VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECNHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

ADVANCED PROGRAMMING (CO2039)

Assignment Report

Student Management System

Advisor: Trương Tuấn Anh
Phan Duy Hãn
Student: Nguyễn Thành Phát – 2252605 (Group CC02)

HO CHI MINH CITY, APRIL 2024

Contents

1 An overview of student management system

In this report, we will use the UML diagram to illustrate the idea of our student management system. In Figure 1, classes involved in UML diagram are `Student`, `UniStu`, `CollegStu`, `StuFactory`, `UniFactory`, `CollegFactory` and `University`. Specifically, both `UniStu` and `CollegStu` inherits class `Student` with different value of attributes. For example:

- Student in `UniStu` has 8 semesters, each semester has 4 courses, and each courses has 3 assignments, 2 tests and 1 exams.
- Student in `CollegStu` has only 4 semester, each semester has 3 courses, and each courses has 1 assignment, 1 test and 1 exam.
- Each class of student has to keep track of status of taking assignment, test and exam.
- The two class `UniStu` and `CollegStu` have different methods for displaying and calculating the GPA based on the scores (we may assume that the score's range is between 0 and 100).

`University` class is used to store the information of all students in the system. The methods of this class demonstrated in Figure 1 allows client to:

- `addStu`: Add the student's information on a specific location (index) in the system and the type of that student (university or college).
- `delStu`: Delete a student appeared firstly in the system based on the name and the type of that student.
- `getBest`: Find the largest GPA in the specific type of student.
- `displayBest`: Display the information of all students having the highest GPA based on the type.
- `display`: Display the information of all students based on the type.
- `study`: Assign random scores of a student based on the name and type.

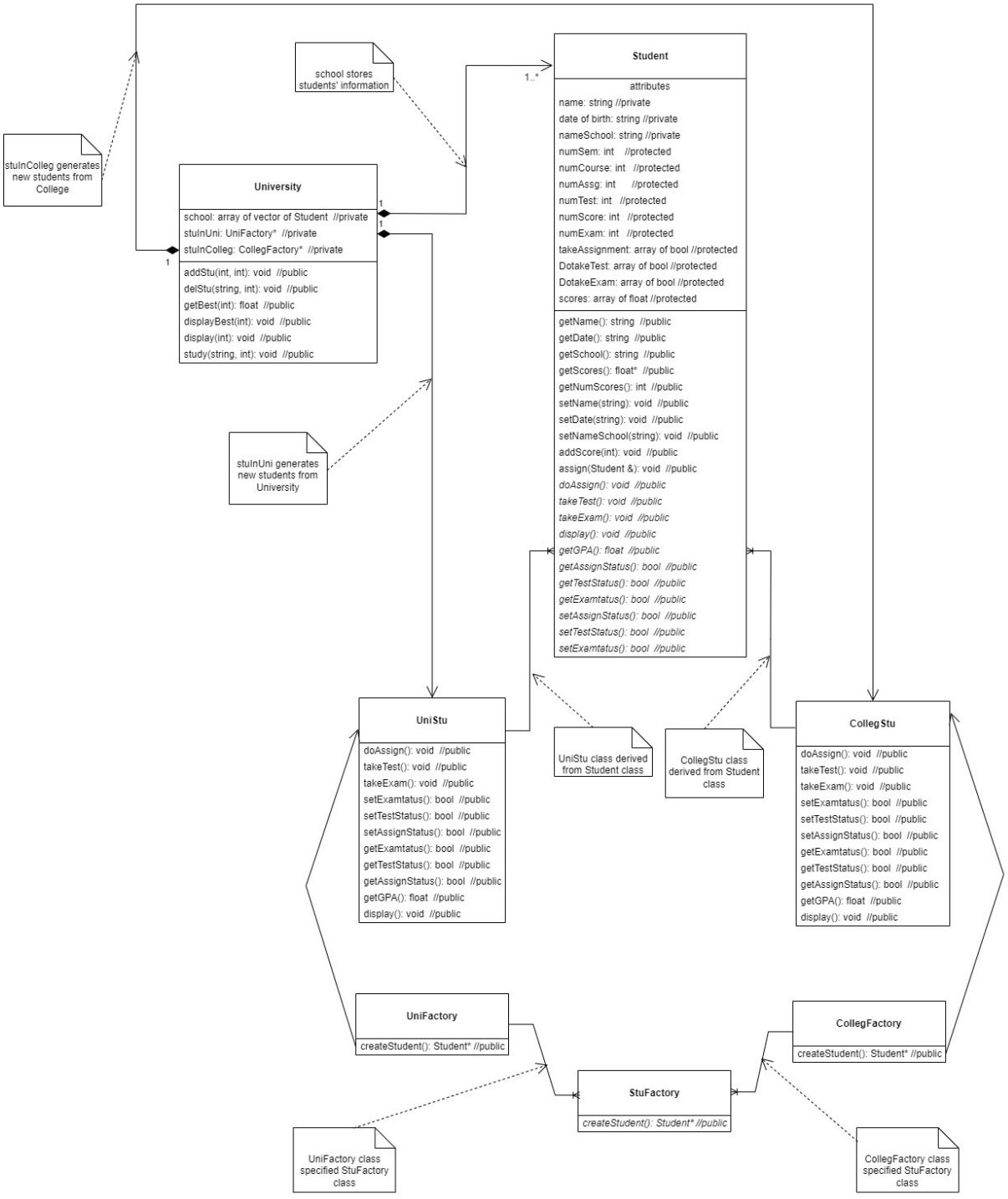

Figure 1: *UML diagram*

2 Vector implementation in the design

In Part 2, many data structures such as stack, queue, vector, map are introduced. In this section, we use these data structures to improve our system.
In Figure 1, considering `University` class, we use an array of vector to store the list of students instead of two-dimensional dynamic array. The reasons we decide to store the list of students of each type by using vector are:

- The number of students in each type does not know beforehand.
- Applying the adding method based on specified position requires the implementation to be more flexible.
- We could use the data structure `map`. Although `map` gives effective traversal for addition, search or deletion (time complexity: $O(log(n))$), `map` needs more space for storing each node (each node has two pointers, which the storage must greater that 8 bytes). The time complexity of addition, search or deletion of `vector` is $O(n)$, which is acceptable.

In our implementation, all of students in the same type (University or College) are grouped in one vector. So, there are only two vectors representing the students in University and ones in College. Observe that the number of vectors is fixed, we may use static array because accessing the n-th element is much effective than using dynamic array.

3 Factory pattern implementation in the design

In Part 3, several design patterns such as singleton, factory, adapter are introduced. In this section, we use these design patterns to improve our system.
In Figure 1, considering `University` class, whenever the `addStu` method is called, we have to allocate more memory for the student by using `new` syntax. This can make our code more complex and hard to read. The advantages of implementing Factory design pattern are:

- Allow the client to create objects without knowing the logic behind them.
- Encapsulate the creation of objects and forcing this creation in fixed manner.
- Whenever creating students, the client just calls the method.

However, in `University` class, we have to declare two pointer representing the two Factories, which adds to this class 8 bytes. This may result to overhead when the number of students are large.

4 Modern features implementation in the design

In Part 4, modern features such as smart pointers, range for loop, auto, decltype, ... are introduced. In this section, we use the features to improve our system.
In Figure 1, considering `University` class, each times we iterate over the list of students, we use modern range-based loop instead the traditional for loop since it is more readable for the client. Moreover, there are several times that we use the iterator to iterate over the list, the keyword `auto` helps us to declare the iterator without specifying its type. The keyword `auto` can help us form a good manner - initializing after declaring the variable (since the `auto` does not accept the uninitialized variables).