**Faculty of Science**

**Course**: CSCI 2020u - Software System Development & Integration

**Component**: Assignment

**Weight**: 10%

**Deadline**: March 5, 2020 (due by 11:59pm)

**Collaboration Policy**

You are permitted to work on this assignment in a team, and submit the results as a team. For this

sort of assignment, with an open-ended component, the collaboration between multiple team

members can be beneficial. Between groups, however, please limit the discussion to the level of

general strategy (not code). Groups of size 2 are recommended. Larger groups will be considered with the proviso that the marker will mark your assignment with higher expectations. In any case, be sure that all members of the team fully understand all code, otherwise they will miss intended learning objectives, which may be a considerable disadvantage at exam time.

**How to Submit**

You will maintain a **git repository** for this assignment, which is a public repository. To submit

the assignment, create a single file 'README.txt' that contains instructions on how to download, compile, and run your codes for each question. A .zip, .7z, or .rar file will not be acceptable.**Also**

**submit this word file (once you complete) into related drop box on Blackboard before**

**deadline.**

**Note***: Comments are mandatory. Failure to properly document your program will result in a*

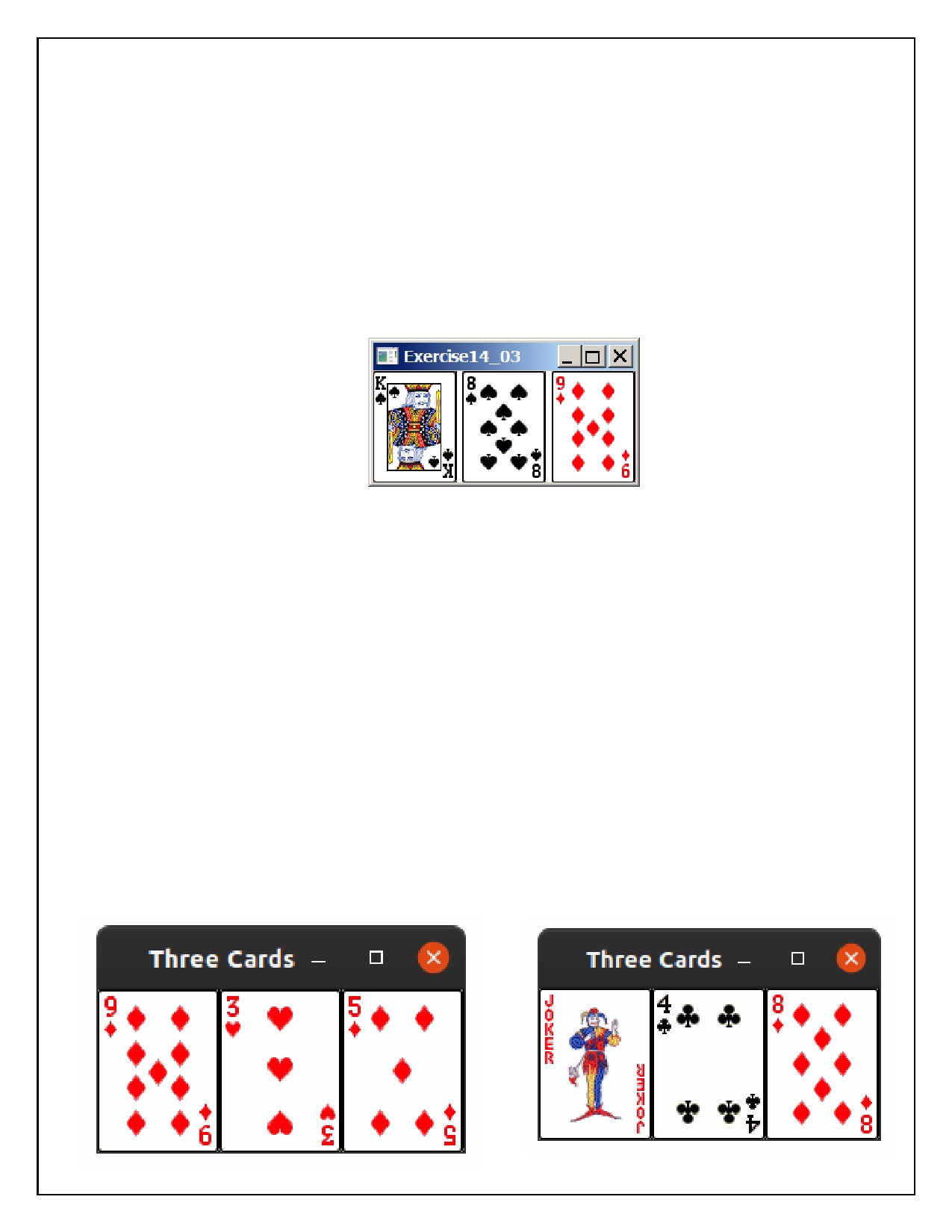
*deduction on the marks you receive for this (and any other) assignment.*

**Remember**:

You need to complete this file and submit it in related **drop box on Blackboard** , in

addition to uploading your codes in your **git repository**, before deadline.

Page **1** of **6**

**Question 1: Displaying Three Cards**

Problem Description:

Display a frame that contains three labels. Each label displays a card, as shown in

the figure below. The card image files are named 1.png, 2.png, ..., 54.png and

stored in the image/card directory. All three cards are distinct and selected randomly.

The image icons can be found in the attached card folder.

Your Task:

1. Create three ImageView and set their icons using the images.

2. Display three images from 54 image cards randomly.

Your Code:

Copy-paste your code here:

Screen shots:

Include two screen shots here:

Page **2** of **6**

|  |
| --- |
| package question1; |
|  |

|  |
| --- |
| /\* |
|  |

|  |
| --- |
| \* Ryan Christopher, 100702835 |
|  |

|  |
| --- |
| \* Mario Velazquez, 100702233 |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \* Software Systems CSCI2020u |
|  |

|  |
| --- |
| \* Assignment: question #1 |
|  |

|  |
| --- |
| \* Due: March 5, 2020 |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \* Display a frame that contains three labels. Each label displays a card, as shown in |
|  |

|  |
| --- |
| \* the figure below. The card image files are named 1.png, 2.png, ..., 54.png and |
|  |

|  |
| --- |
| \* stored in the image/card directory. All three cards are distinct and selected |
|  |

|  |
| --- |
| \* randomly. |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \*/ |
|  |

|  |
| --- |
| import javafx.application.Application; |
|  |

|  |
| --- |
| import javafx.scene.Scene; |
|  |

|  |
| --- |
| import javafx.scene.image.Image; |
|  |

|  |
| --- |
| import javafx.scene.image.ImageView; |
|  |

|  |
| --- |
| import javafx.scene.layout.HBox; |
|  |

|  |
| --- |
| import javafx.stage.Stage; |
|  |

|  |
| --- |
| import java.util.Random; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public class ThreeCards extends Application { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public static void main(String[] args) { |
|  |

|  |
| --- |
| launch(args); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public void start(Stage primaryStage) { |
|  |

|  |
| --- |
| HBox pane = new HBox(); |
|  |

|  |
| --- |
| Random rand = new Random(); |
|  |

|  |
| --- |
| //generate random numbers to determine cards |
|  |

|  |
| --- |
| int[] cards = {rand.nextInt(54)+1,rand.nextInt(54)+1,rand.nextInt(54)+1}; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //check if any cards are the same |
|  |

|  |
| --- |
| for (int i = 0; i<3; i++) { |
|  |

|  |
| --- |
| for (int j=i+1; j<2; j++) { |
|  |

|  |
| --- |
| //if cards are the same, generate new number |
|  |

|  |
| --- |
| if (cards[i] == cards[j]) {cards[j] = rand.nextInt(54)+1; j--;} |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| //if cards are not the same add card to the HBox |
|  |

|  |
| --- |
| pane.getChildren().add(new ImageView(new Image(String.format("image/cards/%d.png",cards[i])))); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| //display window |
|  |

|  |
| --- |
| primaryStage.setTitle("Three Cards"); |
|  |

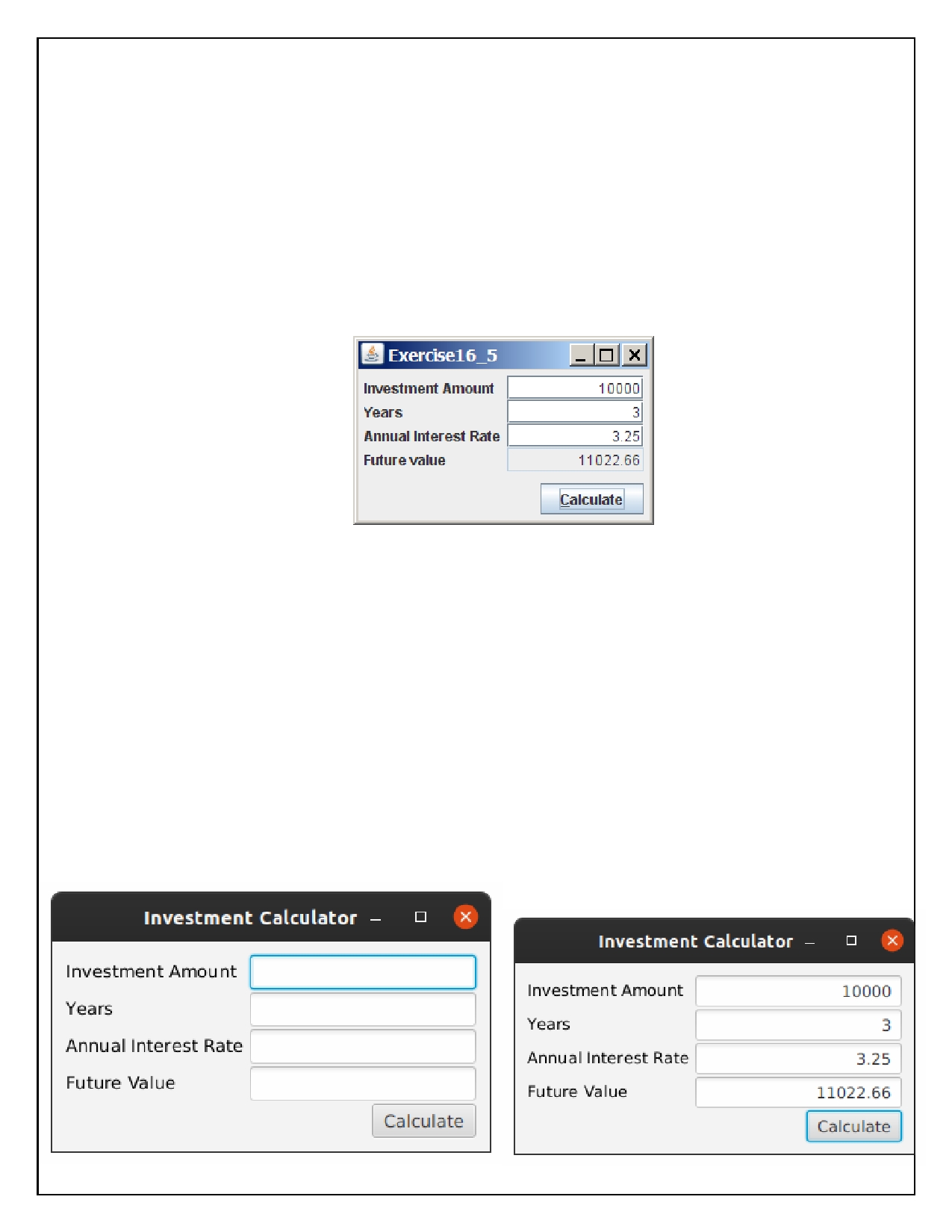
|  |
| --- |
| Scene scene = new Scene(pane); |
|  |

|  |
| --- |
| primaryStage.setScene(scene); |
|  |

|  |
| --- |
| primaryStage.show(); |
|  |

|  |
| --- |
| } |
|  |

}

**Question 2: Investment-Value calculator**

Problem Description:

Write a program that calculates the future value of an investment at a given

interest rate for a specified number of years. The formula for the calculation is as

follows:

futureValue = investmentAmount \* (1 + monthlyInterestRate)

years\*12

Your Task:

Use text fields for interest rate, investment amount, and years. Display the future amount in a text field when the user clicks the Calculate button, as shown in the figure.

Your Code:

Copy-paste your code here.

Screen shots:

Include two screen shots here:

Page **3** of **6**

|  |
| --- |
| package question2; |
|  |

|  |
| --- |
| /\* |
|  |

|  |
| --- |
| \* Ryan Christopher, 100702835 |
|  |

|  |
| --- |
| \* Mario Velazquez, 100702233 |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \* Software Systems CSCI2020u |
|  |

|  |
| --- |
| \* Assignment: question #1 |
|  |

|  |
| --- |
| \* Due: March 5, 2020 |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \* Write a program that calculates the future value of an investment at a given |
|  |

|  |
| --- |
| \* interest rate for a specified number of years. The formula for the calculation is as |
|  |

|  |
| --- |
| \* follows: |
|  |

|  |
| --- |
| \* futureValue = investmentAmount \* (1 + monthlyInterestRate)^years\*12 |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \*/ |
|  |

|  |
| --- |
| import javafx.application.Application; |
|  |

|  |
| --- |
| import javafx.geometry.Insets; |
|  |

|  |
| --- |
| import javafx.geometry.Pos; |
|  |

|  |
| --- |
| import javafx.scene.Scene; |
|  |

|  |
| --- |
| import javafx.scene.control.Button; |
|  |

|  |
| --- |
| import javafx.scene.control.TextField; |
|  |

|  |
| --- |
| import javafx.scene.layout.GridPane; |
|  |

|  |
| --- |
| import javafx.scene.layout.HBox; |
|  |

|  |
| --- |
| import javafx.scene.text.Text; |
|  |

|  |
| --- |
| import javafx.stage.Stage; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public class InvestmentCalculator extends Application { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public static void main(String[] args) { |
|  |

|  |
| --- |
| launch(args); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public void start(Stage primaryStage) { |
|  |

|  |
| --- |
| GridPane pane = new GridPane(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Text[] leftText = {new Text("Investment Amount"), new Text("Years"), |
|  |

|  |
| --- |
| new Text("Annual Interest Rate"), new Text("Future Value")}; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| TextField[] rightFields = {new TextField(), new TextField(), new TextField(), new TextField()}; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for (TextField field : rightFields) {field.setAlignment(Pos.CENTER\_RIGHT);} |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //restrict input for last field since it is used for displaying output only |
|  |

|  |
| --- |
| rightFields[3].setEditable(false); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Button calculate = new Button("Calculate"); |
|  |

|  |
| --- |
| //set button action to calculate the interest by sending the text of the input fields and the field for output |
|  |

|  |
| --- |
| calculate.setOnAction(action -> |
|  |

|  |
| --- |
| calculateInterest(rightFields[0].getText(), rightFields[1].getText(), |
|  |

|  |
| --- |
| rightFields[2].getText(),rightFields[3]) |
|  |

|  |
| --- |
| ); |
|  |

|  |
| --- |
| HBox buttonPane = new HBox(); |
|  |

|  |
| --- |
| buttonPane.getChildren().add(calculate); |
|  |

|  |
| --- |
| buttonPane.setAlignment(Pos.CENTER\_RIGHT); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //GridPane formatting |
|  |

|  |
| --- |
| pane.setPadding(new Insets(10, 10, 10, 10)); |
|  |

|  |
| --- |
| pane.setVgap(2); |
|  |

|  |
| --- |
| pane.setHgap(5); |
|  |

|  |
| --- |
| pane.setAlignment(Pos.CENTER); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //add elements to the pane |
|  |

|  |
| --- |
| for (Text text : leftText) {pane.addColumn(0, text);} |
|  |

|  |
| --- |
| for (TextField fields : rightFields) {pane.addColumn(1, fields);} |
|  |

|  |
| --- |
| pane.addColumn(1, buttonPane); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //final window setup and display |
|  |

|  |
| --- |
| primaryStage.setTitle("Investment Calculator"); |
|  |

|  |
| --- |
| Scene scene = new Scene(pane); |
|  |

|  |
| --- |
| primaryStage.setScene(scene); |
|  |

|  |
| --- |
| primaryStage.show(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\* |
|  |

|  |
| --- |
| \* This method is sent the text in the TextFields and an instance of a TextField, |
|  |

|  |
| --- |
| \* calculates interest based on the numbers entered, |
|  |

|  |
| --- |
| \* then sets the text of the TextField instance. |
|  |

|  |
| --- |
| \*/ |
|  |

|  |
| --- |
| public void calculateInterest(String aS, String yS, String rS, TextField v) { |
|  |

|  |
| --- |
| double aD, yD,rD; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //convert Strings to double |
|  |

|  |
| --- |
| aD = Double.parseDouble(aS); |
|  |

|  |
| --- |
| yD = Double.parseDouble(yS); |
|  |

|  |
| --- |
| rD = Double.parseDouble(rS)/100; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // calculate interest |
|  |

|  |
| --- |
| double valueCalc = aD \* Math.pow((1 + rD/12),yD\*12); |
|  |

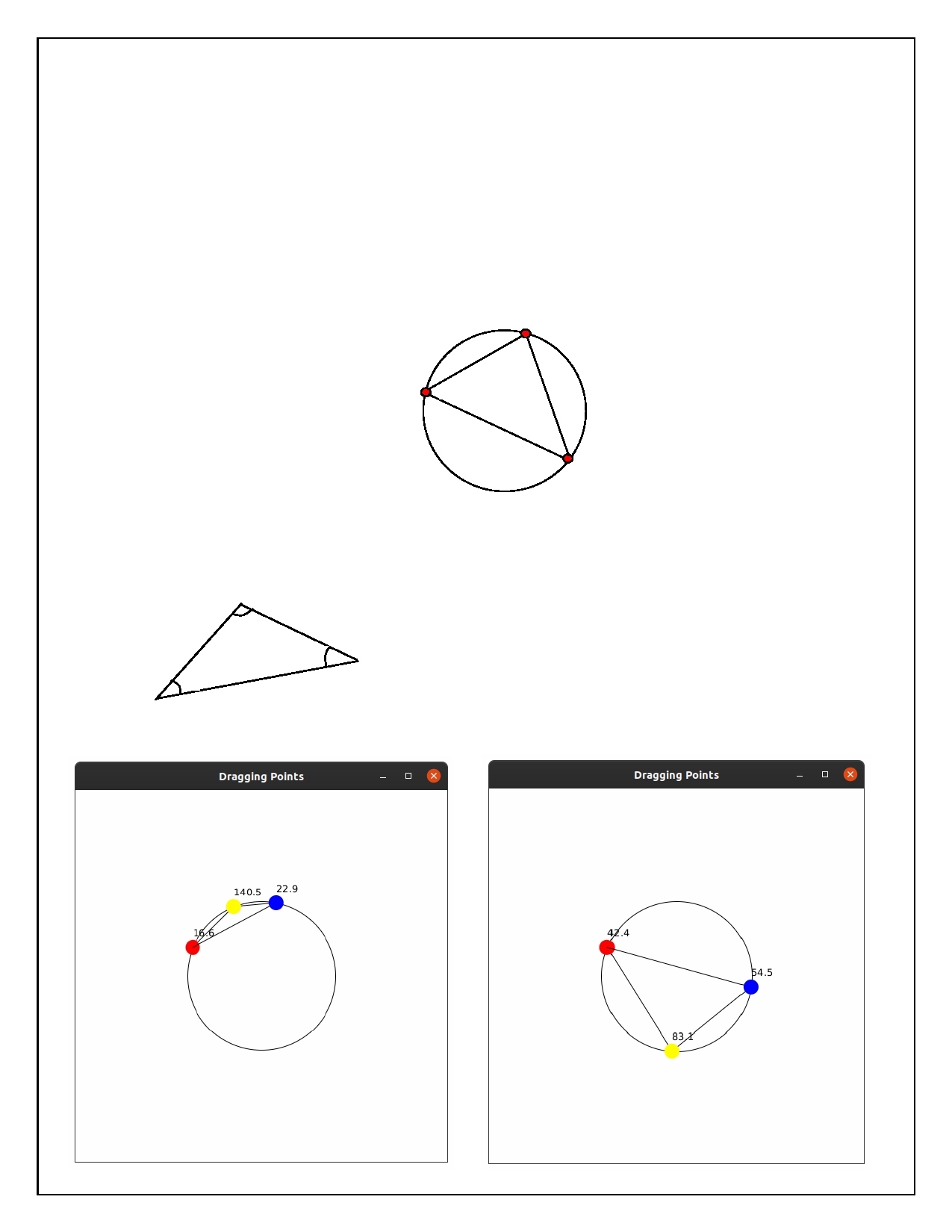
|  |
| --- |
|  |
|  |

|  |
| --- |
| // set text field |
|  |

|  |
| --- |
| v.setText(String.format("%.2f", valueCalc)); |
|  |

|  |
| --- |
| } |
|  |

}

**Question 3: Dragging Points on a Circle**

Problem Description:

Draw a circle with three random points on the circle. Connect the points to form a

triangle. Display the angles in the triangle. Use the mouse to drag a point along

the perimeter of the circle. As you drag it, the triangle and angles are redisplayed dynamically.

60

65

55

Here is the formula to compute angles:

c

x2, y2

B

a

C

A = acos((a \* a - b \* b - c \* c) / (-2 \* b \* c)) B = acos((b \* b - a \* a - c \* c) / (-2 \* a \* c)) C = acos((c \* c - b \* b - a \* a) / (-2 \* a \* b))

A x3, y3

b

x1, y1

Your Code:

Copy-paste your code here:

Screen shots:

Include two screen shots here:

|  |
| --- |
| package question3; |
|  |

|  |
| --- |
| /\* |
|  |

|  |
| --- |
| \* Ryan Christopher, 100702835 |
|  |

|  |
| --- |
| \* Mario Velazquez, 100702233 |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \* Software Systems CSCI2020u |
|  |

|  |
| --- |
| \* Assignment: question #1 |
|  |

|  |
| --- |
| \* Due: March 5, 2020 |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \* Draw a circle with three random points on the circle. Connect the points to form a |
|  |

|  |
| --- |
| \* triangle. Display the angles in the triangle. Use the mouse to drag a point along |
|  |

|  |
| --- |
| \* the perimeter of the circle. As you drag it, the triangle and angles are redisplayed |
|  |

|  |
| --- |
| \* dynamically. |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \*/ |
|  |

|  |
| --- |
| import javafx.application.Application; |
|  |

|  |
| --- |
| import javafx.geometry.Pos; |
|  |

|  |
| --- |
| import javafx.scene.Scene; |
|  |

|  |
| --- |
| import javafx.scene.layout.Pane; |
|  |

|  |
| --- |
| import javafx.scene.layout.StackPane; |
|  |

|  |
| --- |
| import javafx.scene.paint.Color; |
|  |

|  |
| --- |
| import javafx.scene.shape.Circle; |
|  |

|  |
| --- |
| import javafx.scene.shape.Line; |
|  |

|  |
| --- |
| import javafx.scene.text.Text; |
|  |

|  |
| --- |
| import javafx.stage.Stage; |
|  |

|  |
| --- |
| import java.util.Random; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import static java.lang.Math.\*; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public class DraggingPoints extends Application { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public static void main(String[] args) { |
|  |

|  |
| --- |
| launch(args); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public void start(Stage primaryStage) { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int radius=100; |
|  |

|  |
| --- |
| //make circle |
|  |

|  |
| --- |
| Circle mainCircle = new Circle(); |
|  |

|  |
| --- |
| mainCircle.setRadius(radius); |
|  |

|  |
| --- |
| mainCircle.setStroke(Color.BLACK); |
|  |

|  |
| --- |
| mainCircle.setFill(Color.WHITE); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //initialize arrays for the coordinates of the points |
|  |

|  |
| --- |
| Random rand = new Random(); |
|  |

|  |
| --- |
| double[] xPoints = new double[3]; |
|  |

|  |
| --- |
| double[] yPoints = new double[3]; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //loop through the points and create them |
|  |

|  |
| --- |
| for (int i = 0; i<3; i++) { |
|  |

|  |
| --- |
| //generate random x-coordinate |
|  |

|  |
| --- |
| xPoints[i] = rand.nextDouble() \* (radius); |
|  |

|  |
| --- |
| if (rand.nextBoolean()) {xPoints[i]\*=-1;} |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //calculate y-coordinate based on x-coordinate |
|  |

|  |
| --- |
| yPoints[i] = sqrt(radius\*radius - xPoints[i]\*xPoints[i]); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //randomize y (+) or (-) |
|  |

|  |
| --- |
| if (rand.nextBoolean()) { |
|  |

|  |
| --- |
| yPoints[i]\*=-1; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| //put points in proper position on screen |
|  |

|  |
| --- |
| xPoints[i]+=250; |
|  |

|  |
| --- |
| yPoints[i]+=250; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| //initialize and calculate array for angles |
|  |

|  |
| --- |
| double[] angles = calculateAngles(xPoints,yPoints); |
|  |

|  |
| --- |
| //make points |
|  |

|  |
| --- |
| Circle[] points = {new Circle(xPoints[0],yPoints[0],10,Color.RED), |
|  |

|  |
| --- |
| new Circle(xPoints[1],yPoints[1],10,Color.YELLOW), |
|  |

|  |
| --- |
| new Circle(xPoints[2],yPoints[2],10,Color.BLUE) |
|  |

|  |
| --- |
| }; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //make lines connecting the points |
|  |

|  |
| --- |
| //[0]= point 1 to 2, [1]= point 2 to 3, [2]= point 3 to 1 |
|  |

|  |
| --- |
| Line[] lines = {new Line(xPoints[0],yPoints[0],xPoints[1],yPoints[1]), |
|  |

|  |
| --- |
| new Line(xPoints[1],yPoints[1],xPoints[2],yPoints[2]), |
|  |

|  |
| --- |
| new Line(xPoints[2],yPoints[2],xPoints[0],yPoints[0]) |
|  |

|  |
| --- |
| }; |
|  |

|  |
| --- |
| //make text to display the angles between the points |
|  |

|  |
| --- |
| Text[] anglesTxt = {new Text(xPoints[0],yPoints[0]-15,String.format("%.1f", toDegrees(angles[0]))), |
|  |

|  |
| --- |
| new Text(xPoints[1],yPoints[1]-15,String.format("%.1f", toDegrees(angles[1]))), |
|  |

|  |
| --- |
| new Text(xPoints[2],yPoints[2]-15,String.format("%.1f", toDegrees(angles[2]))) |
|  |

|  |
| --- |
| }; |
|  |

|  |
| --- |
| //loop to set up MouseDragged listeners for each point |
|  |

|  |
| --- |
| for (int i=0;i<3;i++) { |
|  |

|  |
| --- |
| int finalI = i; |
|  |

|  |
| --- |
| points[i].setOnMouseDragged(e -> { |
|  |

|  |
| --- |
| //calculate x and y based on angle from (250,250) to cursor |
|  |

|  |
| --- |
| //angle = arctan(x-250/y-250) |
|  |

|  |
| --- |
| double angle = atan2(e.getY()-250,e.getX()-250); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| xPoints[finalI]= (radius\*cos(angle))+250; |
|  |

|  |
| --- |
| yPoints[finalI]= (radius\*sin(angle))+250; |
|  |

|  |
| --- |
| //set point positions |
|  |

|  |
| --- |
| points[finalI].setCenterX(xPoints[finalI]); |
|  |

|  |
| --- |
| points[finalI].setCenterY(yPoints[finalI]); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //reposition lines |
|  |

|  |
| --- |
| lines[finalI].setStartX(xPoints[finalI]); |
|  |

|  |
| --- |
| lines[finalI].setStartY(yPoints[finalI]); |
|  |

|  |
| --- |
| lines[(finalI+2)%3].setEndX(xPoints[finalI]); |
|  |

|  |
| --- |
| lines[(finalI+2)%3].setEndY(yPoints[finalI]); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //calculate angle of connecting lines |
|  |

|  |
| --- |
| double a = hypot(xPoints[1]-xPoints[2],yPoints[1]-yPoints[2]); |
|  |

|  |
| --- |
| double b = hypot(xPoints[0]-xPoints[2],yPoints[0]-yPoints[2]); |
|  |

|  |
| --- |
| double c = hypot(xPoints[1]-xPoints[0],yPoints[1]-yPoints[0]); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| angles[0] = acos((a\*a -b\*b - c\*c)/(-2\*b\*c)); |
|  |

|  |
| --- |
| angles[1] = acos((b\*b -a\*a - c\*c)/(-2\*a\*c)); |
|  |

|  |
| --- |
| angles[2] = acos((c\*c - a\*a -b\*b)/(-2\*a\*b)); |
|  |

|  |
| --- |
| //display updated angles |
|  |

|  |
| --- |
| for (int j=0;j<3;j++) { |
|  |

|  |
| --- |
| anglesTxt[j].setText(String.format("%.1f", toDegrees(angles[j]))); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| //reposition angle text |
|  |

|  |
| --- |
| anglesTxt[finalI].setX(xPoints[finalI]); |
|  |

|  |
| --- |
| anglesTxt[finalI].setY(yPoints[finalI]-15); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| }); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Pane pointsPane = new Pane(); |
|  |

|  |
| --- |
| //add elements to the window |
|  |

|  |
| --- |
| for (int i=0; i<3; i++) { |
|  |

|  |
| --- |
| pointsPane.getChildren().add(anglesTxt[i]); |
|  |

|  |
| --- |
| pointsPane.getChildren().add(lines[i]); |
|  |

|  |
| --- |
| pointsPane.getChildren().add(points[i]); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| StackPane mainPane = new StackPane(); |
|  |

|  |
| --- |
| mainPane.setAlignment(Pos.CENTER); |
|  |

|  |
| --- |
| mainPane.getChildren().addAll(mainCircle,pointsPane); |
|  |

|  |
| --- |
| mainPane.setPrefSize(500,500); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //final window setup and display |
|  |

|  |
| --- |
| primaryStage.setTitle("Dragging Points"); |
|  |

|  |
| --- |
| Scene scene = new Scene(mainPane); |
|  |

|  |
| --- |
| primaryStage.setResizable(false); |
|  |

|  |
| --- |
| primaryStage.setScene(scene); |
|  |

|  |
| --- |
| primaryStage.show(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| /\* |
|  |

|  |
| --- |
| \* This method takes in two double arrays of x and y coordinates and calculates the angles between them, |
|  |

|  |
| --- |
| \* then it returns an array of the calculated angles |
|  |

|  |
| --- |
| \*/ |
|  |

|  |
| --- |
| public double[] calculateAngles(double[] x,double[] y) { |
|  |

|  |
| --- |
| //calculate hypotenuse of angles |
|  |

|  |
| --- |
| double a = hypot(x[1]-x[2],y[1]-y[2]); |
|  |

|  |
| --- |
| double b = hypot(x[0]-x[2],y[0]-y[2]); |
|  |

|  |
| --- |
| double c = hypot(x[1]-x[0],y[1]-y[0]); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| double[] angle = new double[3]; |
|  |

|  |
| --- |
| //formula given in assignment document to calculate angles |
|  |

|  |
| --- |
| angle[0] = acos((a\*a -b\*b - c\*c)/(-2\*b\*c)); |
|  |

|  |
| --- |
| angle[1] = acos((b\*b -a\*a - c\*c)/(-2\*a\*c)); |
|  |

|  |
| --- |
| angle[2] = acos((c\*c - a\*a -b\*b)/(-2\*a\*b)); |
|  |

|  |
| --- |
| return angle; |
|  |

|  |
| --- |
| } |
|  |

}

Page **4** of **6**

**Question 4: Histogram**

Problem Description:

Develop a program that displays a histogram to show the occurrences of each

letter in a text area. The histogram should show the occurrences of each letter in a

text file, as shown in the following figure. Assume that the letters are not case sensitive.

Your Task:

• Place a pane that will display the histogram in the center of the frame.

• Place a label and a text field in a panel, and put the panel in the south side of

the frame. The text file will be entered from this text field.

• Pressing the Enter key on the text field causes the program to count the

occurrences of each letter and display the count in a histogram.

Your Code:

Copy-paste your code here:

Screen shots:

Include two screen shots here:

Page **5** of **6**

|  |
| --- |
| package question4; |
|  |

|  |
| --- |
| /\* |
|  |

|  |
| --- |
| \* Ryan Christopher, 100702835 |
|  |

|  |
| --- |
| \* Mario Velazquez, 100702233 |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \* Software Systems CSCI2020u |
|  |

|  |
| --- |
| \* Assignment: question #1 |
|  |

|  |
| --- |
| \* Due: March 5, 2020 |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \* Develop a program that displays a histogram to show the occurrences of each |
|  |

|  |
| --- |
| \* letter in a text area. The histogram should show the occurrences of each letter in a |
|  |

|  |
| --- |
| \* text file, as shown in the following figure. Assume that the letters are not case |
|  |

|  |
| --- |
| \* sensitive. |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \*/ |
|  |

|  |
| --- |
| import javafx.application.Application; |
|  |

|  |
| --- |
| import javafx.scene.Scene; |
|  |

|  |
| --- |
| import javafx.scene.chart.BarChart; |
|  |

|  |
| --- |
| import javafx.scene.chart.CategoryAxis; |
|  |

|  |
| --- |
| import javafx.scene.chart.NumberAxis; |
|  |

|  |
| --- |
| import javafx.scene.chart.XYChart; |
|  |

|  |
| --- |
| import javafx.scene.control.TextField; |
|  |

|  |
| --- |
| import javafx.scene.input.KeyCode; |
|  |

|  |
| --- |
| import javafx.scene.layout.BorderPane; |
|  |

|  |
| --- |
| import javafx.stage.Stage; |
|  |

|  |
| --- |
| import java.io.File; |
|  |

|  |
| --- |
| import java.io.FileNotFoundException; |
|  |

|  |
| --- |
| import java.util.Scanner; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public class Histogram extends Application { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public static void main(String[] args) { |
|  |

|  |
| --- |
| launch(args); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| private int[] numOccurr = new int[26]; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public void start(Stage primaryStage) { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| BorderPane pane = new BorderPane(); |
|  |

|  |
| --- |
| final CategoryAxis xAxis = new CategoryAxis(); |
|  |

|  |
| --- |
| final NumberAxis yAxis = new NumberAxis(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| BarChart<String, Number> histogram = new BarChart<>(xAxis,yAxis) ; |
|  |

|  |
| --- |
| histogram.setAnimated(false); |
|  |

|  |
| --- |
| histogram.setCategoryGap(0); |
|  |

|  |
| --- |
| xAxis.setLabel("Letter"); |
|  |

|  |
| --- |
| yAxis.setLabel("Occurrence"); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| XYChart.Series<String, Number> series1 = new XYChart.Series<>(); |
|  |

|  |
| --- |
| series1.setName("Letters"); |
|  |

|  |
| --- |
| String[] alphabet ={"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z"}; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| TextField fileInput = new TextField(); |
|  |

|  |
| --- |
| String[] fileName = {""}; |
|  |

|  |
| --- |
| //set on key action for input text field |
|  |

|  |
| --- |
| fileInput.setOnKeyPressed(e ->{ |
|  |

|  |
| --- |
| //when the enter key is pressed on the text field the text is received |
|  |

|  |
| --- |
| if (e.getCode() == KeyCode.ENTER) { |
|  |

|  |
| --- |
| fileName[0] = fileInput.getText(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //create file object and check if the file in the path exists |
|  |

|  |
| --- |
| File inputFile = new File(fileName[0]); |
|  |

|  |
| --- |
| if (!inputFile.exists()) { |
|  |

|  |
| --- |
| //output error msg |
|  |

|  |
| --- |
| System.out.println("File does not exist"); |
|  |

|  |
| --- |
| return; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| else { |
|  |

|  |
| --- |
| System.out.println("Reading file"); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| // create scanner to read the file |
|  |

|  |
| --- |
| Scanner input = null; |
|  |

|  |
| --- |
| try { |
|  |

|  |
| --- |
| input = new Scanner(inputFile); |
|  |

|  |
| --- |
| } catch (FileNotFoundException ex) { |
|  |

|  |
| --- |
| ex.printStackTrace(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| //reset the number of occurrences each time the enter button is pressed |
|  |

|  |
| --- |
| for (int i=0; i<26; i++) { |
|  |

|  |
| --- |
| numOccurr[i] = 0; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //run loop at least once ot receive input and if there is more that the scanner pick ups keep looping |
|  |

|  |
| --- |
| do { |
|  |

|  |
| --- |
| //get input |
|  |

|  |
| --- |
| assert input != null; |
|  |

|  |
| --- |
| String characters = input.next(); |
|  |

|  |
| --- |
| //loop through the string to count each letter |
|  |

|  |
| --- |
| for (int i = 0; i < characters.length(); i++) { |
|  |

|  |
| --- |
| //for each letter in the input check if it is equal to any letter in the alphabet |
|  |

|  |
| --- |
| for (int j = 0; j < 26; j++) { |
|  |

|  |
| --- |
| String currentChar = String.valueOf(characters.charAt(i)); |
|  |

|  |
| --- |
| //if characters are equal add 1 to the corresponding array index and break the loop |
|  |

|  |
| --- |
| if (currentChar.equalsIgnoreCase(alphabet[j])) { |
|  |

|  |
| --- |
| numOccurr[j] += 1; |
|  |

|  |
| --- |
| System.out.println(alphabet[j] + ": " + numOccurr[j]); |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } while (input.hasNext()); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| input.close(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //clear the histogram each time a file is searched to fix adding same series multiple times error |
|  |

|  |
| --- |
| histogram.getData().clear(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //add data to the series |
|  |

|  |
| --- |
| for (int x=0; x<26; x++) { |
|  |

|  |
| --- |
| series1.getData().add(new XYChart.Data<>(alphabet[x], numOccurr[x])); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| //add series to the chart |
|  |

|  |
| --- |
| histogram.getData().addAll(series1); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| }); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| //final window setup |
|  |

|  |
| --- |
| pane.setCenter(histogram); |
|  |

|  |
| --- |
| pane.setBottom(fileInput); |
|  |

|  |
| --- |
| Scene scene = new Scene(pane,800,800); |
|  |

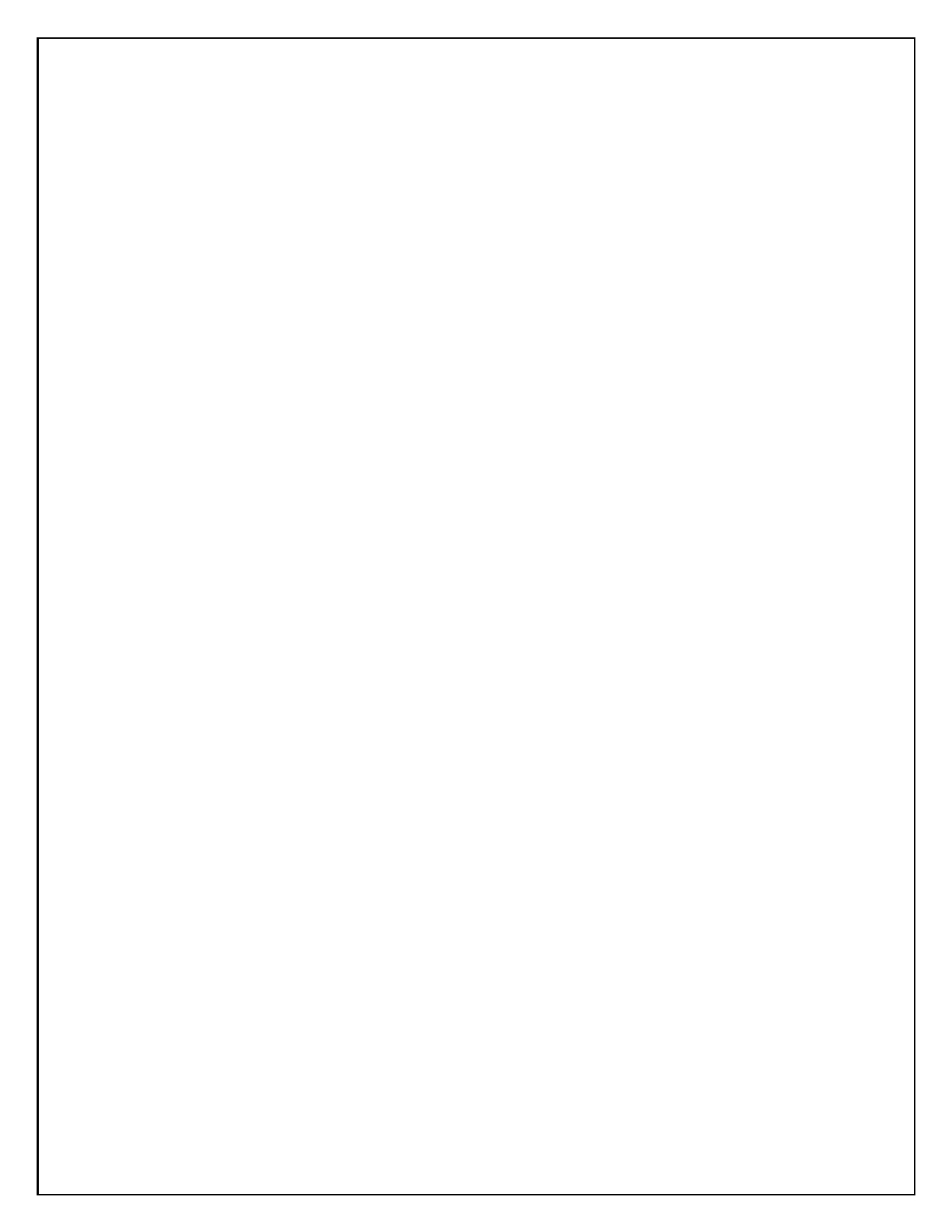
|  |
| --- |
| primaryStage.setTitle("Histogram"); |
|  |

|  |
| --- |
| primaryStage.setScene(scene); |
|  |

|  |
| --- |
| primaryStage.show(); |
|  |

|  |
| --- |
| } |
|  |

}

**Remember**:

You need to complete this file and submit it in related**rop box on Blackboard** , in **d**

addition to uploading your codes in your **git repository**, before deadline.

Page **6** of **6**