CS 474
Object Oriented Programming Languages and Environments
Fall 2018

Homework #2

Complete By: **Thursday, September 20th @ 11:59pm**
Submission: **submitted via Gradescope**

Readings

Chapters 0 and 1 of Object-Oriented Programming with C++ and Smalltalk. The book talks in a general sense about the concepts but is light on specific syntax so experiment with the code given in class to determine what does and doesn't work with class hierarchies.

Programming Exercise

The Human Resources department of a certain company (HumansDefinitelyNotRobots Inc or HDNR Inc) had such great success with their software for keeping track of a directory of employees they were able to sell their application to a variety of other companies. Among them is a particular educational facility which wishes to expand on the application to suit their needs.
It is your task to build a directory application with the following features.

The main screen prompts the user for a choice of numbers, representing the action the user wants to take

Upon entering the character 1, the user is further prompted for the name of a comma separated value file (.csv), and then loads all the names from that file into the directory. The csv contains a comma separated list of attributes specifying the format of the file in the first line, and data about people, one person per line, in the remaining lines. Note that since a single format is specified for all people, many of the fields will be empty. Some fields may contain multiple values, separated by commas. These fields will be surrounded by double quotes.

Upon entering the character 2, the user is further prompted for a name to enter into the directory. This name should be in the format FirstName LastName. The user is then prompted to enter the type of person, then based on that answer asked to enter values in each of the relevant fields.

Upon entering the character 3, the user is prompted for a name search query. This should behave the same as the query from HW1, repeated here. The sequence of characters they enter becomes a substring to be matched against first and last names in the database. If there is a space in the sequence, the program searches for a match among of the first name to the sequence leading up to the space, and the last name to the sequence after the space. All names in the database that match the query should be printed out, with all of the data associated with that user and a blank line between users.

Upon entering the character 4, the user is prompted for the name of a course. The application should output the name of the instructor for the course, if there is one, followed by the teaching assistants, if there are any, followed by the names of all students currently taking the course.

Upon entering the character 5, the application should list all courses currently being taken by students which do not have an instructor, followed by all instructors alongside the courses they are currently instructing.

Upon entering the character 6, all accounts should be updated with a payment. Instructors should be credited according to their salary multiplied by the number of classes they are teaching. TAs should be credited according to the product of their hours worked and salary. Staff should be credited according to their salary.

Upon entering the character 7, the user is prompted for the name of a user. Then, the user is prompted for a list of emails. Then output for each email whether an email could be successfully sent according to the following table (doesn't make sense, just encoding some arbitrary restrictions)

	Student	TA	Instructor	Staff
Student	NO	YES	NO	NO
TA	YES	YES	YES	NO
Instructor	NO	NO	YES	YES
Staff	YES	YES	NO	YES

Upon entering the character 0, the application should exit.

Your solution must make non-trivial use of the following language features: classes, inheritance, polymorphism.
This application will be a console application in either C++ or Java, compiled and run at the command line. The name of the file containing your main function should be {your netid}HW02.
For example, if I were to submit a java solution, I would put my main function in a file called sdeitz2HW02.java.
When finished, submit a zip containing all your source file(s) in a folder called {your netid} at the base directory of the zip (put your files into a folder, then zip that folder) to Gradescope.

Detailed input and output examples to be provided next week, build your application open to generality based on the requirements posed by the interactions defined in this assignment.