Student Management System

Step 1:   
Create Student Class  
\* Attributes: id, name, course, age

// Attribute - instance variables  
 private int id;  
 private String name;  
 private String course;  
 private int age;

Encapsulation

Data hiding: By keeping the private attributes, we are restricting the direct access from outside of the class.

For access: instead of providing direct access, we provide getters and setters.

Ex.(with public)

Student s = new Student(1, “Krishna”, “JAVA”, 23);

s.age=-8; // Wrong age, lead to inconsistency/No validations

Defining the Constructor:

Purpose: it is used to initialize objects when they are created.

Why?

Ensures that when we create the object, it has all the necessary values assigned.

Ex. (with default constructor)

Student s = new Student();

s.setId(1);

s.setName(“Krishna”);

s.setCourse(“JAVA”);

s.setAge(23);

Why this keyword?

The parameters names id, name, course and age are the same as the class attributes.

this.id=id differentiates between the instance variable(this.id) and the parameter(id.)

Define Getters and setters?

Why getters?

* It allows the controlled access to the private variables.
* Read-only fields

Why setters?

* Setter allows the modifications
* We can provide validations as well

Ex. (of validtion)

public void setAge(int age){

if(age > 0){

this.age=age;

}else {

Sout(“Invalid Age”);

}

}

Defining toString() method:

Purpose:

* When we print the object, java internally calls the toString() method.
* Without toString(), it displays the memory reference

Ex.

Student s = new Student(1, “Krishna”, “JAVA”, 23);

Sout(s);

O/P:

[SMS.Student@1234hdddf](mailto:SMS.Student@1234hdddf)

With toString():

Step 2: define StudentManagement interface.

Define abstract methods for CRUD operations.

Interface is just a blueprint of class that defines a contract of methods that must be implemented by class.

Abstraction:

It only defines what needs to be done but doesn’t specify how it should be done.

The actual logic will be implemented in the classes where we are going to use this interface.

We can achieve loose coupling. If we later change the implementation of StudentManagement, other parts of the code will remain unaffected as long as they are interacting with interface.

Interface: Defines a contract that classes must follow

Abstract methods: force implementing classes to define their behavior

Loose coupling: Allows flexible switching between diff implementations

Step 3: define StudentManagementImpl which will implement the StudentManagement

Define the studentList

private List<Student> studentList = new ArrayList<>();

for-each loop is used to iterate over collections.

for (Student student: studentList){  
 System.*out*.println(student);  
}

For(int i=0; i < studentList.size();i++){

Sout(studentList.get(i));

}

Step 4: define StudentManagementSystem (Main Class)

package SMS;  
  
public class Student {  
 // Attribute - instance variables  
 private int id;  
 private String name;  
 private String course;  
 private int age;  
  
 // Constructor  
 public Student(int id, String name, String course, int age) {  
 this.id = id;  
 this.name = name;  
 this.course = course;  
 this.age = age;  
 }  
  
 //Getters and setters  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getCourse() {  
 return course;  
 }  
  
 public void setCourse(String course) {  
 this.course = course;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 // Define toString method  
 @Override  
 public String toString() {  
 return "Student{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", course='" + course + '\'' +  
 ", age=" + age +  
 '}';  
 }  
}

package SMS;  
  
public interface StudentManagement {  
 void addStudent(Student student);  
 void viewAllStudents();  
 void updateStudent(int studentId);  
 void deleteStudent(int studentId);  
 Student findStudentById(int studentId);  
}

package SMS;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
public class StudentManagementImpl implements StudentManagement{  
 private List<Student> studentList = new ArrayList<>();  
  
 @Override  
 public void addStudent(Student student) {  
 studentList.add(student);  
 System.*out*.println("Student Added Successfully");  
 }  
  
 @Override  
 public void viewAllStudents() {  
 if(studentList.isEmpty()){  
 System.*out*.println("The list is empty");  
 } else {  
 System.*out*.println("List of Students:");  
 for (Student student: studentList){  
 System.*out*.println(student);  
 }  
 }  
 }  
  
 @Override  
 public void updateStudent(int studentId) {  
 Student student = findStudentById(studentId);  
 if(student != null){  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Enter new name: ");  
 student.setName(scanner.nextLine());  
 System.*out*.println("Enter new course: ");  
 student.setCourse(scanner.nextLine());  
 System.*out*.println("Enter new Age: ");  
 student.setAge(scanner.nextInt());  
 System.*out*.println("Student updated successfully!");  
 }else {  
 System.*out*.println("Student Not found");  
 }  
 }  
  
 @Override  
 public void deleteStudent(int studentId) {  
 Student student = findStudentById(studentId);  
 if(student!=null){  
 studentList.remove(student);  
 System.*out*.println("Deleted successfully");  
 } else {  
 System.*out*.println("Student not found");  
 }  
 }  
  
 @Override  
 public Student findStudentById(int studentId) {  
 for(Student student : studentList){  
 if(student.getId() == studentId){  
 return student;  
 }  
 }  
 return null;  
 }  
}

package SMS;  
  
import java.util.Scanner;  
  
public class StudentManagementSystem {  
 public static void main(String[] args) {  
 Scanner scanner= new Scanner(System.*in*);  
 StudentManagementImpl management =new StudentManagementImpl();  
 boolean exit = false;  
 System.*out*.println("welcome to sms");  
 while(!exit){  
 System.*out*.println("==========================================================");  
 System.*out*.println("Menu:");  
 System.*out*.println("1. Add Student");  
 System.*out*.println("2. View All Students");  
 System.*out*.println("3. Update Student");  
 System.*out*.println("4. Delete Student");  
 System.*out*.println("5. Exit");  
 System.*out*.println("==========================================================");  
 System.*out*.println("Enter Your Choice: ");  
 int choice = scanner.nextInt();  
  
 switch(choice){  
 case 1:  
 System.*out*.println("Enter Student ID: ");  
 int id = scanner.nextInt();  
 scanner.nextLine(); //consume newLine  
 System.*out*.println("Enter Student Name: ");  
 String name= scanner.nextLine();  
 System.*out*.println("Enter Student Course: ");  
 String course = scanner.nextLine();  
 System.*out*.println("Enter Student Age: ");  
 int age = scanner.nextInt();  
 Student student = new Student(id,name,course, age);  
 management.addStudent(student);  
 break;  
  
 case 2:  
 management.viewAllStudents();  
 break;  
  
 case 3:  
 System.*out*.println("Enter Student Id to update: ");  
 int updateId = scanner.nextInt();  
 management.updateStudent(updateId);  
 break;  
  
 case 4:  
 System.*out*.println("Enter Student Id to delete: ");  
 int deleteId = scanner.nextInt();  
 management.deleteStudent(deleteId);  
 break;  
  
 case 5:  
 exit = true;  
 System.*out*.println("Good Bye!!");  
 break;  
  
 default:  
 System.*out*.println("Invalid Choice!!!!");  
 }  
  
 }  
 }  
}