**Virtual College**

**Functional requirements:**

* A virtual college provides you e-learning facility, it can be used by staff as well as students
* It allows you to take admission to any department.
* It allows students to attend sessions, give exams, view results, view attendance, issue books
* It allows staff to conduct session, mark attendance, conduct exams, issue books
* exams can be practical or theory

**Technical requirements:**

* Have class of users. Student, staff and staff can be the users
* have class of departments. Consider there are 2 types of department eg. Computer and ENTC.
* Students can give exams and view result
* Student/Staff can issue and return books

**Problems and solution:**

* As your system contains complex subsystem you should provide a simple interface. So use Facade pattern.
* Students of diffrent department can give either theory or practical exams. So accordingly method definitions will change. So use Factory pattern and let subclasses decide the definition based on department and type of exam
* Students can issue books. Some permission is required to issue book. So admin acts the mediator who grants permission to issue books. So use Mediator pattern.

**1. college class**

This will contain the description of college.

**2. department class**

This class contains the different departments, which can be for eg. computer or ENTC.

**3. exam interface**

Now department wise there can be theory exam and practical exam. So both computer and ENTC departments can conduct exam for different subject using different marking schemes.So when student belonging to any department wants to give exam, the exam interface will decide which defintion to implement based on theory exam or practical exam So here is the implementation of **Factory design pattern.**

**4. users class**

There can be three kind of users so generalization into 3 classes:-

* + - * 1. student class
        2. staff class
        3. admin class

Each of them will have different rigths to access the virtual college

**5. books class**

Both student and staff can issue books, return books, and pay fine based on delay in submission. Now admin class will act as the **mediator** between student/staff and books.

Student/staff issues book using issueBook() method, admin will check if book is present or not and grant book to student/staff. While returning book, admin will check of delay in submission and call generateFine() in books class and send fine amount to staff/student. Hence admin acts as mediator here, so **Mediator design pattern** is used.

**6. Facade pattern**

College class provides a simple interface to complex system consisting of many users and many departments. So there is implementation of **Facade design pattern.**