CS201 - Fall 2016
Final Project
Design Due: 11:59pm on 11/18/2016
Code Due: 1159pm on 12/02/2016

BACKGROUND:
We have been using encapsulated arrays and lists for the past few weeks in labs to model the CTA. We have learned about inheritance and polymorphism as well. The final project will use these same concepts, but the design for the system will be of your own choosing.
Note: You may discuss the final project with other students, but the work you submit must be your own.

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:

- Add a new station to the CTA system and be able to verify that the data has been added
- 'find' (search) for a specific station; users should be able to verify that the search was successful or unsuccessful
- Delete a station from the CTA system; user should be able to verify that the data was deleted
- Generate a route from a starting station to a destination station (this should be able to handle the possibility of a user needing to transfer from one line to another)
- Find the nearest station to a location
- Exit the program

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 by console displays. You may of course add additional menu options.

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 overridden method *that is NOT a usual instance method and* ***is unique*** *to your application (NOT toString() or equals()).*
- You should be able to demonstrate polymorphism by displaying data to the console *and* by writing data to an output file.
- You should be able to search the data which is also verified by displaying to the console and by writing to an output file.
- 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.

- You should be able to write an output file that shows the results of running your program using the following functionalities: add, delete, modify, search, and polymorphism.
- 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.
- 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 application class and methods in the application class to minimize the size of the main()
- You should NOT have large blocks of code.

IMPLEMENTATION ISSUES:

- Use descriptive name 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 another method.
- 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, author, and the date.
- All identifiers should be commented.
- Each method should have a brief description of its task.

PHASE 1: PROJECT DESIGN

1. Describe the user interface. What are the menu options and how will the user use the application?
2. Describe the programmers' tasks.
    a. Describe how you will read the input file.
    b. Describe how you will process the data from the input file.
    c. Describe how you will store the data (what objects will you store?)
    d. How will you demonstrate polymorphism?
    e. How will you add/delete/modify data?
    f. How will you search data?
3. List the classes you will need to implement your application.
4. Draw a UML class diagram that shows all classes in the program and their relationships. This can be done however you want. I use StarUML, which is available (free) here: http://staruml.io/download, but you can use a graphics program or just draw them by hand and scan them.

5. 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:
    a. Test each option in the menu-driven user interface for
        i. Expected behavior
        ii. Error-resistant input
    b. Test that you can demonstrate polymorphism with your application-specific method
        i. Test results should confirm polymorphism

PHASE 2: PROJECT IMPLEMENTATION AND TESTING

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 student's name, date, and a description of 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.

GRADING:

Project Design (Phase 1) (11/18/2016)	50
Description of the user interface	5
Description of the programmers' tasks	18
Describe how you will read the input	3
Describe how you will process the data from the input file	3
Describe how you will store the data	3
How will you demonstrate polymorphism?	3
How will you add/delete/modify data?	3
How will you search data?	3
Classes: List of names and descriptions	7
UML Class Diagrams	10

Testing	10
Project Code (Phase 2) (12/02/2016)	100
Compiles and runs with no run-time errors, menu is error-proof	10
Include input file(s)	10
Test Plan	12
Copy of the output file from your test plan with labeled test results	5
Inheritance relationship	5
Polymorphism	5
Association relationship	5
Searching works	5
Uses a list	5
Project adds, deletes, and modifies data stored in list	5
Project encapsulates data	3
Project is well coded with good design	10
All classes are complete (getters, setters, toString, equals…)	5
Documentation	15
Total	**150**