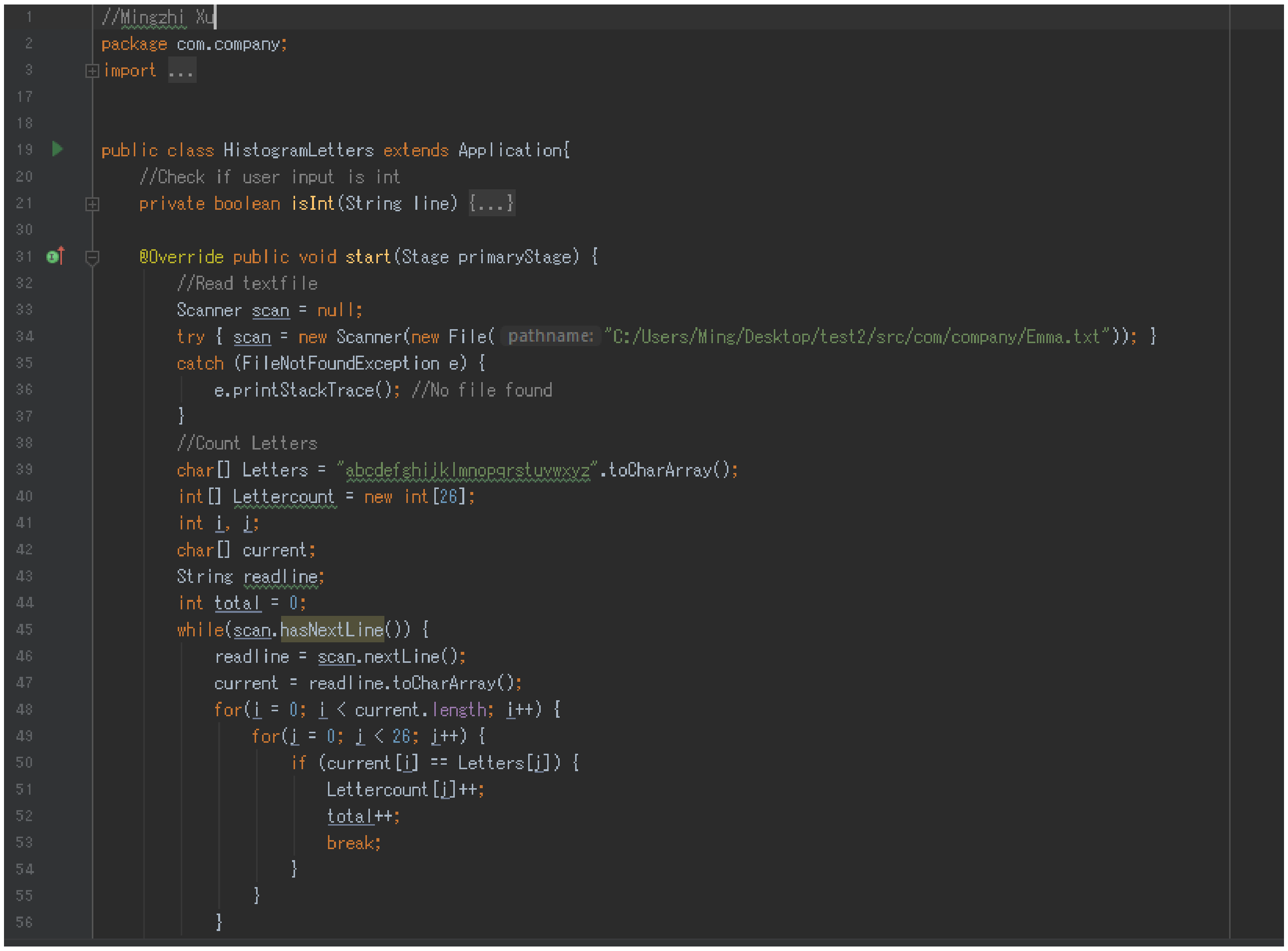
Mingzhi Xu

CSC 22100 Software Design Laboratory Fall 2018

Exercise 2

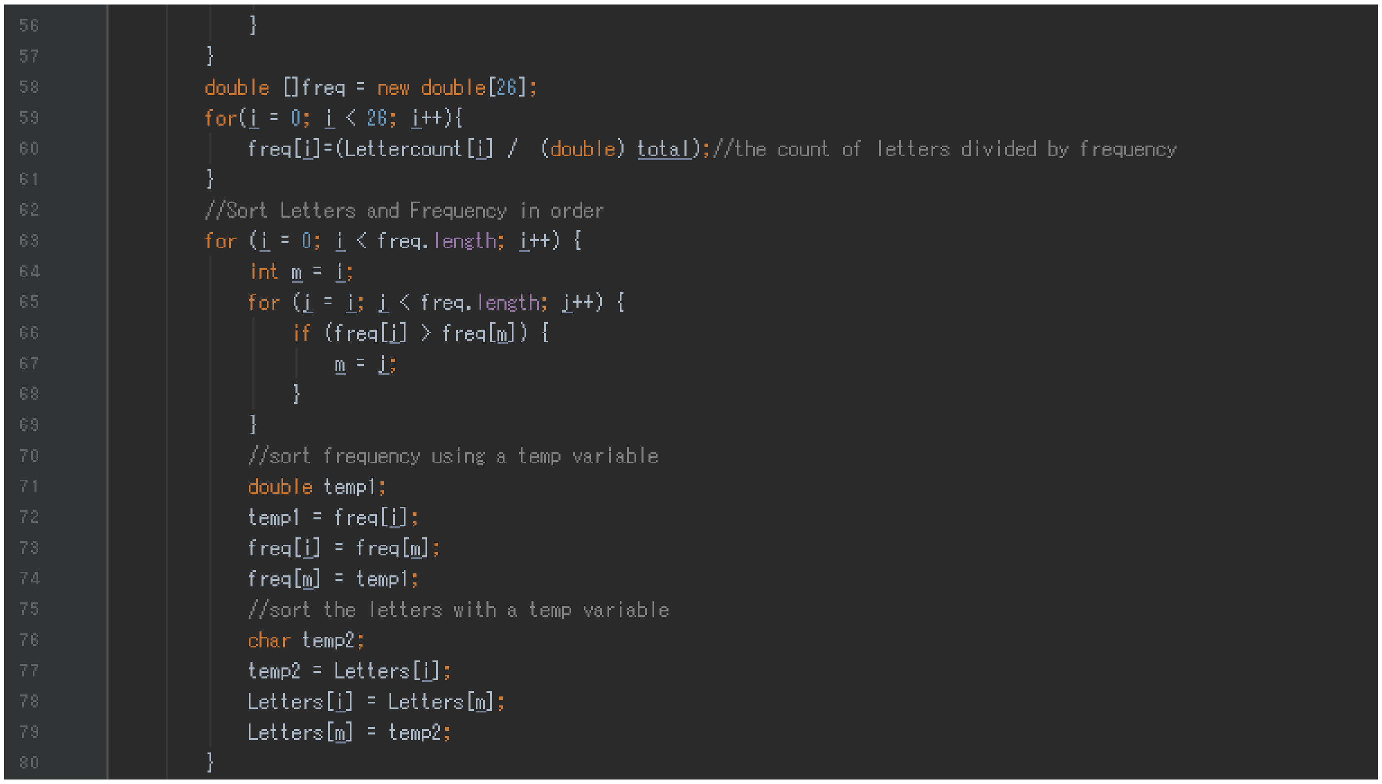
In this exercise, we will implement a Java class PieChart which will display a pie chart of the probabilities of the n most frequent occurrences of an event. We will also implement a Java class HistogramLetters that calculates the n most frequent letters in the given Emma.txt file by Jane Austen and their probabilities and this class will utilize the drawing pane to draw a pie chart of the letter probabilities.

Code for HistogramLetters class



```java
//Mingzhi Xu
package com.company;
import ...


public class HistogramLetters extends Application{
    //Check if user input is int
    private boolean isInt(String line) {...}

    @Override public void start(Stage primaryStage) {
        //Read textfile
        Scanner scan = null;
        try { scan = new Scanner(new File( pathname: "C:/Users/Ming/Desktop/test2/src/com/company/Emma.txt")); }
        catch (FileNotFoundException e) {
            e.printStackTrace(); //No file found
        }
        //Count Letters
        char[] Letters = "abcdefghijklmnopqrstuvwxyz".toCharArray();
        int[] Lettercount = new int[26];
        int i, j;
        char[] current;
        String readline;
        int total = 0;
        while(scan.hasNextLine()) {
            readline = scan.nextLine();
            current = readline.toCharArray();
            for(i = 0; i < current.length; i++) {
                for(j = 0; j < 26; j++) {
                    if (current[i] == Letters[j]) {
                        Lettercount[j]++;
                        total++;
                        break;
                    }
                }
            }
```

Here in the first part of the code, we will need a scanner to read the Emma.txt file in order to start anything. Then we create a char array that contains the letters a to z and an int array to store the counts of each letters that occurs in the Emma.txt file corresponds to the char array.

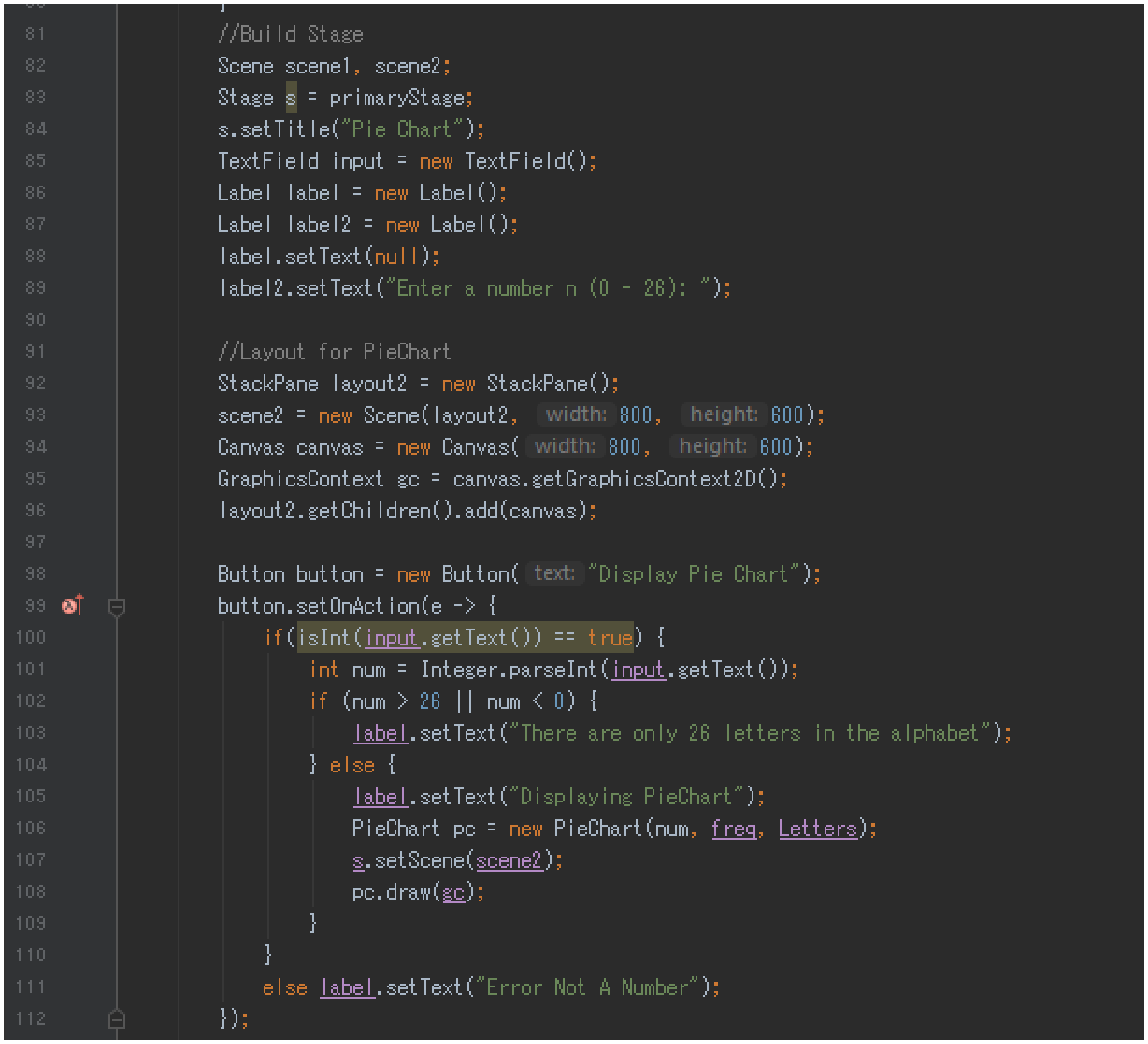


```java
            }
        }
        double []freq = new double[26];
        for(i = 0; i < 26; i++){
            freq[i]=(Lettercount[i] /  (double) total);//the count of letters divided by frequency
        }
        //Sort Letters and Frequency in order
        for (i = 0; i < freq.length; i++) {
            int m = i;
            for (j = i; j < freq.length; j++) {
                if (freq[j] > freq[m]) {
                    m = j;
                }
            }
            //sort frequency using a temp variable
            double temp1;
            temp1 = freq[i];
            freq[i] = freq[m];
            freq[m] = temp1;
            //sort the letters with a temp variable
            char temp2;
            temp2 = Letters[i];
            Letters[i] = Letters[m];
            Letters[m] = temp2;
        }
```

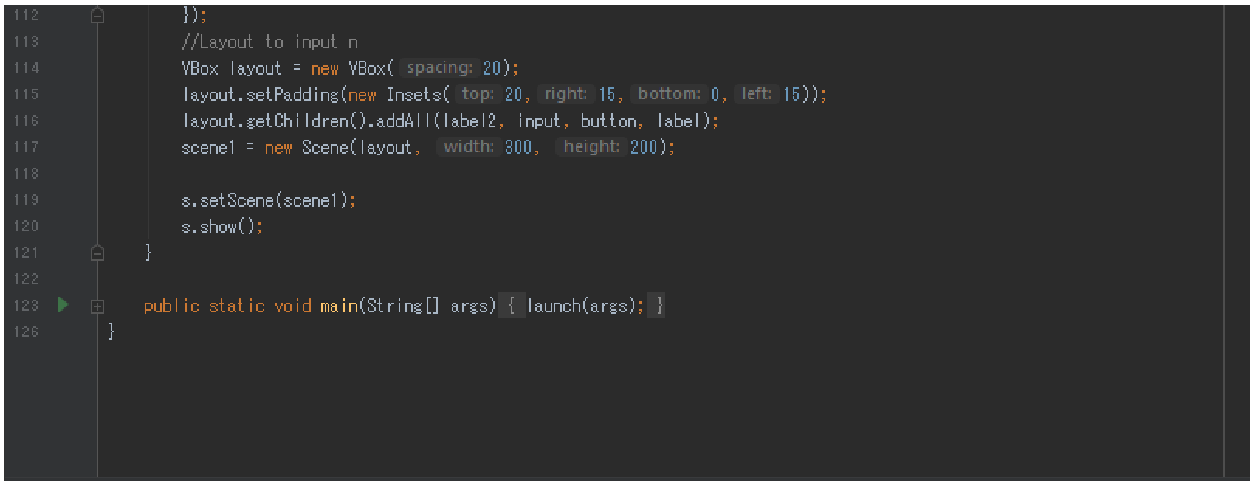
Next, in this part of the code, we will create a double array that calculates and store the frequency probabilities of each letter of the alphabet. Since the frequency of event and the total event is found from the first part, we can simply calculate the probability by using this formula.

$$Probability\ of\ event = \frac{Frequency\ of\ event}{\sum\ Frequencies\ of\ all\ events}$$

We will then sort the frequency in descending order and we will also alternate the latter array to corresponds to the position changes, so it matches the probabilities of event.

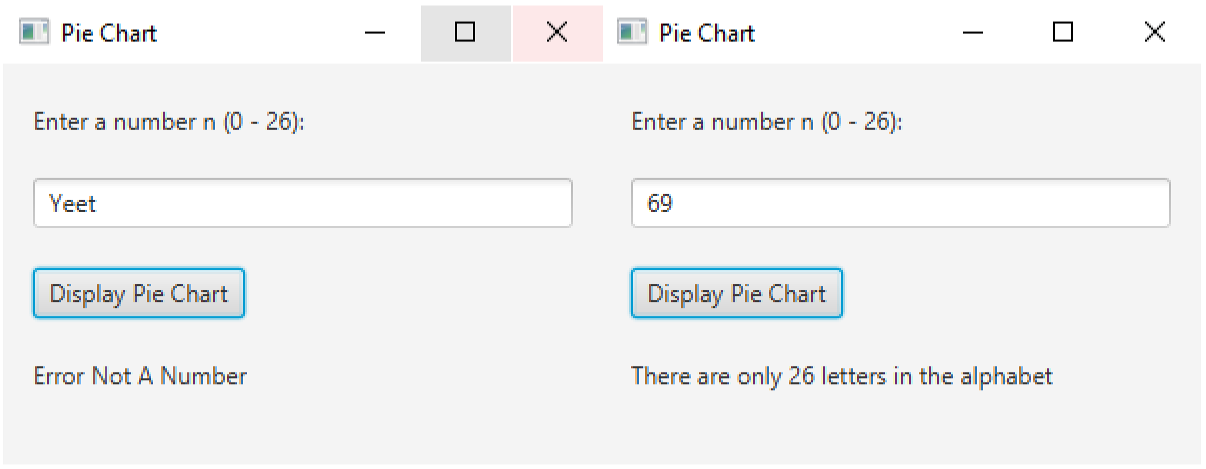


```java
        }
        //Build Stage
        Scene scene1, scene2;
        Stage s = primaryStage;
        s.setTitle("Pie Chart");
        TextField input = new TextField();
        Label label = new Label();
        Label label2 = new Label();
        label.setText(null);
        label2.setText("Enter a number n (0 - 26): ");

        //Layout for PieChart
        StackPane layout2 = new StackPane();
        scene2 = new Scene(layout2, width: 800, height: 600);
        Canvas canvas = new Canvas( width: 800, height: 600);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        layout2.getChildren().add(canvas);

        Button button = new Button( text: "Display Pie Chart");
        button.setOnAction(e -> {
            if(isInt(input.getText()) == true) {
                int num = Integer.parseInt(input.getText());
                if (num > 26 || num < 0) {
                    label.setText("There are only 26 letters in the alphabet");
                } else {
                    label.setText("Displaying PieChart");
                    PieChart pc = new PieChart(num, freq, Letters);
                    s.setScene(scene2);
                    pc.draw(gc);
                }
            }
            else label.setText("Error Not A Number");
        });
```



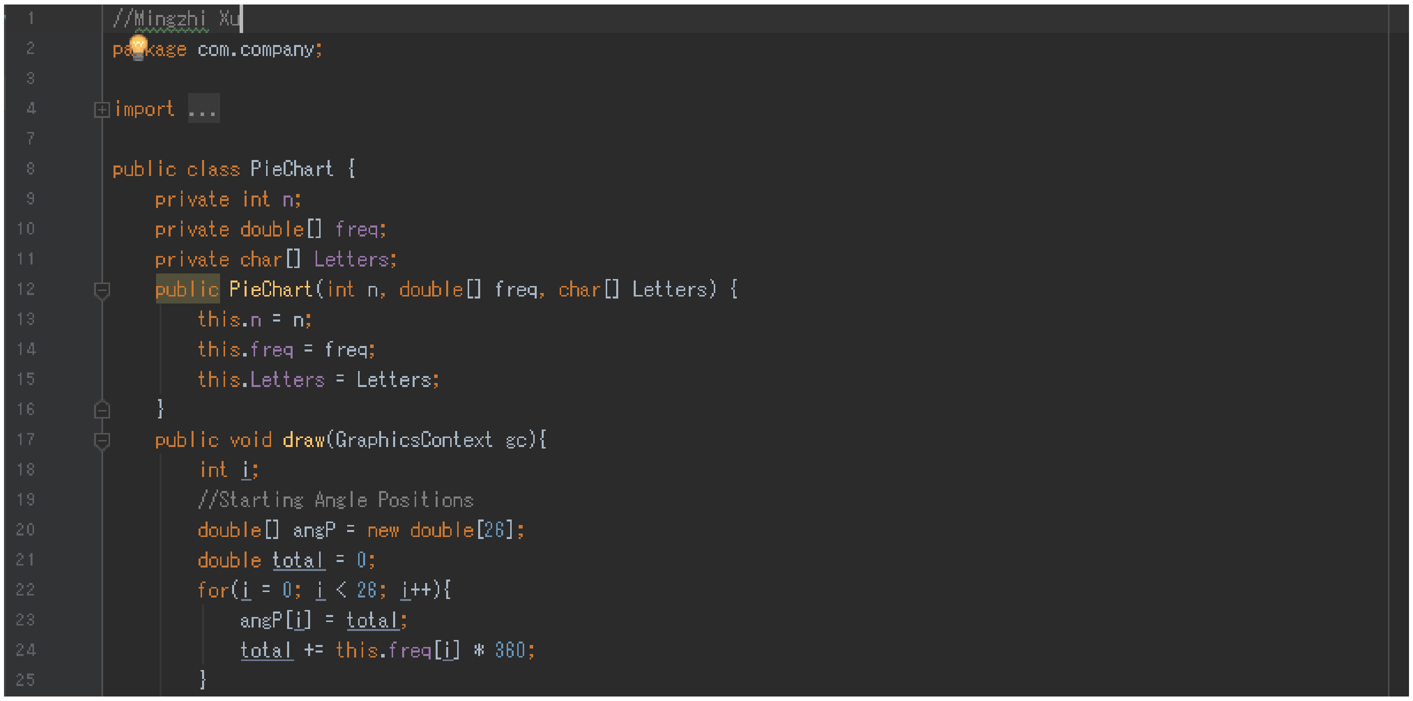
```java
        });
        //Layout to input n
        VBox layout = new VBox( spacing: 20);
        layout.setPadding(new Insets( top: 20, right: 15, bottom: 0, left: 15));
        layout.getChildren().addAll(label2, input, button, label);
        scene1 = new Scene(layout, width: 300, height: 200);

        s.setScene(scene1);
        s.show();
    }

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

This last part of the code from the HistogramLetters class is where the interface will be created and using the PieChart class. First, we make a scene call scene1 which will prompt for user input for n which to be used to determine the number element the pie chart will display later in another scene2. Since there are only 26 letters in the alphabet so we want the user to only enter numbers from 0 to 26, therefore, if the user inputs anything other than a number it will just print the text error not a number, and if the number is not between 0 and 26 then it will display the text that there are only 26 letters in the alphabet. If user input is successfully, then it will go to scene2 where the pie chart will display, which will be shown later in the end of the report.

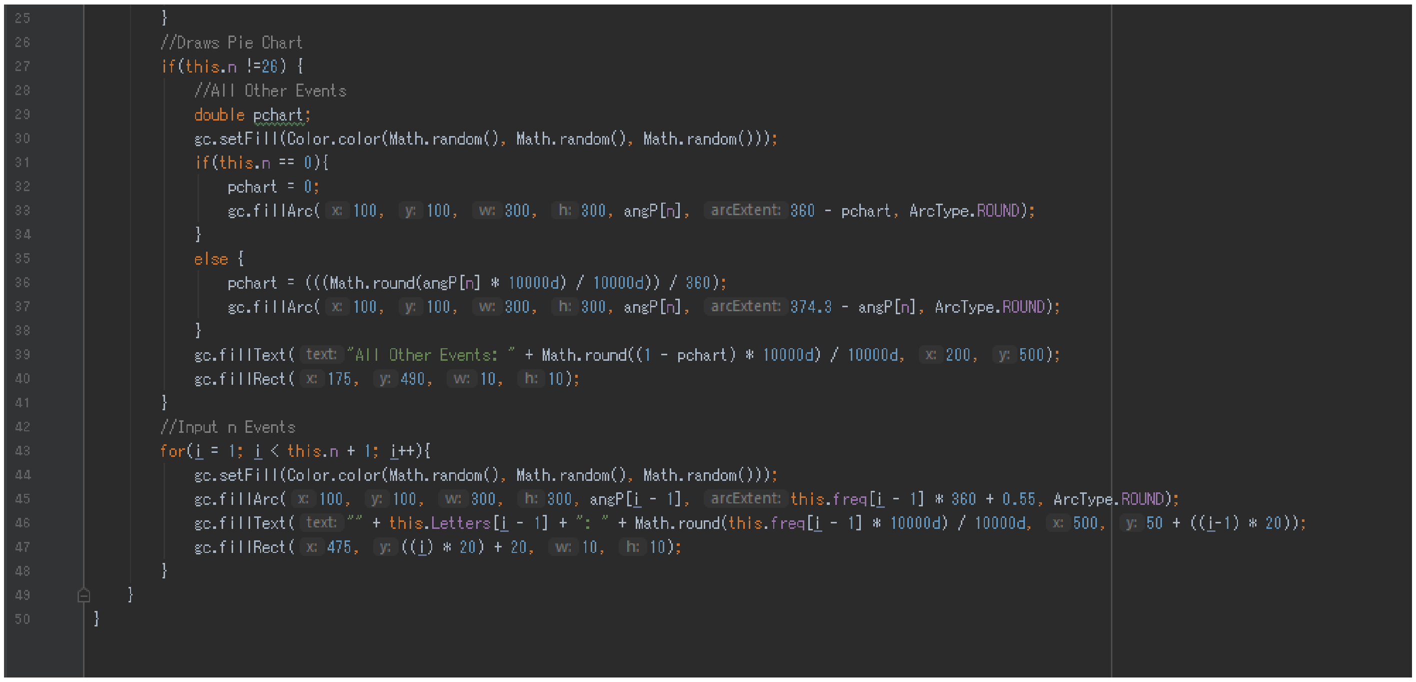

Pie Chart
Enter a number n (0 - 26):
Yeet
Display Pie Chart
Error Not A Number
Pie Chart
Enter a number n (0 - 26):
69
Display Pie Chart
There are only 26 letters in the alphabet

Now we will look at the PieChart class which includes appropriate constructors and a method draw that draws the pie chart. The PieChart object will take an integer n which determines by the user input to display the number of element on the pie chart, double array frequency probabilities which have been determined in the HistogramLetters class, which is same also for the char array Letters which have been sorted in descending order corresponds to the freq array.

Code for PieChart class



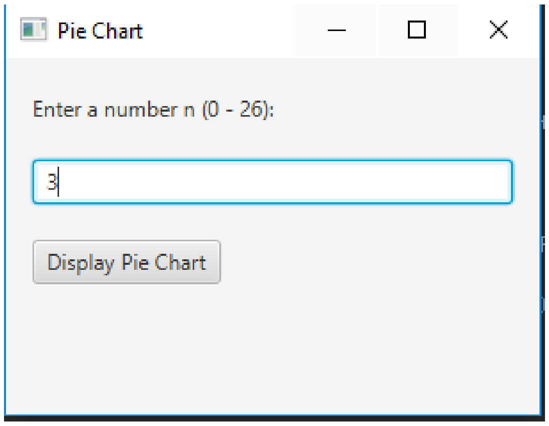
```java
//Mingzhi Xu
package com.company;

import ...

public class PieChart {
    private int n;
    private double[] freq;
    private char[] Letters;
    public PieChart(int n, double[] freq, char[] Letters) {
        this.n = n;
        this.freq = freq;
        this.Letters = Letters;
    }
    public void draw(GraphicsContext gc){
        int i;
        //Starting Angle Positions
        double[] angP = new double[26];
        double total = 0;
        for(i = 0; i < 26; i++){
            angP[i] = total;
            total += this.freq[i] * 360;
        }
```

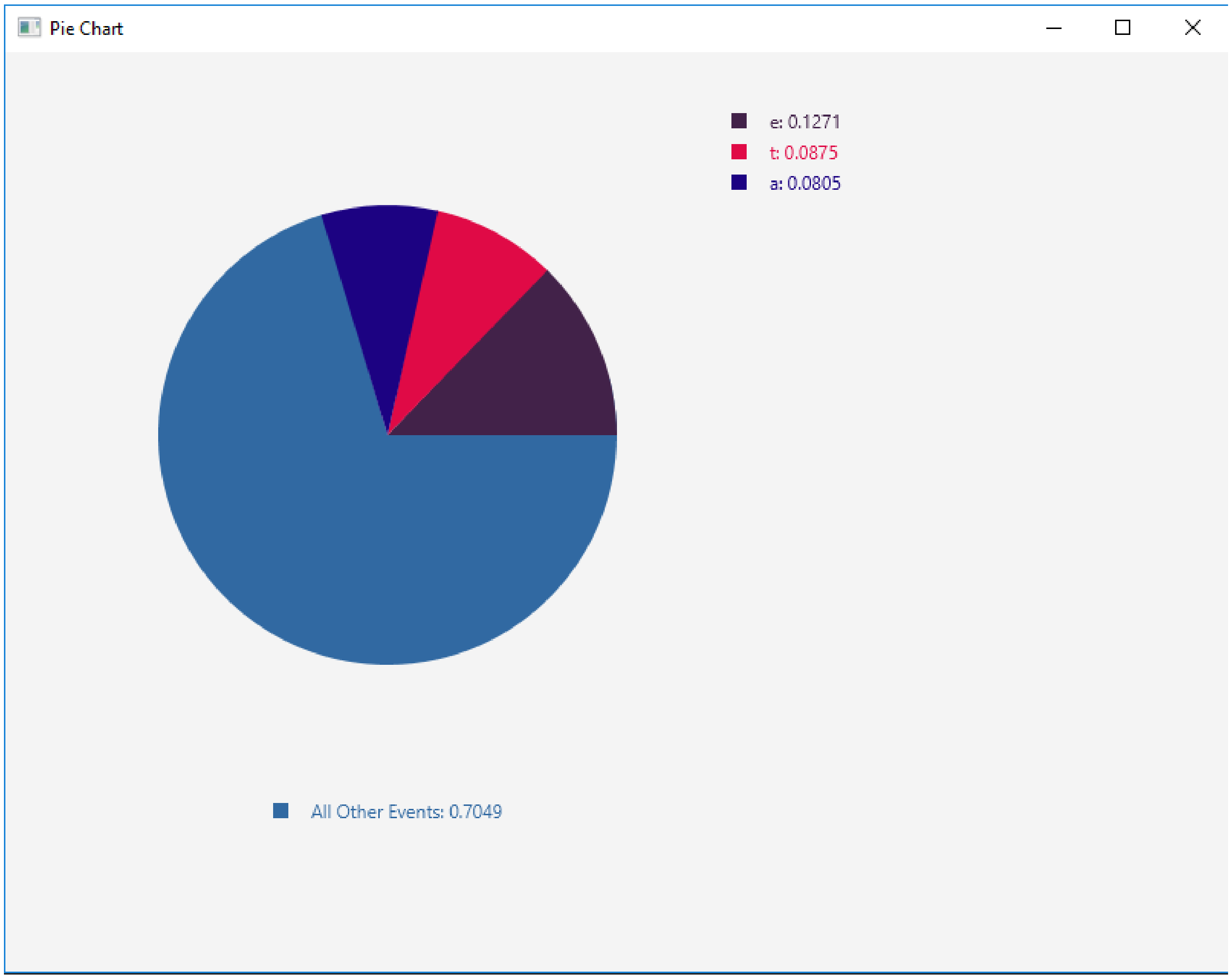


```java
        }
        //Draws Pie Chart
        if(this.n !=26) {
            //All Other Events
            double pchart;
            gc.setFill(Color.color(Math.random(), Math.random(), Math.random()));
            if(this.n == 0){
                pchart = 0;
                gc.fillArc( x: 100,  y: 100,  w: 300,  h: 300, angP[n],  arcExtent: 360 - pchart, ArcType.ROUND);
            }
            else {
                pchart = (((Math.round(angP[n] * 10000d) / 10000d)) / 360);
                gc.fillArc( x: 100,  y: 100,  w: 300,  h: 300, angP[n],  arcExtent: 374.3 - angP[n], ArcType.ROUND);
            }
            gc.fillText( text: "All Other Events: " + Math.round((1 - pchart) * 10000d) / 10000d,  x: 200,  y: 500);
            gc.fillRect( x: 175,  y: 490,  w: 10,  h: 10);
        }
        //Input n Events
        for(i = 1; i < this.n + 1; i++){
            gc.setFill(Color.color(Math.random(), Math.random(), Math.random()));
            gc.fillArc( x: 100,  y: 100,  w: 300,  h: 300, angP[i - 1],  arcExtent: this.freq[i - 1] * 360 + 0.55, ArcType.ROUND);
            gc.fillText( text: "" + this.Letters[i - 1] + ": " + Math.round(this.freq[i - 1] * 10000d) / 10000d,  x: 500,  y: 50 + ((i-1) * 20));
            gc.fillRect( x: 475,  y: ((i) * 20) + 20,  w: 10,  h: 10);
        }
    }
}
```

The only method in the PieChart class will be only the draw method which will draw the pie chart depending on how many elements based on the user input. First, we will make a double array angP which will store the starting angle of each elements then we will determine the color of each element using Math.random. Using methods from Graphics Context such as fillArc, fillText, and fillRect, we can draw the pie chart along with a legend that describes the pie chart.

Here are the results


Pie Chart
Enter a number n (0 - 26):
3
Display Pie Chart


Pie Chart
■ e: 0.1271
■ t: 0.0875
■ a: 0.0805
■ All Other Events: 0.7049