PROJECT
ON
OBJECT ORIENTED PROGRAMMING LANGUAGE

Project Outline

1. **Define the classes and their attributes:**

- Student: id, name, courses, grades
- Course: courseId, courseName, students, teacher
- Teacher: teacherId, name, course

Define Relationships

- A Student can enroll in multiple `Course`s.
- A Course can have multiple `Student`s and one Teacher.
- A Teacher can teach multiple `Course`s.

Implement Key OOPS Concepts

- **Encapsulation**: Use private fields and public getter/setter methods.
- **Inheritance**: If applicable, create a base class and derive specific classes from it.
- **Polymorphism**: Demonstrate method overriding and interfaces.

Step-by-Step Implementation.

1. **Create the Person Class (Base Class)**:

java

```
public class Person {
    private String id;
    private String name;
```

```
    public Person(String id, String name) {
        this.id = id;
        this.name = name;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

2**. Create the Student Class**:

java

```
import java.util.ArrayList;
import java.util.List;
```

```
public class Student extends Person {
    private List<Course> courses;
    private List<String> grades;
    public Student(String id, String name) {
        super(id, name);
        this.courses = new ArrayList<>();
        this.grades = new ArrayList<>();
    }
    public void enrollCourse(Course course) {
        courses.add(course);
    }
    public void addGrade(String grade) {
        grades.add(grade);
    }

    public List<Course> getCourses() {
        return courses;
    }

    public List<String> getGrades() {
        return grades;
    }
}
```

3**. Create the Teacher Class**:

java

```
import java.util.ArrayList;
import java.util.List;
public class Teacher extends Person {
    private List<Course> courses;
    public Teacher(String id, String name) {
        super(id, name);
        this.courses = new ArrayList<>();
    }
    public void assignCourse(Course course) {
        courses.add(course);
    }
    public List<Course> getCourses() {
        return courses;
    }
}
```

4**. Create the Course Class**:

java

```
import java.util.ArrayList;
import java.util.List;

public class Course {
    private String courseId;
    private String courseName;
    private Teacher teacher;
    private List<Student> students;

    public Course(String courseId, String courseName) {
        this.courseId = courseId;
        this.courseName = courseName;
        this.students = new ArrayList<>();
    }

    public void enrollStudent(Student student) {
        students.add(student);
    }

    public void assignTeacher(Teacher teacher) {
        this.teacher = teacher;
    }
```

```
    public String getCourseId() {
        return courseId;
    }

    public String getCourseName() {
        return courseName;
    }

    public Teacher getTeacher() {
        return teacher;
    }

    public List<Student> getStudents() {
        return students;
    }
}
```

5. **Main Class to Demonstrate the Project**:

```
java
public class Main {
    public static void main(String[] args) {
        // Create students
```

```
Student student1 = new Student("S1", "D.venkatesh");
Student student2 = new Student("S2", "E.jalandhar");

// Create teachers
Teacher teacher1 = new Teacher("T1", "Dr. Tharun Reddy");
Teacher teacher2 = new Teacher("T2", "S.Koti Reddy");

// Create courses
Course course1 = new Course("C1", "Mathematics");
Course course2 = new Course("C2", "Physics");

// Assign teachers to courses
course1.assignTeacher(teacher1);
course2.assignTeacher(teacher2);

// Enroll students in courses
course1.enrollStudent(student1);
course1.enrollStudent(student2);

course2.enrollStudent(student2);

// Students enroll in courses
student1.enrollCourse(course1);
student2.enrollCourse(course1);
```

```
        student2.enrollCourse(course2);

        // Assign courses to teachers
        teacher1.assignCourse(course1);
        teacher2.assignCourse(course2);

        // Add grades
        student1.addGrade("A");
        student2.addGrade("B");
        student2.addGrade("A");

        // Print out information
        System.out.println("Course: " + course1.getCourseName() + " taught by " + course1.getTeacher().getName());
        for (Student s : course1.getStudents()) {
            System.out.println("Student: " + s.getName() + " enrolled in " + s.getCourses().get(0).getCourseName());
        }

        System.out.println("Course: " + course2.getCourseName() + " taught by " + course2.getTeacher().getName());
        for (Student s : course2.getStudents()) {
            System.out.println("Student: " + s.getName() + " enrolled in " + s.getCourses().get(1).getCourseName());
        }
    }
```

}

Output:

```
D:\java programes>javac Person.java

D:\java programes>java Main
Course: Mathematics taught by Dr. Tharun Reddy
Student: D.venkatesh enrolled in Mathematics
Student: E.jalandhar enrolled in Mathematics
Course: Physics taught by Prof. s.koti Reddy
Student: E.jalandhar enrolled in Physics
```

Explanation of the Project

Encapsulation: Each class has private attributes and public getter/setter methods.

Inheritance: The Student and Teacher classes inherit from the Person class.

Polymorphism: You can extend this project by adding more methods to demonstrate method overriding, such as a common method in Person that can be overridden in Student and Teacher.

This project demonstrates a basic understanding of OOP principles using Java. You can further extend it by adding more features, such as removing students from courses, updating grades, and using interfaces for better abstraction.