**Problem:**

We will simulate a student handbook system where we can:

1. Add a new student to the handbook.
2. Remove a student from the handbook.
3. Display the list of students.
4. Search for a student by their ID.
5. Sort students by their name.
6. Count the number of students in the handbook.

**1. Create the Student Class**

We'll define a Student class that holds information about the student such as name, student ID, and age.

public class Student {

private String name;

private String studentId;

private int age;

// Constructor

public Student(String name, String studentId, int age) {

this.name = name;

this.studentId = studentId;

this.age = age;

}

// Getters and Setters

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getStudentId() {

return studentId;

}

public void setStudentId(String studentId) {

this.studentId = studentId;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

@Override

public String toString() {

return "Student [ID=" + studentId + ", Name=" + name + ", Age=" + age + "]";

}

}

**2. Create the StudentHandbook Class**

We'll create a class called StudentHandbook that manages an ArrayList of Student objects. This class will have methods to add, remove, display, search, sort, and count students.

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

public class StudentHandbook {

private ArrayList<Student> students;

// Constructor

public StudentHandbook() {

students = new ArrayList<>();

}

// Add a student

public void addStudent(Student student) {

students.add(student);

System.out.println("Student added: " + student);

}

// Remove a student by ID

public void removeStudent(String studentId) {

Student studentToRemove = null;

for (Student student : students) {

if (student.getStudentId().equals(studentId)) {

studentToRemove = student;

break;

}

}

if (studentToRemove != null) {

students.remove(studentToRemove);

System.out.println("Student removed: " + studentToRemove);

} else {

System.out.println("Student with ID " + studentId + " not found.");

}

}

// Display all students

public void displayStudents() {

if (students.isEmpty()) {

System.out.println("No students in the handbook.");

} else {

System.out.println("List of Students:");

for (Student student : students) {

System.out.println(student);

}

}

}

// Search for a student by ID

public void searchStudentById(String studentId) {

for (Student student : students) {

if (student.getStudentId().equals(studentId)) {

System.out.println("Student found: " + student);

return;

}

}

System.out.println("Student with ID " + studentId + " not found.");

}

// Sort students by name

public void sortStudentsByName() {

Collections.sort(students, Comparator.comparing(Student::getName));

System.out.println("Students sorted by name.");

}

// Count the number of students

public void countStudents() {

System.out.println("Total number of students: " + students.size());

}

}

**3. Create the Main Class**

The Main class will serve as the entry point to interact with the StudentHandbook and test the functionality.

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

StudentHandbook handbook = new StudentHandbook();

while (true) {

System.out.println("\nStudent Handbook Menu:");

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

System.out.println("2. Remove Student");

System.out.println("3. Display All Students");

System.out.println("4. Search Student by ID");

System.out.println("5. Sort Students by Name");

System.out.println("6. Count Students");

System.out.println("7. Exit");

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

int choice = scanner.nextInt();

scanner.nextLine(); // Consume the newline character

switch (choice) {

case 1: // Add Student

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

String name = scanner.nextLine();

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

String studentId = scanner.nextLine();

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

int age = scanner.nextInt();

scanner.nextLine(); // Consume the newline character

Student student = new Student(name, studentId, age);

handbook.addStudent(student);

break;

case 2: // Remove Student

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

String removeId = scanner.nextLine();

handbook.removeStudent(removeId);

break;

case 3: // Display All Students

handbook.displayStudents();

break;

case 4: // Search Student by ID

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

String searchId = scanner.nextLine();

handbook.searchStudentById(searchId);

break;

case 5: // Sort Students by Name

handbook.sortStudentsByName();

break;

case 6: // Count Students

handbook.countStudents();

break;

case 7: // Exit

System.out.println("Exiting the Student Handbook.");

scanner.close();

return;

default:

System.out.println("Invalid choice! Please try again.");

}

}

}

}

**4. How the Code Works**

1. **Student Class**:
   * A simple class with fields like name, studentId, and age. It also includes a toString() method to print the student details easily.
2. **StudentHandbook Class**:
   * Manages the list of students using an ArrayList<Student>.
   * Methods to:
     + addStudent(): Adds a new student.
     + removeStudent(): Removes a student by their ID.
     + displayStudents(): Displays all students in the handbook.
     + searchStudentById(): Searches for a student using their ID.
     + sortStudentsByName(): Sorts students alphabetically by their name using Collections.sort() and a custom comparator.
     + countStudents(): Displays the total number of students.
3. **Main Class**:
   * Displays a menu-driven interface for the user to interact with the system.
   * Based on the user input, it will invoke methods from StudentHandbook to add, remove, search, and sort students.

**Example Output**

Student Handbook Menu:

1. Add Student

2. Remove Student

3. Display All Students

4. Search Student by ID

5. Sort Students by Name

6. Count Students

7. Exit

Enter your choice: 1

Enter student name: John Doe

Enter student ID: 123

Enter student age: 20

Student added: Student [ID=123, Name=John Doe, Age=20]

Student Handbook Menu:

1. Add Student

2. Remove Student

3. Display All Students

4. Search Student by ID

5. Sort Students by Name

6. Count Students

7. Exit

Enter your choice: 3

List of Students:

Student [ID=123, Name=John Doe, Age=20]

**Features Implemented:**

* Adding, removing, searching, displaying, sorting, and counting students.
* Using an ArrayList to dynamically manage students.