Imaduddin Sheikh

CTANetwork – Project Design (CS201)

1.

The front-end user interface of the application will consist of a menu, which the user will interact using numeric keys, to select different options available on the menu. The main menu will show instructions for what to press into the console. The menu options will include creating a new station in the CTA train network and added it to the specified route, updating the characteristics of an already stored CTA station, search for a station and its information, return multiple stations' names, find nearest stations with the help of geolocation coordinates, with also provide them with the option of storing a path between stations along the routes, and then exit when the user's tasks are completed.

2.

a) I will separately write a 'private' method that will be responsible for storing the data scanned by the Scanner object 'fileScan' from the Scanner class which will read the station data line-by-line in the Main class outside the application method.

b) I will implement a while-loop with the condition consisting of 'fileScan.hasNext()' method after skipping two lines of the input line until all the file has been read., the data will then be parsed line-by-line through a String 'split' method, and then instantiated into individual CTAStation objects by a parseCTAStation method.

c) Each CTAStation instance will be stored in an ArrayList holding CTAStation, and that ArrayList will be instantiated as part of the CTA Line Station routes, e.g. Green Line, Red Line, Blue Line, with their corresponding CTA stations stored. Each CTARoute will have its own ArrayList containing the CTAStation's.

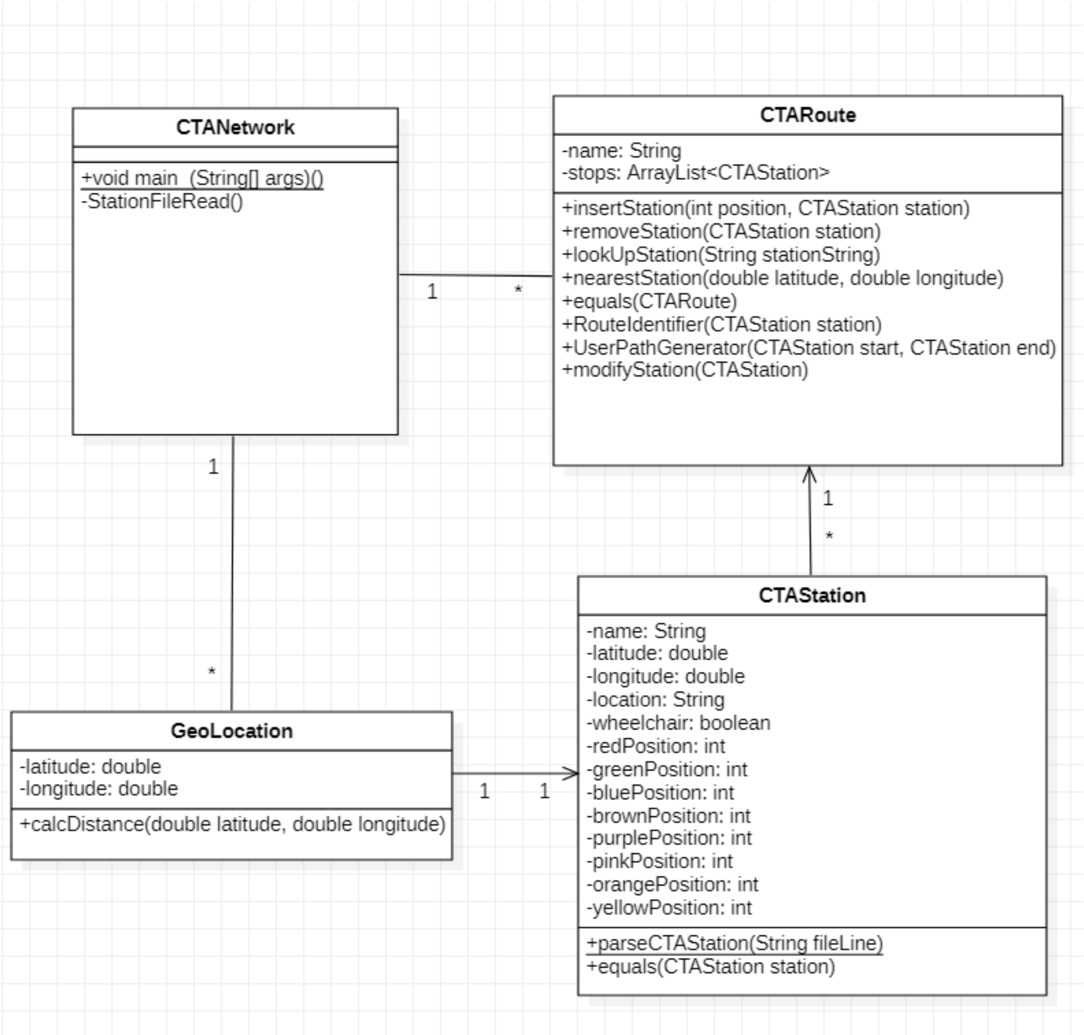
d) The program will call different methods either on the 'mains' method according to the task assigned by the user. Firstly, the program will ask for the route which concerns the user's task. For adding a station, if the program finds out the station is already present, it will output an error. For deleting a station, it will output an error if the concerned station is not present, else delete it. If a station needs to be updated, the program will initially search for the concerned station, and if found, ask the user what to append, and then update information as provided by that user. Each menu option for data updating will be navigated across the menu through the numeric keys.

e) To search for data, the program will first ask for the CTARoute where the concerned data lies or even provide the functionality of searching within all the CTARoute's. Different CTAStation properties will be treated with different methods, e.g. compareTo() method, or even equals() method, depending on the situation. The functionality of the ArrayList, as well as the get(index) methods, will be used frequently during the data search, with the help of for loops and the .size() method.

f)

- CTANetwork class containing the mains application method. This application will run all operations of the CTA application, and will be responsible for calling other methods, and providing a user menu interface.
- CTARoute class, this template will instantiate new routes and will have the methods involved with the data processing and appending of the ArrayLists of CTARoute class that store the CTAStation's.
- CTAStation class will be used to instantiate individual CTA Stations as well as their properties. Its secondary function is to store methods that will be helpful in parsing data read from the input file.
- GeoLocation class will be used for calculations dealing with location.

3.


CTANetwork
+void main (String[] args)()
-StationFileRead()
CTARoute
-name: String
-stops: ArrayList<CTAStation>
+insertStation(int position, CTAStation station)
+removeStation(CTAStation station)
+lookUpStation(String stationString)
+nearestStation(double latitude, double longitude)
+equals(CTARoute)
+RouteIdentifier(CTAStation station)
+UserPathGenerator(CTAStation start, CTAStation end)
+modifyStation(CTAStation)
1
*
1
*
1
*
CTAStation
-name: String
-latitude: double
-longitude: double
-location: String
-wheelchair: boolean
-redPosition: int
-greenPosition: int
-bluePosition: int
-brownPosition: int
-purplePosition: int
-pinkPosition: int
-orangePosition: int
-yellowPosition: int
+parseCTAStation(String fileLine)
+equals(CTAStation station)
GeoLocation
-latitude: double
-longitude: double
+calcDistance(double latitude, double longitude)
1
1

4.

I will start my test plan by testing each, and every method of the individual classes, and work my way up to the code-heavy methods. I will start with the GeoLocation class, then the CTARoute class' methods, CTAStation class, and the CTANetwork class at last. Each method will be tested in a separate Tester mains application class, which will be made for temporary use. Each branch of the menu options will be tested by using arbitrary values such that a certain output is expected which, when verified, will determine the integrity of the methods. The summary of the test plan is to divide the entire program into small blocks of code which will be tested block-by-block so that it is easy to spot the issue. During testing, code blocks will be tested with normal, abnormal and extreme values.