CSc 22100 HW 3

Leji Li

1. The Problem

This assignment is to read the text from a .txt file, count each letter in the file and show the frequnceies of the letters in a pie chart.

MyPieChart: This class can entends or modifies the **MyShape** class, and responses to the drawing of the pie chart graph and the labels of corresponding probabilities.

HistogramAlphaBet: This class takes the part of reading the .txt file, calculating the frequency of each letter and using the **MyPieChart** class to draw the graph.

2. Solution Methods

MyPieChart: drawing a sector in a pie chart uses *fillArc* method.

```
fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)
```

This method works like the *fillOvel*,

x, y are the top left corner of the boundering rectangle.

width and height are the width and height of the boundering rectangle.

startAngle is the lift-up angle accroding to the x axis, in degree.

arcAngle is the angle of the sector, in degree.

For the corresponding frequency label, it should be place in the middle point of the arc. But when the label is in the left side of the pie chart, the label should ends in the middle point of the arc, and starts at the point that has a distance of the label width away from that middle point. So that finding the width of the label is needed.

```
private double[] getTextWidth(Font font, String text){
 Text helper = new Text(text);
 helper.setFont(font);
 helper.setWrappingWidth(0);
 helper.setLineSpacing(0);
 // prefWidth pass-in -1 because node has null content-bias
 double w = helper.prefWidth(-1);
 helper.setWrappingWidth((int)Math.ceil(w));
 return new double[]{
  Math.ceil(helper.getLayoutBounds().getWidth()),
  Math.ceil(helper.getLayoutBounds().getHeight())
  };
```

```
}
```

This method is to find the width of a string in a specific font.

HistogramAlphaBet: Methods in this class are designed as static, so that other class can get access to the methods of this class without instantized an object.

readFile method reads the file and puts the character and its count into an map

```
public static Map<Character, Integer> readText(File file){
 Map<Character, Integer> map = new HashMap<Character, Integer>(26);
 for(char c = 'a'; c <= 'z'; c++){
 map.put(c, 0);
 }
 try {
 Scanner reader = new Scanner(file);
 while (reader.hasNextLine()){
 String line = cleanText(reader.nextLine());
  //System.out.println(line);
  for (int i = 0; i < line.length(); i++) {
  map.put(line.charAt(i), (map.get(line.charAt(i)) +1 ));
  }
  }
  reader.close();
  } catch (FileNotFoundException e) {
  System.out.println("Something wrong when reading the file.");
  e.printStackTrace();
  return null;
  }
  return map;
}
```

getFrequency method convers the map of char-count to char-frequency

```
public static<K> Map<K, Double> getFrequency(Map<K, ? extends Number> map){
 if(null == map){
 return null;
 }
 Map<K, Double> frequency = new HashMap<K, Double>(map.size());
 double sum = 0;
 for(Map.Entry<K, ? extends Number> entry: map.entrySet()){
```

```
 sum += entry.getValue().doubleValue();
 }
  for(Map.Entry<K, ? extends Number> entry: map.entrySet()){
  frequency.put(entry.getKey(), roundOff(entry.getValue().doubleValue() / su
m, 5));
  }
  return frequency;
}
```

sortMap sorts the frequency map

```
public static<K, V extends Comparable<? super V>> Map<K, V> sortMap(Map<K,
V> map){
  if(null == map){
  return null;
  }
  return map.entrySet()
  .stream()
  .sorted(Collections.reverseOrder(Map.Entry.<K, V>comparingByValue()))
  .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue, (e1, e2)
-> e2, LinkedHashMap::new));
}
```

Main: the appliction uses a *BorderLayout,* center part uses to display the pie chart, bottom part has a input so that user can type in nth characters can be shown, and the top part has a button to open a *FileChosser*, so that the application can has more flexiablity.

3. Codes developed

Main.java

```
package com.demo;

import javafx.application.Application;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
```

```
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.File;
import java.util.Map;

public class Main extends Application {

 private Map<Character, Double> frequency = null;
 private BorderPane borderPane;
 private Pane center;
 private TextField input;
 private double pieWidth = 500;
 private double pieHeight = 400;
 private int pieN = 3;

 public static void main(String[] args) {
 // write your code here
 launch(args);
 }

 @Override
 public void start(Stage primaryStage) {
 primaryStage.setTitle("Character Frequency");

 String filePath = "Alice in Wonderland.txt";
 // instantiate a FileChooser object to pick up other file
 FileChooser fileChooser = new FileChooser();
 // read txt file only
 fileChooser.getExtensionFilters().add(
```

```
48   new FileChooser.ExtensionFilter("Text Files", "*.txt")
49   );
50
51   this.borderPane = new BorderPane();
52   HBox top = new HBox();
53   HBox bottom = new HBox();
54   this.center = new Pane();
55
56   center.widthProperty().addListener(new ChangeListener<Number>() {
57   @Override
58   public void changed(ObservableValue<? extends Number> observable, Number o
ldValue, Number newValue) {
59   pieWidth = newValue.doubleValue();
60   updateCenter(pieWidth, pieHeight, pieN);
61   }
62   });
63
64   center.heightProperty().addListener(new ChangeListener<Number>() {
65   @Override
66   public void changed(ObservableValue<? extends Number> observable, Number o
ldValue, Number newValue) {
67   pieHeight = newValue.doubleValue();
68   updateCenter(pieWidth, pieHeight, pieN);
69   }
70   });
71
72   Label pathLabel = new Label(filePath);
73   Button newFile = new Button("New File");
74   top.getChildren().addAll(pathLabel, newFile);
75   top.setAlignment(Pos.CENTER);
76   top.setPrefWidth(500);
77   top.setSpacing(10);
78   top.setPadding(new Insets(10));
79
80   newFile.setOnAction(new EventHandler<ActionEvent>() {
81   @Override
82   public void handle(ActionEvent event) {
83   File selectedFile = fileChooser.showOpenDialog(primaryStage);
```

```
84  if(null == selectedFile)
85  return;
86  pathLabel.setText(selectedFile.getAbsolutePath());
87  frequency = HistogramAlphaBet.sortMap(
88  HistogramAlphaBet.getFrequency(HistogramAlphaBet.readText(selectedFile))
89  );
90  updateCenter(pieWidth, pieHeight, pieN);
91  }
92  });
93
94  Button submit = new Button("Draw");
95  Label display = new Label("Show n most frequent letter: ");
96  this.input = new TextField();
97  submit.setOnAction(new EventHandler<ActionEvent>() {
98  @Override
99  public void handle(ActionEvent event) {
100  pieN = getInputValue();
101  updateCenter(pieWidth, pieHeight, pieN);
102  }
103  });
104  bottom.getChildren().addAll(display, input, submit);
105  bottom.setAlignment(Pos.CENTER);
106  bottom.setPrefWidth(500);
107  bottom.setSpacing(10);
108  bottom.setPadding(new Insets(10));
109
110  frequency = HistogramAlphaBet.sortMap(
111  HistogramAlphaBet.getFrequency(HistogramAlphaBet.readText(new File(filePa
th)))
112  );
113
114  updateCenter(pieWidth, pieHeight, pieN);
115
116  borderPane.setTop(top);
117  borderPane.setCenter(center);
118  borderPane.setBottom(bottom);
119
120  primaryStage.setScene(new Scene(borderPane));
```

```
121   primaryStage.show();
122   }
123
124   /**
125   * get the input value from the textfield
126   * @return return the exact value if the input is valid,
127   * if input less that 0 return 3, if input greater 26 return 26
128   */
129   private int getInputValue(){
130   String s = this.input.getText();
131   int n = 0;
132   try{
133   n = Integer.parseInt(s);
134   }catch (Exception e){
135   n = 3;
136   }
137   if(n <= 0)
138   n = 3;
139   else if(n > 26)
140   n = 26;
141   input.setText(String.valueOf(n));
142   return n;
143   }
144
145   /**
146   * update the central view of the borderpane
147   * @param width new width of the center pie chart
148   * @param height new height of the center pie chart
149   * @param n new n of the center pie chart
150   */
151   private void updateCenter(double width, double height, int n){
152   if(null == center){
153   center = new Pane();
154   }
155   center.getChildren().clear();
156   center.getChildren().add(HistogramAlphaBet.getPieChartCanvas(frequency, w
idth, height, n));
157   }
```

```
158 }
```

HistogramAlphaBet.java

```
1  package com.demo;
2
3  import javafx.scene.canvas.Canvas;
4  import javafx.scene.canvas.GraphicsContext;
5
6  import java.io.File;
7  import java.io.FileNotFoundException;
8  import java.util.*;
9  import java.util.stream.Collectors;
10
11 public class HistogramAlphaBet {
12
13  /**
14  * take out all non-letter characters in a string
15  * @param txt the string that needs to handle
16  * @return a string that contains letter only
17  */
18  private static String cleanText(String txt){
19  return txt.replaceAll("[^a-zA-Z]", "").toLowerCase();
20  }
21
22  /**
23  * rounding a digit in to given decimal place
24  * @param origin
25  * @param decimalPlace
26  * @return
27  */
28  public static double roundOff(double origin, int decimalPlace){
29  double rounding = Math.pow(10, decimalPlace);
30  return ((double)Math.round(origin * rounding)) / rounding;
31  }
32
33  /**
34  * count the number of the characters in alphabet
```

```
35   * @param file the file provides that text context
36   * @return an unsorted map that records that count of each characters in al
phabet
37   */
38   public static Map<Character, Integer> readText(File file){
39   Map<Character, Integer> map = new HashMap<Character, Integer>(26);
40   for(char c = 'a'; c <= 'z'; c++){
41   map.put(c, 0);
42   }
43   try {
44   Scanner reader = new Scanner(file);
45   while (reader.hasNextLine()){
46   String line = cleanText(reader.nextLine());
47   //System.out.println(line);
48   for (int i = 0; i < line.length(); i++) {
49   map.put(line.charAt(i), (map.get(line.charAt(i)) +1 ));
50   }
51   }
52   reader.close();
53   } catch (FileNotFoundException e) {
54   System.out.println("Something wrong when reading the file.");
55   e.printStackTrace();
56   return null;
57   }
58   return map;
59   }
60
61   /**
62   * calculate the frequency of each entries in a map
63   * @param map the map provides the source data, they type of the value shou
ld be
64   * @param <K> the type of key of the map
65   * @return an unsorted map that contains the frequency data of the source m
ap
66   */
67   public static<K> Map<K, Double> getFrequency(Map<K, ? extends Number> map)
{
68   if(null == map){
```

```
69   return null;
70   }
71   Map<K, Double> frequency = new HashMap<K, Double>(map.size());
72   double sum = 0;
73   for(Map.Entry<K, ? extends Number> entry: map.entrySet()){
74   sum += entry.getValue().doubleValue();
75   }
76   for(Map.Entry<K, ? extends Number> entry: map.entrySet()){
77   frequency.put(entry.getKey(), roundOff(entry.getValue().doubleValue() / su
m, 5));
78   }
79   return frequency;
80   }
81
82   /**
83   * sort the map in descending order
84   * @param map source map
85   * @param <K> type of the key
86   * @param <V> type of the value. This type must be comparable
87   * @return a sorted map in descending order, according to the value
88   */
89   public static<K, V extends Comparable<? super V>> Map<K, V> sortMap(Map<K,
V> map){
90   if(null == map){
91   return null;
92   }
93   return map.entrySet()
94   .stream()
95   .sorted(Collections.reverseOrder(Map.Entry.<K, V>comparingByValue()))
96   .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue, (e1, e2)
-> e2, LinkedHashMap::new));
97   }
98
99   public static Canvas getPieChartCanvas(Map<Character, Double> map, double
width, double height, int n){
100   Canvas canvas = new Canvas(width, height);
101   double r = 0.6 * Math.min(width, height) / 2;
102   MyPieChart chart = new MyPieChart(width/2, height/2, r);
```

```
103   chart.setFrequency(map);
104   chart.setDisplayCount(n);
105   GraphicsContext gc = canvas.getGraphicsContext2D();
106   chart.draw(gc);
107   return canvas;
108   }
109  }
```

MyPieChart.java

```
1  package com.demo;
2
3  import javafx.scene.canvas.GraphicsContext;
4  import javafx.scene.paint.Color;
5  import javafx.scene.shape.ArcType;
6  import javafx.scene.text.Font;
7  import javafx.scene.text.Text;
8
9  import java.util.ArrayList;
10  import java.util.Iterator;
11  import java.util.List;
12  import java.util.Map;
13
14  public class MyPieChart extends MyShape {
15   public static final int DEFAULT_DISPLAY_COUNT = 3;
16   private Map<Character, Double> frequency;
17   private double r;
18   private int n = 0;
19   private static List<Color> colorList = new ArrayList<Color>();
20
21   public MyPieChart(double x, double y, double r, int n, Map<Character, Doub
le> frequency){
22   super(x, y);
23   this.r = r;
24   setFrequency(frequency);
25   setDisplayCount(n);
26   }
27
28   public MyPieChart(double x, double y, double r){
```

```
29  this(x, y, r, 3, null);
30  }
31
32  public void setFrequency(Map<Character, Double> frequency){
33  this.frequency = frequency;
34  }
35
36  /**
37  * set the first nth item to show
38  * @param n
39  */
40  public void setDisplayCount(int n){
41  if(n <= 0 ){
42  this.n = DEFAULT_DISPLAY_COUNT;
43  } else if(n > 26){
44  this.n = 26;
45  } else {
46  this.n = n;
47  }
48  // when updating the nth item
49  // update the color list as well
50  if(this.n > colorList.size()){
51  for (int i = colorList.size(); i < this.n; i++){
52  colorList.add(MyColor.randomColor());
53  }
54  }
55  }
56
57  /**
58  * this function is to find how wide a string should occupies in a certain
font
59  * @param font some kind of font
60  * @param text the target string
61  * @return the width and height of the text will takes under the given font
62  * [0] is the width
63  * [1] is the height
64  */
65  private double[] getTextWidth(Font font, String text){
```

```
66  Text helper = new Text(text);
67  helper.setFont(font);
68  helper.setWrappingWidth(0);
69  helper.setLineSpacing(0);
70  // prefWidth pass-in -1 because node has null content-bias
71  double w = helper.prefWidth(-1);
72  helper.setWrappingWidth((int)Math.ceil(w));
73  return new double[]{
74  Math.ceil(helper.getLayoutBounds().getWidth()),
75  Math.ceil(helper.getLayoutBounds().getHeight())
76  };
77  }
78
79  @Override
80  public void draw(GraphicsContext gc) {
81  if(frequency == null || frequency.size() == 0){
82  String hint = "No data to display";
83  Font font = new Font(16);
84  double[] size = getTextWidth(font, hint);
85  gc.setFont(font);
86  gc.setFill(MyColor.Black.toFXPaintColor());
87  gc.fillText(hint, getX() - size[0]/2, getY() - size[1]/2);
88  return;
89  }
90  Iterator iterator;
91  int count = 0;
92  // if n is greater or equal to 25, that is all the character should be sho
wn
93  // take the first n frequencies to sum up the n-frequency
94  // 1- n-frequency is the frequency of the reset as a whole entry
95  if(this.n < 25){
96  iterator = frequency.entrySet().iterator();
97  double sum = 0d;
98  while((iterator.hasNext()) && (count++ < this.n)){
99  sum += ((Map.Entry<Character, Double>)(iterator.next())).getValue();
100  }
101  if(sum < 1d){
102  frequency.put('A', HistogramAlphaBet.roundOff(1d - sum, 5));
```

```
103   frequency = HistogramAlphaBet.sortMap(frequency);
104   setDisplayCount(this.n + 1);
105   }
106   count = 0;
107   }
108   // the sector is a part of a circle
109   // fillArc works like the fillOval
110   // it takes the top left corner of the bounding box
111   // arc width and arc height are the radius of the circle
112   // startAngle is the angle between the x axis and the right side of the s
ector, counterclockwise
113   // arcExtent is the angle of the sector
114   double startingX = this.getX() - this.r; // x of the top left corner
115   double startingY = this.getY() - this.r; // y of the top left corner
116   double sectorAngle = 0d; // arcExtent
117   double angleShift = 90d; // startAngle
118   double midAngle = 0d; // the angle of middle of a sector, use to locate t
he label
119   double labelX = 0d, labelY = 0d; // the starting point of a label
120   Font font = new Font(12); // use to unify the font of the label and find
the length
121   iterator = frequency.entrySet().iterator(); // draw the sectors
122   while(iterator.hasNext() && (count < this.n)){
123   Map.Entry<Character, Double> entry = (Map.Entry<Character, Double>)iterat
or.next();
124   // all angles passed in to the fillArc method should be in degree
125   sectorAngle = 360d * entry.getValue();
126   gc.setFill(colorList.get(count));
127   gc.fillArc(startingX, startingY, this.r * 2, this.r * 2, angleShift, sect
orAngle, ArcType.ROUND);
128
129   // prepare for the label of a sector
130   String labelText ="";
131   char c = entry.getKey();
132   if(c == 'A'){
133   labelText += "All other letters";
134   } else {
135   labelText += c;
136   }
```

```
137   labelText += (": " + entry.getValue());
138
139   // find the location of the label of a sector
140   // it should be placed to the middle of its sector
141   // r*1.05 makes the label has the padding to the piechart
142   midAngle = (angleShift + sectorAngle / 2);
143   double xShift = this.r * 1.05 * Math.cos(Math.toRadians(midAngle));
144   labelX = this.getX() + xShift;
145   labelY = this.getY() - this.r *1.05 * Math.sin(Math.toRadians(midAngle));
146
147   // handle the spacing
148   // since the width of the label is not constant
149   // we need to find out the label width
150   // and to avoid the label overlapping with the piechart or going out of t
he window
151   // we should take the minimum of them, and set this value as the maximum
of the label
152   double labelWidth = getTextWidth(font, labelText)[0];
153   if(xShift < 0){
154   // if the label is in the left side of the piechart
155   // the starting point should be the middle point of the sector arc minus
the label width
156   // r * 0.006 is the margin of the window
157   labelWidth = Math.min(labelWidth, labelX - this.r * 0.06);
158   labelX -= labelWidth;
159   } else {
160   labelWidth = Math.min(labelWidth, (2*getX() - labelX) - this.r * 0.06);
161   }
162
163   gc.setFill(MyColor.Black.toFXPaintColor());
164   gc.setFont(font);
165   gc.fillText(labelText, labelX, labelY, labelWidth);
166
167   // accumulate the starting angle for the next sector
168   angleShift += sectorAngle;
169   count++;
170   }
171   }
```

```
172 }
```

MyShape.java

```
1  package com.demo;
2  import javafx.scene.canvas.GraphicsContext;
3
4  import java.awt.*;
5
6  public abstract class MyShape extends Object {
7   private double x;
8   private double y;
9   private MyColor color;
10
11   public MyShape(double x, double y, MyColor color){
12   this.x = x;
13   this.y = y;
14   this.color = color;
15   }
16   public MyShape(double x, double y){
17   this(x, y, MyColor.Black);
18   }
19   public MyShape(MyColor color){
20   this(0, 0, color);
21   }
22
23   public MyShape(){
24   this(0, 0);
25   }
26
27   public double getX() {
28   return x;
29   }
30
31   public void setX(double x) {
32   this.x = x;
33   }
34
```

```
35   public double getY() {
36   return y;
37   }
38
39   public void setY(double y) {
40   this.y = y;
41   }
42
43   public MyColor getColor() {
44   return color;
45   }
46
47   public void setColor(MyColor color) {
48   this.color = color;
49   }
50
51   public abstract void draw(GraphicsContext gc);
52
53   @Override
54   public String toString(){
55   return "Class MyShapeis the hierarchy's superclass and inherits the Java c
lass Object. An\n" +
56   "implementation of the class defines a point (x, y) and the color of the s
hape. ";
57   }
58  }
```

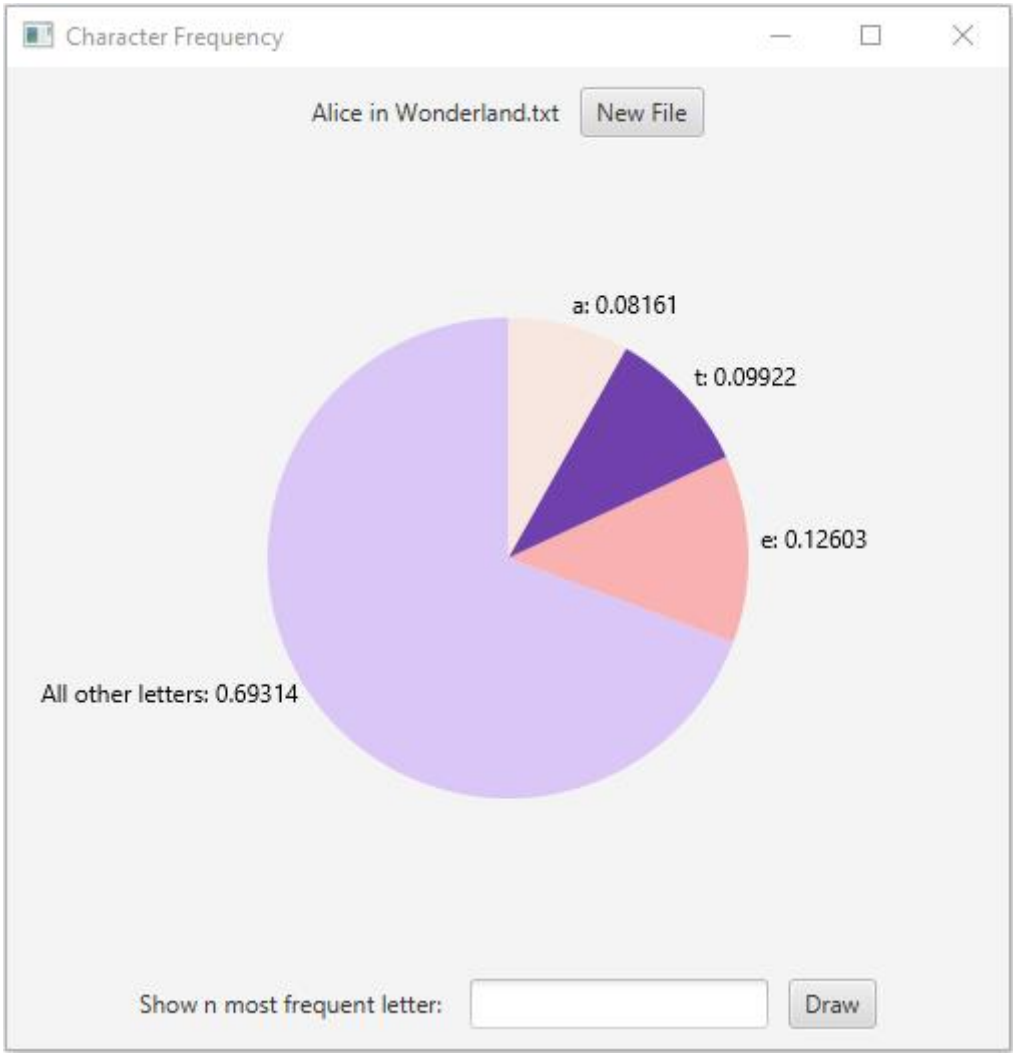
MyColor.java

```
1  package com.demo;
2
3  import javafx.scene.paint.Color;
4
5  public enum MyColor {
6   Red(255, 0, 0),
7   FireBrick(178,34,34),
8   IndianRed(205,92,92),
9   LightCoral(240,128,128),
10   LightPink(255,182,193),
```

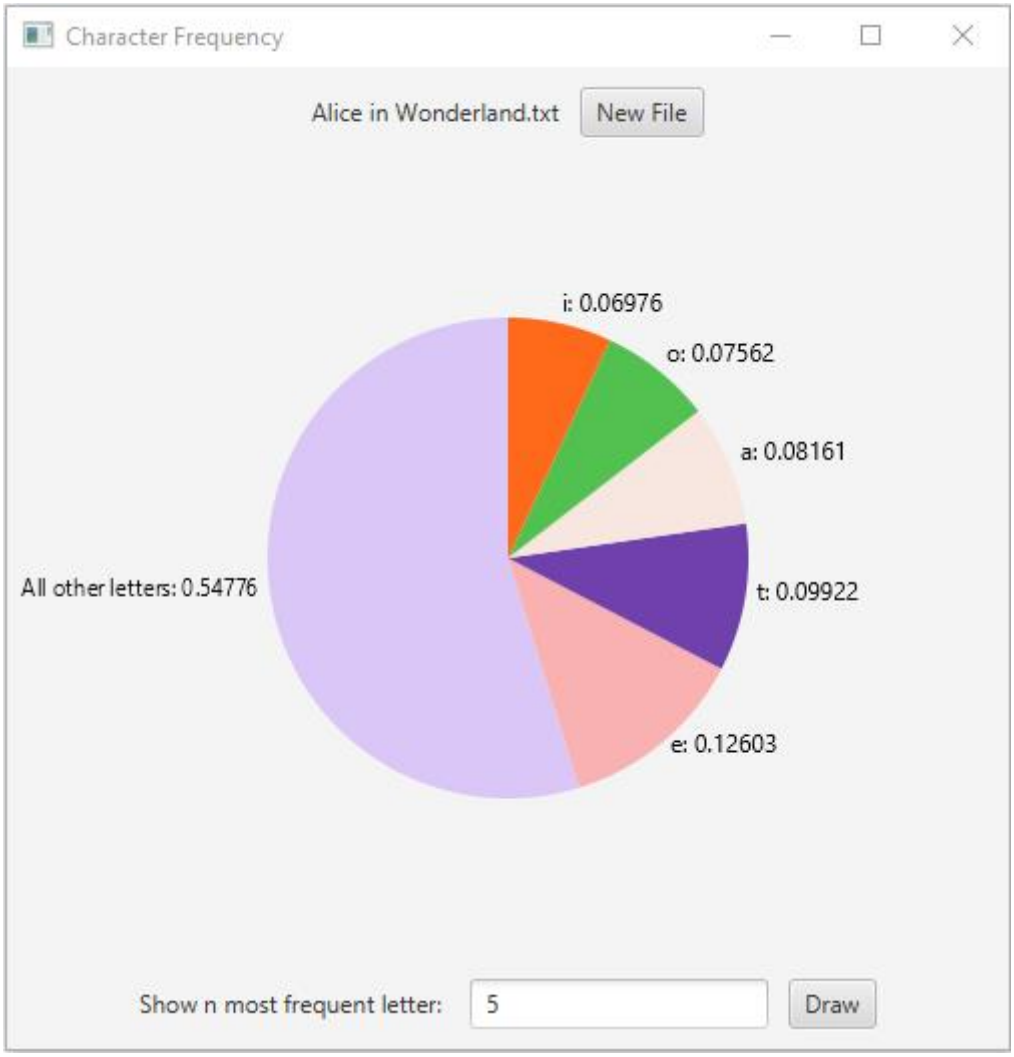
```
Pink(255,192,203),
Green(0,255,0 ),
LightGreen(144,238,144),
PaleGreen(152,251,152),
OliveDrab(85,107,47),
MediumAquamarine(0,250,154),
Turquoise(64,224,208),
Blue(0,0,255),
DarkBlue(0,0,139),
RoyalBlue(65,105,225),
SkyBlue(135,206,235),
Azure(240,255,255),
White(255,255,255),
Black(0,0,0),
Gray(128,128,128),
LightGray(211,211,211),
Yellow( 255,255,0);


private int r, g, b;

private MyColor(){
this(0, 0,0);
}

private MyColor(int r, int g, int b){
setColor(r, g, b);
}

public Color toFXPaintColor(){
return Color.rgb(r, g, b);
}

public void setColor(int hex){
this.r = (hex & 0xFF0000) >> 16;
this.g = (hex & 0xFF00) >> 8;
this.b = hex & 0xFF;
}
```

```
49
50  public void setColor(int r, int g, int b){
51  this.r = r;
52  this.g = g;
53  this.b = b;
54  }
55
56  public int getHexColor(){
57  return ((0xFF0000 & (r << 16)) | (0x00FF00 & (g << 8)) |b);
58  }
59  public MyColor getColor(){
60  return this;
61  }
62
63  public static Color randomColor(){
64  int r = (int)(Math.random() * 256);
65  int g = (int)(Math.random() * 256);
66  int b = (int)(Math.random() * 256);
67
68  return Color.rgb(r, g, b);
69  }
70 }
```

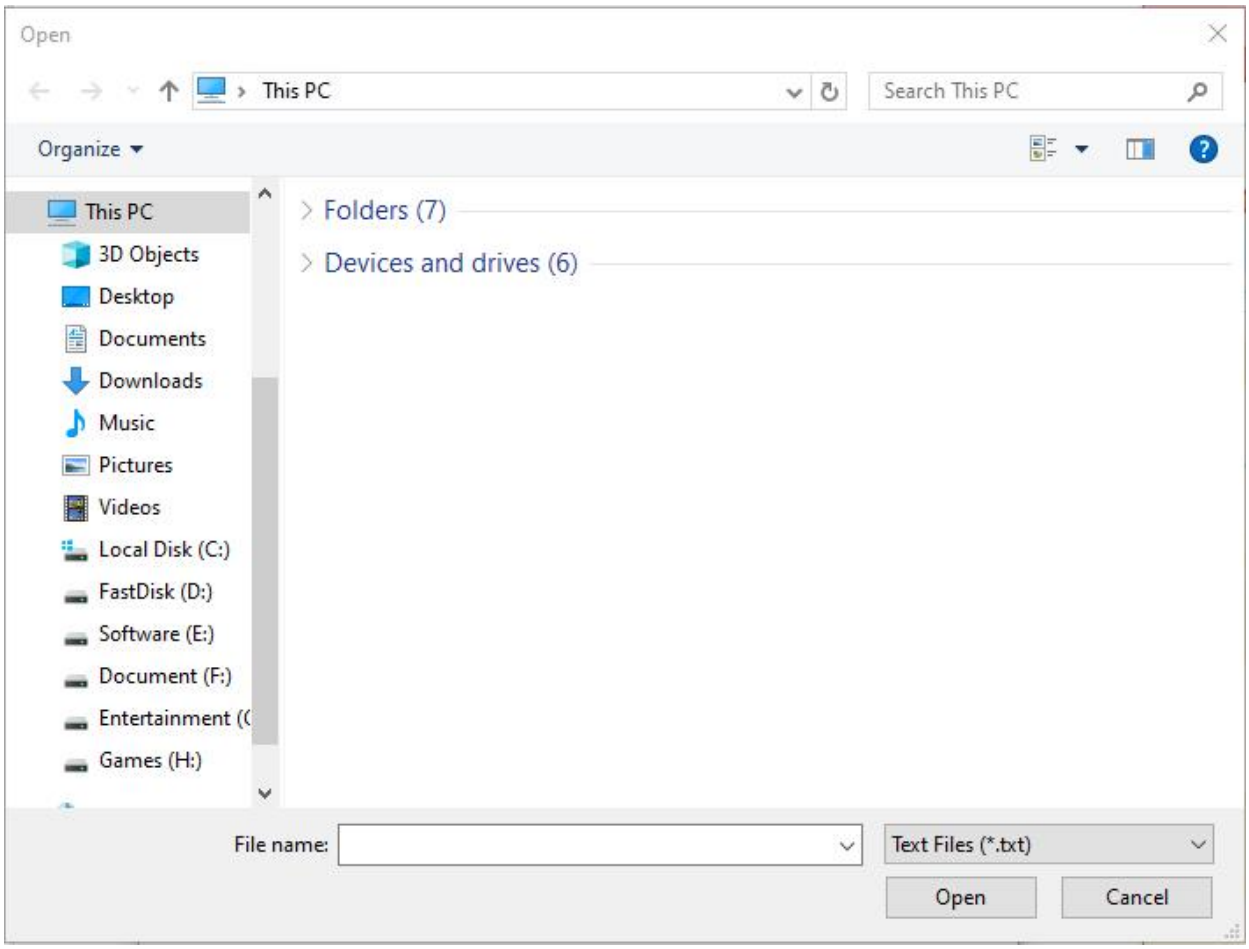
4. Outputs



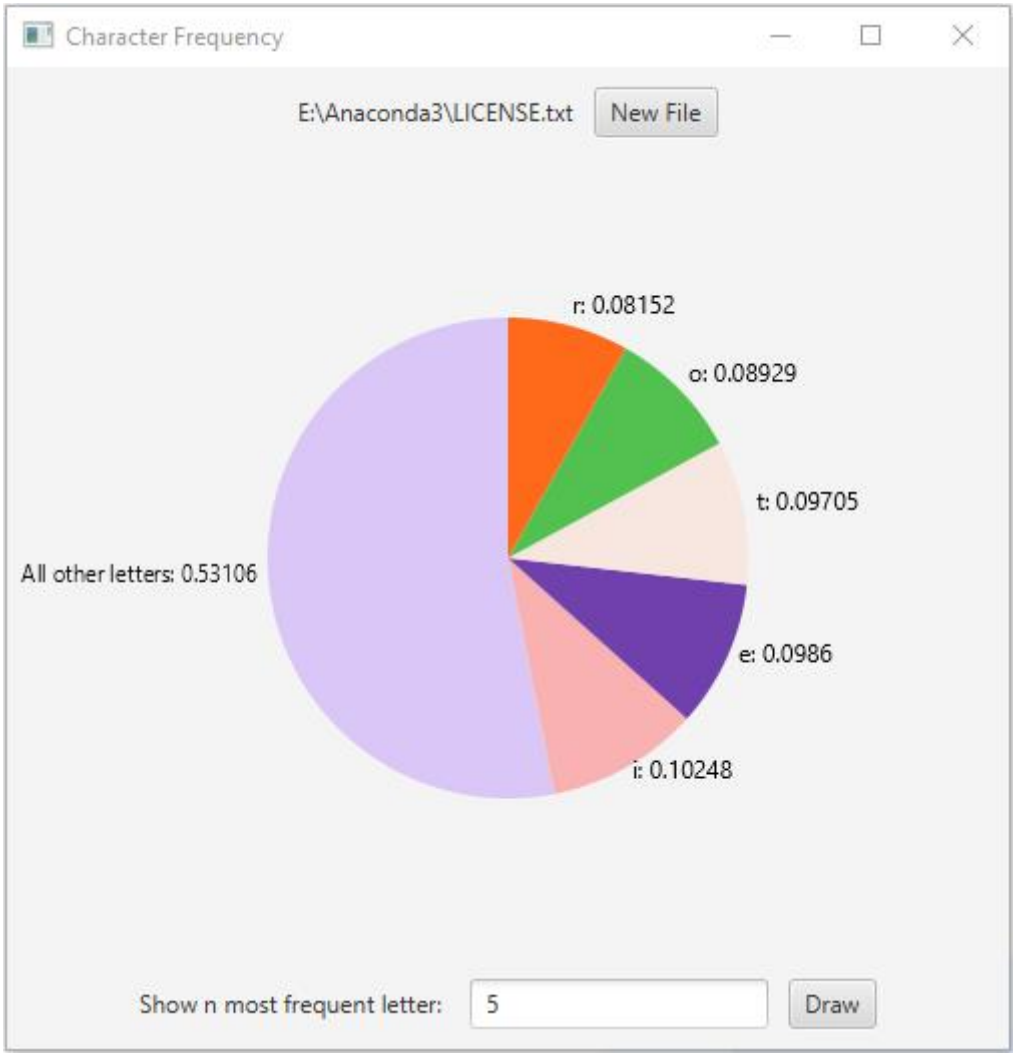
The application will read the "Alice in Wonderland.txt" and shows the top 3 characters having the most frequences by default.



By typing in 5 and click the "Draw" button, the pie chart will update the graph.



By clicking the "New File" on the top, a file chooser will pop up and user can choose an .txt file to analyze.



The full path of the new selected file will be shown at the top, left side of the New File" button.