Indian Institute of Information Technology, Allahabad

Advanced Data Structures and Algorithms (Jul-Dec 2018)

Lab Assignment (3 pages and 5 exercises)

Date of Assignment: 13/08/18 **Due Date: 16/08/18** **Instructor: Dr. S. Maity**

1. Create a class called *IDCard* that contains a person's name, ID and name of a file containing the person's photograph.

   Write *Accessor* and *Mutator* methods for each of these fields. Add the following two overloaded constructors to the class:

   IDCard()

   IDCard(String name, int ID, String filename)

   Test your program by creating different objects using these two constructors and printing and modifying their values on the console using the *Accessor* and *Mutator* methods.

2. Jerry Matthews is the owner of a small start-up music store called *MusicWorld*. The store sells music CDs, among the other things. Jerry plans to eventually develop a sales transaction processing system that allows him to enter data for a CD purchase and output vital information such as quantities, prices, subtotal, tax, and total price. Jerry is a beginning programmer, so he plans to develop the system one small step at a time. To start with, Jerry needs a program that asks the cashier to enter this vital information about the single CD title being purchased: the *identification code*, the *title*, the *price*, and the *quantity*. In addition to displaying these data to the user, the program will *display* the subtotal (price *times* quantity) for each CD title and the total (subtotal *plus* tax) for all CD titles. The *sales tax rate* is 6.25%. Jerry realizes that he will eventually need to add much more functionalities than processing the sale of just one CD title, but he plans to get this much to work before adding various enhancements. He now wants a function of forming the *unique ID number for each transaction*. This will help in receiving the variable that holds the complete date and time information of the transaction. It also manipulates the date into the desired format of the transaction ID. It holds a 24-hour date and time that looks like: "DD/MM/YY HH:MM:SS TMZ" for example "13/08/2018 13:00:00 EST" is equivalent to August 13, 2018 1:00PM EST.

   Determine Class and object. For each object, determine its attributes and methods. Write simple code in C++ for printing appropriate value for identified class instances.

3. Write a *Patient* class which inherits from the *Person* class. You may also need to use the *Money* class. The *Patient* class requires the following:
   a) A variable to store the *patient ID* for the patient
   b) A variable to store the *hospital name*
   c) A variable to store the patient 's *year of joining* the hospital
   d) A variable to store the patient 's *address*
   e) A variable to store the *medical fees* that the patient pays
   f) Constructor methods, which *initializes* the variables
   g) Methods to *set* the hospital, year of joining and address
   h) Methods to *get* the hospital, year of joining and address
   i) A method to *calculate the medical fees*: It should take a *Money* object (the basic fee for the year) as a parameter and use it to return the medical fee for the patient. The basic fee for the year is ₹1000.

4. Define a class *String* that represents a string by a *length* and a *pointer* to a string of characters.
   a) Write a *constructor* for String to allocate appropriate storage for it and to initialize it to a given string. To allocate storage for an array of characters of length N, use the C++ operation

      new char[N]

   b) Write a *constructor* for String to allocate storage of a given size for string but not to initialize its characters.
   c) Write a method *concat* to concatenate one String with another.

5. In method overriding**,** the new method should essentially perform the same functionality as the method that it is replacing. However, by changing the functionality we can improve the method and make its function more appropriate to a specific subclass. To see its use, let's imagine a special category of *Magazine* which has a disc attached to each copy. We can call this a *DiscMag* and we would create a subclass of *Magazine* to deal with *DiscMags*. When a new issue of a *DiscMag* arrives, not only do we want to update the current stock but we want to check if discs are correctly attached. Therefore we want some additional functionality in the *recvNewIssue()* method to remind us to do this. We achieve this by redefining *recvNewIssue()* in the *DiscMag* subclass. Note that when a new issue of *Magazine* arrives, those that do not have a disc we want to invoke the original *recNewIssue()* method defined in the *Magazine* class. When we call the *recvNewIssue()* method on a *DiscMag* object, our program automatically selects the new

overriding version - the caller doesn't need to specify this or even know that it is an overridden method at all. When we call the *recvNewIssue()* method on a *Magazine* it is the method in the super class that is invoked. To implement *DiscMag* we must create a subclass of *Magazine* using *extends*. No additional instance variables or methods are required though it is possible to create some if needed. The *constructor* for *DiscMag* simply passes all its parameters directly to the super class and a version of a *recvNewIssue()* is defined in *DiscMag* to override the one inherited from *Magazine*.

Following the above description and the diagram below, code it using C++.