1. **<u>Describe the user interface. What are the menu options and how will the user use the application?</u>**
    a. **Menu Options**:
        i. **1. Add a Station**
            1. Station Name: [String Input]
            2. Latitude: [Double Input]
            3. Longitude: [Double Input]
            4. Type?: [String input]
                a. 1. Elevated
                b. 2. Surface
                c. 3. Embankment
                d. 4. Subway
            5. Wheelchair?: [Boolean [as String] input]
                a. 1.Yes
                b. 2. No
        ii. 2. Modify a Station
            1. Change From:
                a. Station Name: [String Input]
                b. Latitude: [Double Input]
                c. Longitude: [Double Input]
                d. Type?: [String input]
                    i. 1. Elevated
                    ii. 2. Surface
                    iii. 3. Embankment
                    iv. 4. Subway
                e. Wheelchair?: [Boolean [as String] input]
                    i. 1.Yes
                    ii. 2. No
            2. Change To:
                a. Station Name: [String Input]
                b. Latitude: [Double Input]
                c. Longitude: [Double Input]
                d. Type?: [String input]
                    i. 1. Elevated
                    ii. 2. Surface
                    iii. 3. Embankment
                    iv. 4. Subway
                e. Wheelchair?: [Boolean [as String] input]
                    i. 1.Yes
                    ii. 2. No
        iii. 3. Remove a Station
            1. Station Name: [String Input]
            2. Latitude: [Double Input]

3. Longitude: [Double Input]
4. Type?: [String input]
    a. 1. Elevated
    b. 2. Surface
    c. 3. Embankment
    d. 4. Subway
5. Wheelchair?: [Boolean [as String] input]
    a. 1.Yes
    b. 2. No

iv. 4. Search for a Station

1. Search using station name.
    a. Origin: [String Input]
    b. Destination: [String Input]
    c. Do you require wheelchair access?
        i. 1. Yes
        ii. 2. No
2. Search nearby stations using coordinates.
    a. Latitude (Origin): [Double Input]
    b. Longitude (Origin): [Double Input]
    c. Destination: [String Input]
    d. Do you require wheelchair access?
        i. 1. Yes
        ii. 2. No

v. 5. Exit Application

1. Yes
2. No

2. **<u>Describe the programmers' tasks:</u>**
    a. **Describe how you will read the input file.**
        i. I will use the FileReader object to read from the data file.
    b. **Describe how you will process the data from the input file.**
        i. I will utilize the ArrayList object to search the (Array)List (where I have all my data stored as a string array), to match and output the required Stations as per the demand of the user (based on the user input).
    c. **Describe how you will store the data (what objects will you store?)**
        i. I will store the data in an ArrayList, with objects in the form: [Name: String, Description: String, Latitude: Double, Longitude: Double, Wheelchair Access: Boolean, and Lines: int[ ]].
    d. **How will you add/delete/modify data?**
        i. I will create an ArrayList to store the data in the file, as well as use the FileWriter object, in combination with the Scanner class to process the data in the file. I will get the user input (name, description, latitude and longitude, wheelchair requirements, and the destination) to either add,

remove, or modify the particular station. I will create the methods: searchStation(), removeStation(), modifyStation(), and createStation() in my main client class for this purpose.

e. How will you search the data?

i. I will use an ArrayList to store the data in the input file as a number of string arrays. I will then use the FileReader object to read the file ArrayList, and the Scanner class to scan the lines (string arrays) in the ArrayList to sort out the needed station by using the name, description, latitude and longitude, wheelchair requirements, and the destination from user input.

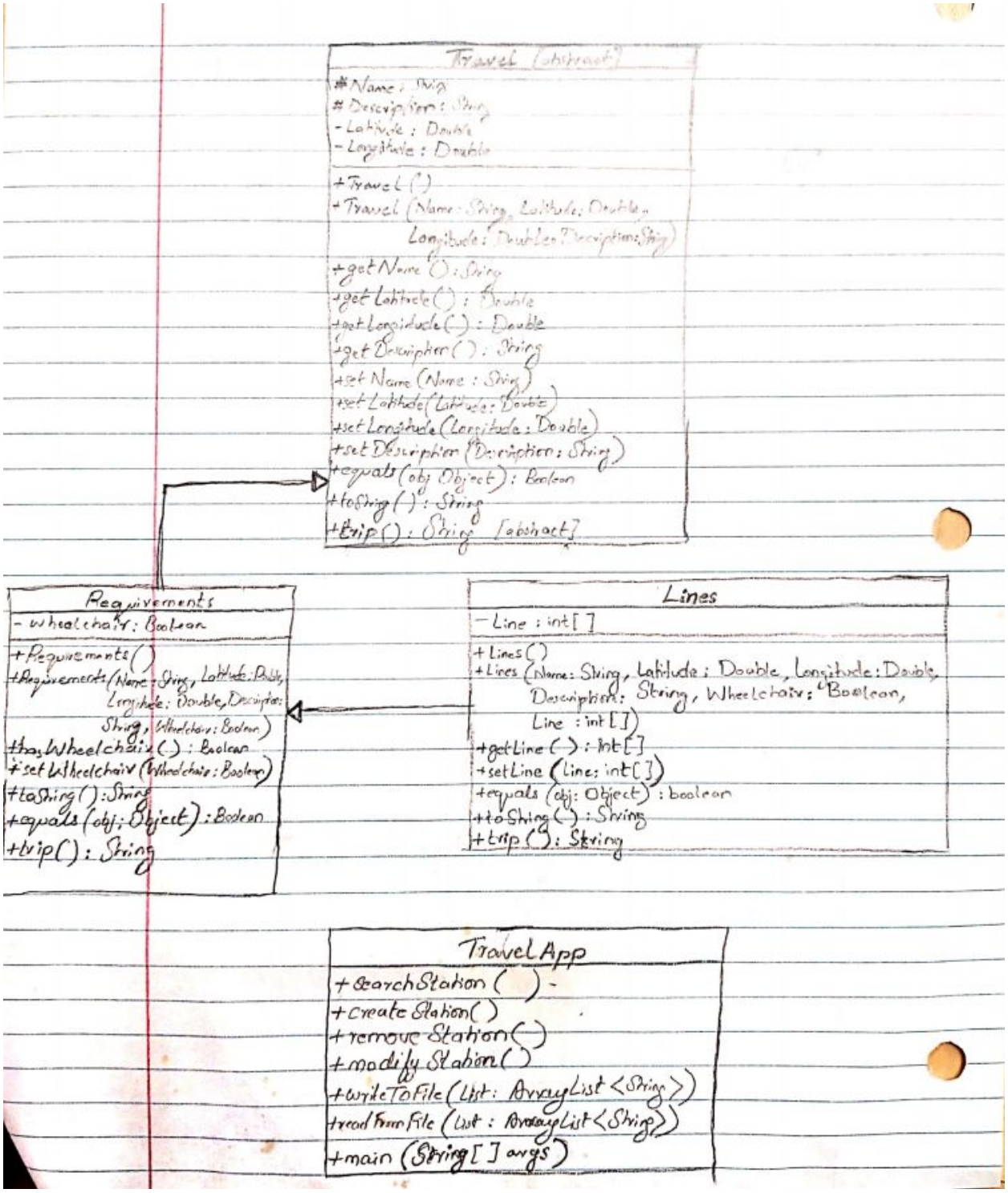
f. List the classes you will need to implement your application.

i. **Travel**: This will be my superclass, and where my program begins. This class will contain the fundamental information and building blocks to support other subclasses. In this class, I will be adding the basic constructors, mutators, equalsTo() method, and toString() methods to be inherited from by the other classes.

ii. **Requirements**: This subclass will extend on the Travel superclass, and add a new “user necessity” factor to the equation. This new factor will be the need of a ‘Wheelchair’, and it will allow the user to narrow down the searches based on whether the station has wheelchair access or not.

iii. **Lines**: The subclass will extend on the Requirements class, and add the additional information about the different ‘lines’ that contain certain stations. This class will determine whether a station lies on a color train line (ex: Red line, Blue line), and give searches and travelling routes based on that information.

iv. **TravelApp**: This will be my main (client) class that will utilize the methods and objects created in the earlier class system, to allow the user to add, remove, modify, and search for a station. Based on the user’s travel plan, this class will output the recommended route to the user.


Travel (abstract)
#Name : String
#Description : String
-Latitude : Double
-Longitude : Double
+Travel ()
+Travel (Name : String, Latitude: Double, Longitude : Double, Description: String)
+getName () : String
+getLatitude () : Double
+getLongitude () : Double
+getDescription () : String
+setName (Name : String)
+setLatitude (Latitude : Double)
+setLongitude (Longitude : Double)
+setDescription (Description : String)
+equals (obj: Object) : Boolean
+toString () : String
+trip () : String [abstract]
Requirements
- wheelchair : Boolean
+Requirements ()
+Requirements (Name : String, Latitude: Double, Longitude : Double, Description: String, Wheelchair: Boolean)
+hasWheelchair () : Boolean
+setWheelchair (Wheelchair : Boolean)
+toString () : String
+equals (obj: Object) : Boolean
+trip () : String
Lines
-Line : int[]
+Lines ()
+Lines (Name: String, Latitude : Double, Longitude: Double, Description: String, Wheelchair: Boolean, Line : int[])
+getLine () : int[]
+setLine (Line: int[])
+equals (obj: Object) : boolean
+toString () : String
+trip () : String
TravelApp
+SearchStation ()
+createStation ()
+removeStation ()
+modifyStation ()
+writeToFile (List: ArrayList<String>)
+readFromFile (List : ArrayList<String>)
+main (String[] args)

Main Menu

User Input	Correct?	Output
Input (Valid)	Yes	(Moves to next precudure)
Input (Invalid)	No	"Invaid command, please try again"
No Input	No/ N/A	[Prompt user again]

Search Station (Name)

_	User Input	Correct?	Output
Origin	Input (String)	Yes	(Moves to next precudure)
	Input (Invalid)	No	"Invaid command, please try again"
	No Input	No/ N/A	[Prompt user again]
Destination	Input (String)	Yes	(Moves to next precudure)
	Input (Invalid)	No	"Invaid command, please try again"
	No Input	No/ N/A	[Prompt user again]
Wheelchair	Input (Boolean [as int])	Yes	(Moves to next precudure)
	Input (Invalid)	No	"Invaid command, please try again"
	No Input	No/ N/A	[Prompt user again]

Search Station (Coordinates)

_	User Input	Correct?	Output
Latitude	Input (Double)	Yes	(Moves to next precudure)
	Input (Invalid)	No	"Invaid command, please try again"
	No Input	No/ N/A	[Prompt user again]
Longitude	Input (Double)	Yes	(Moves to next precudure)
	Input (Invalid)	No	"Invaid command, please try again"
	No Input	No/ N/A	[Prompt user again]
Destination	Input (String)	Yes	(Moves to next precudure)
	Input (Invalid)	No	"Invaid command, please try again"
	No Input	No/ N/A	[Prompt user again]
Wheelchair	Input (Boolean [as int])	Yes	(Moves to next precudure)
	Input (Invalid)	No	"Invaid command, please try again"
	No Input	No/ N/A	[Prompt user again]

*Other commands in other methods are just the above value types rearranged. So, the output will be the same for them as well.