

ASSIGNMENT: STUDENT INFORMATION SYSTEM

SUBMITTED BY-SRINIDHI.V

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:

TASK 2: IMPLEMENT CONSTRUCTORS

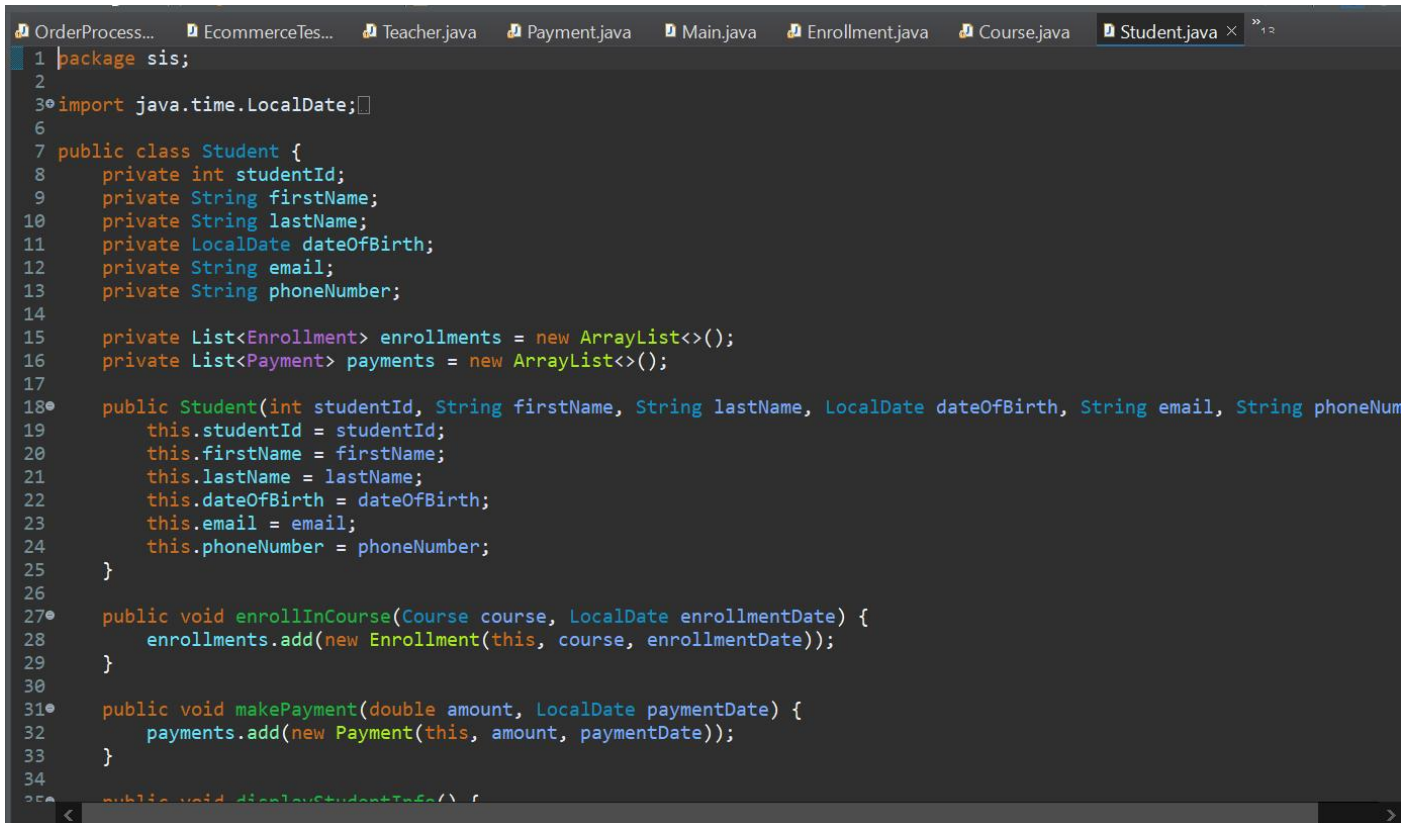
Implement constructors for each class to initialize their attributes. Constructors are special methods that are called when an object of a class is created. They are used to set initial values for the attributes of the class. Below are detailed instructions on how to implement constructors for each class 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 class with the following attributes:

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



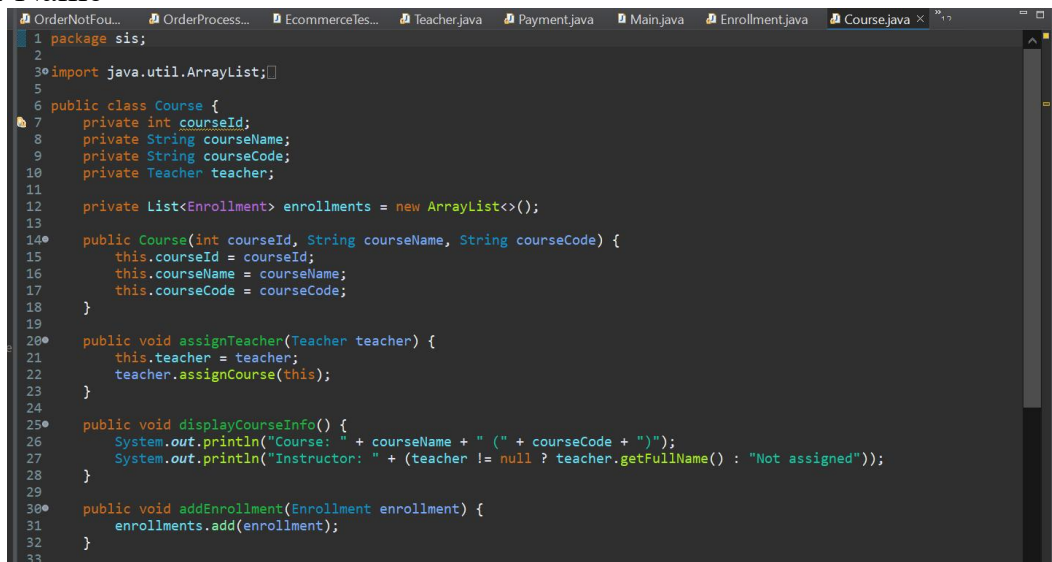
```

1 package sis;
2
3 import java.time.LocalDate;
4
5
6
7 public class Student {
8     private int studentId;
9     private String firstName;
10    private String lastName;
11    private LocalDate dateOfBirth;
12    private String email;
13    private String phoneNumber;
14
15    private List<Enrollment> enrollments = new ArrayList<>();
16    private List<Payment> payments = new ArrayList<>();
17
18    public Student(int studentId, String firstName, String lastName, LocalDate dateOfBirth, String email, String phoneNum
19        this.studentId = studentId;
20        this.firstName = firstName;
21        this.lastName = lastName;
22        this.dateOfBirth = dateOfBirth;
23        this.email = email;
24        this.phoneNumber = phoneNumber;
25    }
26
27    public void enrollInCourse(Course course, LocalDate enrollmentDate) {
28        enrollments.add(new Enrollment(this, course, enrollmentDate));
29    }
30
31    public void makePayment(double amount, LocalDate paymentDate) {
32        payments.add(new Payment(this, amount, paymentDate));
33    }
34
35    public void displayStudentInfo() {

```

Course class with the following attributes:

- Course ID
- Course Name
- Course Code
- Instructor Name



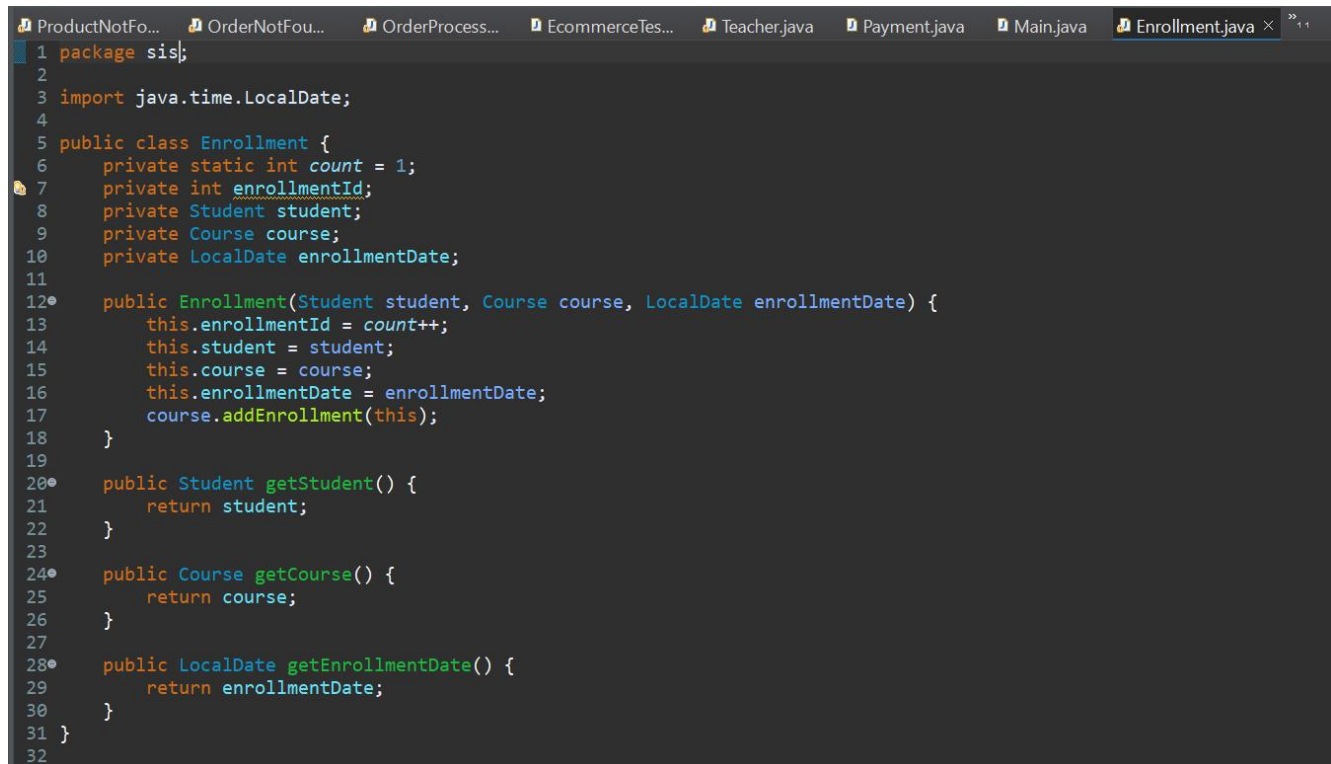
```

1 package sis;
2
3 import java.util.ArrayList;
4
5
6 public class Course {
7     private int courseId;
8     private String courseName;
9     private String courseCode;
10    private Teacher teacher;
11
12    private List<Enrollment> enrollments = new ArrayList<>();
13
14    public Course(int courseId, String courseName, String courseCode) {
15        this.courseId = courseId;
16        this.courseName = courseName;
17        this.courseCode = courseCode;
18    }
19
20    public void assignTeacher(Teacher teacher) {
21        this.teacher = teacher;
22        teacher.assignCourse(this);
23    }
24
25    public void displayCourseInfo() {
26        System.out.println("Course: " + courseName + " (" + courseCode + ")");
27        System.out.println("Instructor: " + (teacher != null ? teacher.getFullName() : "Not assigned"));
28    }
29
30    public void addEnrollment(Enrollment enrollment) {
31        enrollments.add(enrollment);
32    }
33

```

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

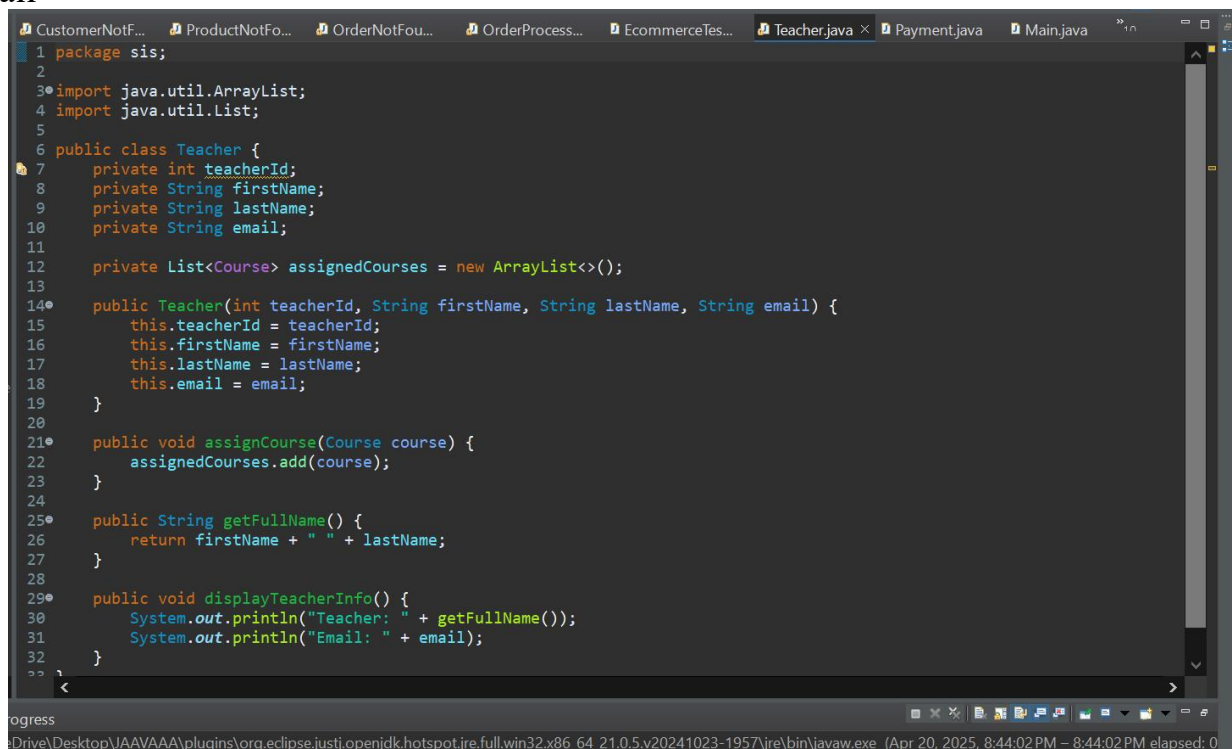
- Enrollment ID
- Student ID (reference to a Student)
- Course ID (reference to a Course)
- Enrollment Date



```
1 package sis;
2
3 import java.time.LocalDate;
4
5 public class Enrollment {
6     private static int count = 1;
7     private int enrollmentId;
8     private Student student;
9     private Course course;
10    private LocalDate enrollmentDate;
11
12    public Enrollment(Student student, Course course, LocalDate enrollmentDate) {
13        this.enrollmentId = count++;
14        this.student = student;
15        this.course = course;
16        this.enrollmentDate = enrollmentDate;
17        course.addEnrollment(this);
18    }
19
20    public Student getStudent() {
21        return student;
22    }
23
24    public Course getCourse() {
25        return course;
26    }
27
28    public LocalDate getEnrollmentDate() {
29        return enrollmentDate;
30    }
31 }
32
```

Teacher class with the following attributes:

- Teacher ID
- First Name
- Last Name
- Email

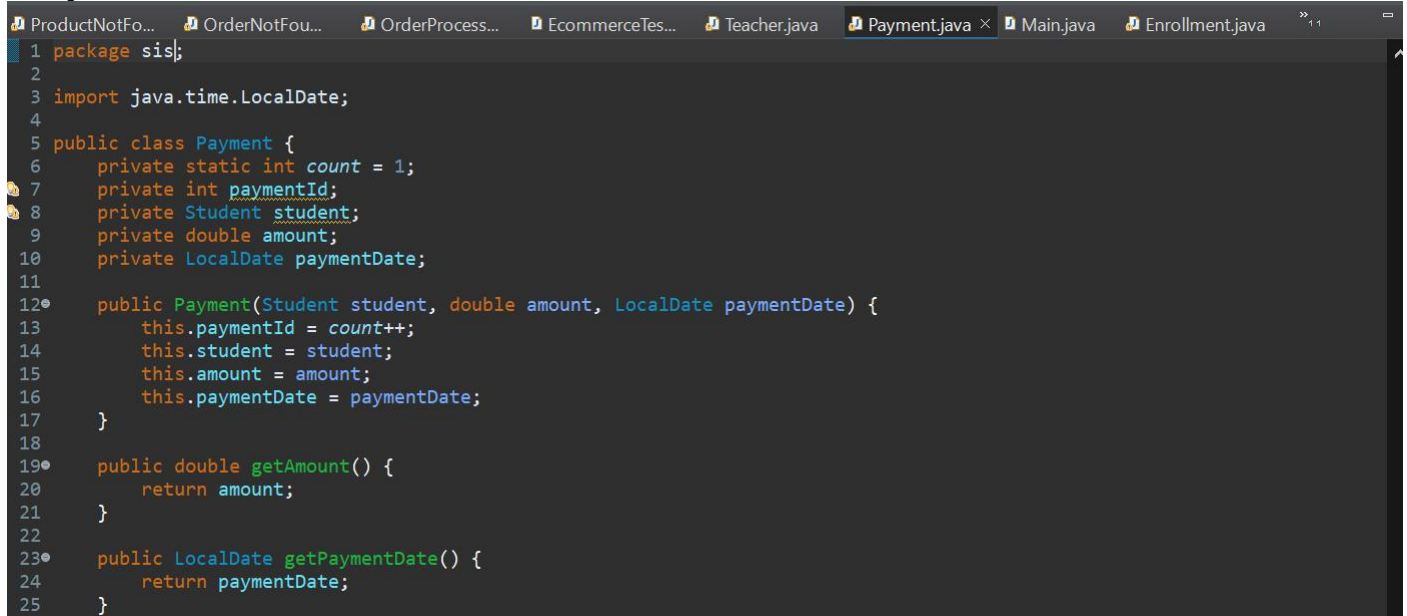


```
1 package sis;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class Teacher {
7     private int teacherId;
8     private String firstName;
9     private String lastName;
10    private String email;
11
12    private List<Course> assignedCourses = new ArrayList<>();
13
14    public Teacher(int teacherId, String firstName, String lastName, String email) {
15        this.teacherId = teacherId;
16        this.firstName = firstName;
17        this.lastName = lastName;
18        this.email = email;
19    }
20
21    public void assignCourse(Course course) {
22        assignedCourses.add(course);
23    }
24
25    public String getFullName() {
26        return firstName + " " + lastName;
27    }
28
29    public void displayTeacherInfo() {
30        System.out.println("Teacher: " + getFullName());
31        System.out.println("Email: " + email);
32    }
33 }
34
```

Progress
Drive\Desktop\JAAVAAA\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.21.0.5.v20241023-1957\jre\bin\javaw.exe (Apr 20, 2025, 8:44:02 PM – 8:44:02 PM elapsed: 0

Payment class with the following attributes:

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



```
1 package sis;
2
3 import java.time.LocalDate;
4
5 public class Payment {
6     private static int count = 1;
7     private int paymentId;
8     private Student student;
9     private double amount;
10    private LocalDate paymentDate;
11
12    public Payment(Student student, double amount, LocalDate paymentDate) {
13        this.paymentId = count++;
14        this.student = student;
15        this.amount = amount;
16        this.paymentDate = paymentDate;
17    }
18
19    public double getAmount() {
20        return amount;
21    }
22
23    public LocalDate getPaymentDate() {
24        return paymentDate;
25    }
26 }
```

TASK 3: : IMPLEMENT METHODS

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

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.

```
Student Information
Student ID: 1
First Name: John
Last Name: Doe
Date of Birth: 2000-01-01
Email: john.doe@example.com
Phone Number: 123-456-7890
Courses Enrolled:
1. Computer Science
Payments Made:
Amount: 500.0 on 2025-04-20
```

Enrollment Class:

- GetStudent(): Retrieves the student associated with the enrollment.
- GetCourse(): Retrieves the course associated with the enrollment.

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.

Payment Class:

- GetStudent(): Retrieves the student associated with the payment.
- GetPaymentAmount(): Retrieves the payment amount.
- GetPaymentDate(): Retrieves the payment date.

```
Student Information
Student ID: 2
First Name: Jane
Last Name: Smith
Date of Birth: 2001-02-02
Email: jane.smith@example.com
Phone Number: 987-654-3210
Courses Enrolled:
1. Mathematics
Payments Made:
Amount: 300.0 on 2025-04-20
```


TASK 4:EXCEPTIONS HANDLING AND CUSTOM EXCEPTIONS

```
import java.util.*;
import java.time.LocalDate;

// Custom Exceptions
class DuplicateEnrollmentException extends Exception {
    public DuplicateEnrollmentException(String message) { super(message); }
}
class CourseNotFoundException extends Exception {
    public CourseNotFoundException(String message) { super(message); }
}
class StudentNotFoundException extends Exception {
    public StudentNotFoundException(String message) { super(message); }
}
class TeacherNotFoundException extends Exception {
    public TeacherNotFoundException(String message) { super(message); }
}
class PaymentValidationException extends Exception {
    public PaymentValidationException(String message) { super(message); }
}
class InvalidStudentDataException extends Exception {
    public InvalidStudentDataException(String message) { super(message); }
}
class InvalidCourseDataException extends Exception {
    public InvalidCourseDataException(String message) { super(message); }
}
class InvalidEnrollmentDataException extends Exception {
    public InvalidEnrollmentDataException(String message) { super(message); }
}
class InvalidTeacherDataException extends Exception {
    public InvalidTeacherDataException(String message) { super(message); }
}
class InsufficientFundsException extends Exception {
    public InsufficientFundsException(String message) { super(message); }
}
```

Task 4: Exceptions handling and Custom Exceptions

```
Enrollment report for course: Computer Science
Enrolled Students:
```

```
1. John Doe
```

```
Enrollment report for course: Mathematics
Enrolled Students:
```

```
1. Jane Smith
```

```
Payment report for student: John Doe
Payments:
```

```
Amount: 500.0 on 2025-04-20
```

```
Payment report for student: Jane Smith
Payments:
```

```
Amount: 300.0 on 2025-04-20
```

TASK 5:COLLECTIONS

```
Course: Computer Science
Enrolled Students: John Doe
Assigned Teacher: Dr. Emily Jones
```

```
Course: Mathematics
Enrolled Students: Jane Smith
Assigned Teacher: Prof. Mark Brown
```

```
Student Payment History:
Student: John Doe - Payments:
Amount: 500.0 on 2025-04-20
Student: Jane Smith - Payments:
Amount: 300.0 on 2025-04-20
```

```
Teachers and their Assigned Courses:
Teacher: Dr. Emily Jones - Courses: Computer Science
Teacher: Prof. Mark Brown - Courses: Mathematics
```

TASK 6: Create Methods for Managing Relationship

```
package sis;
```

```
import java.util.*;
```

```
public class SISManager {
    private List<Enrollment> enrollments = new ArrayList<>();
    private List<Payment> payments = new ArrayList<>();

    public void AddEnrollment(Student student, Course course, String enrollmentDate) {
        Enrollment enrollment = new Enrollment(student, course, enrollmentDate);
        student.getEnrollments().add(enrollment);
        course.getEnrollments().add(enrollment);
        enrollments.add(enrollment);
        System.out.println("Enrolled " + student.getFirstName() + " in " +
course.getCourseName());
    }

    public void AssignCourseToTeacher(Course course, Teacher teacher) {
        course.AssignTeacher(teacher);
        teacher.getAssignedCourses().add(course);
        System.out.println("Assigned " + teacher.getFirstName() + " to course " +
course.getCourseName());
    }
}
```

```

public void AddPayment(Student student, double amount, String paymentDate) {
    Payment payment = new Payment(student, amount, paymentDate);
    student.getPayments().add(payment);
    payments.add(payment);
    System.out.println("Payment of " + amount + " added for " + student.getFirstName());
}

public List<Enrollment> GetEnrollmentsForStudent(Student student) {
    return student.getEnrollments();
}

public List<Course> GetCoursesForTeacher(Teacher teacher) {
    return teacher.getAssignedCourses();
}
}






```

TASK 7: DATABASE CONNECTIVITY







student_id	first_name	last_name	dob	email	phone_number
1	John	Doe	1995-08-15	john.doe@example.com	123-456-7890
2	Jane	Johnson	1996-05-10	jane.johnson@example.com	987-654-3210
NULL	NULL	NULL	NULL	NULL	NULL

course_id	course_name	course_code
1	Introduction to Programming	CS101
2	Mathematics 101	MATH101
3	Advanced Database Management	CS302
NULL	NULL	NULL

enrollment_id	student_id	course_id
1	1	1
2	1	2
3	2	1
NULL	NULL	NULL

Result Grid				
Filter Rows: <input type="text"/>				
Edit:   				
Export/Import:  				
Wrap				
	teacher_id	name	email	expertise
▶	1	Sarah Smith	sarah.smith@example.com	Computer Science
	2	Mark Brown	mark.brown@example.com	Mathematics
*	NULL	NULL	NULL	NULL

Result Grid		
Filter Rows: <input type="text"/>		
Export: 		
Wrap Cell Content: 		
	course_id	teacher_id
▶	1	1
	2	2

Result Grid				
Filter Rows: <input type="text"/>				
Edit:   				
Export/Import:  				
Wrap Cell Content: 				
	payment_id	student_id	amount	payment_date
▶	1	2	500.00	2023-04-10
*	NULL	NULL	NULL	NULL

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
- Course 2: Mathematics 101

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.

```
Task 8: Student Enrollment
Student Details:
First Name: John
Last Name: Doe
Date of Birth: 1995-08-15
Email: john.doe@example.com
Phone Number: 123-456-7890
Enrolled Courses:
Course 1: Introduction to Programming
Course 2: Mathematics 101
Student record created and courses enrolled.
```

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

Course to be assigned:

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

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.

```
Task 9: Teacher Assignment
Teacher Details:
Name: Sarah Smith
Email: sarah.smith@example.com
Expertise: Computer Science
Assigned Course:
Course Name: Advanced Database Management
Course Code: CS302
Teacher assigned to the course.
```

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: 101
- Payment Amount: \$500.00
- Payment Date: 2023-04-10

The system should perform the following tasks:

- Retrieve Jane Johnson's student record from the database based on her student ID.
- Record the payment information in the database, associating it with Jane's student record.
- Update Jane's outstanding balance in the database based on the payment amount.

```
Task 10: Payment Record
Student Details:
Student ID: 101
Payment Amount: $500.00
Payment Date: 2023-04-10
Payment recorded and outstanding balance updated.
```

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 101

The system should perform the following tasks:

- Retrieve enrollment records from the database for the specified course.
- Generate an enrollment report listing all students enrolled in Computer Science 101.
- Display or save the report for the administrator.

```
Course: Computer Science 101
Generating enrollment report...
Report: List of students enrolled in Computer Science 101
Enrollment report generated for the administrator.
```