```
// Main.java
// This file demonstrates the core
functionalities of the Student Management
System.
// It assumes the model, dao, service, and
exception classes (Student, Course,
// Enrollment, StudentDAO, CourseDAO,
EnrollmentDAO, StudentService,
CourseService,
// EnrollmentService,
DataNotFoundException,
InvalidInputException, BusinessException)
// are present in their respective 'com.sms'
subpackages as defined previously.

package com.sms;

import com.sms.model.Student;
import com.sms.model.Course;
import com.sms.model.Enrollment;
import com.sms.service.StudentService;
import com.sms.service.CourseService;
import com.sms.service.EnrollmentService;
import
com.sms.exception.DataNotFoundExceptio
n;
import
com.sms.exception.InvalidInputException;
import
com.sms.exception.BusinessException;

import java.time.LocalDate;
import java.util.List;

public class Main {

    public static void main(String[] args) {
        // Initialize our service layer instances.
        // In a real application, these would
typically be injected by a framework
        // like Spring for better dependency
management.
```

```
        StudentService studentService = new
StudentService();
        CourseService courseService = new
CourseService();
        EnrollmentService enrollmentService
= new EnrollmentService();

        System.out.println("🚀 **Student
Management System - Core Feature
Demonstration** 🚀");

System.out.println("----------------------------------------
--------------------");

        // --- 1. Adding New Students ---
        System.out.println("\n--- Adding
Students ---");
        try {
            // Add valid students
            Student s1 = new Student("STD001",
"Aarav Sharma", LocalDate.of(2003, 7, 21),
"101 Maple Ave", "9876543210",
"aarav.s@example.com");
        Student s2 = new Student("STD002",
"Priya Singh", LocalDate.of(2002, 11, 5),
"202 Oak St", "8765432109",
"priya.s@example.com");
        Student s3 = new Student("STD003",
"Rahul Kumar", LocalDate.of(2004, 3, 10),
"303 Pine Rd", "7654321098",
"rahul.k@example.com");

        studentService.addStudent(s1);
        studentService.addStudent(s2);
        studentService.addStudent(s3);

        // Demonstrate **Error Handling**:
Attempt to add a student with an invalid
email format
        System.out.println("\nAttempting to
add student with an invalid email:");
        studentService.addStudent(new
Student("STD004", "Invalid Emailer",
LocalDate.of(2000, 1, 1), "Someplace",
"1234567890", "invalid-email"));
        } catch (InvalidInputException e) {
            System.err.println("❌ **Error during
student addition**: " + e.getMessage());
        } catch (Exception e) { // Catch any
other unexpected errors
            System.err.println("❌ **An
unexpected error occurred**: " +
e.getMessage());
        }

        // --- 2. Adding New Courses ---
        System.out.println("\n--- Adding
Courses ---");
        try {
            // Add valid courses
            Course c1 = new Course("CS101",
"Programming Fundamentals", "Learn Java
basics.", 4, "Computer Science");
```

```
            Course c2 = new Course("MA201",
"Linear Algebra", "Concepts of vector
spaces.", 3, "Mathematics");
            Course c3 = new Course("PH101",
"Classical Mechanics", "Study of motion
and forces.", 3, "Physics");

            courseService.addCourse(c1);
            courseService.addCourse(c2);
            courseService.addCourse(c3);

            // Demonstrate **Error Handling**:
Attempt to add a course with zero credit
hours
            System.out.println("\nAttempting to
add course with zero credit hours:");
            courseService.addCourse(new
Course("BUS001", "Business Ethics",
"Ethics in corporate world.", 0, "Business"));
        } catch (InvalidInputException e) {
            System.err.println("❌ **Error during
course addition**: " + e.getMessage());
        }

        // --- 3. Enrolling Students in Courses ---
        System.out.println("\n--- Enrolling
Students ---");
        try {
            // Perform valid enrollments

enrollmentService.enrollStudentInCourse("
STD001", "CS101");

enrollmentService.enrollStudentInCourse("
STD001", "MA201"); // Aarav takes two
courses

enrollmentService.enrollStudentInCourse("
STD002", "CS101"); // Priya takes CS101

enrollmentService.enrollStudentInCourse("
STD003", "PH101"); // Rahul takes PH101
```

```
            // Demonstrate **Error Handling**: 
Enroll student in a non-existent course
            System.out.println("\nAttempting to 
enroll in a non-existent course:");

enrollmentService.enrollStudentInCourse("
STD001", "NONEXST");

        } catch (DataNotFoundException | 
BusinessException e) {
            System.err.println("❌ **Error during 
enrollment**: " + e.getMessage());
        }

        // --- 4. Assigning Grades ---
        System.out.println("\n--- Assigning 
Grades ---");
        try {
            // Assign valid grades
```

```
enrollmentService.assignGrade("STD001",
"CS101", 88.5);

enrollmentService.assignGrade("STD002",
"CS101", 92.0);

enrollmentService.assignGrade("STD003",
"PH101", 75.0);

            // Demonstrate **Error Handling**:
Assign an invalid grade (out of range)
            System.out.println("\nAttempting to
assign an invalid grade (105.0):");

enrollmentService.assignGrade("STD001",
"MA201", 105.0);
        } catch (InvalidInputException |
DataNotFoundException e) {
            System.err.println("❌ **Error during
grade assignment**: " + e.getMessage());
        }
        // --- 5. Retrieving Information ---
        System.out.println("\n--- Retrieving 
Information ---");

        System.out.println("\n**All Registered 
Students:**");

studentService.getAllStudents().forEach(Sy
stem.out::println);

        System.out.println("\n**All Available 
Courses:**");

courseService.getAllCourses().forEach(Sys
tem.out::println);

        System.out.println("\n**Enrollments 
for STD001 (Aarav Sharma):**");
        try {
            List<Enrollment> aaravsEnrollments
```

```
=
enrollmentService.getEnrollmentsByStuden
tId("STD001");
            if (aaravsEnrollments.isEmpty()) {
                System.out.println("No
enrollments found for STD001.");
            } else {

aaravsEnrollments.forEach(System.out::pri
ntln);
            }
        } catch (InvalidInputException e) {
            System.err.println("❌ **Error
retrieving enrollments**: " +
e.getMessage());
        }


        // --- 6. Updating Student Information
---
        System.out.println("\n--- Updating
Student STD002 ---");
        try {
            Student priya =
studentService.getStudentById("STD002");
            priya.setAddress("555 New Street,
Cityville");

priya.setPhoneNumber("9991112222");

studentService.updateStudent(priya);
            System.out.println("Updated
Student STD002: " +
studentService.getStudentById("STD002"));

            // Demonstrate **Error Handling**:
Attempt to update a non-existent student
            System.out.println("\nAttempting to
update a non-existent student:");
            studentService.updateStudent(new
Student("STD999", "Ghost Student",
LocalDate.of(1990,1,1), "Nowhere", "0",
"ghost@none.com"));
        } catch (DataNotFoundException |
InvalidInputException e) {
            System.err.println("❌ **Error
updating student**: " + e.getMessage());
        }

        // --- 7. Dropping Students from
Courses ---
        System.out.println("\n--- Dropping
Student STD001 from MA201 ---");
        try {

enrollmentService.dropStudentFromCourse
("STD001", "MA201");
            System.out.println("Enrollments for
STD001 after drop:");

enrollmentService.getEnrollmentsByStuden
tId("STD001").forEach(System.out::println);
```

```
            // Demonstrate **Error Handling**:
Attempt to drop a non-existent enrollment
            System.out.println("\nAttempting to
drop a non-existent enrollment:");

enrollmentService.dropStudentFromCourse
("STD001", "NONEXST");
        } catch (DataNotFoundException |
InvalidInputException e) {
            System.err.println("❌ **Error
dropping enrollment**: " + e.getMessage());
        }

        // --- 8. Deleting a Student and a
Course ---
        System.out.println("\n--- Deleting
Student STD002 ---");
        try {
            // In a real system, deleting a
student would often require deleting their
enrollments first
```

```
            // to maintain referential integrity.
The service layer would orchestrate this.

studentService.deleteStudent("STD002");
            System.out.println("Students
remaining: " +
studentService.getAllStudents().size());
        } catch (DataNotFoundException e) {
            System.err.println("❌ **Error
deleting student**: " + e.getMessage());
        }

        System.out.println("\n--- Deleting
Course PH101 ---");
        try {
            // Similarly, deleting a course with
active enrollments would typically be
prevented or
            // require prior unenrollment.

courseService.deleteCourse("PH101");
```

```
            System.out.println("Courses
remaining: " +
courseService.getAllCourses().size());
        } catch (DataNotFoundException e) {
            System.err.println("❌ **Error
deleting course**: " + e.getMessage());
        }


System.out.println("\n-----------------------------------
-------------------------");
        System.out.println("✅
**Demonstration Complete!** You can see
the core functionalities and error handling
in action.");

System.out.println("---------------------------------------
---------------------");
    }
}
```