Nick Amundsen

Devon McDonald

As far as we know, the program appears to compile and run the java applet fairly well. It might actually be that there could be a better way to manage the resources associated with JPanel and JFrame but we are not yet aware of how to handle that. In terms of a simple program designed to display a graph and work out a function, I would say it works fairly well. However inefficient it might be, it still handles the tasks set out for it very quickly and that counts for something. The only downside I could see to having the graph displayed like that is if there is too many points to be displayed it could get a little cramped. Unless the window size got larger then there would be no way to remedy that problem. No matter the size of the input list the calculating routes methods should work just fine however.

The most challenging part of the program was having to take knowledge we had known with the DrawingPanel class used in other classes we took and using that for the JPanel class. Trying to understand how the event driven system works and using that with the paint() and repaint() methods as well as actionPerformed(). Creating the MyGraph() object initiates the methods from the superclass, and then once we call the displayGraph() the JFrame application is started and the JPanel is added to it, calling the Paint() method automatically. Overriding the paint method and then calling displayVertices() and displayEdges() allowed us to properly map out each vertex and edge using class constants for the collections of edge objects and vertex objects. The vertex objects store the x and y position for graphing purposes only, and then obviously they also store the name. The real information is stored in each edge object, such as distance, time, and cost. Implementing all of these collections in Dijktra’s algorithm was tough but not the hardest part of the entire program.

To test the graphics part of the program we used an expanded version of the sample test file and made more routes between places. It took many trial and error tests as we went along to get the cosine and sine functions to work properly to display the nodes evenly in a circle. Then getting the edges to appear so you can actually tell which direction flights were traveling was also challenging. If we had more time we could have made the lines be drawn as arrows instead but for simplicity we ran a midpoint formula and drew a green line for the direction the flight is going. Testing the algorithm for finding the best path just involved calculating the path by hands and then comparing our answers to the answer from our function. All pretty straightforward testing, we did not attempt any huge numbers of nodes but it would probably handle it fine.