**Project**

Background

For the past few weeks, we have been learning about lists, inheritance, polymorphism, and recursion and using them to model parts of the CTA. For this project, you will use these concepts to design and implement a program that models the entire CTA and allows users to create/remove stations, search for stations, and navigate between stations.

**Note**: You are welcome to discuss the final project with other students, but the work you submit must be your own and may not use code written by others.

The Problem

You must create a program that models the CTA and allows users to interact with the data it contains. How the system is designed is entirely up to you, but it must meet the following requirements.

User Interface Requirements

1. Your program must be able to create a new station that is added to the CTA.
2. Your program must be able to modify an existing station on the CTA.
3. Your program must be able to remove a station from the CTA.
4. Your program must be able to search for a station within the CTA when given a name by the user. This search should be able to return multiple stations if more than one has the same name.
5. Your program must be able to find the nearest station within the CTA when given a latitude and longitude by the user.
6. Your program must be able to generate a path between two stations, specified by the user by their names. This means that your program may need to ask the user to clarify which station they mean when given a name that is shared by multiple stations. Your program should also be able to handle the situation where the stations are not on the same line and be able to indicate to the user where to switch from one line to another.
7. Your program should be able to exit when the user requests to finish.

The user should be continually prompted to choose between the above options and can exit the menu with the appropriate choice. The verification of each operation should be visible in the console. You may of course add additional menu options beyond the 7 specified here.

Programming Requirements

* You should have at least 1 inheritance relationship.
* You should have at least 1 association relationship where the 'outer' class definition has an instance of another user-defined class.
* You should be able to have at least 1 over-ridden method that is **NOT** a usual instance method and **is unique** to your application (i.e. not toString or equals)
* You should be able to write data to the console *and* by write data to an output file when requested by the user. For example, the user may wish to 'save' their transit path.
* You should read and store an input file of data
* Your data should be stored in an encapsulated list
* You should be able to add, delete, modify, and search the data
* Your user interface should be error-proof (Your code should never quit with an error message!)

Design Issues

* The entire project should be one package for ease of use.
* Each class should be in a separate file named with unique names.
* Each class should have all usual instance methods, i.e. mutators, accessors, constructors, and toString.
* Each class should **ONLY** have the necessary data members and the methods that manipulate these data members.
* Each method should have a specific task **NOT** multiple tasks.
* Use the object-oriented language features to make your code more efficient (inheritance, association).
* You should have **MINIMAL** code in your main method and use additional methods in the application class to make the code easier to read.
* You should **NOT** have large blocks of code. In this case, "large blocks" would be anything over 100 lines for a single method.

Implementation Issues

* Use descriptive names for all variables
* Use appropriate programming conventions
* Use methods to break up large pieces of code
* No data handling in main. Your file read should be handled by other methods
* Your file I/O should implement try-catches

Documentation

* Each class should have documentation at the top describing the role of that class in the project, the author (you), and the date.
* All identifiers should be commented
* Each method should have a brief description of its tasks

Phase I: Project Design

04/10/2020

1. Describe the user interface. What are the menu options and how will the user use the application?
2. Describe the programmers' tasks:
   * Describe how you will read the input file.
   * Describe how you will process the data from the input file.
   * Describe how you will store the data (what objects will you store?)
   * How will you add/delete/modify data?
   * How will you search the data?
   * List the classes you will need to implement your application.
3. Draw a UML diagram that shows all classes in the program and their relationships. This can be done however you want. If you want a specific application, [StarUML](http://staruml.io/download) is a good one. But you are welcome to use any graphics program or just draw them by hand and scan them.
4. Think how you will determine if your code functions are expected. Develop a test plan based on the above description; how will you test that the expected outputs have been achieved? **Be sure this test plan is complete.** Your test plan should minimally test each option in the menu-driven user interface for expected behavior and error-resistance.

Phase II: Project Implementation and Testing

05/08/2020

**Step 1**: Implement each of the classes in your design as well as the application. Be sure to document your code. Each class should have the your name, date, and a description o the class in comments at the top of the file.

**Step 2**: Test each of the classes according to your proposed test plan. Be sure to work with small pieces of the whole before trying to put the entire project together. Nothing is more discouraging than a large project with numerous errors that compound one another to the point that it is impossible to know where to begin the debugging process.

**Step 3**: Test your application

**Step 4**: Carefully check your results with the expected results and debug the project.

**Step 5**: Before submission, ask a few friends to test your application to see if your user interface is user-friendly and 'unbreakable'. Correct any problems you identify.