Student Information System (SIS)

Implement OOPs

A Student Information System (SIS) manages information about students, courses, student enrollments, teachers, and payments. Each student can enroll in multiple courses, each course can have multiple students, each course is taught by a teacher, and students make payments for their courses. Students have attributes such as name, date of birth, email, and phone number. Courses have attributes such as course name, course code, and instructor name. Enrollments track which students are enrolled in which courses. Teachers have attributes such as names and email. Payments track the amount and date of payments made by students.

Task 1: Define Classes

Define the following classes based on the domain description:

Student class with the following attributes:

- Student ID
- First Name
- Last Name
- Date of Birth
- Email
- Phone Number

```
package entity;

import java.time.LocalDate;
import java.util.List;


public class Student {
    private int studentId;
    private String firstName;
    private String lastName;
    private LocalDate dateOfBirth;
    private String email;
    private String phoneNumber;
```

Course class with the following attributes:

- Course ID
- Course Name

- Course Code
- Instructor Name

```
package entity;

import java.util.List;

public class Course {
    private int courseId;
    private String courseName;
    private String courseCode;
    private String instructorName;
```

Enrollment class to represent the relationship between students and courses. It should have attributes:

- Enrollment ID
- Student ID (reference to a student)
- Course ID (reference to a Course)
- Enrollment Date

```
package entity;
import java.time.LocalDate;

public class Enrollment {
    private int enrollmentId;
    private int studentId;     // Reference to Student object
    private int courseId;      // Reference to Course object
    private LocalDate enrollmentDate;
```

Teacher class with the following attributes:

- Teacher ID
- First Name
- Last Name
- Email

```
package entity;

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

public class Teacher {
    private int teacherId;
    private String firstName;
    private String lastName;
    private String email;
```

Payment class with the following attributes:

- Payment ID
- Student ID (reference to a student)
- Amount
- Payment Date

```
package entity;
import java.time.LocalDate;

public class Payment {
    private int paymentId;
    private int studentId;  // Reference to Student object
    private double amount;
    private LocalDate paymentDate;
```

Task 2: Implement Constructors

Implement constructors for each class to initialize their attributes. Constructors are special methods that are called when an object of a class is created. They are used to set initial values for the attributes of the class. Below are detailed instructions on how to implement constructors for each class in your Student Information System (SIS) assignment:

Student Class Constructor

In the Student class, you need to create a constructor that initializes the attributes of a student when an instance of the Student class is created

SIS Class Constructor

If you have a class that represents the Student Information System itself (e.g., SIS class), you may also implement a constructor for it. This constructor can be used to set up any initial configuration for the SIS.

Repeat the above process for each class Course, Enrollment, Teacher, Payment by defining constructors that initialize their respective attributes.

STUDENT

```
package entity;

import java.time.LocalDate;
import java.util.List;

public class Student {
    private int studentId;
    private String firstName;
    private String lastName;
    private LocalDate dateOfBirth;
    private String email;
    private String phoneNumber;
public Student() {
                    super();
```

```
		// TODO Auto-generated constructor stub
	}
	public Student(int studentId, String firstName, String lastName, LocalDate date, String email,
			String phoneNumber) {
		super();
		this.studentId = studentId;
		this.firstName = firstName;
		this.lastName = lastName;
		this.dateOfBirth = date;
		this.email = email;
		this.phoneNumber = phoneNumber;
	}
	public int getStudentId() {
		return studentId;
	}
	public void setStudentId(int studentId) {
		this.studentId = studentId;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public LocalDate getDateOfBirth() {
		return dateOfBirth;
	}
	public void setDateOfBirth(LocalDate dateOfBirth) {
		this.dateOfBirth = dateOfBirth;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getPhoneNumber() {
		return phoneNumber;
	}
	public void setPhoneNumber(String phoneNumber) {
		this.phoneNumber = phoneNumber;
	}
	@Override
	public String toString() {
		return "Student [studentId=" + studentId + ", firstName=" + firstName + ", lastName=" + lastName
				+ ", dateOfBirth=" + dateOfBirth + ", email=" + email + ", phoneNumber=" +
phoneNumber + "]";
	}
}
```

COURSE

```
package entity;

import java.util.List;

public class Course {
   private int courseId;
   private String courseName;
   private String courseCode;
   private String instructorName;
public Course() {
		super();
		// TODO Auto-generated constructor stub
	}

	public Course(int courseId, String courseName, String courseCode, String instructorName) {
		super();
		this.courseId = courseId;
		this.courseName = courseName;
		this.courseCode = courseCode;
		this.instructorName = instructorName;
	}
	public int getCourseId() {
		return courseId;
	}
	public void setCourseId(int courseId) {
		this.courseId = courseId;
	}
	public String getCourseName() {
		return courseName;
	}
	public void setCourseName(String courseName) {
		this.courseName = courseName;
	}
	public String getCourseCode() {
		return courseCode;
	}
	public void setCourseCode(String courseCode) {
		this.courseCode = courseCode;
	}
	public String getInstructorName() {
		return instructorName;
	}
	public void setInstructorName(String instructorName) {
		this.instructorName = instructorName;
	}
```

```
	@Override
	public String toString() {
		return "Course [courseId=" + courseId + ", courseName=" + courseName + ",
courseCode=" + courseCode
				+ ", instructorName=" + instructorName + "]";
	}
}
```

TEACHER

```
package entity;

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

public class Teacher {
   private int teacherId;
   private String firstName;
   private String lastName;
   private String email;
	public Teacher() {
		super();
		// TODO Auto-generated constructor stub
	}
	public Teacher(int teacherId, String firstName, String lastName, String email) {
		super();
		this.teacherId = teacherId;
		this.firstName = firstName;
		this.lastName = lastName;
		this.email = email;
		this.assignedCourses = new ArrayList<>();
	}
	public int getTeacherId() {
		return teacherId;
	}
	public void setTeacherId(int teacherId) {
		this.teacherId = teacherId;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
```

```
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	@Override
	public String toString() {
		return "Teacher [teacherId=" + teacherId + ", firstName=" + firstName + ", lastName="
+ lastName + ", email="
				+ email + "]";
	}
}
```

ENROLLMENT

```
package entity;
import java.time.LocalDate;

public class Enrollment {
	private int enrollmentId;
   private int studentId;     // Reference to Student object
   private int courseId;       // Reference to Course object
   private LocalDate enrollmentDate;
   //default constructor
	public Enrollment() {
		super();
	}

	public Enrollment(int enrollmentId, int studentId, int courseId, LocalDate enrollmentDate,
Student student, Course course) {
		super();
		this.enrollmentId = enrollmentId;
		this.studentId = studentId;
		this.courseId = courseId;
		this.enrollmentDate = enrollmentDate;
		this.student = student;  // setting the student for this enrollment
      this.course = course;  // setting the course for this enrollment
	}

	public int getEnrollmentId() {
		return enrollmentId;
```

```
	}

	public void setEnrollmentId(int enrollmentId) {
		this.enrollmentId = enrollmentId;
	}

	public int getStudentId() {
		return studentId;
	}

	public void setStudentId(int studentId) {
		this.studentId = studentId;
	}

	public int getCourseId() {
		return courseId;
	}

	public void setCourseId(int courseId) {
		this.courseId = courseId;
	}

	public LocalDate getEnrollmentDate() {
		return enrollmentDate;
	}

	public void setEnrollmentDate(LocalDate enrollmentDate) {
		this.enrollmentDate = enrollmentDate;
	}
	@Override
	public String toString() {
		return "Enrollment [enrollmentId=" + enrollmentId + ", studentId=" + studentId + ",
courseId=" + courseId
				+ ", enrollmentDate=" + enrollmentDate + "]";
	}
}
```

PAYMENT

```
package entity;
import java.time.LocalDate;

public class Payment {
	private int paymentId;
   private int studentId;  // Reference to Student object
   private double amount;
```

```
    private LocalDate paymentDate;
    //default constructor
        public Payment() {
                super();
        }

        public Payment(int paymentId, int studentId, double amount, LocalDate paymentDate,Student
student) {
                super();
                this.paymentId = paymentId;
                this.studentId = studentId;
                this.amount = amount;
                this.paymentDate = paymentDate;
                this.student = student;
        }

        public int getPaymentId() {
                return paymentId;
        }

        public void setPaymentId(int paymentId) {
                this.paymentId = paymentId;
        }

        public int getStudentId() {
                return studentId;
        }

        public void setStudentId(int studentId) {
                this.studentId = studentId;
        }

        public double getAmount() {
                return amount;
        }

        public void setAmount(double amount) {
                this.amount = amount;
        }

        public LocalDate getPaymentDate() {
                return paymentDate;
        }

        public void setPaymentDate(LocalDate paymentDate) {
                this.paymentDate = paymentDate;
```

```
	}

	@Override
	public String toString() {
		return "Payment [paymentId=" + paymentId + ", studentId=" + studentId + ",
amount=" + amount + ", paymentDate="
				+ paymentDate + "]";
	}
}
```

Task 3: Implement Methods

Implement methods in your classes to perform various operations related to the Student Information System (SIS). These methods will allow you to interact with and manipulate data within your system. Below are detailed instructions on how to implement methods in each class:

Implement the following methods in the appropriate classes:

Student Class:

• EnrollInCourse(course: Course): Enrolls the student in a course.

• UpdateStudentInfo(firstName: string, lastName: string, dateOfBirth: DateTime, email: string, phoneNumber: string): Updates the student's information.

• MakePayment(amount: decimal, paymentDate: DateTime): Records a payment made by the student.

• DisplayStudentInfo(): Displays detailed information about the student.

• GetEnrolledCourses(): Retrieves a list of courses in which the student is enrolled.

• GetPaymentHistory(): Retrieves a list of payment records for the student.

```
package entity;

import java.time.LocalDate;
import java.util.List;


public class Student {
	private int studentId;
   private String firstName;
   private String lastName;
   private LocalDate dateOfBirth;
   private String email;
   private String phoneNumber;

   // Collection relationships
   private List<Enrollment> enrollments;
   private List<Payment> payments;
```

```
//getter for this
public List<Enrollment> getEnrollments() {
   return enrollments;
}

public List<Payment> getPayments() {
   return payments;
}

	public Student() {
		super();
		// TODO Auto-generated constructor stub
	}

	public Student(int studentId, String firstName, String lastName, LocalDate date, String email,
				String phoneNumber) {
		super();
		this.studentId = studentId;
		this.firstName = firstName;
		this.lastName = lastName;
		this.dateOfBirth = date;
		this.email = email;
		this.phoneNumber = phoneNumber;
	}
	public int getStudentId() {
		return studentId;
	}
	public void setStudentId(int studentId) {
		this.studentId = studentId;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public LocalDate getDateOfBirth() {
		return dateOfBirth;
	}
	public void setDateOfBirth(LocalDate dateOfBirth) {
		this.dateOfBirth = dateOfBirth;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getPhoneNumber() {
		return phoneNumber;
```

```
	}
	public void setPhoneNumber(String phoneNumber) {
		this.phoneNumber = phoneNumber;
	}
	@Override
	public String toString() {
		return "Student [studentId=" + studentId + ", firstName=" + firstName + ", lastName=" + lastName
				+ ", dateOfBirth=" + dateOfBirth + ", email=" + email + ", phoneNumber=" +
phoneNumber + "]";
	}
}
```

Course Class:

• AssignTeacher(teacher: Teacher): Assigns a teacher to the course.

• UpdateCourseInfo(courseCode: string, courseName: string, instructor: string): Updates course information.

• DisplayCourseInfo(): Displays detailed information about the course.

• GetEnrollments(): Retrieves a list of student enrollments for the course.

• GetTeacher(): Retrieves the assigned teacher for the course.

```
package entity;

import java.util.List;

public class Course {
   private int courseId;
   private String courseName;
   private String courseCode;
   private String instructorName;

   //collection relationship
   private List<Enrollment> enrollment;

   //getter for this
   public List<Enrollment> getEnrollments() {
      return enrollment;
   }

	public Course() {
		super();
		// TODO Auto-generated constructor stub
	}


	public Course(int courseId, String courseName, String courseCode, String instructorName) {
		super();
		this.courseId = courseId;
		this.courseName = courseName;
		this.courseCode = courseCode;
```

```
		this.instructorName = instructorName;
	}


	public int getCourseId() {
		return courseId;
	}
	public void setCourseId(int courseId) {
		this.courseId = courseId;
	}
	public String getCourseName() {
		return courseName;
	}
	public void setCourseName(String courseName) {
		this.courseName = courseName;
	}
	public String getCourseCode() {
		return courseCode;
	}
	public void setCourseCode(String courseCode) {
		this.courseCode = courseCode;
	}
	public String getInstructorName() {
		return instructorName;
	}
	public void setInstructorName(String instructorName) {
		this.instructorName = instructorName;
	}
	@Override
	public String toString() {
		return "Course [courseId=" + courseId + ", courseName=" + courseName + ", courseCode=" +
courseCode
				+ ", instructorName=" + instructorName + "]";
	}

}
```

Enrollment Class:

• GetStudent(): Retrieves the student associated with the enrollment.

• GetCourse(): Retrieves the course associated with the enrollment.

```
package entity;
import java.time.LocalDate;

public class Enrollment {
	private int enrollmentId;
   private int studentId;     // Reference to Student object
   private int courseId;       // Reference to Course object
   private LocalDate enrollmentDate;

   //collection relationship
```

```
    private Student student;
    private Course course;

    //getter for this

    public Student getStudent() {
        return student;
    }

    public Course getCourse() {
        return course;
    }

    //default constructor
            public Enrollment() {
                    super();
            }

            public Enrollment(int enrollmentId, int studentId, int courseId, LocalDate enrollmentDate, Student student,
Course course) {
                    super();
                    this.enrollmentId = enrollmentId;
                    this.studentId = studentId;
                    this.courseId = courseId;
                    this.enrollmentDate = enrollmentDate;
                    this.student = student;  // setting the student for this enrollment
        this.course = course;  // setting the course for this enrollment
            }

            public int getEnrollmentId() {
                    return enrollmentId;
            }

            public void setEnrollmentId(int enrollmentId) {
                    this.enrollmentId = enrollmentId;
            }

            public int getStudentId() {
                    return studentId;
            }

            public void setStudentId(int studentId) {
                    this.studentId = studentId;
            }

            public int getCourseId() {
                    return courseId;
            }

            public void setCourseId(int courseId) {
                    this.courseId = courseId;
            }

            public LocalDate getEnrollmentDate() {
                    return enrollmentDate;
            }
```

```
	public void setEnrollmentDate(LocalDate enrollmentDate) {
		this.enrollmentDate = enrollmentDate;
	}
	@Override
	public String toString() {
		return "Enrollment [enrollmentId=" + enrollmentId + ", studentId=" + studentId + ", courseId=" +
courseId
				+ ", enrollmentDate=" + enrollmentDate + "]";
	}
}
```

Teacher Class:

• UpdateTeacherInfo(name: string, email: string, expertise: string): Updates teacher information.

• DisplayTeacherInfo(): Displays detailed information about the teacher.

• GetAssignedCourses(): Retrieves a list of courses assigned to the teacher.

```
package entity;

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

public class Teacher {
   private int teacherId;
   private String firstName;
   private String lastName;
   private String email;


   //collection relationship
   private List<Course> assignedCourses;

   //getter for this
   public List<Course> getAssignedCourses() {
      return assignedCourses;
   }

	public Teacher() {
		super();
		// TODO Auto-generated constructor stub
	}
	public Teacher(int teacherId, String firstName, String lastName, String email) {
		super();
		this.teacherId = teacherId;
		this.firstName = firstName;
		this.lastName = lastName;
		this.email = email;
		this.assignedCourses = new ArrayList<>();
	}
	public int getTeacherId() {
```

```
                return teacherId;
        }
        public void setTeacherId(int teacherId) {
                this.teacherId = teacherId;
        }
        public String getFirstName() {
                return firstName;
        }
        public void setFirstName(String firstName) {
                this.firstName = firstName;
        }
        public String getLastName() {
                return lastName;
        }
        public void setLastName(String lastName) {
                this.lastName = lastName;
        }
        public String getEmail() {
                return email;
        }
        public void setEmail(String email) {
                this.email = email;
        }
        @Override
        public String toString() {
                return "Teacher [teacherId=" + teacherId + ", firstName=" + firstName + ", lastName=" + lastName + ",
email="
                                + email + "]";
        }
}
```

Payment Class:

• GetStudent(): Retrieves the student associated with the payment.

• GetPaymentAmount(): Retrieves the payment amount.

• GetPaymentDate(): Retrieves the payment date.

```
package entity;
import java.time.LocalDate;

public class Payment {
        private int paymentId;
   private int studentId;  // Reference to Student object
   private double amount;
   private LocalDate paymentDate;


   //collection relationship
   private Student student;

   //getter for this
```

```
        public Student getStudent() {
           return student;
        }

   //default constructor
        public Payment() {
                super();
        }

        public Payment(int paymentId, int studentId, double amount, LocalDate paymentDate,Student student) {
                super();
                this.paymentId = paymentId;
                this.studentId = studentId;
                this.amount = amount;
                this.paymentDate = paymentDate;
                this.student = student;
        }

        public int getPaymentId() {
                return paymentId;
        }

        public void setPaymentId(int paymentId) {
                this.paymentId = paymentId;
        }

        public int getStudentId() {
                return studentId;
        }

        public void setStudentId(int studentId) {
                this.studentId = studentId;
        }

        public double getAmount() {
                return amount;
        }

        public void setAmount(double amount) {
                this.amount = amount;
        }

        public LocalDate getPaymentDate() {
                return paymentDate;
        }

        public void setPaymentDate(LocalDate paymentDate) {
                this.paymentDate = paymentDate;
        }

        @Override
        public String toString() {
                return "Payment [paymentId=" + paymentId + ", studentId=" + studentId + ", amount=" + amount + ",
paymentDate="
                                        + paymentDate + "]";
        }       }
```

SIS Class (if you have one to manage interactions):

• EnrollStudentInCourse(student: Student, course: Course): Enrolls a student in a course.

• AssignTeacherToCourse(teacher: Teacher, course: Course): Assigns a teacher to a course.

• RecordPayment(student: Student, amount: decimal, paymentDate: DateTime): Records a payment made by a student.

• GenerateEnrollmentReport(course: Course): Generates a report of students enrolled in a specific course.

• GeneratePaymentReport(student: Student): Generates a report of payments made by a specific student.

• CalculateCourseStatistics(course: Course): Calculates statistics for a specific course, such as the number of enrollments and total payments.

Use the Methods

In your driver program or any part of your code where you want to perform actions related to the Student Information System, create instances of your classes, and use the methods you've implemented.

Repeat this process for using other methods you've implemented in your classes and the SIS class.

```
package entity;

import java.sql.Date;
import java.time.LocalDate;
import java.util.List;

public class Sis {
            public Sis() {
       // Initialize lists, DB connection, or setup configs if any
       System.out.println("Student Information System initialized.");
   }
             // Add Enrollment method
            public void addPayment(Student student, double amount, Date utilDate) {
               LocalDate paymentDate = utilDate.toInstant().atZone(java.time.ZoneId.systemDefault()).toLocalDate();
               int studentId = student.getStudentId();
               Payment payment = new Payment(0, studentId, amount, paymentDate, student);
               student.getPayments().add(payment); // use plural getter
            }


             // Assign Course to Teacher method
   public void assignCourseToTeacher(Course course, Teacher teacher) {
      teacher.getAssignedCourses().add(course);  // Add course to teacher's course list
   }

   // Add Payment method
   public void addPayment(Student student, double amount, LocalDate paymentDate) {
```

```
        int studentId = student.getStudentId(); // Assuming the Student class has a method to get studentId
        Payment payment = new Payment(0, studentId, amount, paymentDate, student);
        student.getPayments().add(payment); // Add payment to student's payment history
    }

            // Get Enrollments for a Student method
              public List<Enrollment> getEnrollmentsForStudent(Student student) {
                 return student.getEnrollments();  // Retrieve student's list of enrollments
              }

            // Get Courses for a Teacher method
              public List<Course> getCoursesForTeacher(Teacher teacher) {
                 return teacher.getAssignedCourses();  // Retrieve teacher's list of assigned courses
              }
}
```

Task 4: Exceptions handling and Custom Exceptions

Implementing custom exceptions allows you to define and throw exceptions tailored to specific situations or business logic requirements.

Create Custom Exception Classes You'll need to create custom exception classes that are inherited from the System.Exception class or one of its derived classes (e.g., System.ApplicationException). These custom exception classes will allow you to encapsulate specific error scenarios and provide meaningful error messages.

Throw Custom Exceptions

In your code, you can throw custom exceptions when specific conditions or business logic rules are violated. To throw a custom exception, use the throw keyword followed by an instance of your custom exception class.

• **DuplicateEnrollmentException**: Thrown when a student is already enrolled in a course and tries to enroll again. This exception can be used in the EnrollStudentInCourse method.

```
package exception;

public class DuplicateEnrollmentException extends Exception {
    public DuplicateEnrollmentException(String message) {
        super(message);
    }
}
```

• **CourseNotFoundException**: Thrown when a course does not exist in the system, and you attempt to perform operations on it (e.g., enrolling a student or assigning a teacher).

```
package exception;

public class CourseNotFoundException extends Exception {
```

```
    public CourseNotFoundException(String message) {
        super(message);
    }
}
```

• **StudentNotFoundException**: Thrown when a student does not exist in the system, and you attempt to perform operations on the student (e.g., enrolling in a course, making a payment).

```
package exception;

public class StudentNotFoundException extends Exception {
    public StudentNotFoundException(String message) {
        super(message);
    }
}
```

• **TeacherNotFoundException**: Thrown when a teacher does not exist in the system, and you attempt to assign them to a course.

```
package exception;

public class TeacherNotFoundException extends Exception {
    public TeacherNotFoundException(String message) {
        super(message);
    }}
```

• **PaymentValidationException**: Thrown when there is an issue with payment validation, such as an invalid payment amount or payment date.

```
package exception;

public class PaymentValidationException extends Exception {
    public PaymentValidationException(String message) {
        super(message);
}
}
```

• **InvalidStudentDataException**: Thrown when data provided for creating or updating a student is invalid (e.g., invalid date of birth or email format).

```
package exception;

public class InvalidStudentDataException extends Exception {
    public InvalidStudentDataException(String message) {
        super(message);
    }
}
```

• **InvalidCourseDataException**: Thrown when data provided for creating or updating a course is invalid (e.g., invalid course code or instructor name).

```
package exception;

public class InvalidCourseDataException extends Exception {
    public InvalidCourseDataException(String message) {
        super(message);
    }
}
```

• **InvalidEnrollmentDataException**: Thrown when data provided for creating an enrollment is invalid (e.g., missing student or course references).

```
package exception;

public class InvalidEnrollmentDataException extends Exception {
    public InvalidEnrollmentDataException(String message) {
        super(message);
    }
}
```

• **InvalidTeacherDataException**: Thrown when data provided for creating or updating a teacher is invalid (e.g., missing name or email).

```
package exception;

public class InvalidTeacherDataException extends Exception {
    public InvalidTeacherDataException(String message) {
        super(message);
    }
}
```

• **InsufficientFundsException**: Thrown when a student attempts to enroll in a course but does not have enough funds to make the payment.

```
package exception;

public class InsufficientFundsException extends Exception {
    public InsufficientFundsException(String message) {
        super(message);
    }
}
```

Task 5: Collections

Implement Collections:

Implement relationships between classes using appropriate data structures (e.g., lists or dictionaries) to maintain associations between students, courses, enrollments, teachers, and payments.

These relationships are essential for the Student Information System (SIS) to track and manage student enrollments, teacher assignments, and payments accurately.

Define Class-Level Data Structures

You will need class-level data structures within each class to maintain relationships. Here's how to define them for each class:

Student Class:

Create a list or collection property to store the student's enrollments. This property will hold references to Enrollment objects.

Example: List<Enrollment> Enrollments { get; set; }

```
package entity;

import java.time.LocalDate;
import java.util.List;


public class Student {
		private int studentId;
    private String firstName;
    private String lastName;
    private LocalDate dateOfBirth;
    private String email;
    private String phoneNumber;

    // Collection relationships
    private List<Enrollment> enrollments;
    private List<Payment> payments;

    //getter for this
    public List<Enrollment> getEnrollments() {
        return enrollments;
    }

    public List<Payment> getPayments() {
        return payments;
    }

		public Student() {
			super();
			// TODO Auto-generated constructor stub
		}

		public Student(int studentId, String firstName, String lastName, LocalDate date, String email,
				String phoneNumber) {
			super();
			this.studentId = studentId;
```

```
                this.firstName = firstName;
                this.lastName = lastName;
                this.dateOfBirth = date;
                this.email = email;
                this.phoneNumber = phoneNumber;
        }
        public int getStudentId() {
                return studentId;
        }
        public void setStudentId(int studentId) {
                this.studentId = studentId;
        }
        public String getFirstName() {
                return firstName;
        }
        public void setFirstName(String firstName) {
                this.firstName = firstName;
        }
        public String getLastName() {
                return lastName;
        }
        public void setLastName(String lastName) {
                this.lastName = lastName;
        }
        public LocalDate getDateOfBirth() {
                return dateOfBirth;
        }
        public void setDateOfBirth(LocalDate dateOfBirth) {
                this.dateOfBirth = dateOfBirth;
        }
        public String getEmail() {
                return email;
        }
        public void setEmail(String email) {
                this.email = email;
        }
        public String getPhoneNumber() {
                return phoneNumber;
        }
        public void setPhoneNumber(String phoneNumber) {
                this.phoneNumber = phoneNumber;
        }
        @Override
        public String toString() {
                return "Student [studentId=" + studentId + ", firstName=" + firstName + ", lastName=" + lastName
                                + ", dateOfBirth=" + dateOfBirth + ", email=" + email + ", phoneNumber=" +
phoneNumber + "]";
        }
}
```

Course Class:

Create a list or collection property to store the course's enrollments. This property will hold references to Enrollment objects.

Example: List<Enrollment> Enrollments { get; set; }

```
package entity;

import java.util.List;

public class Course {
   private int courseId;
   private String courseName;
   private String courseCode;
   private String instructorName;

   //collection relationship
   private List<Enrollment> enrollment;

   //getter for this
   public List<Enrollment> getEnrollments() {
      return enrollment;
   }

	public Course() {
		super();
		// TODO Auto-generated constructor stub
	}


	public Course(int courseId, String courseName, String courseCode, String instructorName) {
		super();
		this.courseId = courseId;
		this.courseName = courseName;
		this.courseCode = courseCode;
		this.instructorName = instructorName;
	}


	public int getCourseId() {
		return courseId;
	}
	public void setCourseId(int courseId) {
		this.courseId = courseId;
	}
	public String getCourseName() {
		return courseName;
	}
	public void setCourseName(String courseName) {
		this.courseName = courseName;
	}
	public String getCourseCode() {
		return courseCode;
	}
	public void setCourseCode(String courseCode) {
		this.courseCode = courseCode;
	}
	public String getInstructorName() {
		return instructorName;
	}
```

```
	public void setInstructorName(String instructorName) {
		this.instructorName = instructorName;
	}
	@Override
	public String toString() {
		return "Course [courseId=" + courseId + ", courseName=" + courseName + ", courseCode=" +
courseCode
				+ ", instructorName=" + instructorName + "]";
	}

}
```

Enrollment Class:

Include properties to hold references to both the Student and Course objects.

Example: Student Student { get; set; } and Course Course { get; set; }

```
package entity;
import java.time.LocalDate;

public class Enrollment {
	private int enrollmentId;
   private int studentId;     // Reference to Student object
   private int courseId;       // Reference to Course object
   private LocalDate enrollmentDate;

   //collection relationship
   private Student student;
   private Course course;

   //getter for this

   public Student getStudent() {
      return student;
   }

   public Course getCourse() {
      return course;
   }

   //default constructor
	public Enrollment() {
		super();
	}

	public Enrollment(int enrollmentId, int studentId, int courseId, LocalDate enrollmentDate, Student student,
Course course) {
		super();
		this.enrollmentId = enrollmentId;
		this.studentId = studentId;
		this.courseId = courseId;
		this.enrollmentDate = enrollmentDate;
```

```
                this.student = student;  // setting the student for this enrollment
        this.course = course;  // setting the course for this enrollment
        }

        public int getEnrollmentId() {
                return enrollmentId;
        }

        public void setEnrollmentId(int enrollmentId) {
                this.enrollmentId = enrollmentId;
        }

        public int getStudentId() {
                return studentId;
        }

        public void setStudentId(int studentId) {
                this.studentId = studentId;
        }

        public int getCourseId() {
                return courseId;
        }

        public void setCourseId(int courseId) {
                this.courseId = courseId;
        }

        public LocalDate getEnrollmentDate() {
                return enrollmentDate;
        }

        public void setEnrollmentDate(LocalDate enrollmentDate) {
                this.enrollmentDate = enrollmentDate;
        }
        @Override
        public String toString() {
                return "Enrollment [enrollmentId=" + enrollmentId + ", studentId=" + studentId + ", courseId=" +
courseId
                                + ", enrollmentDate=" + enrollmentDate + "]";
        }
}
```

Teacher Class:

Create a list or collection property to store the teacher's assigned courses. This property will hold references to Course objects.

Example: List<Course> AssignedCourses { get; set; }

```
package entity;

import java.util.ArrayList;
```

```
import java.util.List;

public class Teacher {
   private int teacherId;
   private String firstName;
   private String lastName;
   private String email;


   //collection relationship
   private List<Course> assignedCourses;

   //getter for this
   public List<Course> getAssignedCourses() {
      return assignedCourses;
   }

            public Teacher() {
                     super();
                     // TODO Auto-generated constructor stub
            }
            public Teacher(int teacherId, String firstName, String lastName, String email) {
                     super();
                     this.teacherId = teacherId;
                     this.firstName = firstName;
                     this.lastName = lastName;
                     this.email = email;
                     this.assignedCourses = new ArrayList<>();
            }
            public int getTeacherId() {
                     return teacherId;
            }
            public void setTeacherId(int teacherId) {
                     this.teacherId = teacherId;
            }
            public String getFirstName() {
                     return firstName;
            }
            public void setFirstName(String firstName) {
                     this.firstName = firstName;
            }
            public String getLastName() {
                     return lastName;
            }
            public void setLastName(String lastName) {
                     this.lastName = lastName;
            }
            public String getEmail() {
                     return email;
            }
            public void setEmail(String email) {
                     this.email = email;
            }
            @Override
            public String toString() {
```

```
                return "Teacher [teacherId=" + teacherId + ", firstName=" + firstName + ", lastName=" + lastName + ",
email="
                                + email + "]";
        }
}
```

Payment Class:

Include a property to hold a reference to the Student object.

Example: Student Student { get; set; }

```
package entity;
import java.time.LocalDate;

public class Payment {
        private int paymentId;
   private int studentId;  // Reference to Student object
   private double amount;
   private LocalDate paymentDate;


   //collection relationship
   private Student student;

   //getter for this

        public Student getStudent() {
           return student;
        }

   //default constructor
        public Payment() {
                super();
        }

        public Payment(int paymentId, int studentId, double amount, LocalDate paymentDate,Student student) {
                super();
                this.paymentId = paymentId;
                this.studentId = studentId;
                this.amount = amount;
                this.paymentDate = paymentDate;
                this.student = student;
        }

        public int getPaymentId() {
                return paymentId;
        }

        public void setPaymentId(int paymentId) {
                this.paymentId = paymentId;
        }

        public int getStudentId() {
```

```
		return studentId;
	}

	public void setStudentId(int studentId) {
		this.studentId = studentId;
	}

	public double getAmount() {
		return amount;
	}

	public void setAmount(double amount) {
		this.amount = amount;
	}

	public LocalDate getPaymentDate() {
		return paymentDate;
	}

	public void setPaymentDate(LocalDate paymentDate) {
		this.paymentDate = paymentDate;
	}

	@Override
	public String toString() {
		return "Payment [paymentId=" + paymentId + ", studentId=" + studentId + ", amount=" + amount + ",
paymentDate="
				+ paymentDate + "]";
	}
}
```

Task 6: Create Methods for Managing Relationships

To add, remove, or retrieve related objects, you should create methods within your SIS class or each relevant class.

• **AddEnrollment**(student, course, enrollmentDate): In the SIS class, create a method that adds an enrollment to both the Student's and Course's enrollment lists. Ensure the Enrollment object references the correct Student and Course.

• **AssignCourseToTeacher**(course, teacher): In the SIS class, create a method to assign a course to a teacher. Add the course to the teacher's AssignedCourses list.

• **AddPayment**(student, amount, paymentDate): In the SIS class, create a method that adds a payment to the Student's payment history. Ensure the Payment object references the correct Student.

• **GetEnrollmentsForStudent**(student): In the SIS class, create a method to retrieve all enrollments for a specific student.

• **GetCoursesForTeacher**(teacher): In the SIS class, create a method to retrieve all courses assigned to a specific teacher.

```
package dao;

import entity.Course;
import entity.Student;
import entity.Teacher;
import exception.DuplicateEnrollmentException;

import java.util.Date;

public interface SisService {
    void enrollStudentInCourse(Student student, Course course) throws DuplicateEnrollmentException;
    void assignTeacherToCourse(Teacher teacher, Course course);
    void recordPayment(Student student, double amount, Date paymentDate);
    void generateEnrollmentReport(Course course);
    void generatePaymentReport(Student student);
    void calculateCourseStatistics(Course course);
    }
```

Task 7: Database Connectivity

```
package util;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnUtil {

            private static final String fileName = "db.properties";
              public static Connection getDbConnection() {
                  Connection con = null;
                  String connString=null;
                  try {
                      connString = DBPropertyUtil.getConnectionString(fileName); // Get URL from properties
                  } catch (IOException e) {
                      System.out.println("Connection string could not be retrieved.");
                      e.printStackTrace();
                  }
                  if (connString != null) {
                       try {
                        con = DriverManager.getConnection(connString); // Get actual Connection object
                       }
                       catch (SQLException e) {
                      System.out.println("Database connection failed.");
                      e.printStackTrace();
                  }
                  }
                  return con;
              }
}
```

```
package util;

import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;

public class DBPropertyUtil {
	//this method takes the filename which contains db connection like
	//user name, pwd, port number, protocol and db name as an argument
	//and returns a connection
	public static String getConnectionString(String fileName)throws IOException {
		String connStr=null;
		Properties props=new Properties();
		FileInputStream fis=new FileInputStream(fileName);
		props.load(fis);

		String user = props.getProperty("user");
		String password = props.getProperty("password");
	String port = props.getProperty("port");
	String database = props.getProperty("database");
	String protocol = props.getProperty("protocol");
	String system = props.getProperty("system");

	connStr=protocol+"//"+system+":"+port+"/"+database+"?user="+user+"&password="+password;
		return connStr;
	}

}
```

Main Module code: -

```
package main;

import dao.SisService;
import dao.SisImpl;
import entity.Course;
import entity.Student;
import entity.Teacher;
import exception.DuplicateEnrollmentException;

import java.sql.Date;
import java.time.LocalDate;
import java.util.Scanner;

public class MainModule {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      SisService sis = new SisImpl(); // Service implementation

      String continueChoice;
      do {
         System.out.println("\n--- Welcome to the Student Information System ---");
```

```
System.out.println("1. Create student and Enroll Student in Course");
System.out.println("2. Add Course");
System.out.println("3. Add Teacher");
System.out.println("4. Assign Teacher to Course");
System.out.println("5. Record Payment by Student");
System.out.println("6. Generate Enrollment Report");
System.out.println("7. Exit");
System.out.print("Enter choice: ");
int choice = sc.nextInt();
sc.nextLine(); // Clear buffer

try {
   switch (choice) {
      case 1:
         // Task: Enroll Student in Course
         System.out.print("Enter Student First Name: ");
         String firstName = sc.nextLine();
         System.out.print("Enter Student Last Name: ");
         String lastName = sc.nextLine();
         System.out.print("Enter Student Date of Birth (yyyy-mm-dd): ");
         LocalDate dob = LocalDate.parse(sc.nextLine());
         System.out.print("Enter Student Email: ");
         String email = sc.nextLine();
         System.out.print("Enter Student Phone Number: ");
         String phone = sc.nextLine();

         Student student = new Student();
         student.setFirstName(firstName);
         student.setLastName(lastName);
         student.setDateOfBirth(dob);
         student.setEmail(email);
         student.setPhoneNumber(phone);

         // Task: Create Courses
         System.out.print("Enter Course ID to Enroll in: ");
         int courseId = sc.nextInt();
         sc.nextLine(); // Clear buffer
         Course course = new Course();
         course.setCourseId(courseId);  // Course ID should match DB

         try {
            sis.enrollStudentInCourse(student, course);
            System.out.println("done!! " + firstName + " " + lastName + " enrolled in course " + courseId);
         } catch (DuplicateEnrollmentException e) {
            System.out.println(" ⚠️  " + e.getMessage());
         }
         break;


      case 2:
         // Add Course
         System.out.print("Enter Course Name: ");
         String cName = sc.nextLine();
         System.out.print("Enter Course Code: ");
         String cCode = sc.nextLine();
```

```
                Course newCourse = new Course();
                newCourse.setCourseName(cName);
                newCourse.setCourseCode(cCode);

                sis.addCourse(newCourse);
                System.out.println(" Course added successfully.");
                break;

            case 3:
                // Add Teacher
                System.out.print("Enter Teacher First Name: ");
                String tFirst = sc.nextLine();
                System.out.print("Enter Teacher Last Name: ");
                String tLast = sc.nextLine();
                System.out.print("Enter Email: ");
                String tEmail = sc.nextLine();

                Teacher newTeacher = new Teacher();
                newTeacher.setFirstName(tFirst);
                newTeacher.setLastName(tLast);
                newTeacher.setEmail(tEmail);

                sis.addTeacher(newTeacher);
                System.out.println(" Teacher added successfully.");
                break;

            case 4:
                // Task: Assign Teacher to Course
                System.out.print("Enter Teacher First Name: ");
                String teacherFirstName = sc.nextLine();
                System.out.print("Enter Teacher Last Name: ");
                String teacherLastName = sc.nextLine();


                // Create Teacher Object
                Teacher teacher = new Teacher();
                teacher.setFirstName(teacherFirstName);
                teacher.setLastName(teacherLastName);


                System.out.print("Enter Course Code to Assign Teacher: ");
                String courseCode = sc.nextLine();

                Course assignCourse = new Course();
                assignCourse.setCourseCode(courseCode);

                sis.assignTeacherToCourse(teacher, assignCourse);
                System.out.println("yeah!! " + teacherFirstName + " " + teacherLastName + " assigned to course " +
courseCode);
                break;

            case 5:
                // Task: Record Payment by Student
                System.out.print("Enter Student ID: ");
                int studentId = sc.nextInt();
                sc.nextLine(); // Clear buffer
```

```
                    System.out.print("Enter Payment Amount: ");
                    double amount = sc.nextDouble();
                    sc.nextLine(); // Clear buffer
                    System.out.print("Enter Payment Date (yyyy-mm-dd): ");
                    Date paymentDate = Date.valueOf(sc.nextLine());

                    Student payingStudent = new Student();
                    payingStudent.setStudentId(studentId);
                    sis.recordPayment(payingStudent, amount, paymentDate);
                    System.out.println(" Payment of ₹" + amount + " recorded for student ID " + studentId + " on " +
paymentDate);
                    break;

                case 6:
                    // Task: Generate Enrollment Report

                    System.out.print("Enter Course ID for Report: ");
                    int reportCourseId = Integer.parseInt(sc.nextLine());
                    Course reportCourse = new Course();
                    reportCourse.setCourseId(reportCourseId);
                    sis.generateEnrollmentReport(reportCourse);

                    break;


                case 7:
                    // Exit
                    System.out.println("Exiting... Goodbye!");
                    return;

                default:
                    System.out.println(" ⚠️ Invalid choice.");
                }
            } catch (Exception e) {
                System.out.println("Unexpected Error: " + e.getMessage());
                e.printStackTrace();
            }

            // Continue or exit loop
            System.out.print("Do you want to continue (yes/no)? ");
            continueChoice = sc.nextLine();

        } while (continueChoice.equalsIgnoreCase("yes"));

        sc.close();
        System.out.println("Byebye.....");
    }
}
```

Task 8: Student Enrollment

In this task, a new student, John Doe, is enrolling in the SIS. The system needs to record John's information, including his personal details, and enroll him in a few courses. Database connectivity is required to store this information.

John Doe's details:

- First Name: John
- Last Name: Doe
- Date of Birth: 1995-08-15
- Email: john.doe@example.com
- Phone Number: 123-456-7890

John is enrolling in the following courses:

- Course 1: Introduction to Programming

The system should perform the following tasks:

- Create a new student record in the database.
- Enroll John in the specified courses by creating enrollment records in the database.

```
--- Welcome to the Student Information System ---
1. Create student and Enroll Student in Course
2. Add Course
3. Add Teacher
4. Assign Teacher to Course
5. Record Payment by Student
6. Generate Enrollment Report
7. Exit
Enter choice: 1
Enter Student First Name: John
Enter Student Last Name: Doe
Enter Student Date of Birth (yyyy-mm-dd): 1995-08-15
Enter Student Email: doe@example.com
Enter Student Phone Number: 1234567890
Enter Course ID to Enroll in: 1
done!! John Doe enrolled in course 1
Do you want to continue (yes/no)?
```

Task 9: Teacher Assignment

In this task, a new teacher, Sarah Smith, is assigned to teach a course. The system needs to update the course record to reflect the teacher assignment.

Teacher's Details:

- Name: Sarah Smith
- Email: sarah.smith@example.com

• Expertise: Computer Science

```
--- Welcome to the Student Information System ---
1. Create student and Enroll Student in Course
2. Add Course
3. Add Teacher
4. Assign Teacher to Course
5. Record Payment by Student
6. Generate Enrollment Report
7. Exit
Enter choice: 4
Enter Teacher First Name: Sarah
Enter Teacher Last Name: Smith
Enter Course Code to Assign Teacher: 3
☑ Sarah Smith assigned to course 3
Do you want to continue (yes/no)?
```

Course to be assigned:

• Course Name: Advanced Database Management
• Course Code: CS302

```
--- Welcome to the Student Information System ---
1. Create student and Enroll Student in Course
2. Add Course
3. Add Teacher
4. Assign Teacher to Course
5. Record Payment by Student
6. Generate Enrollment Report
7. Exit
Enter choice: 2
Enter Course Name: Advanced Database Management
Enter Course Code: CS302
☑ Course added successfully.
Do you want to continue (yes/no)?
```

The system should perform the following tasks:

• Retrieve the course record from the database based on the course code.
• Assign Sarah Smith as the instructor for the course.
• Update the course record in the database with the new instructor information.

```
--- Welcome to the Student Information System ---
1. Create student and Enroll Student in Course
2. Add Course
3. Add Teacher
4. Assign Teacher to Course
5. Record Payment by Student
6. Generate Enrollment Report
7. Exit
Enter choice: 3
Enter Teacher First Name: Sarah
Enter Teacher Last Name: Smith
Enter Email: sarah.smith@example.com
☑ Teacher added successfully.
Do you want to continue (yes/no)?
```

Task 10: Payment Record

In this task, a student, Jane Johnson, makes a payment for her enrolled courses. The system needs to record this payment in the database.

Jane Johnson's details:

• Student ID: 4
• Payment Amount: $500.00
• Payment Date: 2023-04-10

```
--- Welcome to the Student Information System ---
1. Create student and Enroll Student in Course
2. Add Course
3. Add Teacher
4. Assign Teacher to Course
5. Record Payment by Student
6. Generate Enrollment Report
7. Exit
Enter choice: 5
Enter Student ID: 4
Enter Payment Amount: 500
Enter Payment Date (yyyy-mm-dd): 2023-04-10
💰 Payment of ₹500.0 recorded for student ID 4 on 2023-04-10
Do you want to continue (yes/no)?
```

Task 11: Enrollment Report Generation

In this task, an administrator requests an enrollment report for a specific course, "Computer Science 101." The system needs to retrieve enrollment information from the database and generate a report.

Course to generate the report for:

• Course Name: Computer Science /The Course Id for this course is 1 so I use this to retrieve the data.

```
--- Welcome to the Student Information System ---
1. Create student and Enroll Student in Course
2. Add Course
3. Add Teacher
4. Assign Teacher to Course
5. Record Payment by Student
6. Generate Enrollment Report
7. Exit
Enter choice: 6
Enter Course ID for Report: 1
Enrollment Report for Course ID 1:
- Smrthi Shankar
- sita ram
- John Doe
Do you want to continue (yes/no)?
```