Nathaniel J. Sands
CSc 22100-P
CCNY Spring 2020
Prof. Auda
4/16/2020

Exercise 2

The goal of this exercise is to develop a program that will read a text file, analyze the frequency with which each letter occurs in the text, and then graphically display the results in a colored pie chart with labels. To this end, we develop a **Text** class to import and manipulate text files, a **HistogramAlphaBet** class that generates a frequency map, and a **MyPieChart** class that translates the frequency map to a graphic.

Classes and Methods

The **Text** class constructor takes a string parameter which gives the path of the text file to be read into memory. The text is read using the *Files.readString* method and stored in a String instance variable *contents*. All non-alphabetical characters are removed, and the remaining alphabetical characters are sent to lower case. This modified string of characters is stored in *charString*. The variable *numChars* records the number of characters in this string. Accessor methods allow *charString* and *numChars* to be output.

An **Event** class is implemented to handle statistical information. It stores an event *name* and its *probability*. The getProb method returns the probability of the event, and the *getLabel* methods generate a label for use with the final graphic.

The bulk of the text processing is done by the **HistorgramAlphaBet** class. The constructor takes a **Text** object parameter. It reads in the character string, and creates a **HashMap** object which stores the number of times each letter in the alphabet occurs within the text. This map is sorted according to the frequency count of each letter (in descending order). An array of **Event** objects *events* is generated, each of which records a letter and its probability in the text (# of occurences / total # of characters). The reason the information is transferred from a map to an array is to allow for the information to be accessed by index, rather than by key.

The **MyPieChart** class is translates the frequency map information into a pie chart drawn with JavaFX methods. The constructor takes the x- and y-coordinates of the chart's center on the drawing canvas, the desired radius of the chart, and an array of **Event** objects. The object contains a method *drawNMostFreq* wich creates a pie chart showing the probabilities of the *n* most frequent letters in the text. It iterates *n* times through the array of events, and uses the probability of each letter to determine the angle of its corresponding arc (theta = probability * 360 degrees). A filled arc is drawn using a color randomly selected from the Enum class **MyColor**, and a corresponding text label is added to the graphic using the getLabel and

strokeText methods. The pie chart is completed by drawing a single arc representing the combined frequency of the remaining 26 – *n* letters.

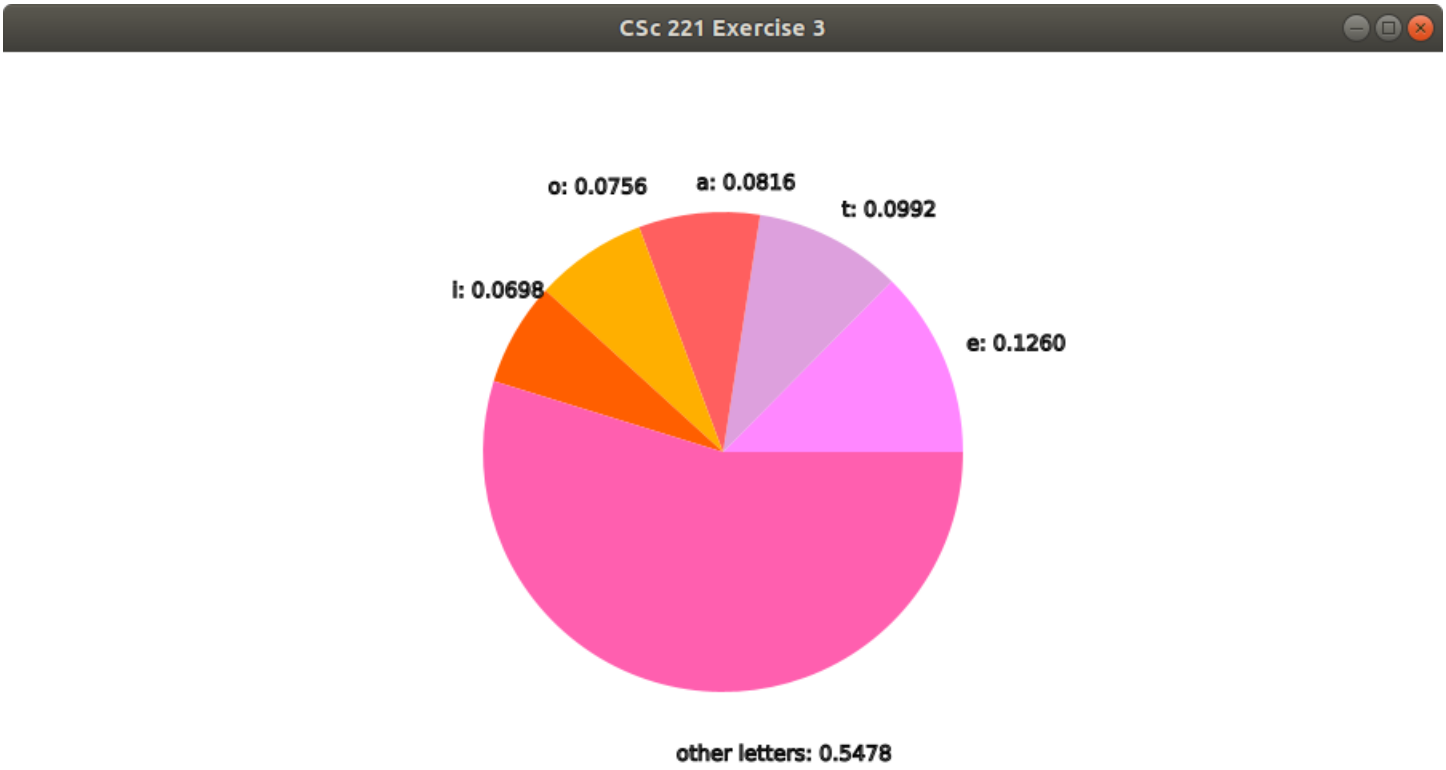
A Globals class is used to store constants pertaining to the canvas size, and center points.

The following packages were imported:

```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.shape.ArcType;
import javafx.stage.Stage;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import java.util.Random;
import java.lang.Math;
import java.nio.file.Files;
import java.nio.file.InvalidPathException;
import java.nio.file.Paths;
import java.nio.file.Path;
import java.io.IOException;
import java.util.Map;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Set;
import java.util.stream.Collectors;
```

Results

The text of “Alice in Wonderland” was imported and a pie chart showing the frequency of the 5 most frequent letters was drawn. See attached image and code.


CSc 221 Exercise 3
o: 0.0756
a: 0.0816
t: 0.0992
l: 0.0698
e: 0.1260
other letters: 0.5478

```
/*
Nathaniel J. Sands
CSc 22100
CCNY Spring 2020
Exercise 3
04/16/20
*/

package sample;

import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.shape.ArcType;
import javafx.stage.Stage;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import java.util.Random;
import java.lang.Math;
import java.nio.file.Files;
import java.nio.file.InvalidPathException;
import java.nio.file.Paths;
import java.nio.file.Path;
import java.io.IOException;
import java.util.Map;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Set;
import java.util.stream.Collectors;

/* GLOBAL VARIABLES */
class Globals {
    final static double MAX_X = 900.0;
    final static double MAX_Y = 500.0;
    final static double CENTER_X = MAX_X / 2;
    final static double CENTER_Y = MAX_Y / 2;
}

/* COLORS */
enum MyColor {
    PLUM(221,160,221), MAGENTA(255,0,255),
    MEDIUMTURQUOISE(72,209, 204), ORANGE(255,165,0),
    RED1 ( 255,0,0),DEEPPINK2 ( 255,0,95),DEEPPINK1 ( 255,0,135),
    MAGENTA2 ( 255,0,215),
    MAGENTA1 ( 255,0,255),ORANGERED1 ( 255,95,0),
    INDIANRED1 ( 255,95,95),
    HOTPINK ( 255,95,175),
    MEDIUMORCHID1 ( 255,95,255),DARKORANGE ( 255,135,0),
    SALMON1 ( 255,135,95),LIGHTCORAL ( 255,135,135),
    PALEVIOLETRED1 ( 255,135,175),ORCHID2 ( 255,135,215),
```

```
    ORCHID1 ( 255,135,255),ORANGE1 ( 255,175,0),
    SANDYBROWN ( 255,175,95),LIGHTSALMON1 ( 255,175,135),
    LIGHTPINK1 ( 255,175,175),PINK1 ( 255,175,215),
    PLUM1 ( 255,175,255),GOLD1 ( 255,215,0),
    LIGHTGOLDENROD2 ( 255,215,95),NAVAJOWHITE1 ( 255,215,175),
    MISTYROSE1 ( 255,215,215),THISTLE1 ( 255,215,255),
    YELLOW1 ( 255,255,0),LIGHTGOLDENROD1 ( 255,255,95),
    KHAKI1 ( 255,255,135),WHEAT1 ( 255,255,175),
    CORNSILK1 ( 255,255,215),GREY100 ( 255,255,255);

    private int red, green, blue;
    private static final MyColor[] VALUES = values();
    private static final int SIZE = VALUES.length;
    private static final Random RANDOM = new Random();

    public static MyColor randomColor() {
        return VALUES[RANDOM.nextInt(SIZE)];
    }
    MyColor(int red, int green, int blue) {
        this.red = red;
        this.green = green;
        this.blue = blue;
    }
    public Color getRGB() {return Color.rgb(red, green, blue);}

}
/*TEXT*/
class Text {
    private String contents;
    private String charString;
    private Integer numChars;
    private Path path;

    Text(String p) {
        try {
            path = Paths.get(p);
        } catch (InvalidPathException exc) {
            System.out.println("Bad path.");
            System.exit(-1);
        }
        try {
            contents = Files.readString(path);
            charString = contents.replaceAll("[^a-zA-Z]", "").toLowerCase();
            numChars = charString.length();
        } catch (IOException exc) {
            System.out.println("I/O error.");
            System.exit(-1);
        }
    }
    void print() {
        if (contents != "")
```

```
            System.out.print(contents);
        else
            System.out.println("File empty.");
        return;
    }
    int numChars() { return numChars; }
    String charString() { return charString; }
}
/*EVENT*/
class Event {
    private String name;
    private double probability;
    Event(String n, double prob) {
        name = n;
        probability = prob;
    }
    String getLabel() {
        return String.format(name + ": %.4f", probability);
    }
    double getProb() {
        return probability;
    }
}
/*HISTOGRAMALPHABET*/
class HistogramAlphaBet {
    public Map<Character, Integer> m = new HashMap<Character,Integer>();
    public Map<Character, Integer> sortedMap;
    public Character[] keyArray;
    private int keyArraySize;
    private double numChars;
    private Event [] events;

    HistogramAlphaBet(Text text) {
        String s = text.charString();
        for (int i=0; i < s.length(); i++) {
            char ch = s.charAt(i);
            m.putIfAbsent(ch,0);
            m.put(ch, m.get(ch)+1);
        }
        numChars = text.numChars();
        sortedMap = m
                .entrySet()
                .stream()
                .sorted(Map.Entry.comparingByValue())
                .collect(Collectors.toMap(Map.Entry::getKey,
                 Map.Entry::getValue, (e1, e2)->e2, LinkedHashMap::new));
        Set<Character> keySet = sortedMap.keySet();
        keyArraySize = keySet.size();
        keyArray = keySet.toArray(new Character[keySet.size()]);
        int i = 0;
        int j = keyArraySize - 1;
```

```
        while (j > i) {
            Character temp = keyArray[j];
            keyArray[j] = keyArray[i];
            keyArray[i] = temp;
            i++; j--;
        }
        events = new Event[keyArraySize];
        for (int k=0; k < keyArraySize; k++) {
            events[k] = new Event(Character.toString(keyArray[k]),
                    m.get(keyArray[k])/numChars);
        }
    }
    int freq(char ch) { return m.get(ch); }
    char nthMostFreqChar(int n)  { return keyArray[n-1]; }
    Event [] getEvents() { return events; }

}

/*MYPIECHART*/
class MyPieChart {
    private double x;
    private double y;
    private double r;
    private Event [] frequencies;

    MyPieChart(double x, double y, double r,
               Event [] freq) {
        this.x = x;
        this.y = y;
        this.r = r;
        frequencies = freq;
    }
    public void drawNMostFreq(int n, GraphicsContext gc) {
        double theta = 0;
        double dTheta = 0;
        double probSoFar = 0;
        double textRadius=1.1*r;
        for (int i=0; i < n; i++) {
            if (theta >= 100 && theta <= 215)
                textRadius = 1.3*r;
            else textRadius = 1.1*r;
            probSoFar += frequencies[i].getProb();
            dTheta = frequencies[i].getProb() * 360;

            gc.setFill(MyColor.randomColor().getRGB());
            gc.fillArc(x-r,y-r,
                    2*r, 2*r,theta,dTheta, ArcType.ROUND);
            double textX = textRadius * Math.cos((theta+.5*dTheta)*Math.PI/
             180);
            double textY = (-1) * textRadius * Math.sin((theta+.
             5*dTheta)*Math.PI/180);
```

```
            gc.strokeText(frequencies[i].getLabel(),x+textX, y+textY);
            theta += dTheta;
        }
        dTheta = (1-probSoFar)*360;
        gc.setFill(MyColor.randomColor().getRGB());
        gc.fillArc(x-r,y-r,
                2*r, 2*r,theta,dTheta, ArcType.ROUND);
        double textX = textRadius * Math.cos((theta+.5*dTheta)*Math.PI/180);
        double textY = (-1) * textRadius * Math.sin((theta+.5*dTheta)*Math.PI/
         180);
        String s = String.format("other letters: %.4f", 1-probSoFar);
        gc.strokeText(s,x+textX, y+textY);
    }
}


public class Main extends Application {

    @Override
    public void start(Stage stage) {

        stage.setTitle("CSc 221 Exercise 3");
        Group root = new Group();
        Canvas canvas = new Canvas(Globals.MAX_X, Globals.MAX_Y);
        GraphicsContext gc = canvas.getGraphicsContext2D();

        Text alice = new Text("alice.txt");
        HistogramAlphaBet freqMap = new HistogramAlphaBet(alice);
        MyPieChart chart = new MyPieChart(Globals.CENTER_X,
                Globals.CENTER_Y, 150, freqMap.getEvents());
        chart.drawNMostFreq(5,gc);

        root.getChildren().add(canvas);
        stage.setScene(new Scene(root));
        stage.show();

    }


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