Student Record Management System POC

### **Overview:**

This is a **Java console-based application** that allows users to perform basic **CRUD operations** (Create, Read, Update, Delete) on student records.

It serves as a foundational project to demonstrate:

* Object-Oriented Programming concepts
* Java Collection Framework (ArrayList)
* Basic user interaction via the console

### **Key Features:**

1. **Add Student**: Input and store new student details (ID, Name, Age, Course).
2. **View Students**: Display all existing student records in a formatted way.
3. **Update Student**: Modify an existing student’s details using their ID.
4. **Delete Student**: Remove a student record based on the given ID.
5. **Exit**: Close the application safely.

### **Technologies Used:**

* Java SE
* Core OOP Principles (Encapsulation, Abstraction)
* Scanner for CLI input
* ArrayList for dynamic data storage

**File Structure:**

StudentRecordManagementSystem/ [ Package ]

├── Student.java [ Class ]

└── StudentManagementApp.java [ Class ]

### **Code Summary:**

#### **Student.java**

Defines the Student class with:

* Properties: id, name, age, course
* Getters & Setters
* toString() for clean console output

#### **StudentManagementApp.java**

Main class with logic for:

* Menu display
* Handling user input
* Performing CRUD operations on ArrayList<Student>

## **Functionality**

### 

## **Menu display ( Taking user choice )**

do {

System.out.println("\n=== Student Management System ===");

System.out.println("1. Add Student");

System.out.println("2. View Students");

System.out.println("3. Update Student");

System.out.println("4. Delete Student");

System.out.println("5. Exit");

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

choice = scanner.nextInt();

scanner.nextLine(); // consume newline

switch (choice) {

case 1 -> addStudent();

case 2 -> viewStudents();

case 3 -> updateStudent();

case 4 -> deleteStudent();

case 5 -> System.out.println("Exiting...");

default -> System.out.println("Invalid choice! Try again.");

}

} while (choice != 5);

}

## **2. Add Student**

private static void addStudent() {

System.out.print("Enter ID: ");

int id = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter Name: ");

String name = scanner.nextLine();

System.out.print("Enter Age: ");

int age = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter Course: ");

String course = scanner.nextLine();

students.add(new Student(id, name, age, course));

System.out.println("Student added successfully.");

}

### **Method: addStudent()**

### **Console Interaction:**

=== Student Management System ===

1. Add Student

2. View Students

3. Update Student

4. Delete Student

5. Exit

Enter choice: 1

Enter ID: 101

Enter Name: Alice

Enter Age: 20

Enter Course: Java

Student added successfully.

## **2. View Students**

private static void viewStudents() {

if (students.isEmpty()) {

System.out.println("No students found.");

return;

}

System.out.println("\n--- Student List ---");

for (Student s : students) {

System.out.println(s);

}

}

### **Method: viewStudents()**

### **Console Interaction:**

=== Student Management System ===

Enter choice: 2

--- Student List ---

ID: 101 | Name: Alice | Age: 20 | Course: Java

### **Internal Flow:**

* Check if students list is empty
* If not, loop through list and print toString() of each Student

## **3. Update Student**

private static void updateStudent() {

System.out.print("Enter student ID to update: ");

int id = scanner.nextInt();

scanner.nextLine();

for (Student s : students) {

if (s.getId() == id) {

System.out.print("Enter new name: ");

s.setName(scanner.nextLine());

System.out.print("Enter new age: ");

s.setAge(scanner.nextInt());

scanner.nextLine();

System.out.print("Enter new course: ");

s.setCourse(scanner.nextLine());

System.out.println("Student updated successfully.");

return;

}

}

System.out.println("Student not found.");

}

### **Method: updateStudent()**

### **Console Interaction:**

=== Student Management System ===

Enter choice: 3

Enter student ID to update: 101

Enter new name: Alice Johnson

Enter new age: 21

Enter new course: Advanced Java

Student updated successfully.

### **Internal Flow:**

* Loop through students list
* Match ID
* Use setName(), setAge(), and setCourse() to update values

## **4. Delete Student**

private static void deleteStudent() {

System.out.print("Enter student ID to delete: ");

int id = scanner.nextInt();

scanner.nextLine();

for (Student s : students) {

if (s.getId() == id) {

students.remove(s);

System.out.println("Student deleted successfully.");

return;

}

}

System.out.println("Student not found.");

}

### **Method: deleteStudent()**

### **Console Interaction:**

=== Student Management System ===

Enter choice: 4

Enter student ID to delete: 101

Student deleted successfully.

### **Internal Flow:**

* Loop through students list
* Match ID
* Call students.remove(student)
* Break the loop

## **✅ 5. Exit**

### **Console Interaction:**

=== Student Management System ===

Enter choice: 5

Exiting...

### **Internal Flow:**

* Exits loop and ends program