Lab Assignment 2

Subject: Object-Oriented Programming
Semester: Spring 2025

Class: FA24-BSE-A
Assignment Date: 25-03-2025
Due Date: 25-03-2025 **Maximum Marks:** 10

CLO4 - Implement a small module utilizing object oriented design.

You are tasked with developing a School Management System using object-oriented principles. The system should manage details about a school, its teachers, students, and classes. A **School** has a name, an address, and a list of classes. The school is also associated with a **Principal** who is a person with a name, age, and experience in years. The relationship between the school and the principal is one of **composition** — the school cannot exist without a principal. If the school is deleted, the principal object should also be deleted.

Each **ClassRoom** in the school has a class name (e.g., "Grade 10"), a class code (e.g., "G10"), and a teacher. A classroom can have multiple students, but for this assignment, you are required to limit the number of students to a maximum of **five**. You are expected to use a simple array for storing student objects. The relationship between the school and the classroom is also one of **composition** — if the school is deleted, all associated classrooms should also be deleted.

A **Teacher** is a person with a name, age, and the subject they teach. A teacher can only be assigned to a single classroom at a time. The relationship between a teacher and a classroom is an example of **aggregation** — a teacher can exist independently of a classroom, and deleting a classroom does not delete the teacher object. Similarly, a **Student** is a person with a name, age, and a roll number. A student can be enrolled in a single class at a time. The relationship between a student and a classroom is also an example of **aggregation** — a student can exist without being enrolled in a class, and deleting a classroom does not delete the student object.

To model this system effectively, you are required to create a `Person` class that will act as a **parent class** for the `Teacher`, `Student`, and `Principal` classes. The `toString()` method should be overridden in each child class to display class-specific details in a formatted way. You should also implement the `equals()` method in the `Student` and `Teacher` classes. The `equals()` method in the `Student` class should compare two student objects based on their roll number, whereas the `equals()` method in the `Teacher` class should compare two teacher objects based on their teacher ID.

This assignment will test your ability to design and implement an object-oriented solution using **inheritance**, **composition**, **aggregation**, and **method overriding**. You are expected to define meaningful constructors and methods, handle logical cases like class limits, and ensure that the output is cleanly formatted.

Step 1: Create a UML Class Diagram

Before starting the coding task, you are required to first design a **UML class diagram** on paper. The diagram should reflect the relationships between the classes (`inheritance`, `composition`, and `aggregation`) and define the attributes and methods for each class. Make sure that the relationships are clearly labeled. The class diagram should be neatly drawn and properly formatted. Submit a **scanned copy** or a **clear photograph** of the completed UML class diagram along with your code submission.

Step 2: Implement the Classes

Once the UML diagram is complete and approved by the instructor, proceed with the implementation of the system. Define meaningful constructors and methods for each class. Make sure that the constructors properly initialize the objects and that the `toString()` and `equals()` methods are correctly overridden. Ensure that the relationships between the objects are implemented as described in the scenario.

Step 3: Write Code to Perform the Following Tasks

1. Create a **School** object with a **Principal** and at least two classes.
2. Create a **Teacher** object and assign them to a class.
3. Enroll up to five **Student** objects in each class.
4. Prevent enrolling a student into a class if the class is already full — display an appropriate message if the student cannot be added.
5. Display the details of the school, including the principal, class, teacher, and student details in a structured format using the `toString()` method.
6. Demonstrate the use of the `equals()` method by comparing two student objects and two teacher objects.
7. Make sure that objects are connected logically — a student should reference their class, and a teacher should reference the class they are assigned to.

Submission Instructions

1. You are required to upload your **complete code** and the **UML class diagram** to a Git repository before the end of the lab session.
2. Once the code is uploaded, generate a link to the repository and submit it using the form below:
   GitHub Submission Form
3. **No updates to the repository** will be allowed after submission. Ensure that you have reviewed your code and tested it for errors before submitting.
4. The submission will be marked based on both the code quality and the accuracy of the UML diagram.