TASK 5

STUDENT COURSE REGISTRATION SYSTEM

1.Course Database: Store course information, including course code, title, description, capacity, and schedule.

2.Student Database: Store student information, including student ID, name, and registered courses.

3.Course Listing: Display available courses with details and available slots.

4.Student Registration: Allow students to register for courses from the available options.

5.Course Removal: Enable students to drop courses they have registered for.

PROGRAM:

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

// Course class to store course information
class Course {
    private String courseCode;
    private String title;
    private String description;
    private int capacity;
    private String schedule;
    private int enrolledStudents;

    public Course(String courseCode, String title, String description, int capacity, String schedule) {
        this.courseCode = courseCode;
```

```
    this.title = title;
    this.description = description;
    this.capacity = capacity;
    this.schedule = schedule;
    this.enrolledStudents = 0;
}

public String getCourseCode() {
    return courseCode;
}

public String getTitle() {
    return title;
}

public String getDescription() {
    return description;
}

public int getCapacity() {
    return capacity;
}

public String getSchedule() {
    return schedule;
}

public int getEnrolledStudents() {
    return enrolledStudents;
}
```

```
    public void enrollStudent() {
        enrolledStudents++;
    }

    public void dropStudent() {
        enrolledStudents--;
    }

    public int availableSlots() {
        return capacity - enrolledStudents;
    }

    @Override
    public String toString() {
        return "Course Code: " + courseCode + "\nTitle: " + title + "\nDescription: " + description
+
                "\nCapacity: " + capacity + "\nEnrolled Students: " + enrolledStudents +
"\nSchedule: " + schedule + "\n";
    }
}

// Student class to store student information
class Student {
    private int studentId;
    private String name;
    private List<Course> registeredCourses;

    public Student(int studentId, String name) {
        this.studentId = studentId;
        this.name = name;
        this.registeredCourses = new ArrayList<>();
```

```
    }

    public int getStudentId() {
        return studentId;
    }

    public String getName() {
        return name;
    }

    public List<Course> getRegisteredCourses() {
        return registeredCourses;
    }

    public void registerCourse(Course course) {
        if (course.availableSlots() > 0) {
            registeredCourses.add(course);
            course.enrollStudent();
            System.out.println("Student " + name + " registered successfully for course " +
course.getCourseCode());
        } else {
            System.out.println("Course " + course.getCourseCode() + " is full. Registration
failed.");
        }
    }

    public void dropCourse(Course course) {
        if (registeredCourses.remove(course)) {
            course.dropStudent();
            System.out.println("Student " + name + " dropped course " + course.getCourseCode()
+ " successfully.");
        } else {
```

```
            System.out.println("Student " + name + " is not registered for course " +
course.getCourseCode() + ".");
        }
    }

    @Override
    public String toString() {
        return "Student ID: " + studentId + "\nName: " + name + "\nRegistered Courses: " +
registeredCourses.size() + "\n";
    }
}

// Main class to manage the course and student databases
public class CourseRegistrationSystem {
    private static List<Course> courses = new ArrayList<>();
    private static List<Student> students = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        initializeCourses();

        boolean exit = false;
        while (!exit) {
            System.out.println("\n--- Course Registration System Menu ---");
            System.out.println("1. View available courses");
            System.out.println("2. Register a student for a course");
            System.out.println("3. Drop a course for a student");
            System.out.println("4. View student details");
            System.out.println("5. Exit");

            System.out.print("Enter your choice: ");
```

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

        switch (choice) {
            case 1:
                displayCourses();
                break;
            case 2:
                registerCourse();
                break;
            case 3:
                dropCourse();
                break;
            case 4:
                viewStudentDetails();
                break;
            case 5:
                exit = true;
                break;
            default:
                System.out.println("Invalid choice. Please enter a number between 1 and 5.");
        }
    }

    System.out.println("Exiting the Course Registration System. Goodbye!");
    scanner.close();
}

private static void initializeCourses() {
    // Initialize some sample courses
```

```
        courses.add(new Course("CS101", "Introduction to Computer Science", "Basic concepts of programming", 50, "Mon/Wed/Fri 10:00-11:30"));
        courses.add(new Course("ENG201", "English Composition", "Writing skills and composition", 40, "Tue/Thu 13:00-14:30"));
        courses.add(new Course("MAT301", "Advanced Mathematics", "Calculus and algebra", 60, "Mon/Wed 14:00-15:30"));
    }

    private static void displayCourses() {
        System.out.println("\n--- Available Courses ---");
        for (Course course : courses) {
            System.out.println(course.toString());
        }
    }

    private static void registerCourse() {
        System.out.print("Enter student ID: ");
        int studentId = scanner.nextInt();
        scanner.nextLine();  // Consume newline character

        System.out.print("Enter student name: ");
        String studentName = scanner.nextLine();

        Student student = findOrCreateStudent(studentId, studentName);

        System.out.print("Enter course code to register: ");
        String courseCode = scanner.nextLine();

        Course course = findCourse(courseCode);
        if (course != null) {
            student.registerCourse(course);
        } else {
```

```
            System.out.println("Course with code " + courseCode + " not found.");
        }
    }

    private static void dropCourse() {
        System.out.print("Enter student ID: ");
        int studentId = scanner.nextInt();
        scanner.nextLine();  // Consume newline character

        Student student = findStudent(studentId);
        if (student != null) {
            System.out.print("Enter course code to drop: ");
            String courseCode = scanner.nextLine();

            Course course = findCourse(courseCode);
            if (course != null) {
                student.dropCourse(course);
            } else {
                System.out.println("Course with code " + courseCode + " not found.");
            }
        } else {
            System.out.println("Student with ID " + studentId + " not found.");
        }
    }

    private static void viewStudentDetails() {
        System.out.print("Enter student ID: ");
        int studentId = scanner.nextInt();
        scanner.nextLine();  // Consume newline character

        Student student = findStudent(studentId);
```

```
    if (student != null) {
        System.out.println("\nStudent Details:\n" + student.toString());
        System.out.println("Registered Courses:");
        for (Course course : student.getRegisteredCourses()) {
            System.out.println(course.toString());
        }
    } else {
        System.out.println("Student with ID " + studentId + " not found.");
    }
}

private static Student findOrCreateStudent(int studentId, String studentName) {
    Student student = findStudent(studentId);
    if (student == null) {
        student = new Student(studentId, studentName);
        students.add(student);
    }
    return student;
}

private static Student findStudent(int studentId) {
    for (Student student : students) {
        if (student.getStudentId() == studentId) {
            return student;
        }
    }
    return null;
}

private static Course findCourse(String courseCode) {
    for (Course course : courses) {
```

```
            if (course.getCourseCode().equalsIgnoreCase(courseCode)) {
                return course;
            }
        }
        return null;
    }
}
```

OUTPUT:

```
--- Course Registration System Menu ---
1. View available courses
2. Register a student for a course
3. Drop a course for a student
4. View student details
5. Exit
Enter your choice: 1

--- Available Courses ---
Course Code: CS101
Title: Introduction to Computer Science
Description: Basic concepts of programming
Capacity: 50
Enrolled Students: 0
Schedule: Mon/Wed/Fri 10:00-11:30

Course Code: ENG201
Title: English Composition
Description: Writing skills and composition
```

Capacity: 40

Enrolled Students: 0

Schedule: Tue/Thu 13:00-14:30

Course Code: MAT301

Title: Advanced Mathematics

Description: Calculus and algebra

Capacity: 60

Enrolled Students: 0

Schedule: Mon/Wed 14:00-15:30

--- Course Registration System Menu ---

1. View available courses

2. Register a student for a course

3. Drop a course for a student

4. View student details

5. Exit

Enter your choice: 2

Enter student ID: 1

Enter student name: ABC

Enter course code to register: CS101

Student ABC registered successfully for course CS101

--- Course Registration System Menu ---

1. View available courses

2. Register a student for a course

3. Drop a course for a student

4. View student details

5. Exit

Enter your choice: 3

Enter student ID: 2

Student with ID 2 not found.

--- Course Registration System Menu ---

1. View available courses

2. Register a student for a course

3. Drop a course for a student

4. View student details

5. Exit

Enter your choice: 4

Enter student ID: 1

Student Details:

Student ID: 1

Name: ABC

Registered Courses: 1

Registered Courses:

Course Code: CS101

Description: Basic concepts of programming

Capacity: 50

Enrolled Students: 1

Schedule: Mon/Wed/Fri 10:00-11:30

Title: Introduction to Computer Science

--- Course Registration System Menu ---

1. View available courses

2. Register a student for a course

3. Drop a course for a student

4. View student details

5. Exit

Enter your choice: 5

Exiting the Course Registration System. Goodbye!

=== Code Execution Successful ===