Project Title: Student Grading System

Overview

In this project, you will create a **Student Grading System** in Java. The application will enable users to add students and courses, enroll students in courses, record grades, and view all students, courses, and grades—all by interacting with a console-based menu. All self creating ids.

1. **Student Class**
   Purpose: Represents a student with personal details.

   Fields:

   - `String studentId:` *A unique identifier for each student.*
   - `String name:` *The student's full name.*
   - `String email:` *The student's email address.*

   Methods:

   - `getStudentId()`: *Access and modify the student ID.*
   - `getName()`, `setName(String name):` *Access and modify the student's name.*
   - `getEmail()`, `setEmail(String email):` *Access and modify the student's email.*
2. **Course Class**
   Purpose: Represents a course offered in the system.
   Fields:
   - `String courseId:` *A unique identifier for each course.*
   - `String courseName:` *The name of the course.*
   - `int creditHours:` *The number of credit hours the course is worth.*

   Methods:

   - `getCourseId()`, `setCourseId(String courseId)` *Access and modify the course ID.*
   - `getCourseName()`, `setCourseName(String courseName)` *Access and modify the course name.*

    - `getCreditHours()`, `setCreditHours(int creditHours)`
      Access and modify the credit hours.
3. **3. Grade Class**
   Purpose: Represents a grade assigned to a student for a specific course.
   Fields:
    - `Student student`
      The student who received the grade.
    - `Course course`
      The course for which the grade was given.
    - `double gradeValue`
      The numerical value of the grade (e.g., on a 100-point scale).
4. **Methods:**
    - `getStudent()`, `setStudent(Student student)`
      Access and modify the student associated with the grade.
    - `getCourse()`, `setCourse(Course course)`
      Access and modify the course associated with the grade.
    - `getGradeValue()`, `setGradeValue(double gradeValue)`
      Access and modify the grade value.
5. **4. GradingSystem Class**
   Purpose: Manages the overall system, including students, courses, and grades. Handles file operations for data persistence.
   Fields:
    - `List<Student> students`
      A list of all students in the system.
    - `List<Course> courses`
      A list of all courses available.
    - `List<Grade> grades`
      A list of all grades recorded.
6. **Methods:**
    - **Student Management:**
        - `addStudent(Student student)`
          Add a new student to the system.
        - `removeStudent(String studentId)`
          Remove a student from the system.
        - `getStudentById(String studentId)`
          Retrieve a student by their ID.
    - **Course Management:**

        - `addCourse(Course course)`
          Add a new course to the system.
        - `removeCourse(String courseId)`
          Remove a course from the system.
        - `getCourseById(String courseId)`
          Retrieve a course by its ID.
    - **Enrollment Management:**
        - `enrollStudentInCourse(String studentId, String courseId)`
          Enroll a student in a specific course.
    - **Grade Management:**
        - `addGrade(String studentId, String courseId, double gradeValue)`
          Record a grade for a student in a specific course.
    - **File Operations:**
        - `saveData(String filename)`
          Save all system data to a file.
        - `loadData(String filename)`
          Load system data from a file.

7. **Main Class**
   Purpose: Serves as the entry point of the application, facilitating user interaction through a console-based menu.
   Requirements:
    - **Initialize the Grading System:**
        - Create an instance of the `GradingSystem` class.
        - Load existing data from files if available.
    - **User Interface:**
        - Implement a simple text-based menu with options.
        - Use a loop to continuously display the menu until the user chooses to exit.
    - **Handle User Choices:**
        - **Add Student:**
          Prompt the user for student ID, name, and email. Create a new `Student` object and add it to the system.
        - **Add Course:**
          Prompt the user for course ID, name, and credit hours. Create a new `Course` object and add it to the system.

        - **Enroll Student in Course:**
          Prompt the user for student ID and course ID. Enroll the specified student in the specified course.
        - **Record Grade:**
          Prompt the user for student ID, course ID, and grade value. Record the grade for the student in the specified course.
        - **Show All Students:**
          Display a list of all students with their details.
        - **Show All Courses:**
          Display a list of all courses with their details.
        - **Show All Grades:**
          Display a list of all recorded grades.
        - **Exit:**
          Save all data to files to ensure persistence and terminate the application.
    - **Exit Procedure:**
        - Before exiting, ensure all current data is saved to files.
        - Provide a confirmation message indicating that data has been saved and the application is closing.

8. **Data Persistence**
   Purpose: Ensure that all data (students, courses, grades) is saved to files so that information is retained between sessions.