CT5707 物件導向程式分析與設計

Project

The purpose of this C++ homework is to give you some critical insights into object-oriented design.

You are required to solve the same problem using three different approaches to object-oriented design:

1. An approach that seems very natural at first sight. This approach involves the use of a diamond hierarchy.
2. A more flexible approach using mixin classes.
3. And, finally, an approach that uses role-playing classes. This will be the most sophisticated approach and will involve the use of the RTTI feature of C++. RTTI stands for Run Time Type Identification.

Your goal is to design an object-oriented system that could be used in a university for registering students for courses, for keeping track of what teachers are teaching which courses, etc. Your program should make provision for the following classes:

a) `Person`
b) `Student`
c) `GraduateStudent`
d) `GraduateTeachingAssistant`
e) `Teacher`

In addition to whatever else you might want to put into it, the class Person should hold the following data

```
string name;
string address;
unsigned long ssn;
Date birthdate;
```

In addition to inheriting directly (or accessing indirectly) the attributes of Person, the Student class should contain the following data members:

```
StudentStatus status;
Department department;
Course* enrolledFor[ MaxCoursesForStudent ];
int numCourses;
```

Similarly, in addition to inheriting directly (or accessing indirectly) the attributes of Person, the Teacher class should contain the following data members:

```
Rank rank;
double salary;
Department department;
Course* coursesOffered[ MaxCoursesForTeacher ];
string* graders[ MaxGraders ];
```

The GraduateStudent class, to be derived from the Student class, should account for the fact that every graduate student has an advisor and that a graduate student is not allowed to enroll for low-level courses. The following attributes are to be defined directly for the GraduateStudent class:

```
Teacher advisor;
unsigned short numCourses;
```

The last class, GradTeachAsst, is to be derived from both the GraduateStudent class and the Teacher class.

Please keep in the mind the following:

I) All the data members should be in the private section of each class. And because the data members will be private, you'd need to define and implement the 'get' and the 'set' methods for most if not all the data members.

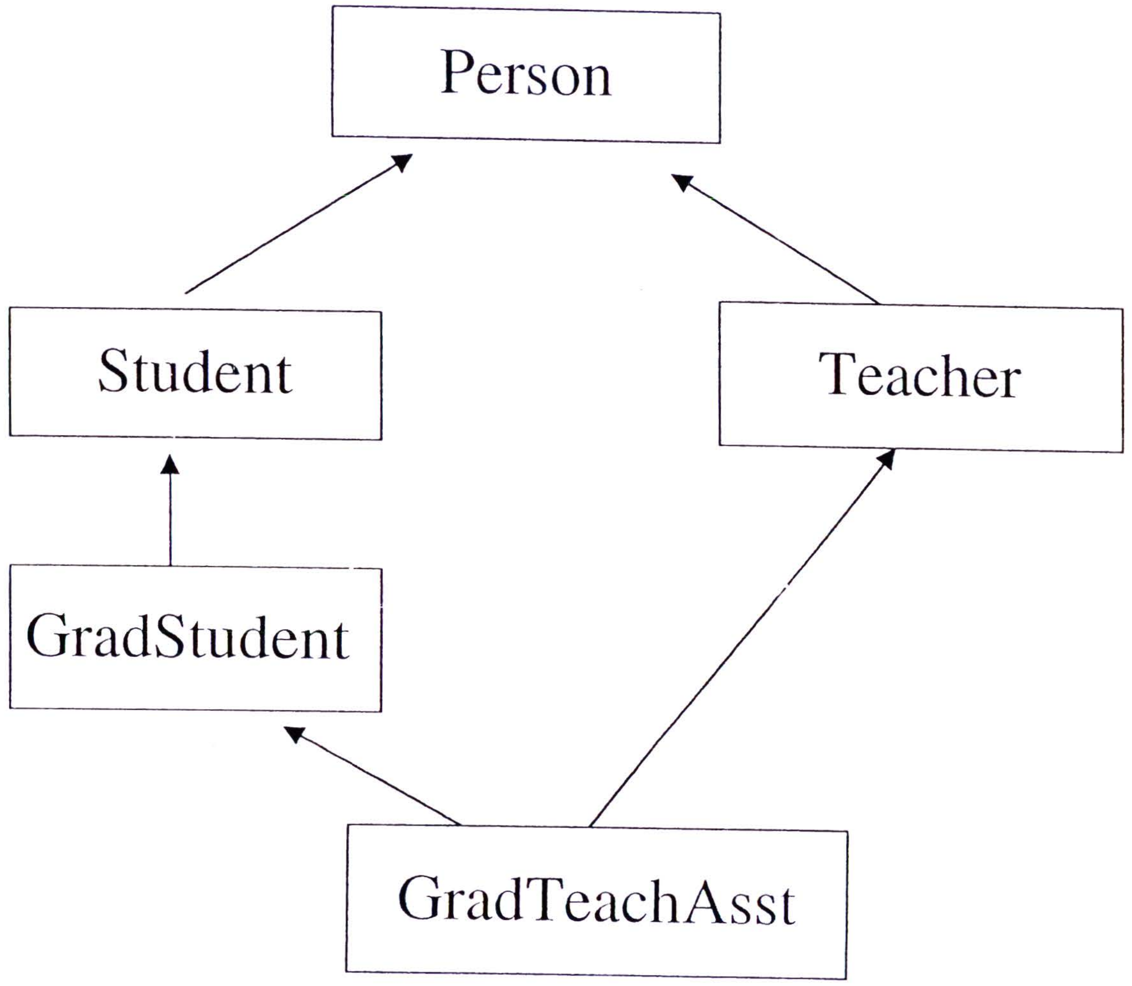
II) **Each class should have a print() method that must behave polymorphically with respect to the entire hierarchy.**

III) To cope with the problems caused by repeated inheritance, do not forget about the concept of a virtual base.

The class hierarchy for the first approach will be as shown in Figure 1. Note that a GraduateTeachAsst will inherit two data members with the same name: the 'department' data member from the Student class and the 'department' data member from the Teacher class. These are different and distinct data members for a GradTeachAsst. A graduate teaching assistant can be a student in one department and a TA in a different department. Make sure that the constructor for GradTeachAsst reflects this fact. The constructor for GradTeachAsst also has to bear the primary responsibility for constructing the Person sub-object.

The class hierarchy for the second more flexible approach will be as shown in Figure 2. The two mixin classes that you must implement are M_QualifiedToTeach and M_CanBecomeStudent. The third mixin class, M_QualifiedToDoResearch, is shown to underscore the fact that this kind of a hierarchy is relatively easier to extend.

And, finally, the class hierarchy for the last approach will look like what's shown in Figure 3. Note the nature of the association link between the class Person and the class UniversityMember. A Person now contains an aggregation of the UniversityMember objects. Each UniversityMember object represents a role, which can be a Student role, a Teacher role, etc. There will be only one active role at one time.



```
Student Status   status ;
status  =  Fulltime ;

cout<< status << endl ;     // 0
cout << status Labels [status] <<endl;  // "full-time"
```

Figure 1

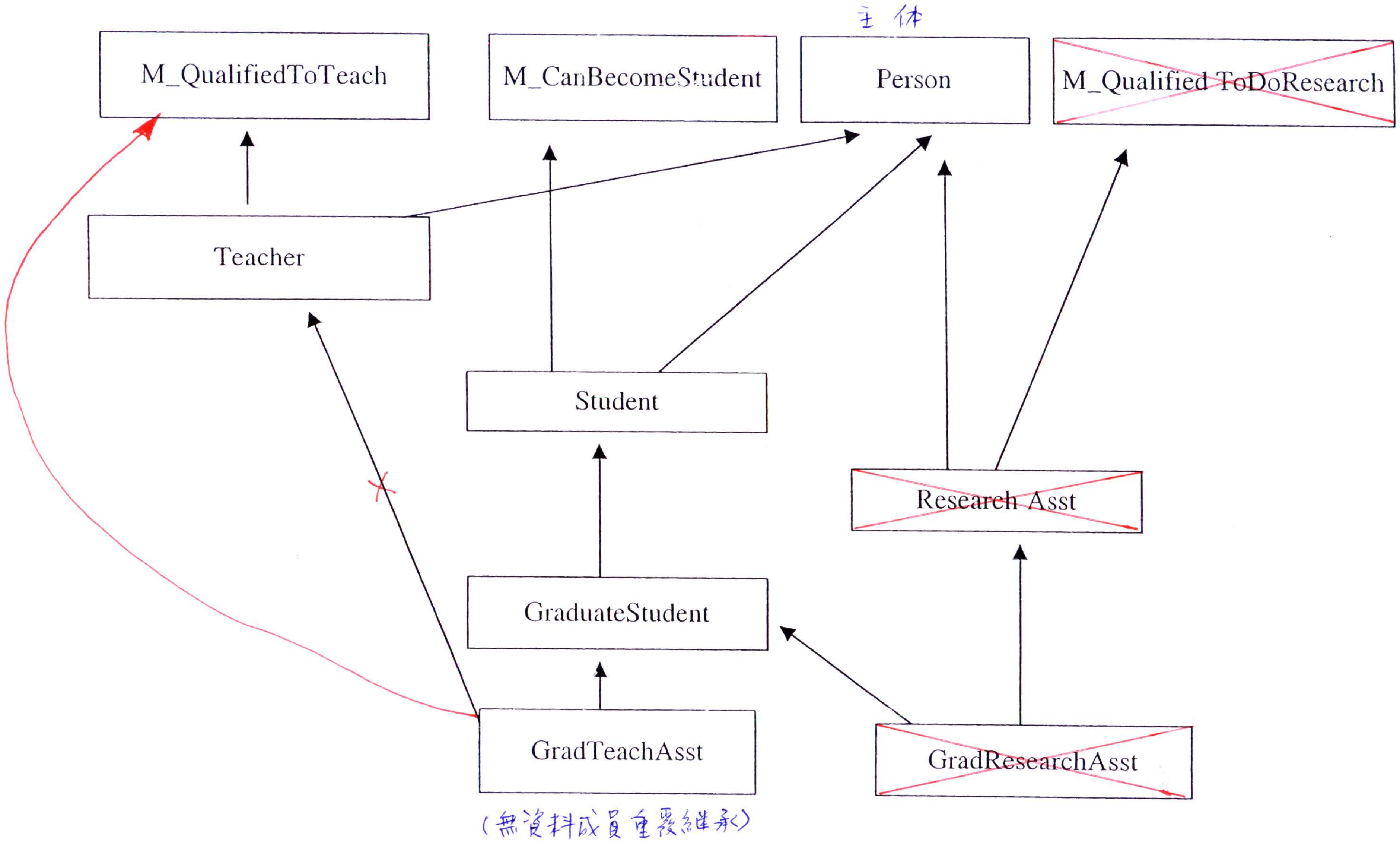
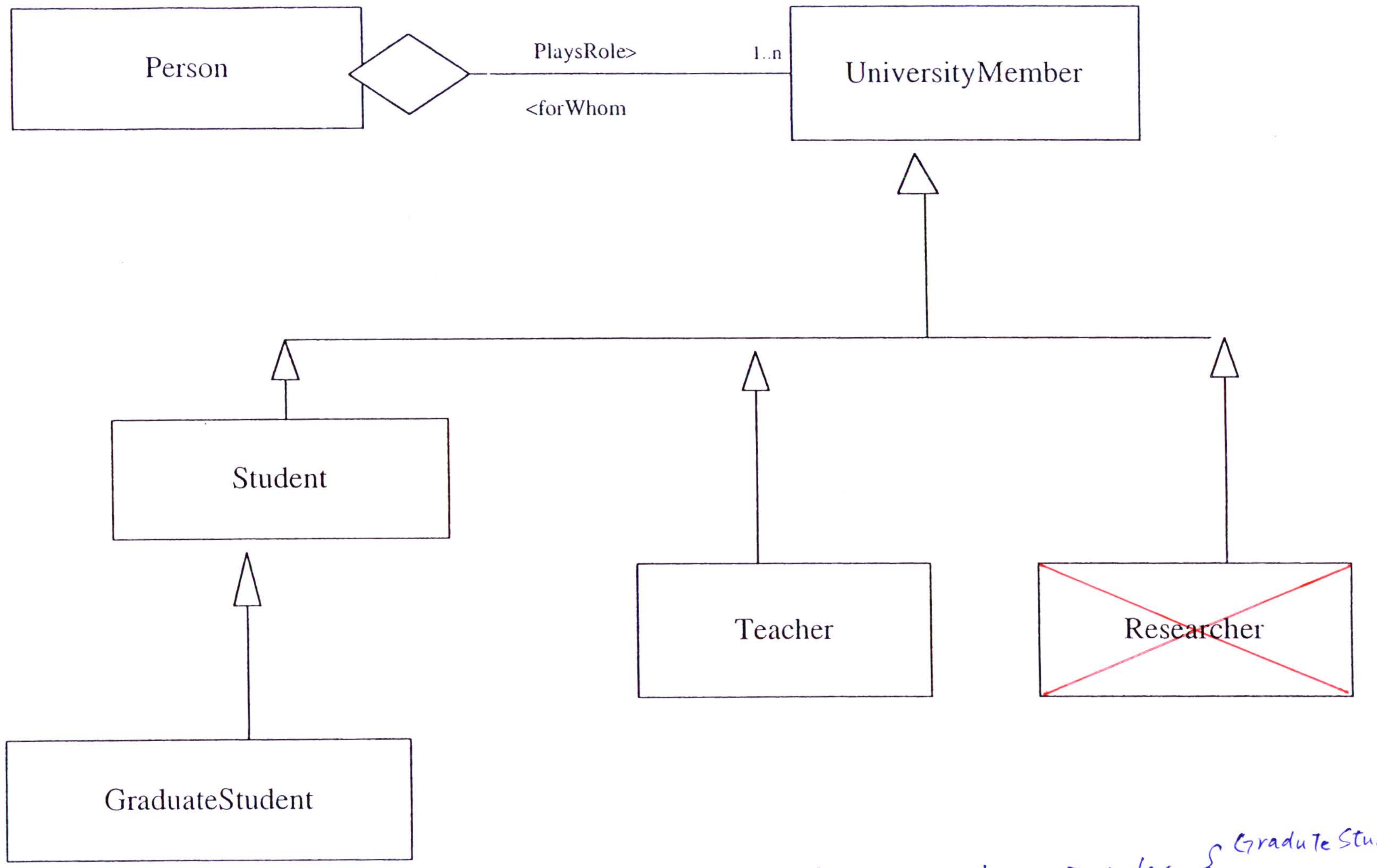

主 体
M_QualifiedToTeach
M_CanBecomeStudent
Person
~~M_QualifiedToDoResearch~~
Teacher
Student
~~Research Asst~~
GraduateStudent
GradTeachAsst
~~GradResearchAsst~~
(無資料成員重覆繼承)

Figure 2



Grad TA : has 2 roles { Gradute Student
Teacher

Figure 3