This document provides a detailed explanation of the Java code implementation for the Course Enrollment and Grade Management System. The system demonstrates object-oriented programming concepts such as encapsulation, use of instance and static methods, and access control through access modifiers.

## 1. Student Class

The Student class is responsible for storing individual student data and managing their enrollments and grades.  
- Private instance variables include: studentName, studentId, registeredCourses, and coursePerformance.  
- Public getter and setter methods are provided for accessing and updating the student's name and ID.  
- The method enrollInCourse() allows a student to enroll in a course.  
- The assignCourseGrade() method assigns a grade to a course the student is enrolled in.  
- coursePerformance is a HashMap used to associate a course with a corresponding grade.

## 2. Course Class

The Course class represents individual courses offered by the university.  
- Private instance variables include: courseCode, courseTitle, and courseLimit.  
- Public getter methods provide access to the course information.  
- A static variable totalEnrollment tracks the total number of students enrolled across all courses.  
- The enrollStudentInCourse() method ensures the course's capacity is not exceeded.  
- The getTotalEnrollment() static method returns the total enrollment across all courses.

## 3. CourseManagement Class

This class acts as a utility and coordinator for managing the course system.  
- It uses static lists and maps to manage available courses and student performance.  
- The addNewCourse() method allows administrators to add courses.  
- The registerStudentToCourse() method enrolls a student in a course.  
- The updateStudentGrade() method assigns a grade to a student for a course.  
- The computeOverallGrade() method calculates the average grade of a student based on their enrolled courses.

## 4. Admin Interface (Main Method)

The main method provides a command-line interface for interacting with the system.  
- It displays a menu with options to add courses, enroll students, assign grades, and calculate overall grades.  
- User input is taken via a Scanner object and used to invoke appropriate methods.  
- Input validation is performed, and appropriate messages are displayed in case of invalid input or errors.  
- Example: If a course is full, a message is displayed, and the student is not enrolled.

6. Conclusion

The system is designed with modularity and scalability in mind. By leveraging encapsulation, access control, and static features of Java, the system ensures clear separation of concerns, robust data management, and ease of future maintenance.