

Programming Assignment Week #7

OO Inheritance: Member Functions and Variables

Adding members extends the definition of a class. It is not necessary to do this by changing the class directly; instead, you can inherit members of a class when designing another class. Inheritance lets you create classes from existing classes. The new classes are called derived classes, and the existing classes are called base classes.

Each derived class, in turn, could become a base class for a future derived class. In single inheritance, the derived class is derived from a single base class; in multiple inheritance, the derived class is derived from more than one base class.

The public members of a class become public members of the derived class. The private members of a class become private members of the derived class. However, public members of a base class can be inherited either as public members or as private members of the derived class.

Objectives

In this lab, you define derived classes from base classes.

After completing this lab, you will be able to:

- Write a derived class when given the base class.
- Create an instantiation of the derived class (declare objects) and the base class.
- Recognize the difference between objects of the base class and objects of the derived class.
- Be able to write and extend constructors for derived and base classes.

Instructions

Step 1

Download the 3 files called Assg6Test.cpp, Person.cpp and Person.h from the class web site. Create a project and add these files to it in your IDE. The Person.[h/cpp] header and definition files implements a basic data type that defines a person, with 3 pieces of information the persons first and last name, and their age. The test program contains a main() function that shows how to instantiate objects of type Person, and use the accessor methods of the class.

You should get the files into a project and make sure they compile and run successfully. Once you have that working move on to step 2.

Step 2

We are going to use inheritance to extend the Person base class. We will create a new data type, called Student, which extends or inherits from the Person class. To do this you will need to add 2 new files to your project, call them Student.h and Student.cpp. As previously, the Student.h header file will contain the header information and declarations of the Student data type, while the Student.cpp implementation file will be where you implement the methods for your Student class.

Your Student class should do the following:

- It should be an extension of the Person class. That means you need to inherit the Person class as the base class of your Student class. A Student is a type of Person.
- Add member variables to your Student class. A Student is a Person, so it inherits the first name, last name and age attributes of its Person base class. A student should also have the following attributes:
 - gpa: a float, holds the student's grade point average, a float ranging from 0.0 to 4.0
 - grade: an integer holding the grade level, e.g. 12 for high school senior, 13 for college freshman, 14 for college sophomore, 17 for first year graduate student, etc.
 - classes: An array of up to 5 strings, that will be used to record the current classes a student is enrolled in. For example, the student might have the following strings in their classes array `classes[0] = "CSci 152"; classes[1] = "Hist 101";`
- All member variables should be private by convention and good programming practices. This will mean you will need accessor and mutator methods in order to get at these values:
 - Add accessor and mutator methods for the gpa and grade. Given an object of type Student, you should be able to:

```
Student s;  
s.setGpa(3.5);  
cout << "The GPA of s is: " << s.getGpa() <<  
endl;  
s.setGrade(13);  
cout << "s is in " << s.getGrade() << " grade  
" << endl;
```

- The classes is a little different in terms of accessing and changing. Create a mutator function that takes a string (the name of a class) and a slot number (from 0 to 4) and assigns that slot in the classes array to the given class name. For example:

```
Student s;  
s.setClass("CSci 152", 0);  
s.setClass("Hist 101", 1);
```

- For an accessor, write a function called `getClasses()` that returns a string. Your function should create a string representation of all of the classes the student is enrolled in as indicated by their classes array and return that as a result.
- Your Student class should also contain a new Constructor. Figure out how to chain constructors in C++. Your constructor should allow the programmer to provide, in addition to the `firstName`, `lastName` and `age`, also the `gpa` and `grade` of the student (ignore initializing the classes for now). For example, to construct a variable of type Student using this constructor I should be able to do something like:

```
// firstName, lastName, age, gpa, grade (
// yes I am in an equivalent of the 23rd grade :- (
Student s("Derek", "Harter", 35, 3.7, 23);
```

Try and make sure that you chain your constructor appropriately so that you are only assigning the values of the `gpa` and `grade` in your constructor, and are using the Person base class constructor to assign the `firstName`, `lastName` and `age` values.

- Feel free, if you are getting the hang of things, to add more appropriate student attributes to your Student data type. How about some test scores, and maybe a `letterGrade` function that returns a letter grade based on the GPA or the average of their test scores, etc.
- Also another thing to try, try and override the Person base classes' `str()` method. Use the `str()` function as an example of what to do, but have your Student `str()` method return a string with all of this extra info about a student, like `gpa`, `grade`, `classes`, etc.

Step 3

Add appropriate test code to the `main()` function to test our your new Student class. Here is just a small example of what you should be able to do with your student, but you should add more to this:

```
Student jane; // do have an appropriate default constructor
for your Student data type?
Student nate("Nathan", "Allen", 22, 3.2, 16);
Student sue("Susan", "Monday", 19, 2.8, 14);

cout << "sue full name: " << sue.getFullName() << endl;
cout << "jane gpa: " << jane.getGpa() << endl;
nate.setGrade(15);
cout << "nate grade: " << nate.getGrade() << endl;
nate.setClass("Astronomy 444", 0);
nate.setClass("Physics 389", 1);
cout << "Nate classes: " << nate.getClasses() << endl;

cout << "The full info for the student" << endl;
```

```
cout << jane.str() << endl;  
cout << sue.str() << endl;  
cout << nate.str() << endl;
```

Assignment 6 Finished

You have now completed Assignment 6. If your program compiles and runs correctly and you have successfully uploaded your source file to the eCollege online submission site, then you are done.