EECS 297/397 Special Topics: C/C++ Programming Spring 2020

Assignment 5: Inheritance

Instructor: Orhan Özgüner Due: March 20 before 11:59 PM

This assignment consists of five required parts and one bonus. You will be constructing a framework for a university course-management system similar to what was done in the previous assignment, only this time you will be constructing the personnel hierarchy of the department.

All of the fields in the classes you will create must be `private`, and you should modify them using `get()` and `set()` methods, and constructors.

You are provided `main.cpp` which includes all of the object creations, and a function call at the end which is intended to display the structure of the department. When you open it, there will be errors because the classes it calls are missing. You will create classes and source/header files to fill in those errors. Your submission (including the bonus) should only include the `.h` and `.cpp` files discussed below. We will use our own copy of main.cpp to grade.

Part 1

Create the base class Person (`person.h` and `person.cpp`). A Person should contain only a numerical ID and a string for a name. You should have both a full constructor (i.e. `Person(int, string)`) and an empty constructor (i.e. `Person()`) which initializes the member values to some sort of defaults that a real Person could never have (like an ID of 0 and a name that is an empty string).

Part 2

Create the classes Student and Employee (`student.h` / `student.cpp`, and `employee.h` / `employee.cpp`). Both Students and Employees should inherit from Person. A Student should also have

- A numerical graduation year
- A major (a string).
- A GPA (number that will normally take on a range between 0.0 and 4.0)

An Employee should also have

- A numerical Salary
- A Department the Employee works for (a string).

Part 3

Create two types of Employee: Faculty and Staff (`faculty.h / faculty.cpp`, and `staff.h / staff.cpp`). Staff just have a Title (“Administrative Assistant”, “Student Affairs Rep”, etc.), while Faculty have a Position (“Associate Professor”, “Assistant Professor”, “Senior Professor” and so on) and also a Research Focus (“Bioinformatics”, “Complexity Theory”, and so on).

Since both of these classes are intended to be actually created, make sure that you have both a full constructor (one that fills in all the fields with its arguments) and an empty constructor (`Staff()` or `Faculty()`) that fills in the fields with sensible defaults.

Faculty and Staff should both have a method called `bool works_together(Employee other)`. If `works_together()` is called on a Staff member, it should return true if their departments are the same and false otherwise. If `works_together()` is called on a Faculty member, it should return true if `other` is a Staff member with no department, *or* `other` is another Faculty member with the same Department *and* the same ResearchFocus.

Part 4

Create a ResearchAssistant class (`research_assistant.h` and `research_assistant.cpp`). A ResearchAssistant has the properties of both a Student and a Staff, since either profession can take that role. The ResearchAssistant itself also has a Research Area field.

It should thus have two constructors:

```
ResearchAssistant(std::string area, std::string title, std::string department, int
salary, int id, std::string name)
```

creates a ResearchAssistant that is like a Staff, and

```
ResearchAssistant(std::string area, std::string major, double gpa, int grad_year, int
id, std::string name)
```

creates a ResearchAssistant that is like a Student. The fields not applicable to the ResearchAssistant in question should be initialized automatically to default values no real person in that role could have. You are free to either call the parent constructors or fill in the parent fields directly (both methods were shown in the Vehicle example provided).

Part 5

Create an Administrator class (`administrator.h` and `administrator.cpp`). An Administrator is both a Staff and a Faculty object, but not all of the properties of both classes are free to be set. Instead, its Title and Position values are both automatically set to "Department Administrator" and its Research Focus is the same as the Department it is in. Thus, its constructor should look like this:

```
Administrator(std::string department, int salary, int id, std::string name)
```

Administrators should also include `std::vector`s of Students, ResearchAssistants, Faculty, and Staff. There should be corresponding `AddStudent`, `AddResearcher`, `AddFaculty`, and `AddStaff` methods.

The Administrator must also include a `print()` method which outputs all of the Person objects it has stored in the order of Students, ResearchAssistants, Faculty, and Staff, one per line.

The output should then look something like this:

```
   Students:
12360:  Tom Shkurti.  Class of 2022.  Majoring in Computer Science.  GPA = 4.000000.
12361:  Iayn Bogdanov.  Class of 1974.  Majoring in Computer Science.  GPA = 3.500000.
   Staff:
12350:  Cynthia Hatcher.  Assistant to the Chair.  Works in Computer Science for $20000.
12351:  David Jarvi.  Technician.  Works in Computer Science for $20000.
   Faculty:
12346:  Murat Cenk Cavusoglu.  Professor.  Works in Computer Science for $50000.  Studies
Surgical Robotics
12347:  Orhan Ozguner.  Adjunct Professor.  Works in Computer Science for $50000.  Studies
Surgical Robotics
12349:  Mehmet Koyuturk.  Professor.  Works in Computer Science for $50000.  Studies
Bioinformatics
   Research assistants:
12362:  Ammar Nahari.  Class of 2022.  Majoring in Computer Science.  GPA = 3.000000.
Studying Robotics
12375:  Russel Jackson.  Lab Manager.  Works in Computer Science for $1000.  Studying
Robotics
```

You do not need to precisely match the spacing or phrasing of the example, although you do need to include all of the fields it prints.

Part 6 (Bonus)

Create a constructor in the ResearchAssistant class with the following signature:

```
   ResearchAssistant(
std::string area, Student student_in, Administrator & administrator_to_notify )
```

This will create a new ResearchAssistant with the student's Name, ID Number, Major, and other identifying information, give it a new research area, and then *look through the Administrator's lists, delete a student with that ID, and add itself to the ResearchAssistant list*, effectively promoting the Student to a ResearchAssistant.