## Concepts Reinforced:

* **Access Modifiers**
* **Encapsulation**
* **Inheritance**
* **Static and Instance Variables**
* **Constructor Chaining**
* **Method Overriding**
* **Object Composition**

## Real-Time Problem Statement

### ****Project Title: "Smart Student Record System"****

### Context:

You are building a **Student Information Management System** for a university. The system should register students, store academic and contact details securely, and allow for special scholarships for high-performing students.

The solution must follow **OOP principles** like **encapsulation**, **inheritance**, **access modifiers**, and use **composition** to manage addresses and academic information.

## System Overview

### Person (Imported from another module):

* Has private mobile and default laptop
* Used to demonstrate **access control** and **encapsulation**

### Student:

* Stores name, email, and academic details
* Has private roll number and CGPA
* Composed with Address
* Uses **constructor chaining**, **static blocks**, **overridden toString()**
* Contains validation for email and CGPA

### ScholarshipStudent:

* Subclass of Student
* Has field: float scholarshipAmount
* Calculates reduced fees using CGPA
* Demonstrates **method overriding**

### Address:

* Static country = "India"
* Fields: doorNo, street, city, pincode
* Returns short address format

## Hands-On Tasks 🛠️

### Task 1: Mask Roll Number

* Add method getMaskedRoll() in Student
* Returns: "ROLL-XXX7" (only last digit shown)

### Task 2: Calculate Scholarship Deduction

* Add method in ScholarshipStudent:
* public float getPayableFees(float totalFees)
* CGPA ≥ 9 → ₹30,000 scholarship
* CGPA ≥ 8 → ₹20,000
* Else → ₹10,000

### Task 3: Add PartTimeStudent Subclass

* New subclass with:
  + int workHours
  + float stipendPerHour
* Add method calculateMonthlyStipend()

### Task 4: Protect Roll Number

* Try to access rollNo directly from Tester → Should fail
* Access via getter/setter

### Task 5: Welcome Messages via Overriding

* Base method:
* public void printWelcome()
* Override:
  + ScholarshipStudent: "Welcome Scholar with CGPA 9.2!"
  + PartTimeStudent: "Welcome Working Student – 20hrs/week"

### Task 7: Short Address Format

* Method in Address:
* public String getShortAddress()

→ "5, Coimbatore - 641014"

### Task 8: Track Student Count

* Static variable studentCount in Student
* Increment inside constructor
* Display count at the end in Tester

### Task 9: Validate CGPA

* CGPA must be between 0.0 and 10.0

### Task 10: Convert Student to Scholar

* Method in Tester:
* public static ScholarshipStudent convertToScholar(Student s)
* Pass student data and set scholarship based on CGPA

## Mini Menu App in Tester

**public** **class** Tester {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.in);

Student[] students = **new** Student[100];

**int** index = 0;

**while** (**true**) {

System.out.println("\n🎓 === Student Record Menu ===");

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

System.out.println("2. Register Scholarship Student");

System.out.println("3. Show Student Count");

System.out.println("4. Show Payable Fees (for Scholars)");

System.out.println("5. Export Student as CSV");

System.out.println("6. Exit");

System.out.print("Choose option: ");

**int** choice = sc.nextInt();

sc.nextLine(); // consume newline

**switch** (choice) {

**case** 1: {

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

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

String name = sc.nextLine();

System.out.print("Email: ");

String email = sc.nextLine();

System.out.print("Roll No: ");

String roll = sc.nextLine();

System.out.print("CGPA: ");

**float** cgpa = sc.nextFloat(); sc.nextLine();

Address addr = readAddress(sc);

students[index++] = **new** Student(name, email, roll, cgpa, addr);

**break**;

}

**case** 2: {

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

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

String name = sc.nextLine();

System.out.print("Email: ");

String email = sc.nextLine();

System.out.print("Roll No: ");

String roll = sc.nextLine();

System.out.print("CGPA: ");

**float** cgpa = sc.nextFloat(); sc.nextLine();

Address addr = readAddress(sc);

System.out.print("Scholarship Amount: ");

**float** scholarship = sc.nextFloat(); sc.nextLine();

students[index++] = **new** ScholarshipStudent(name, email, roll, cgpa, addr, scholarship);

**break**;

}

**case** 3: {

System.out.println("📊 Total Students Registered: " + Student.getStudentCount());

**break**;

}

**case** 4: {

System.out.print("Enter student index (0 to " + (index - 1) + "): ");

**int** i = sc.nextInt(); sc.nextLine();

**if** (students[i] **instanceof** ScholarshipStudent) {

System.out.print("Enter Total Fees: ");

**float** fees = sc.nextFloat(); sc.nextLine();

ScholarshipStudent s = (ScholarshipStudent) students[i];

System.out.println("Payable Fees: ₹" + s.getPayableFees(fees));

} **else** {

System.out.println("❌ Not a Scholarship Student!");

}

**break**;

}

**case** 5: {

**for** (**int** i = 0; i < index; i++) {

System.out.println(students[i].toCSV());

}

**break**;

}

**case** 6: {

System.out.println("👋 Exiting System.");

**return**;

}

**default**:

System.out.println("Invalid option.");

}

}

}

// Helper to read address

**private** **static** Address readAddress(Scanner sc) {

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

System.out.print("Door No: ");

String door = sc.nextLine();

System.out.print("Street: ");

String street = sc.nextLine();

System.out.print("City: ");

String city = sc.nextLine();

System.out.print("Pincode: ");

String pin = sc.nextLine();

**return** **new** Address(door, street, city, pin);

}

}