = new Label(HALF\_LIFE\_STRING + SPACE\_STRING + CARBON\_ISOTOPE + SPACE\_STRING + "is:" + SPACE\_STRING + CARBON\_HALF\_LIFE

+ SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand" + SPACE\_STRING + HALF\_LIFE\_GRAMS + SPACE\_STRING + G\_UNITS);

Label userDecayLabel;

Pane animationPane = new Pane();

VBox titlePane = new VBox();

int halfLifeYears = CARBON\_HALF\_LIFE;

double lambda;

int projectedDecay;

int projectedParticlesAmount;

int particleRow;

Circle[] halfLifeParticles;

Circle[] initialParticles;

Circle[] userParticles;

double halfLifeXPoint;

double initialXPoint;

double yPoint;

double userXPoint;

Rectangle halfLifeRect;

Rectangle userDecayRect;

Rectangle initialDecayRect;

ToggleGroup isotopeGroup = new ToggleGroup();

RadioButton carbonIsotope;

RadioButton niobiumIsotope;

RadioButton thoriumIsotope;

RadioButton molybdenumIsotope;

RadioButton curiumIsotope;

ComboBox usersChoices;

Tab graphTab;

Tab tableTab;

String isotope = CARBON\_ISOTOPE;

@Override

public BorderPane getPane() {

carbonIsotope = new RadioButton(CARBON\_ISOTOPE + SPACE\_STRING);

carbonIsotope.setToggleGroup(isotopeGroup);

carbonIsotope.setSelected(true);

niobiumIsotope = new RadioButton(NIOBIUM\_ISOTOPE + SPACE\_STRING);

niobiumIsotope.setToggleGroup(isotopeGroup);

thoriumIsotope = new RadioButton(THORIUM\_ISOTOPE + SPACE\_STRING);

thoriumIsotope.setToggleGroup(isotopeGroup);

molybdenumIsotope = new RadioButton(MOLYBDENUM\_ISOTOPE + SPACE\_STRING);

molybdenumIsotope.setToggleGroup(isotopeGroup);

curiumIsotope = new RadioButton(CURIUM\_ISOTOPE + SPACE\_STRING);

curiumIsotope.setToggleGroup(isotopeGroup);

//Rectangle Panes

VBox leftBox = new VBox();

Group leftGroup = new Group();

leftGroup.getChildren().add(leftBox);

VBox centerBox = new VBox();

Group centerGroup = new Group();

centerGroup.getChildren().add(centerBox);

VBox rightBox = new VBox();

Group rightGroup = new Group();

rightGroup.getChildren().add(rightBox);

//Display particles

initialParticles();

halfLifeParticles();

calculations();

userInputs();

//Set label for user default decay

userDecayLabel = new Label(DEFAULT\_USER\_INPUT + SPACE\_STRING + getUserInputYears() + SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand"

+ SPACE\_STRING + projectedParticlesAmount + SPACE\_STRING + G\_UNITS);

//Set labels invisible

initialDecayLabel.setVisible(false);

halfLifeLabel.setVisible(false);

userDecayLabel.setVisible(false);

//Defining the rectangles

initialDecayRect = new Rectangle(25, 125, 250, 250);

leftBox.getChildren().addAll(initialDecayRect, initialDecayLabel);

leftGroup.setLayoutX(25);

leftGroup.setLayoutY(125);

initialDecayRect.setStroke(Color.GREEN);

initialDecayRect.setStrokeWidth(3);

initialDecayRect.setFill(null);

halfLifeRect = new Rectangle(375, 125, 250, 250);

centerBox.getChildren().addAll(halfLifeRect, halfLifeLabel);

centerGroup.setLayoutX(375);

centerGroup.setLayoutY(125);

halfLifeRect.setStroke(Color.YELLOW);

halfLifeRect.setStrokeWidth(3);

halfLifeRect.setFill(null);

userDecayRect = new Rectangle(725, 125, 250, 250);

rightBox.getChildren().addAll(userDecayRect, userDecayLabel);

rightGroup.setLayoutX(725);

rightGroup.setLayoutY(125);

userDecayRect.setStroke(Color.RED);

userDecayRect.setStrokeWidth(3);

userDecayRect.setFill(null);

//Initially Rectangle animation

hideInitial();

FadeTransition fadeInitialRect = new FadeTransition(Duration.seconds(2), initialDecayRect);

fadeInitialRect.setFromValue(0);

fadeInitialRect.setToValue(1);

fadeInitialRect.setAutoReverse(false);

fadeInitialRect.setCycleCount(1);

ParallelTransition parallelInitialRect = new ParallelTransition();

FadeTransition[] fadeInitialParticles = new FadeTransition[100];

for (int i = 0; i < 100; i++) {

fadeInitialParticles[i] = new FadeTransition(Duration.seconds(2), initialParticles[i]);

fadeInitialParticles[i].setAutoReverse(false);

fadeInitialParticles[i].setFromValue(0);

fadeInitialParticles[i].setToValue(1);

fadeInitialParticles[i].setCycleCount(1);

parallelInitialRect.getChildren().add(fadeInitialParticles[i]);

}

parallelInitialRect.getChildren().add(fadeInitialRect);

//Half life Rectangle angimation

hideHalfLife();

FadeTransition fadeHalfLifeRect = new FadeTransition(Duration.seconds(2), halfLifeRect);

fadeHalfLifeRect.setFromValue(0);

fadeHalfLifeRect.setToValue(1);

fadeHalfLifeRect.setAutoReverse(false);

fadeHalfLifeRect.setCycleCount(1);

ParallelTransition parallelHalfLifeRect = new ParallelTransition();

FadeTransition[] fadeHalfLifeParticles = new FadeTransition[50];

for (int i = 0; i < 50; i++) {

fadeHalfLifeParticles[i] = new FadeTransition(Duration.seconds(2), halfLifeParticles[i]);

fadeHalfLifeParticles[i].setAutoReverse(false);

fadeHalfLifeParticles[i].setFromValue(0);

fadeHalfLifeParticles[i].setToValue(1);

fadeHalfLifeParticles[i].setCycleCount(1);

parallelHalfLifeRect.getChildren().add(fadeHalfLifeParticles[i]);

}

parallelHalfLifeRect.getChildren().add(fadeHalfLifeRect);

//User Decay Rectangle animation

hideUserDecay();

FadeTransition fadeUserRect = new FadeTransition(Duration.seconds(2), userDecayRect);

fadeUserRect.setFromValue(0);

fadeUserRect.setToValue(1);

fadeUserRect.setAutoReverse(false);

fadeUserRect.setCycleCount(1);

ParallelTransition parallelUserRect = new ParallelTransition();

FadeTransition[] fadeUserParticles = new FadeTransition[projectedParticlesAmount];

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

fadeUserParticles[i] = new FadeTransition(Duration.seconds(2), userParticles[i]);

fadeUserParticles[i].setAutoReverse(false);

fadeUserParticles[i].setFromValue(0);

fadeUserParticles[i].setToValue(1);

fadeUserParticles[i].setCycleCount(1);

parallelUserRect.getChildren().add(fadeUserParticles[i]);

}

parallelUserRect.getChildren().add(fadeUserRect);

//Declare and define button actions

Button pause = new Button(PAUSE\_STRING);

pause.setDisable(true);

Button start = new Button(START\_STRING);

start.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

initialDecayLabel.setVisible(true);

initialDecayRect.setVisible(true);

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

initialParticles[i].setVisible(true);

}

parallelInitialRect.play();

parallelInitialRect.setOnFinished(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

halfLifeLabel.setVisible(true);

halfLifeRect.setVisible(true);

for (int i = 0; i < 50; i++) {

halfLifeParticles[i].setVisible(true);

}

parallelHalfLifeRect.play();

}

});

parallelHalfLifeRect.setOnFinished(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

userDecayLabel.setVisible(true);

userDecayRect.setVisible(true);

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

userParticles[i].setVisible(true);

}

parallelUserRect.play();

}

});

tableGraphTab();

start.setDisable(true);

pause.setDisable(false);

}

});

pause.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

start.setDisable(false);

pause.setDisable(true);

}

});

Button done = new Button(DONE\_STRING);

done.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

MainClass.returnToMain();

start.setDisable(false);

}

});

Button reset = new Button(RESET\_STRING);

Alert resetPopUp = new Alert(Alert.AlertType.CONFIRMATION, "Choose your option: ", ButtonType.YES, ButtonType.NO, ButtonType.CANCEL);

resetPopUp.setTitle("Changing Sample Isotopes");

resetPopUp.setHeaderText("Would you like to change the radioactive isotopes?");

reset.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

initialDecayLabel.setVisible(false);

halfLifeLabel.setVisible(false);

userDecayLabel.setVisible(false);

parallelHalfLifeRect.playFromStart();

parallelHalfLifeRect.stop();

parallelInitialRect.playFromStart();

parallelInitialRect.stop();

parallelUserRect.playFromStart();

parallelUserRect.stop();

hideUserDecay();

tableTab.setContent(null);

graphTab.setContent(null);

Optional<ButtonType> result = resetPopUp.showAndWait();

if (result.isPresent() && result.get() == ButtonType.YES) {

pause.fire();

removeParticles();

resetPopUp.close();

showResetPopup();

hideUserDecay();

} else if (result.isPresent() && result.get() == ButtonType.NO) {

pause.fire();

hideUserDecay();

} else {

}

}

});

Button help = new Button(HELP\_STRING);

Alert helpMsg = new Alert(Alert.AlertType.INFORMATION, null, ButtonType.OK);

helpMsg.setTitle(HELP\_STRING);

helpMsg.setContentText(HELP\_MESSAGE\_NYC\_2);

help.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

helpMsg.show();

}

});

//Add to pane

titlePane.getChildren().addAll(mainTitle, formulaLabel, infoLabel);

animationPane.getChildren().addAll(leftGroup, centerGroup, rightGroup);

//Add panes to main pane

BorderPane root = new BorderPane();

HBox buttonPane = new HBox();

buttonPane.getChildren().addAll(start, done, pause, reset, help);

buttonPane.setAlignment(Pos.CENTER);

buttonPane.setSpacing(10);

//Declare and define graphs and tab Pane

TabPane graphPane = new TabPane();

graphTab = new Tab("Graph Tab");

tableTab = new Tab("Table Tab");

graphTab.setClosable(false);

tableTab.setClosable(false);

graphPane.getTabs().addAll(graphTab, tableTab);

//Bottom pane containing button and graphs

SplitPane bottomPane = new SplitPane();

bottomPane.setMinHeight(250);

bottomPane.getItems().addAll(buttonPane, graphPane);

bottomPane.setDividerPositions(0.5f);

bottomPane.setPrefHeight(160);

bottomPane.setPrefWidth(200);

//Set positions

root.setBottom(bottomPane);

root.setCenter(animationPane);

root.setTop(titlePane);

return root;

}

public void showResetPopup() {

Dialog changeValues = new Dialog();

changeValues.setContentText("Pick Isotopes");

changeValues.setHeaderText("Choose an isotope to see its radioactive decay: ");

changeValues.getDialogPane().getButtonTypes().addAll(ButtonType.OK, ButtonType.CANCEL);

HBox radioActiveIsotopes = new HBox();

HBox inputYears = new HBox();

VBox alignmentPane = new VBox();

Label finalYears = new Label(LABEL\_FINAL\_YEARS\_INFO);

usersChoices = new ComboBox();

usersChoices.getItems().addAll(1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000);

usersChoices.setValue(1000);

Label units = new Label(YEARS\_UNIT);

radioActiveIsotopes.getChildren().addAll(carbonIsotope, niobiumIsotope, thoriumIsotope, molybdenumIsotope, curiumIsotope);

radioActiveIsotopes.setSpacing(5.0);

inputYears.getChildren().addAll(finalYears, usersChoices, units);

inputYears.setSpacing(5.0);

alignmentPane.getChildren().addAll(radioActiveIsotopes, inputYears);

alignmentPane.setSpacing(5);

changeValues.getDialogPane().setContent(alignmentPane);

Optional<ButtonType> result = changeValues.showAndWait();

if (result.isPresent() && result.get() == ButtonType.OK) {

setUserInputYears(Integer.parseInt(usersChoices.getValue().toString()));

changeValues.close();

if (carbonIsotope.isSelected()) {

setIsotopeProperty(CARBON\_ISOTOPE);

initialDecayLabel.setText(getIsotopeProperty() + SPACE\_STRING + "initially at" + SPACE\_STRING + INITIAL\_SAMPLE + SPACE\_STRING + G\_UNITS);

halfLifeLabel.setText(HALF\_LIFE\_STRING + SPACE\_STRING + getIsotopeProperty() + SPACE\_STRING + "is:" + SPACE\_STRING + CARBON\_HALF\_LIFE

+ SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand" + SPACE\_STRING + HALF\_LIFE\_GRAMS + SPACE\_STRING + G\_UNITS);

halfLifeYears = CARBON\_HALF\_LIFE;

isotope = CARBON\_ISOTOPE;

carbonIsotope.setSelected(true);

calculations();

userInputs();

userDecayLabel.setText(USER\_YEARS + SPACE\_STRING + getUserInputYears() + SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand"

+ SPACE\_STRING + projectedParticlesAmount + SPACE\_STRING + G\_UNITS);

}

if (thoriumIsotope.isSelected()) {

setIsotopeProperty(THORIUM\_ISOTOPE);

initialDecayLabel.setText(getIsotopeProperty() + SPACE\_STRING + "initially at" + SPACE\_STRING + INITIAL\_SAMPLE + SPACE\_STRING + G\_UNITS);

halfLifeLabel.setText(HALF\_LIFE\_STRING + SPACE\_STRING + getIsotopeProperty() + SPACE\_STRING + "is:" + SPACE\_STRING + THORIUM\_HALF\_LIFE

+ SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand" + SPACE\_STRING + HALF\_LIFE\_GRAMS + SPACE\_STRING + G\_UNITS);

halfLifeYears = THORIUM\_HALF\_LIFE;

isotope = THORIUM\_ISOTOPE;

thoriumIsotope.setSelected(true);

calculations();

userInputs();

userDecayLabel.setText(USER\_YEARS + SPACE\_STRING + getUserInputYears() + SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand"

+ SPACE\_STRING + projectedParticlesAmount + SPACE\_STRING + G\_UNITS);

}

if (molybdenumIsotope.isSelected()) {

setIsotopeProperty(MOLYBDENUM\_ISOTOPE);

initialDecayLabel.setText(getIsotopeProperty() + SPACE\_STRING + "initially at" + SPACE\_STRING + INITIAL\_SAMPLE + SPACE\_STRING + G\_UNITS);

halfLifeLabel.setText(HALF\_LIFE\_STRING + SPACE\_STRING + getIsotopeProperty() + SPACE\_STRING + "is:" + SPACE\_STRING

+ MOLYBDENUM\_HALF\_LIFE + SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand" + SPACE\_STRING + HALF\_LIFE\_GRAMS + SPACE\_STRING + G\_UNITS);

halfLifeYears = MOLYBDENUM\_HALF\_LIFE;

isotope = MOLYBDENUM\_ISOTOPE;

molybdenumIsotope.setSelected(true);

calculations();

userInputs();

userDecayLabel.setText(USER\_YEARS + SPACE\_STRING + getUserInputYears() + SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand"

+ SPACE\_STRING + projectedParticlesAmount + SPACE\_STRING + G\_UNITS);

}

if (curiumIsotope.isSelected()) {

setIsotopeProperty(CURIUM\_ISOTOPE);

initialDecayLabel.setText(getIsotopeProperty() + SPACE\_STRING + "initially at" + SPACE\_STRING + INITIAL\_SAMPLE + SPACE\_STRING + G\_UNITS);

halfLifeLabel.setText(HALF\_LIFE\_STRING + SPACE\_STRING + getIsotopeProperty() + SPACE\_STRING + "is:" + SPACE\_STRING + CURIUM\_HALF\_LIFE

+ SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand" + SPACE\_STRING + HALF\_LIFE\_GRAMS + SPACE\_STRING + G\_UNITS);

halfLifeYears = CURIUM\_HALF\_LIFE;

isotope = CURIUM\_ISOTOPE;

curiumIsotope.setSelected(true);

calculations();

userInputs();

userDecayLabel.setText(USER\_YEARS + SPACE\_STRING + getUserInputYears() + SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand"

+ SPACE\_STRING + projectedParticlesAmount + SPACE\_STRING + G\_UNITS);

}

if (niobiumIsotope.isSelected()) {

setIsotopeProperty(NIOBIUM\_ISOTOPE);

initialDecayLabel.setText(getIsotopeProperty() + SPACE\_STRING + "initially at" + SPACE\_STRING + INITIAL\_SAMPLE + SPACE\_STRING + G\_UNITS);

halfLifeLabel.setText(HALF\_LIFE\_STRING + SPACE\_STRING + getIsotopeProperty() + SPACE\_STRING + "is:" + SPACE\_STRING + NIOBIUM\_HALF\_LIFE

+ SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand" + SPACE\_STRING + HALF\_LIFE\_GRAMS + SPACE\_STRING + G\_UNITS);

halfLifeYears = NIOBIUM\_HALF\_LIFE;

isotope = NIOBIUM\_ISOTOPE;

niobiumIsotope.setSelected(true);

calculations();

userInputs();

userDecayLabel.setText(USER\_YEARS + SPACE\_STRING + getUserInputYears() + SPACE\_STRING + YEARS\_UNIT + SPACE\_STRING + "\nand"

+ SPACE\_STRING + projectedParticlesAmount + SPACE\_STRING + SPACE\_STRING + G\_UNITS);

}

} else {

changeValues.close();

}

}

public void hideInitial() {

initialDecayRect.setVisible(false);

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

initialParticles[i].setVisible(false);

}

}

public void hideHalfLife() {

halfLifeRect.setVisible(false);

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

halfLifeParticles[i].setVisible(false);

}

}

public void hideUserDecay() {

userDecayRect.setVisible(false);

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

userParticles[i].setVisible(false);

}

}

public void removeParticles() {

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

animationPane.getChildren().remove(userParticles[i]);

}

}

public void userInputs() {

userXPoint = 739.75;

userParticles = new Circle[projectedParticlesAmount];

int temp3 = 0;

if (projectedParticlesAmount % 10 != 0) {

particleRow = (projectedParticlesAmount + 1);

} else {

particleRow = projectedParticlesAmount;

}

if (projectedParticlesAmount <= 10) {

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

if (temp3 == projectedParticlesAmount) {

break;

} else {

userParticles[temp3] = new Circle(userXPoint, yPoint, 9.75);

userXPoint += 24.5;

temp3++;

}

}

} else {

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

if (i > 0) {

yPoint += 24.5;

} else {

yPoint = 139.75;

}

for (int j = 0; j < 10; j++) {

if (temp3 == projectedParticlesAmount) {

break;

} else if (userXPoint > 935.75 + 24.5) {

userXPoint = 739.75;

userParticles[temp3] = new Circle(userXPoint, yPoint, 9.75);

userXPoint += 24.5;

temp3++;

} else {

userParticles[temp3] = new Circle(userXPoint, yPoint, 9.75);

userXPoint += 24.5;

temp3++;

}

}

}

}

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

animationPane.getChildren().add(userParticles[i]);

}

}

public void halfLifeParticles() {

halfLifeXPoint = 389.75;

halfLifeParticles = new Circle[50];

int temp2 = 0;

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

if (i > 0) {

yPoint += 24.5;

} else {

yPoint = 139.75;

}

for (int j = 0; j < 10; j++) {

if (halfLifeXPoint > 585.75 + 24.5) {

halfLifeXPoint = 389.75;

halfLifeParticles[temp2] = new Circle(halfLifeXPoint, yPoint, 9.75);

halfLifeXPoint += 24.5;

temp2++;

} else {

halfLifeParticles[temp2] = new Circle(halfLifeXPoint, yPoint, 9.75);

halfLifeXPoint += 24.5;

temp2++;

}

}

}

for (int i = 0; i < 50; i++) {

animationPane.getChildren().add(halfLifeParticles[i]);

}

}

public void initialParticles() {

initialXPoint = 39.75;

yPoint = 139.75;

int temp = 0;

initialParticles = new Circle[100];

for (int i = 0; i < 10; i++) {

if (i > 0) {

yPoint += 24.5;

} else {

yPoint = 139.75;

}

for (int j = 0; j < 10; j++) {

if (initialXPoint > 235.75 + 24.5) {

initialXPoint = 39.75;

initialParticles[temp] = new Circle(initialXPoint, yPoint, 9.75);

animationPane.getChildren().add(initialParticles[temp]);

initialXPoint += 24.5;

temp++;

} else {

initialParticles[temp] = new Circle(initialXPoint, yPoint, 9.75);

initialXPoint += 24.5;

animationPane.getChildren().add(initialParticles[temp]);

temp++;

}

}

}

}

public void calculations() {

lambda = -0.693 / halfLifeYears;

projectedDecay = (int) (INITIAL\_SAMPLE \* Math.pow(E\_CONSTANT, lambda \* userInputYears.getValue()));

projectedParticlesAmount = (int) projectedDecay;

particleRow = projectedParticlesAmount / 10;

}

public void tableGraphTab() {

//Table of Values

int[] xGraphPoints = new int[10];

int temp = 1000;

VBox xTabPoints = new VBox();

xTabPoints.setAlignment(Pos.CENTER);

Label xTableLabel = new Label(YEARS\_UNIT.toUpperCase());

xTabPoints.getChildren().add(xTableLabel);

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

xGraphPoints[i] = temp;

xTabPoints.getChildren().add(new Label(Integer.toString(xGraphPoints[i])));

temp += 1000;

}

int[] yGraphPoints = new int[10];

VBox yTabPoints = new VBox();

yTabPoints.setAlignment(Pos.CENTER);

Label yTableLabel = new Label(G\_UNITS.toUpperCase());

yTabPoints.getChildren().add(yTableLabel);

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

yGraphPoints[i] = (int) (INITIAL\_SAMPLE \* Math.pow(E\_CONSTANT, lambda \* xGraphPoints[i]));

yTabPoints.getChildren().add(new Label(Integer.toString(yGraphPoints[i])));

}

HBox xyTable = new HBox();

xyTable.setSpacing(5);

xyTable.setAlignment(Pos.CENTER);

xyTable.getChildren().addAll(xTabPoints, yTabPoints);

tableTab.setContent(xyTable);

//Graphing

NumberAxis xAxis = new NumberAxis(0, 10000, 1000);

NumberAxis yAxis = new NumberAxis(0, 100, 20);

ScatterChart<Number, Number> scatterGraph = new ScatterChart<>(xAxis, yAxis);

xAxis.setLabel(YEARS\_UNIT);

yAxis.setLabel(G\_UNITS);

scatterGraph.setTitle(NYC\_2 + " of " + isotope);

scatterGraph.setLegendVisible(false);

XYChart.Series points = new XYChart.Series();

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

points.getData().add(new XYChart.Data(xGraphPoints[i], yGraphPoints[i]));

}

scatterGraph.getData().addAll(points);

graphTab.setContent(scatterGraph);

}

private String getIsotopeProperty() {

return isotopeProperty.getValue();

}

private void setIsotopeProperty(String selection) {

isotopeProperty.setValue(selection);

}

private int getUserInputYears() {

return userInputYears.getValue();

}

private void setUserInputYears(int years) {

userInputYears.setValue(years);

}

}













