DESIGN MODEL OF VIRTUAL COLLEGE SYSTEM AND ITS DESIGN PATTERNS

**Singleton Pattern:**

**Singleton design pattern ensures that only one instance of the class is created. It allows global point of access to the object. The instance can be used from anywhere and it is impossible to directly invoke the constructer each time.**

**Therefore, we declare constructor as private so that we cannot create an object outside the class. Thus it serves the purpose of control the object creation by keeping the constructor as private.**

In this case, the **Principal** is a singleton. It means that there can be at most only one Principal at a given time. Regardless of the personal identity, the title Principal is the global point of access that defines the person and the task which are related to him.

**Factory Pattern:**

**If we have a base class and its n derived classes and based on the data provided, we have to return the object of one of the derived-classes we use factory pattern.**

**FacultyType** class will hold type of faculty and provide faculty type to all other classes. There can be three types of faculty i.e. lecturer, professor and assistant professor.

public enum FacultyType{

LECTURER, PROFESSOR, ASST\_PROFFESOR

}

**Faculty** class is the parent class of all the instances and it will contain common logic applicable to all the faculty types.

public abstract class Faculty {

    public Faculty(FacultyType post) {

        this.post = post;

        arrangeParts();

    }

// Do subclass level processing in this method

    protected abstract void identify();

    private FacultyType post = null;

    public FacultyType conduct\_exam() {

. . .

    }

    public void give\_assigns() {

. . .

            }

}

Class **Lecturer** is concrete implementation of faculty type lecturer.

Class **Professor** is concrete implementation of faculty type professor.

Class **Asst\_Professor** is concrete implementation of faculty type assistant professor.

**FacultyFactory** is the main class implementation using factory pattern. It instantiates the faculty instance only once after determining its type.

public class FacultyFactory {

    public static Faculty instantiate(FacultyType post) {

        Faculty faculty = null;

        switch (post) {

        case LECTURER:

           faculty = new Lecturer();

            break;

        case PROFESSOR:

            faculty = new Professorr();

            break;

        case ASST\_PROFESSOR:

            faculty = new Asst\_professor();

            break;

                }

        return faculty;

    }

}