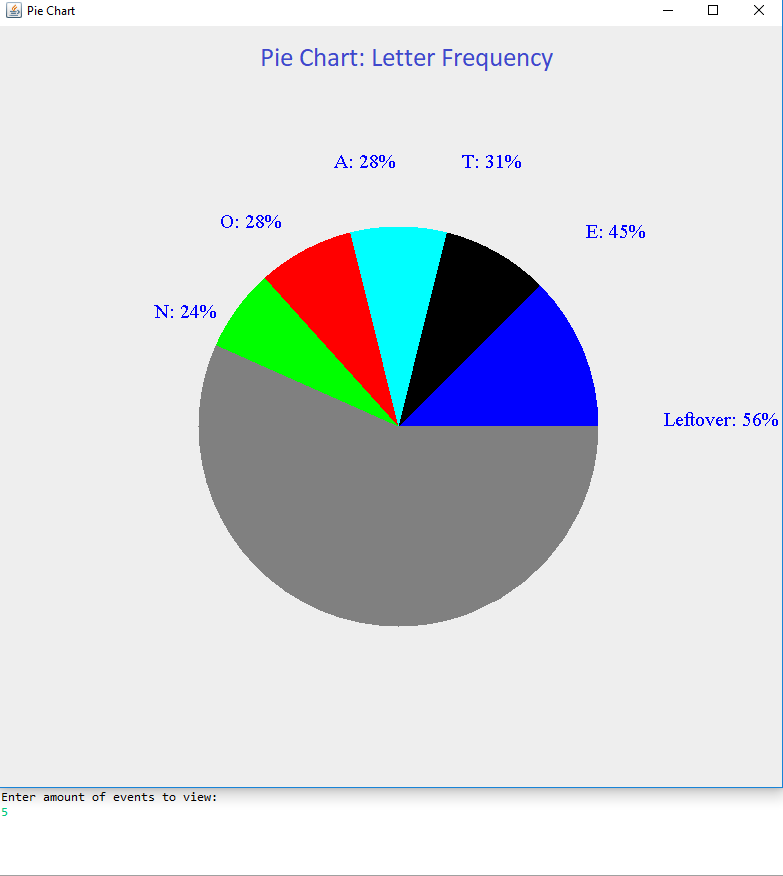
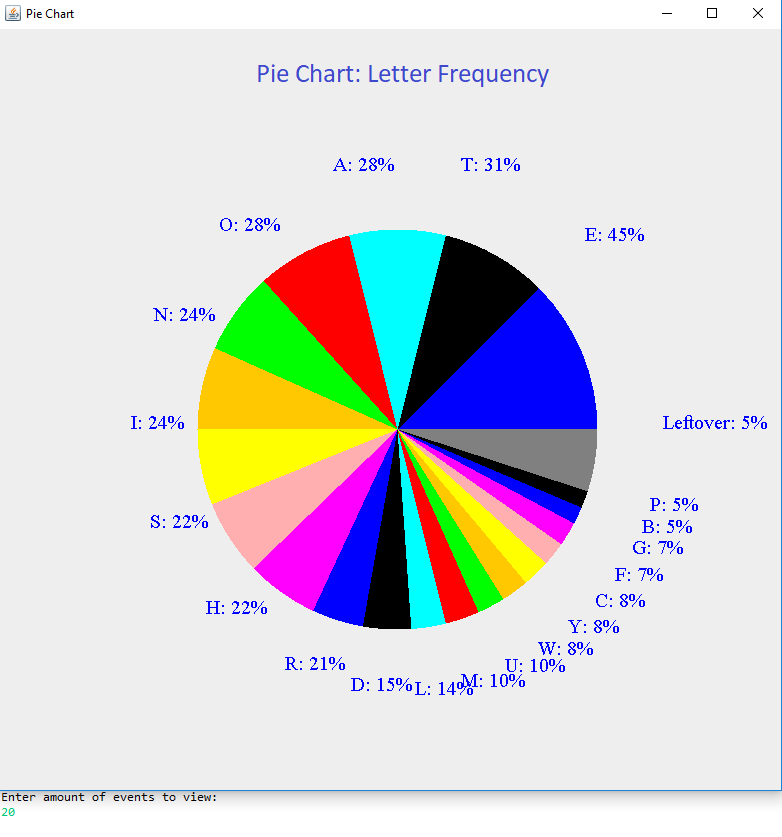
Comer, Nicholas G.

Assignment 3: Report



For this assignment, I was tasked with creating a pie chart using Java that would be able to suit the needs of the user. This meant that I would take in some unknown event and make the pie chart based around the events that were inputted. The pie chart would show the probabilities of the n most frequent occurrences of an event specified in step 3 of the exercise. The probability was defined as the frequency of the single event divided by the frequency of all events. In the pie chart, the area of each segment was to be proportional to the probability of the corresponding event. Each segment had to have a different color. The last segment was to represent all other events and their cumulative probability. The pie chart would have to include appropriate constructors and a method draw that would draw the actual pie chart. The user would need to be able to input the amount of events to look at, and the probability would be displayed by the pie chart slice it corresponds to. Finally, I would need to implement a Java class HistogramLetters that calculates the n most frequent letters in a given text file “Emma.txt”, and find the probabilities of each letter. This class would use the pie chart class.

The first step was to make the pie chart. Originally, I planned on using JavaFX. However, it proved unruly in terms of being able to take input from other classes. Basically, it would work better if I only used one class for both the input and the drawing. Therefore, I turned to the method I had been using for the past few assignments, which was tried and true. This method was using the Java 2DGraphics context. This would also allow me to use a draw method, which is something that the JavaFX pie chart did not require. The first step of creating the Pie Chart class was to look at the different variables that I would need to keep track of. The things I decided I needed the computer to remember were a map of the events and the amount of times they occurred, the total amount of events occurring, the width and height of the canvas, the actual amount of events to be viewed. These were all made into private/protected variables. The constructor handled taking the input and putting them into the instance variables. Then, I made the function for determining the probability of any given event. Basically, all I did was divide the frequency I found by inputting the given key by the total frequency of all events. Next, I made a helper method called get most frequent. This method would be used when I draw the pie graph. Originally, I tried to use an iterator to go through the map, but that proved to be quite difficult, so I went with just using a for each loop. Besides that, the method is pretty standard. I kept track of what the maximum entry was by comparing the value to each other. Finally, I returned the key to the maximum value entry in the map. I had another helper method called remove most frequent. All this method did is exactly what its title implied. I removed the most frequent entry from the map, using the get most frequent method. Finally, I arrived at the draw method. First I created a variable called current angle in order to keep track of where on the pie chart I was while drawing. Then I loop through the map for as many times as the user told me with the nMostFrequent variable. With every iteration the color of the slice of the pie chart changes. Originally, I brute forced it by using a ton of if statements, but I figured out that I could just use modulus mathematics in order to get the different colors to show up on a rotation. After setting the color, I filled the arc starting from the center of the canvas and reaching a length of half the canvas. The starting angle of the arc would be the current angle variable I defined earlier. The central angle of the arc would be the probability of the most frequent event multiplied by 360. Then I incremented the current angle so that the next slice would appear in the correct position. The last thing I did in the loop was to create the string that tells the user what the event of the slice was and its probability. Outside of the loop, I drew the slice of the pie chart that represented the remaining events, and added the corresponding label.

Next came the histogram letters class. This one was tough to figure out because I was so used to just using the paint method outside of the main method. However, I could not do that this time because the main method is where the file “Emma.txt” was read, which meant that I needed to look inside the main method to determine what the events for the pie chart would be. At first, I attempted to pass variables outside of the main method. This didn’t work. Then I tried to read the file without using the main method. This is possible, but not with my meager grasp of the programming. Finally, I attempted to use the paint method inside the main method. This might be possible, but I do not know how to make a graphics object of my own. I ended up deciding to just rely on JFrame, which meant I could call the pie chart’s draw method inside the main method. Now that I figured out how I was going to draw, let’s look at how I went about reading the text file. I decided to use a buffered reader, as it seemed like the simplest way to go. I created the hash map that would come to represent the letters and their occurrences. Then I made a while loop, that basically went as long as the char being read was not equal to negative one. If the char being read was a lower case letter, I changed it to its upper-case equivalent using mathematics on ASCII. Then, if the hash map already contained the letter as a key, I incremented the value corresponding to the letter by one. Otherwise, I created the new entry in the hash map. Either way, I incremented a variable related to the total amount of characters in the text. After the reading of the file was done, and I had variables for both the hash map of the frequencies of letters and the total amount of characters, I created a new instance of a pie chart object. Using the JPanel method paint component, I was able to call the pie chart object’s draw method. From there things worked out. The last part of the entire project that I attended to was making sure that the user could input the amount of events they wanted to view. To that end I implemented a scanner that runs before the pie chart is drawn.