Course Registration System

Objective:
Design and implement a Course Registration System that allows students to register for courses, manage course offerings, and handle the interactions between students and courses using object-oriented principles, with a focus on classes and inheritance.

Problem Description:
The Course Registration System will manage the process of students registering for courses, track enrolled students for each course, and provide functionalities for course management. The system will be designed using object-oriented principles, emphasizing the use of classes and inheritance to represent different entities and behaviors within the system.

Key Components:

1. Base Class: `Person`
   - Attributes:
     - `name`: The name of the person.
     - `email`: The email address of the person.
     - `id`: A unique identifier for the person.
   - Methods:
     - `displayInfo()`: Displays the personal information of the person.
   - Description: The `Person` class serves as a base class representing any person in the system, such as students and instructors. It provides common attributes and methods that can be inherited by more specific classes.

2. Derived Classes:
   - `Student` Class (inherits from `Person`):
     - Additional Attributes:
       - `studentID`: A unique identifier for the student.
       - `enrolledCourses`: A list of courses the student is currently enrolled in.
     - Additional Methods:
       - `enrollInCourse(Course course)`: Enrolls the student in a given course.
       - `dropCourse(Course course)`: Drops the student from a given course.
       - `listEnrolledCourses()`: Lists all courses the student is currently enrolled in.
     - Description: The `Student` class extends `Person` and includes attributes and methods specific to students, such as enrolling in and dropping courses.

   - `Instructor` Class (inherits from `Person`):
     - Additional Attributes:
       - `instructorID`: A unique identifier for the instructor.
       - `teachingCourses`: A list of courses the instructor is teaching.
     - Additional Methods:
       - `assignCourse(Course course)`: Assigns the instructor to teach a given course.
       - `listTeachingCourses()`: Lists all courses the instructor is currently teaching.

- Description: The `Instructor` class extends `Person` and includes attributes and methods specific to instructors, such as assigning and listing courses they are teaching.

3. Course Class:
   - Attributes:
     - `courseName`: The name of the course.
     - `courseCode`: A unique code for the course (e.g., "CS101").
     - `instructor`: The instructor assigned to the course.
     - `enrolledStudents`: A list of students enrolled in the course.
     - `maxEnrollment`: The maximum number of students that can enroll in the course.
   - Methods:
     - `enrollStudent(Student student)`: Enrolls a student in the course, if there is space available.
     - `dropStudent(Student student)`: Removes a student from the course.
     - `listEnrolledStudents()`: Lists all students currently enrolled in the course.
     - `assignInstructor(Instructor instructor)`: Assigns an instructor to the course.
   - Description: The `Course` class manages the details of a course, including enrollment of students and assignment of instructors. It interacts with instances of the `Student` and `Instructor` classes.

4. Department Class:
   - Attributes:
     - `departmentName`: The name of the department (e.g., "Computer Science").
     - `coursesOffered`: A collection of courses offered by the department.
     - `instructors`: A collection of instructors in the department.
   - Methods:
     - `addCourse(Course course)`: Adds a new course to the department's offerings.
     - `removeCourse(Course course)`: Removes a course from the department's offerings.
     - `listCourses()`: Lists all courses currently offered by the department.
     - `assignInstructorToCourse(Instructor instructor, Course course)`: Assigns an instructor to a specific course.
   - Description: The `Department` class manages the courses and instructors within a department. It facilitates the addition and removal of courses and the assignment of instructors to courses.

5. Course Registration System Launcher:
   - Purpose: Implement a `CourseRegistrationLauncher` class with a `main()` method to simulate the course registration system. This class will:
     - Create instances of `Student`, `Instructor`, and `Course`.
     - Add courses to a department.
     - Assign instructors to courses.
     - Allow students to enroll in and drop courses.
     - Display the list of courses, enrolled students, and assigned instructors.

Example Usage:

- Create a `Student` and `Instructor` class using inheritance from the `Person` base class to leverage shared attributes and behaviors.
- Develop a `Course` class to manage the core functionality of course enrollment, student and instructor assignment, and course information display.
- Implement a `Department` class to handle the organization of courses and instructors within a department, demonstrating the relationships between classes.
- Simulate the course registration process using a launcher class to bring all the components together and facilitate user interactions.

This problem statement encourages the use of inheritance to manage different types of people within the system while promoting code reuse and modular design. The system design demonstrates how to organize and interact with various classes to build a robust course registration system in Java.