EE 205 Data Structures and Algorithms **Fall 2018**

Project 1: Student Management System

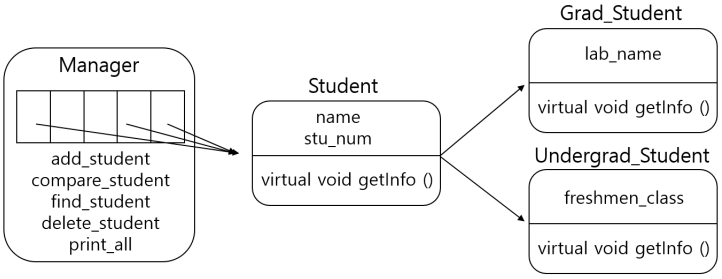
Lecturer: Yung Yi TA: Jihwan Bang

Introduction

In this project, you will implement a simple student management system using basic knowledge of C++ programming learned in class. This is a preliminary project to other projects so that you can easily adapt to C++ programming to build a variety of data structures and algorithms in earnest. This project require knowledge about basic concept of object-oriented programming, inheritance, function and operator overloading, overriding, virtual function, and so on.

Your simple student management system will be able to manage students in college for various purposes. Accepting correct data will result in generating the desired output in correct format without significant delay. The system basically requires the functionality of adding, deleting, finding students with specific information. Note that detailed name of the methods or members need not be the same as the examples given in this guide.

Class Structure



Class Structure

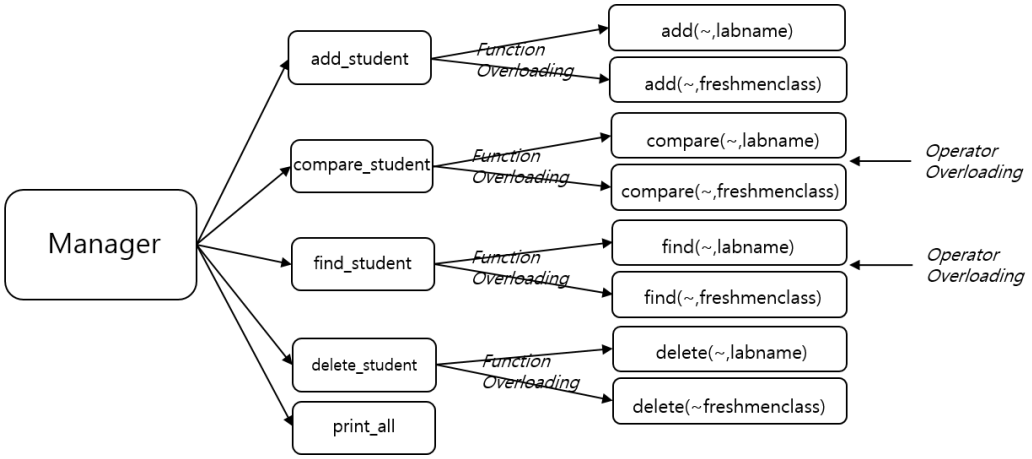
Firstly, you have to build a student manager class which manages the entire set of students and handles the main functions system. The **Manager** class maintains an array of `Student` class pointers. This array should be capable of handling up to 300 students. The `Manager` class maintains 6 methods to deal with the students in the system: `add_student`, `compare_student`, `find_student`, `delete_student`, `print_all` methods. Details of these methods will be provided in **Required Functionality** section.

The **Student** class needs to have the following members: `name`, `stu_num`. These members can be accessed by child classes, but not by external classes. The **getInfo()** method which simply prints all the information about the object needs to be implemented as a pure virtual function in this class. This is not a recommendation, but mandatory. These methods will be implemented on child class in detail. Also, initialization method should be implemented as constructor.

The **Grad_Student** class inherits `Student` class. It additionally has the following member: `lab_name`. This member also cannot be seen by external class. The **getInfo()** method which simply prints all the information about the object needs to be implemented as a pure virtual function in this class. This is not a recommendation, but mandatory. These methods will be implemented on child class in detail. Also, initialization method should be implemented as constructor.

The **Undergrad_Student** inherits the `Student` class. It additionally has the following member: `freshmen_class`. This member also cannot be seen by external classes. The **getInfo()** method which simply prints all the information about the object needs to be implemented as a pure virtual function in this class. This is not a recommendation, but mandatory. These methods will be implemented on child class in detail. Also, initialization method should be implemented as constructor.

Required Functionality



Required Functionality

Main function needs the following functionality: `add_student`(), `compare student`(), `find student`(), `delete student`(), `print all`(). All the details about input argument format and output format is given in **'student.cpp'** file.

1. **add_student()** method adds student object given by the argument in student array. This method should to be capable of adding both `graduate_student` object and `undergraduate_student` object using the function overloading. This is not a recommendation, but mandatory.

2. **compare_student()** method compares whether the (index argument)th object in the student array is the same to the object with given arguments followed by index.

3. **find_student()** method finds all the student objects from the student array which has the same member as given argument. Before implementing this method, you first have to implement student comparing operator == by **operator overloading**. This is not a recommendation, but mandatory. This overloaded boolean operator compares whether two compared objects have exactly the same members, and returns true if they are all the same, false otherwise. This method needs to be capable of comparing `undergraduate_student` and `graduate_student`, `graduate_student` and `graduate_student`, and so on (i.e. any two objects of child classes of the Student class).

Using this operator, you then need to implement `find_student` method. This method should to be capable of finding both `graduate_student` class and `undergraduate_student` class using function overloading.

4. **delete_student()** method deletes the student object that matches to the argument object from the student array. If there are multiple matches, your program should delete the first matched student object. You should be careful so that there exists no null pointer in the middle of the student array. This method should be capable of deleting both a `graduate_student` object and an `undergraduate_student` object.

5. **print_all()** methods simply prints all the information of all the student objects exist in the array.

Submission

Starter Code We have provided 3 files: main.cpp, student.cpp, student.h. In student.h file, you need to define member variable and format of methods. In student.cpp file, you need to actually implement the methods. In main file, you need to handle the exception case for invalid and improper argument.

Makefile You need to build your own makefile. You can easily learn how to build makefile on any website. It will not burden you much since this project does not need you to build a lot of files. Simply typing 'make' in your terminal should construct your final objective file properly. It will be included in grading policy.

Readme Please turn in a Readme file along with your code which states any specific consideration we should take to grade your code properly or anything that you thinks to be ambiguous in interpreting this project. E.g. I have discussed with Joonki Hong / I changed the input format / I use set function instead of constructor...

Submission Format Put main.cpp, student.cpp, student.h along with makefile and readme file in a single directory and compress it. Your final submission file should be compressed single file with name format `'20xxyyyy_pj1.tar.gz'`.

Grading Policy

Submission 0.5 Makefile Compiling 0.5 Main Function 1

Class Initialization 0.5 Virtual Function 0.5 Proper Inheritance 1

Add Student 1 Operator Overloading 0.5 Find Student 1

Delete Student 1.5 Change Student 1.5 Print All 0.5