```java
import java.util.*;

/**
* Represents a student enrolled in the university.
*/
class Student {
   private String name;
   private String studentId;
   private Map<Course, Double> enrolledCourses; // Stores enrolled courses
and grades

   public Student(String name, String studentId) {
       this.name = name;
       this.studentId = studentId;
       this.enrolledCourses = new HashMap<>();
   }

   public String getName() {
       return name;
   }

   public String getStudentId() {
       return studentId;
   }

   public Map<Course, Double> getEnrolledCourses() {
       return Collections.unmodifiableMap(enrolledCourses);
   }

   /**
    * Enrolls a student in a given course.
    */
   public boolean enrollInCourse(Course course) {
       if (course.addStudent()) {
           enrolledCourses.put(course, null); // Initially, no grade
assigned
           return true;
       }
       return false;
   }
```

```
    /**
     * Assigns a grade to the student for a given course.
     */
    public boolean assignGrade(Course course, double grade) {
        if (enrolledCourses.containsKey(course)) {
            enrolledCourses.put(course, grade);
            return true;
        }
        return false;
    }

    /**
     * Calculates the overall GPA of the student.
     */
    public double calculateOverallGrade() {
        double total = 0;
        int count = 0;

        for (Double grade : enrolledCourses.values()) {
            if (grade != null) {
                total += grade;
                count++;
            }
        }
        return (count > 0) ? total / count : 0.0;
    }
}

/**
* Represents a university course.
*/
class Course {
    private String courseCode;
    private String name;
    private int maxCapacity;
    private int enrolledStudents;

    private static int totalEnrolledStudents = 0; // Track global student
enrollments
```

```
    public Course(String courseCode, String name, int maxCapacity) {
        this.courseCode = courseCode;
        this.name = name;
        this.maxCapacity = maxCapacity;
        this.enrolledStudents = 0;
    }

    public String getCourseCode() {
        return courseCode;
    }

    public String getName() {
        return name;
    }

    public int getMaxCapacity() {
        return maxCapacity;
    }

    public int getEnrolledStudents() {
        return enrolledStudents;
    }

    public static int getTotalEnrolledStudents() {
        return totalEnrolledStudents;
    }

    /**
     * Adds a student to the course if capacity allows.
     */
    public boolean addStudent() {
        if (enrolledStudents < maxCapacity) {
            enrolledStudents++;
            totalEnrolledStudents++;
            return true;
        }
        return false;
    }
}
```

```java
/**
* Manages courses, student enrollments, and grade assignments.
*/
class CourseManagement {
   private static List<Course> courses = new ArrayList<>();
   private static List<Student> students = new ArrayList<>();

   /**
    * Adds a new course to the system.
    */
   public static void addCourse(String courseCode, String name, int
maxCapacity) {
       courses.add(new Course(courseCode, name, maxCapacity));
   }

   /**
    * Registers a new student.
    */
   public static void registerStudent(String name, String studentId) {
       students.add(new Student(name, studentId));
   }

   /**
    * Enrolls a student in a course.
    */
   public static boolean enrollStudent(String studentId, String
courseCode) {
       Student student = findStudentById(studentId);
       Course course = findCourseByCode(courseCode);

       if (student != null && course != null) {
           return student.enrollInCourse(course);
       }
       return false;
   }

   /**
    * Assigns a grade to a student for a course.
    */
```

```java
    public static boolean assignGrade(String studentId, String courseCode,
double grade) {
        Student student = findStudentById(studentId);
        Course course = findCourseByCode(courseCode);

        if (student != null && course != null) {
            return student.assignGrade(course, grade);
        }
        return false;
    }

    /**
     * Retrieves a student's overall GPA.
     */
    public static double calculateOverallGrade(String studentId) {
        Student student = findStudentById(studentId);
        return (student != null) ? student.calculateOverallGrade() : 0.0;
    }

    private static Student findStudentById(String studentId) {
        return students.stream().filter(s ->
s.getStudentId().equals(studentId)).findFirst().orElse(null);
    }

    private static Course findCourseByCode(String courseCode) {
        return courses.stream().filter(c ->
c.getCourseCode().equals(courseCode)).findFirst().orElse(null);
    }
}

/**
* Command-line interface for the administrator.
*/
public class AdminInterface {
    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        while (true) {
            System.out.println("\n=== Course Enrollment & Grade Management
===");
```

```
            System.out.println("1. Add Course");
            System.out.println("2. Register Student");
            System.out.println("3. Enroll Student in Course");
            System.out.println("4. Assign Grade");
            System.out.println("5. Calculate Overall Grade");
            System.out.println("6. Exit");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    addCourse();
                    break;
                case 2:
                    registerStudent();
                    break;
                case 3:
                    enrollStudent();
                    break;
                case 4:
                    assignGrade();
                    break;
                case 5:
                    calculateOverallGrade();
                    break;
                case 6:
                    System.out.println("Exiting... Goodbye!");
                    return;
                default:
                    System.out.println("Invalid option. Try again.");
            }
        }
    }

    private static void addCourse() {
        System.out.print("Enter Course Code: ");
        String code = scanner.nextLine();
        System.out.print("Enter Course Name: ");
```

```
        String name = scanner.nextLine();
        System.out.print("Enter Maximum Capacity: ");
        int capacity = scanner.nextInt();
        CourseManagement.addCourse(code, name, capacity);
        System.out.println("Course added successfully!");
    }

    private static void registerStudent() {
        System.out.print("Enter Student Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Student ID: ");
        String id = scanner.nextLine();
        CourseManagement.registerStudent(name, id);
        System.out.println("Student registered successfully!");
    }

    private static void enrollStudent() {
        System.out.print("Enter Student ID: ");
        String id = scanner.nextLine();
        System.out.print("Enter Course Code: ");
        String code = scanner.nextLine();
        if (CourseManagement.enrollStudent(id, code)) {
            System.out.println("Student enrolled successfully!");
        } else {
            System.out.println("Enrollment failed. Course might be full or
invalid ID.");
        }
    }

    private static void assignGrade() {
        System.out.print("Enter Student ID: ");
        String id = scanner.nextLine();
        System.out.print("Enter Course Code: ");
        String code = scanner.nextLine();
        System.out.print("Enter Grade: ");
        double grade = scanner.nextDouble();
        if (CourseManagement.assignGrade(id, code, grade)) {
            System.out.println("Grade assigned successfully!");
        } else {
            System.out.println("Failed to assign grade.");
```

```
        }
    }

    private static void calculateOverallGrade() {
        System.out.print("Enter Student ID: ");
        String id = scanner.nextLine();
        System.out.println("Overall GPA: " +
CourseManagement.calculateOverallGrade(id));
    }
}
```

Code Overview

This Java program manages students, courses, enrollments, and grades using a **CLI (Command-Line Interface).**

---

1. Student Class

- Stores **name, student ID, and enrolled courses with grades.**
- Methods:
    - `enrollInCourse(course)`: Adds a course if allowed.
    - `assignGrade(course, grade)`: Updates grade for a course.
    - `calculateOverallGrade()`: Computes **GPA (average grade).**

---

2. Course Class

- Stores **course code, name, max capacity, and enrolled students.**
- Methods:
    - `addStudent()`: Enrolls a student **if capacity allows.**
    - `getTotalEnrolledStudents()`: Tracks total students in all courses.

---

3. CourseManagement Class

- Handles **course and student registration, enrollments, and grades.**
- Methods:
    - `addCourse(courseCode, name, capacity)`: Adds a new course.
    - `registerStudent(name, studentId)`: Registers a new student.
    - `enrollStudent(studentId, courseCode)`: Enrolls a student in a course.
    - `assignGrade(studentId, courseCode, grade)`: Assigns a grade.
    - `calculateOverallGrade(studentId)`: Returns **GPA.**

---

4. AdminInterface (Main Program)

- **CLI menu** for admin to:
    1. Add Course
    2. Register Student
    3. Enroll Student
    4. Assign Grade
    5. Calculate GPA
    6. Exit

```
└─$ java AdminInterface

=== Course Enrollment & Grade Management ===
1. Add Course
2. Register Student
3. Enroll Student in Course
4. Assign Grade
5. Calculate Overall Grade
6. Exit
Choose an option: 2
Enter Student Name: steve
Enter Student ID: 1
Student registered successfully!

=== Course Enrollment & Grade Management ===
1. Add Course
2. Register Student
3. Enroll Student in Course
4. Assign Grade
5. Calculate Overall Grade
6. Exit
Choose an option: 1
Enter Course Code: 1
Enter Course Name: cyber security
Enter Maximum Capacity: 50
Course added successfully!

=== Course Enrollment & Grade Management ===
1. Add Course
2. Register Student
3. Enroll Student in Course
4. Assign Grade
5. Calculate Overall Grade
6. Exit
Choose an option: 3
Enter Student ID: 1
Enter Course Code: 1
Student enrolled successfully!

=== Course Enrollment & Grade Management ===
1. Add Course
2. Register Student
3. Enroll Student in Course
4. Assign Grade
5. Calculate Overall Grade
6. Exit
Choose an option:
```