Chengliang Tan

Exercise 3 report

In Main.java, I implemented the GUI for selecting n and generating the pie chart. I created two scenes, scene1 and scene2. In scene1, there is a choice box that contains 26 options, since we have 26 different alphabets, and there is also a button called “button1” that jumps to scene2 after clicking. Both button1 and choicebox is added to layout1 using the layout1.getChildren.addAll() method. layout1 is a VBox type layout, which display all its input vertically. In scene2, there are 2 buttons, button2 and button3. Button2 will get the n value we selected in the choice box using the getChoice() method, inside the handle() method of button2, I called the main function of the class HistogramAlphaBet to count character frequencies from Emma.txt, then, the handle() method will create a pieChart object using the float array arcExtent[] and draw the object using a GraphicalContext variable gc. In the handle() method I also used fillText() to create a legend for the pie chart. I used a for loop to create the legend for the n selected characters, then create a legend for the sum all unselected characters. The formula I used to calculate the position of the legend is x = pie1.getX() + pie1.getW() * Math.*cos*(Math.*toRadians*(((pie1.getArcExtent(i) / 2) + pie1.getStartAngle(i))))/2
And y = pie1.getY() - pie1.getH() * Math.*sin*(Math.*toRadians*(((pie1.getArcExtent(i) / 2) +pie1.getStartAngle(i)))) / 2, where pie1 is the pieChart object we created eailier in the program. The other button is button3 which will take us back to scene1. I have written a getChoice() method to get the value from the choice box. I also create a canvas object of width 400 and

height 400 to display the pieChart object. Then I added button2, button3, and canvas to layout2 which also displays all inputs vertically.

In pieChart.java, I built the pieChart class that has private variables int n, double x, y, w, h, float[] startAngle, float[] arcExtent, and ArcType closure. The constructor for the pieChart class takes in the variables int n, double x, double y, double w, double h, and float[] inArcExtent as input. The variable n represents how many segments the pieChart will have, x and y is the center coordinate of the pieChart, w and h is the width and height of the pieChart, and the arcExtent is an array containing the probabilities of each character. I created a setStartAngle() method where I created a for loop where i goes from 0 to n and set startAngle[i] = startAngle[i-1] + arcExtent[i-1]. The getSum() function is for getting the total frequencies of the 26-n characters. For the draw method, I first wrote a randomColor() method that randomizes the rgb values and generates a random color every time its called. The gc.setfill(randomColor()) method will make sure that every time I create an arc the color of the arc will be different. I used the fillArc() method and to draw the first segment of the peichart, a for loop to create n-1 segments for the pieChart, and then another fillArc() method to draw the last segment which represents the 26-n character's frequency. I used the variable int n, double x, double y, double w, double h, the arcExtent float array and the startAngle float array as the parameter of the fillArc() method.

For the HistogramAlphabet class, I used a BufferReader to read the txt file Emma.txt. Create a totalChar variable that keeps track of the total count of all characters in the text file. The count integer array is a frequency table use to keep track of the individual count of each character in the text file, where each index represents a character and the value stored under that index represents that character's count. The porb array is for keeping track of the probability of

each character. To read the text file, I used a while loop and the while loop would not stop if the int nextLetter = reader.read != -1. The char variable current is used to store the current letter that the BufferedReader reads. Then there is an if statement that checks the current character is actually a character from ‘a’ to ‘z’. If yes, then count[current - 97]++ will increase the count of that character by 1, and totalChar++ will increase the total number of characters by 1.

After filling out the count array, I created a hashmap and put the characters and the corresponding count into the map using my makeMap() method, this method will take the count array as input and create a map with character from ‘a’ to ‘z’ as its keys and their occurrences as values. This map will be sorted by their keys in ascending order. To sort the map by values, I used the method sortByComparator() which takes a map and a boolean value as input. The input map will be the map you want to sort by values, and if the boolean input is true, then the method will sort the map by ascending value order, if false, the method will sort the map by descending value order. The method makeProbMap() also takes the count array as input, but this map will store the character as the key and the probabilities of each character as the values.