Class roles and brief description

Edge

Implements EdgeIn. It stores the nodes and its label to be used by the multigraph. We have getter and setter methods within this class for the nodes A and B, as well as having a getter method for the label.

EdgeIn

Edge interface which represents the edge between two connected nodes on a specific label(line). Contains two methods which are two setters. The first, sets the edge label and the second sets the nodes on each side of the edge. Thus, taking the ID's of NodeA and NodeB to create the link.

Role: Because the Multigraph interface makes use of nodes and edges it assumes these classes exists therefore we need to have interfaces for them.

Node

Implements NodeIn. The two methods are implemented from the NodeIn interface. There are another two methods as well as the constructor. The two methods get the ID and Name, the constructor simply takes in the NodeID and nodeName and sets them to their according private string variables.

NodeIn

Node interface which represents a node on the multigraph. This interface contains two setter methods, the first takes in the NodeID, the second takes the String nodeName.

Role: Because the Multigraph interface makes use of nodes and edges it assumes these classes exists therefore we need to have interfaces for them.

Multigraph

Implements MultigraphADT. It holds the edges and nodes in an ArrayList which is filled with the setters given by the implemented MultigraphADT interface.  Our initial constructor sets the nodeList and the edgeList.

This class is also responsible for finding the shortest route between two nodes and returning it as an ArrayList to the menu to output. It uses two to do so, searchShortestPath does a breadth first search storing all possible routes in arraylists and will the shortest path found.

MultigraphADT

The multigraph interface which represents the graph and holds all of the nodes and edges, it is implemented by the Multigraph class. There is a method for adding nodes to the graph and same with edges.

Menu

Menu is responsible for handling getting user input and displaying the result of a search, it loops until the user types the exit command.

The methods include getInput() which handles getting the user input, displayOutput which handles the arraylist given by the output of the search. It also contains a getValidStation method which checks if the input given by the user is correct. Additionally, there is a clarifyMultipleInput method which is used to handle the case where two stations have tha same name.

The last method is the main which runs the loop which deals with user queries and creates an instance of MetroMapParser to create the multigraph from the file.

MetroMapParser

MapParser parses and tokenises data from the input file and generating the multigraph from it. It contains a constructor which requires the name of a file as input to read from. generateGraphFromFile() creates a multigraph then parses a .txt file line by line and tokenises each line to fill the multigraph with nodes and edges.

BadFileException

MapParser throws this in a huff if the file given to MetroMapParser is formatted incorrectly.

IO Exception

Is thrown by MetroMapParser if there is an issue with trying to read from the file such as reading from a networked file and are disconnected from it.