



## Faculty of Science

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

**Component:** Assignment

**Weight:** 10%

**Deadline:** March 6, 2019 (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.

## 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:

```
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import java.util.concurrent.ThreadLocalRandom;

public class Question1 extends Application {
    @Override // Override the start method in the Application class
    public void start(Stage primaryStage) {

        GridPane pane = new GridPane();
        pane.setAlignment(Pos.CENTER);
        pane.setHgap(5);
        pane.setVgap(5);

        for (int x = 0; x < 3; x += 1) {
            Image image1 = new Image(StringMaker());
            ImageView imageView = new ImageView(image1);
            pane.add(imageView, x, 0);
        }
    }
}
```

```

// Create a scene and place it in the stage
Scene scene = new Scene(pane);
primaryStage.setTitle("Question_1"); // Set the stage title
primaryStage.setScene(scene); // Place the scene in the stage
primaryStage.show(); // Display the stage

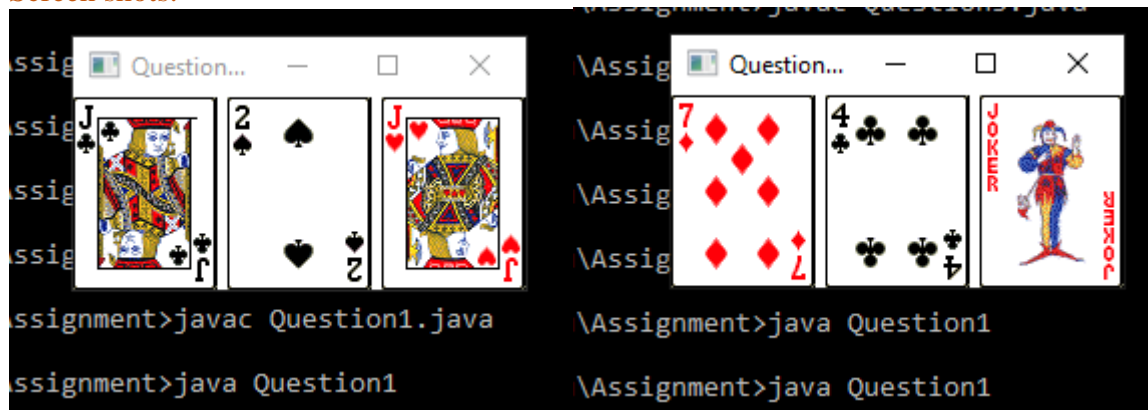
}

public static void main(String[] args) {
    launch(args);
}

public static String StringMaker() {
    int randomNum = ThreadLocalRandom.current().nextInt(1, 55);
    return "file:///C:/Users/HaydenLaptop/Documents/School 2018-2019/Winter/csci2020u/Assignment/Cards/" + randomNum + ".png";
}
}

```

Screen shots:

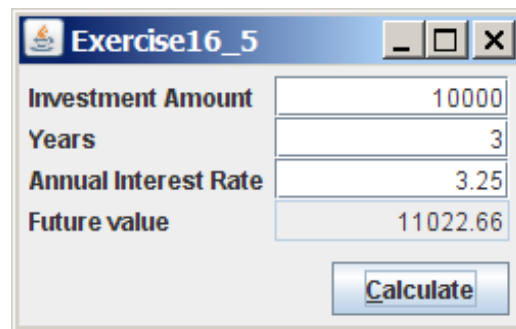


## Question 2: Investment-Value calculator

### 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:

$$\text{futureValue} = \text{investmentAmount} * (1 + \text{monthlyInterestRate})^{\text{years} * 12}$$



Investment Amount	10000
Years	3
Annual Interest Rate	3.25
Future value	11022.66

Calculate

### 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:

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class Question2 extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        GridPane gridPane = new GridPane();

        //Creating text fields and respective labels
        Label lAmount = new Label("Investment Amount");
```

```

TextField tfAmount = new TextField();
tfAmount.setAlignment(Pos.BASELINE_RIGHT);

Label lYears = new Label("Years");
TextField tfYears = new TextField();
tfYears.setAlignment(Pos.BASELINE_RIGHT);

Label lRate = new Label("Annual Interest Rate");
TextField tfRate = new TextField();
tfRate.setAlignment(Pos.BASELINE_RIGHT);

Label lFutureValue = new Label("Future Value");
TextField tfFutureValue = new TextField();
tfFutureValue.setAlignment(Pos.BASELINE_RIGHT);

//adding the calculate button
Button bCalculate = new Button("Calculate");
GridPane.setHalignment(bCalculate, HPos.RIGHT);

//adding handler for the calculate button
bCalculate.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        double amount = Double.parseDouble(tfAmount.getText());
        double years = Double.parseDouble(tfYears.getText());
        double interestRate = Double.parseDouble(tfRate.getText());
        double futureValue = calculateFutureVal(amount, years, interestRate);
        tfFutureValue.setText(String.valueOf(futureValue));
    }
});

//adding stuff to gridPane
gridPane.add(lAmount, 0, 0);
gridPane.add(tfAmount, 1, 0);

gridPane.add(lYears, 0, 1);
gridPane.add(tfYears, 1, 1);

gridPane.add(lRate, 0, 2);
gridPane.add(tfRate, 1, 2);

gridPane.add(lFutureValue, 0, 3);
gridPane.add(tfFutureValue, 1, 3);

gridPane.add(bCalculate, 1, 4);

gridPane.setPadding(new Insets(10));
gridPane.setVgap(7);

Scene scene = new Scene(gridPane);
primaryStage.setTitle("Question 2");
primaryStage.setScene(scene);
primaryStage.show();
}

public static double calculateFutureVal(double amount, double years, double
interestRate){
    double monthlyInterestRate = interestRate/1200;

```

```

        double futureValue = amount*Math.pow((1+monthlyInterestRate), years*12);
        return futureValue;
    }

    public static void main(String[] args) {
        Launch(args);
    }
}

```

Screen shots:

Question 2

Investment Amount

Years

Annual Interest Rate

Future Value

Question 2

Investment Amount

Years

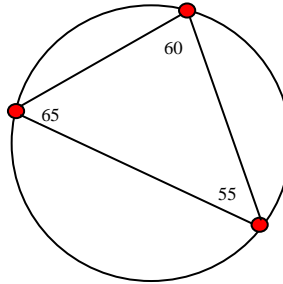
Annual Interest Rate

Future Value

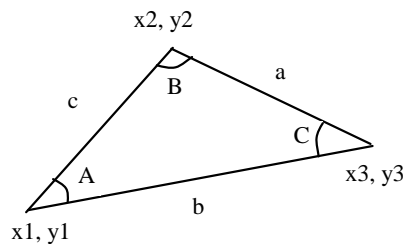
## Question 3: Dragging Points on a Circle

### 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.



Here is the formula to compute angles:



$$\begin{aligned} A &= \arccos((a^2 + b^2 - c^2) / (2 * a * b)) \\ B &= \arccos((b^2 + c^2 - a^2) / (2 * b * c)) \\ C &= \arccos((c^2 + a^2 - b^2) / (2 * c * a)) \end{aligned}$$

### Your Code:

Copy-paste your code here:

```
import javafx.application.Application;
import javafx.geometry.Point2D;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Line;

import java.util.ArrayList;
import java.util.Random;

public class Question3 extends Application{

    public void start(Stage primaryStage) throws Exception{
```

```

Pane pane = new Pane();
pane.setPrefSize(600, 600);

//creating the bigger circle
Circle bCircle = new Circle();
bCircle.setCenterX(300);
bCircle.setCenterY(300);
bCircle.setRadius(150);
bCircle.setStroke(Color.BLACK);
bCircle.setFill(Color.WHITE);
pane.getChildren().add(bCircle);

//creating three random smaller points/circles
ArrayList<Circle> circles = new ArrayList<>();
ArrayList<Point2D> points = new ArrayList<>();
for(int i = 0; i<3; i++){
    //generating a random angle
    Random random = new Random();
    int angle = random.nextInt(160)+20;

    Circle sCircle = new Circle();
    sCircle.setRadius(10);
    sCircle.setCenterX(bCircle.getRadius()*Math.cos(angle) +
bCircle.getCenterX());
    sCircle.setCenterY(bCircle.getRadius()*Math.sin(angle) +
bCircle.getCenterY());
    points.add(new Point2D(sCircle.getCenterX(), sCircle.getCenterY()));
    sCircle.setStroke(Color.BLACK);
    sCircle.setFill(Color.RED);
    circles.add(sCircle);
    pane.getChildren().add(sCircle);
}

//connecting the smaller circles with lines
Line l12 = new Line(circles.get(0).getCenterX(), circles.get(0).getCenterY(),
circles.get(1).getCenterX(), circles.get(1).getCenterY());
Line l23 = new Line(circles.get(1).getCenterX(), circles.get(1).getCenterY(),
circles.get(2).getCenterX(), circles.get(2).getCenterY());
Line l31 = new Line(circles.get(2).getCenterX(), circles.get(2).getCenterY(),
circles.get(0).getCenterX(), circles.get(0).getCenterY());
ArrayList<Line> lines = new ArrayList<>();
lines.add(l12);
lines.add(l23);
lines.add(l31);
pane.getChildren().addAll(l12, l23, l31);

//getting measurement of sides for angle calculation
double[] sides = calculateSides(lines);

//calculating angles
double[] angles = calculateAngle(sides[0], sides[2], sides[1]);

//displaying the angles
ArrayList<Text> texts = new ArrayList<>();
for(int i=0; i<3; i++){
    Text text = new Text(String.format("%.2f",angles[i]));
    text.setX(points.get(i).getX()+15);
    text.setY(points.get(i).getY());
    texts.add(text);
    pane.getChildren().add(text);
}

```



```

    }

    //adding handlers for moving smaller circles
    for(int i =0; i<3; i++){
        final int j = i;
        circles.get(i).setOnMouseDragged(event -> {
            //get coordinates wrt center of bigger circle
            double x = event.getX()-bCircle.getCenterX();
            double y = event.getY()-bCircle.getCenterY();

            //calculate the angle from center
            double tanTheta = Math.atan(y/x);
            if(x<0){
                tanTheta+=Math.PI;
            }
            //change the center
            double updatedX = bCircle.getRadius()*Math.cos(tanTheta);
            double updatedY = bCircle.getRadius()*Math.sin(tanTheta);
            circles.get(j).setCenterX(updatedX+bCircle.getCenterX());
            circles.get(j).setCenterY(updatedY+bCircle.getCenterY());

            //update lines
            if(j==0){
                lines.get(0).setStartX(updatedX+300);
                lines.get(0).setStartY(updatedY+300);

                lines.get(2).setEndX(updatedX+300);
                lines.get(2).setEndY(updatedY+300);
            }

            if(j==1){
                lines.get(1).setStartX(updatedX+300);
                lines.get(1).setStartY(updatedY+300);

                lines.get(0).setEndX(updatedX+300);
                lines.get(0).setEndY(updatedY+300);
            }

            if(j==2){
                lines.get(2).setStartX(updatedX+300);
                lines.get(2).setStartY(updatedY+300);

                lines.get(1).setEndX(updatedX+300);
                lines.get(1).setEndY(updatedY+300);
            }

            //updating the angles
            double[] updatedSides = calculateSides(lines);
            double[] updatedAngles = calculateAngle(updatedSides[0],
updatedSides[2], updatedSides[1]);

            for(int k =0; k<3; k++){
                texts.get(k).setText(String.format("%.2f", updatedAngles[k]));
                texts.get(k).setX(lines.get(k).getStartX()+10);
                texts.get(k).setY(lines.get(k).getStartY()+20);
            }
        });
    }
}

```

```

        primaryStage.setTitle("Question 3");
        primaryStage.setScene(new Scene(pane));
        primaryStage.show();
    }

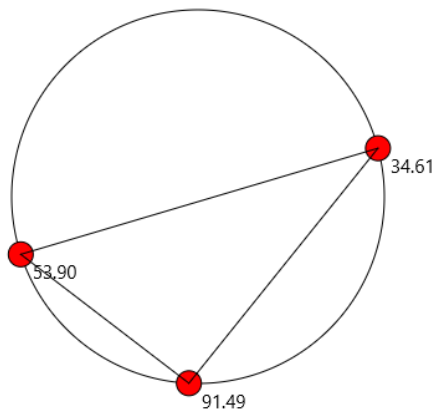
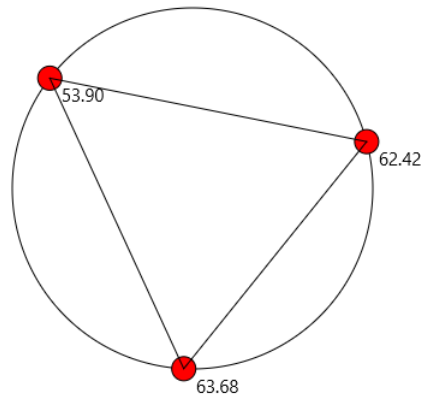
    public static double[] calculateAngle(double side1, double side2, double side3){
        double[] angles = new double[3];
        angles[0] = Math.acos((side3*side3-side1*side1-side2*side2)/(-
2*side1*side2))*180/Math.PI;
        angles[1] = Math.acos((side2*side2-side1*side1-side3*side3)/(-
2*side1*side3))*180/Math.PI;
        angles[2] = Math.acos((side1*side1-side2*side2-side3*side3)/(-
2*side2*side3))*180/Math.PI;
        return angles;
    }

    public static double[] calculateSides(ArrayList<Line> lines){
        Line l12 = lines.get(0);
        Line l23 = lines.get(1);
        Line l31 = lines.get(2);
        double[] sides = new double[3];
        sides[0] = Math.sqrt(Math.pow((l12.getStartX()-
l12.getEndX()),2)+Math.pow(l12.getStartY()-l12.getEndY(), 2));
        sides[1] = Math.sqrt(Math.pow((l23.getStartX()-
l23.getEndX()),2)+Math.pow(l23.getStartY()-l23.getEndY(), 2));
        sides[2] = Math.sqrt(Math.pow((l31.getStartX()-
l31.getEndX()),2)+Math.pow(l31.getStartY()-l31.getEndY(), 2));
        return sides;
    }

    public static void main(String[] args) {
        Launch(args);
    }
}

```

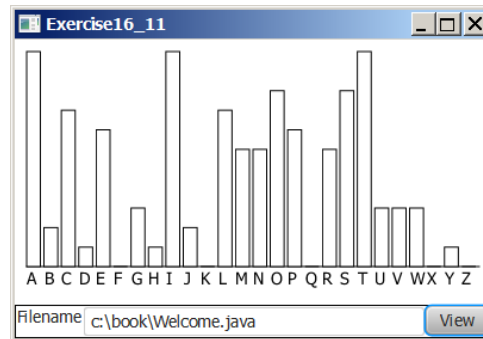
Screen shots:



## 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.

COMPILE THIS ONE WITH “javac Question4.java -Xlint”

### Your Code:

```
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.layout.BorderPane;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import javafx.application.Platform;
import java.lang.*;

public class Question4 extends Application {
    @Override
    public void start(Stage primaryStage) {
        CategoryAxis xAxis = new CategoryAxis();
        NumberAxis yAxis = new NumberAxis();
```

```

BarChart<String,Number> bc =
    new BarChart<>(xAxis,yAxis);
bc.setTitle("Question_4");
xAxis.setLabel("Letters");
yAxis.setLabel("Times");

int[] times = new int[26];

XYChart.Series series1 = new XYChart.Series();
series1.setName("Letters");
for (int i = 65; i <= 90; i += 1) {
    series1.getData().add(new XYChart.Data(Character.toString((char)i), times[i-65]));
}

TextField tf = new TextField();
tf.setOnAction(e -> {
    String text = String.valueOf(tf.getText());
    for (int i = 0; i < 26; i += 1) {
        times[i] = 0;
    }
    for (int i = 0; i < tf.getLength(); i += 1) {
        if (65 <= (int)Character.toUpperCase(text.charAt(i)) &&
(int)Character.toUpperCase(text.charAt(i)) <= 90) {
            times[(int)Character.toUpperCase(text.charAt(i))-65] += 1;
        }
    }
    series1.getData().clear();
    for (int i = 65; i <= 90; i += 1) {
        series1.getData().add(new XYChart.Data(Character.toString((char)i), times[i-65]));
    }
});

BorderPane borderPane = new BorderPane();
borderPane.setCenter(bc);
borderPane.setBottom(tf);
bc.setAnimated(false);
bc.getData().add(series1);

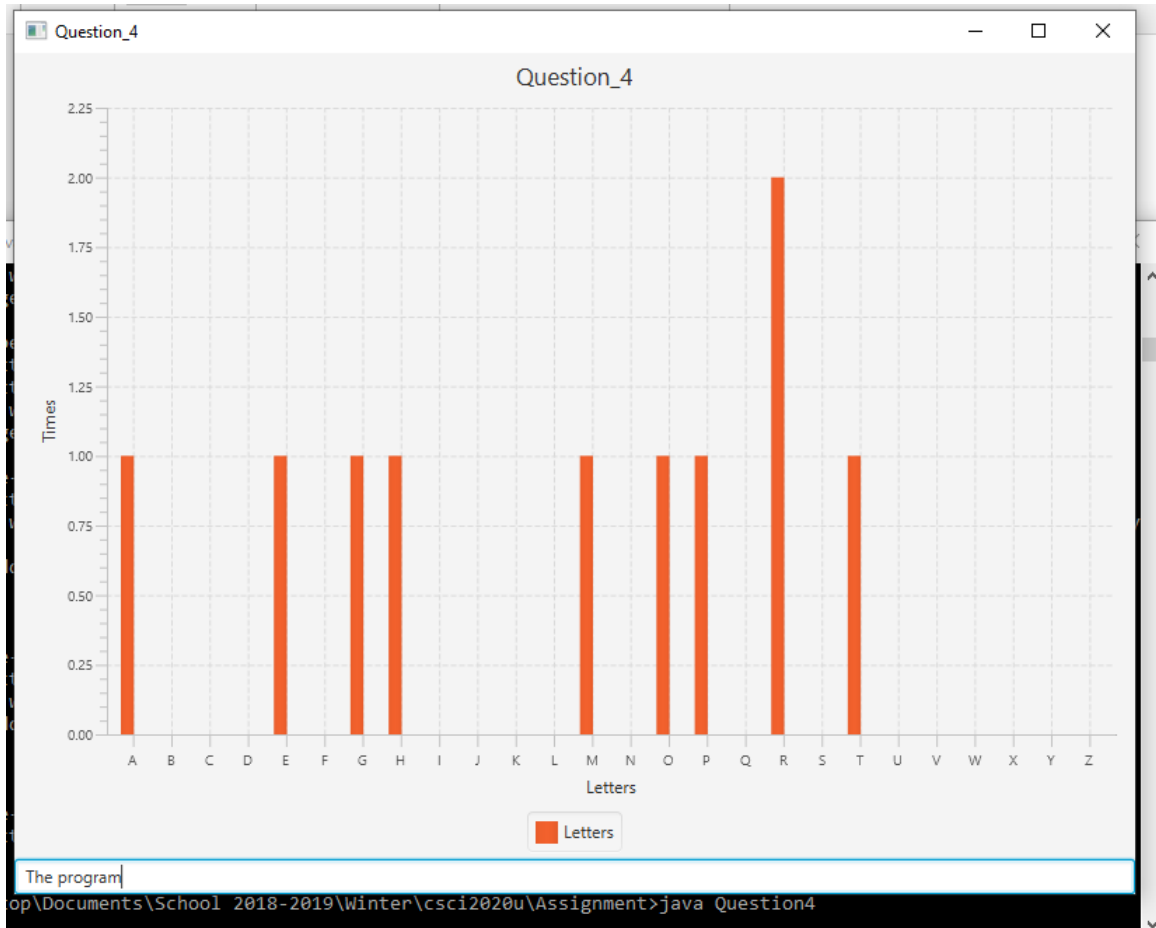
Scene scene = new Scene(borderPane,800,600);
primaryStage.setTitle("Question_4");
primaryStage.setScene(scene);
primaryStage.show();
}

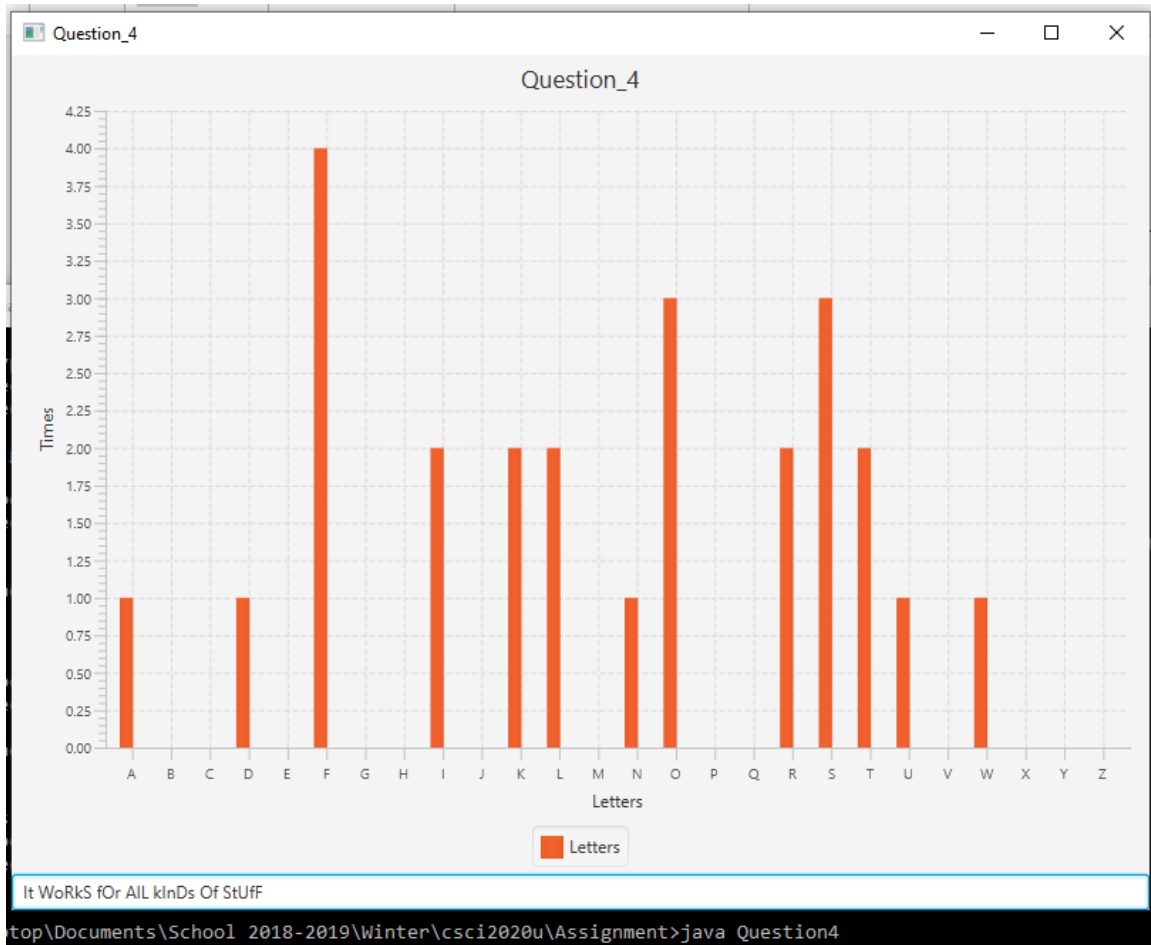
public static void main(String[] args) {
    launch(args);
}

```

}

### Screen shots:





**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.