by Kunal Sir

INHERITANCE ASSIGNMENT ANSWER

1. SINGLE INHERITANCE (Parent → Child)

Q1. Vehicle → Car Management System

A transport company maintains basic vehicle information.
`Vehicle` has: registration number, manufacturer, base price.
`Car` adds: fuel type and number of seats.

Task:
Create classes `Vehicle` and `Car`. The Car class should print complete details including on-road price = base price + ₹15,000 road tax.

✓ Sample Input

```
Car
MH12AB1234
Tata Motors
550000
Petrol
5
```

✓ Sample Output

```
Registration: MH12AB1234
Manufacturer: Tata Motors
Base Price: 550000
Fuel Type: Petrol
Seats: 5
On-Road Price: 565000
```

```java
// Vehicle and Car - Single Inheritance
class Vehicle {
    String regNo;
    String manufacturer;
    int basePrice;

    Vehicle(String regNo, String manufacturer, int basePrice) {
        this.regNo = regNo;
        this.manufacturer = manufacturer;
        this.basePrice = basePrice;
    }

    void display() {
        System.out.println("Registration: " + regNo);
        System.out.println("Manufacturer: " + manufacturer);
        System.out.println("Base Price: " + basePrice);
    }
}

class Car extends Vehicle {
    String fuelType;
    int seats;

    Car(String regNo, String manufacturer, int basePrice, String fuelType, int
seats) {
        super(regNo, manufacturer, basePrice);
        this.fuelType = fuelType;
        this.seats = seats;
    }

    void displayCarDetails() {
        super.display();
        System.out.println("Fuel Type: " + fuelType);
        System.out.println("Seats: " + seats);
        System.out.println("On-Road Price: " + (basePrice + 15000));
    }
```

```
    public static void main(String[] args) {
        // sample input values
        Car c = new Car("MH12AB1234", "Tata Motors", 550000, "Petrol", 5);
        c.displayCarDetails();
    }
}


Output :

Registration: MH12AB1234
Manufacturer: Tata Motors
Base Price: 550000
Fuel Type: Petrol
Seats: 5
On-Road Price: 565000
```

Q2. BankAccount → SavingsAccount

`BankAccount`: account number, holder name, balance
`SavingsAccount`: interestRate, method applyInterest()

Apply interest:
`newBalance = balance + (balance * rate / 100)`

✓ Sample Input

```
SavingsAccount
Account No: 12345
Name: Rohan
Balance: 10000
Interest Rate: 5
```

✓ Sample Output

```
Before Interest: 10000
After Interest: 10500
```

```java
// BankAccount and SavingsAccount
class BankAccount {
    String accountNo;
    String holderName;
    double balance;

    BankAccount(String accountNo, String holderName, double balance) {
        this.accountNo = accountNo;
        this.holderName = holderName;
        this.balance = balance;
    }

    void showBalance() {
        System.out.println("Before Interest: " + (int)balance);
    }
}
```

```
class SavingsAccount extends BankAccount {
    double interestRate; // percent

    SavingsAccount(String accountNo, String holderName, double balance, double
interestRate) {
        super(accountNo, holderName, balance);
        this.interestRate = interestRate;
    }

    void applyInterest() {
        balance = balance + (balance * interestRate / 100.0);
    }

    public static void main(String[] args) {
        SavingsAccount sa = new SavingsAccount("12345", "Rohan", 10000, 5);
        sa.showBalance();
        sa.applyInterest();
        System.out.println("After Interest: " + (int)sa.balance);
    }
}

Output :

Before Interest: 10000
After Interest: 10500
```

Q3. Employee → Manager Salary Calculation

`Employee`: id, name, baseSalary
`Manager`: teamAllowance, overridden calculateSalary()

✓ Sample Input

```
Manager
ID: 101
Name: Meera
Base Salary: 50000
Team Allowance: 8000
```

✓ Sample Output

```
Employee ID: 101
Name: Meera
Final Salary: 58000
```

```java
// Employee and Manager
class Employee {
    int id;
    String name;
    int baseSalary;

    Employee(int id, String name, int baseSalary) {
        this.id = id;
        this.name = name;
        this.baseSalary = baseSalary;
    }

    int calculateSalary() {
        return baseSalary;
    }

    void printBasic() {
        System.out.println("Employee ID: " + id);
        System.out.println("Name: " + name);
    }
}
```

```
class Manager extends Employee {
    int teamAllowance;

    Manager(int id, String name, int baseSalary, int teamAllowance) {
        super(id, name, baseSalary);
        this.teamAllowance = teamAllowance;
    }

    int calculateSalary(int dummy) { // demonstrating manager salary calculation
        return baseSalary + teamAllowance;
    }

    public static void main(String[] args) {
        Manager m = new Manager(101, "Meera", 50000, 8000);
        m.printBasic();
        System.out.println("Final Salary: " + m.calculateSalary(0));
    }
}


Employee ID: 101
Name: Meera
Final Salary: 58000
```

Q4. Product → ElectronicProduct Billing

`Product`: brand, price
`ElectronicProduct`: warrantyYears, finalPrice = price + warrantyYears*500

✓ Sample Input

```
ElectronicProduct
Brand: Samsung
Price: 20000
Warranty: 2 years
```

✓ Sample Output

```
Brand: Samsung
Base Price: 20000
Warranty Years: 2
Final Price: 21000
```

```java
// Product and ElectronicProduct
class Product {
    String brand;
    int price;

    Product(String brand, int price) {
        this.brand = brand;
        this.price = price;
    }

    void display() {
        System.out.println("Brand: " + brand);
        System.out.println("Base Price: " + price);
    }
}

class ElectronicProduct extends Product {
    int warrantyYears;

    ElectronicProduct(String brand, int price, int warrantyYears) {
        super(brand, price);
        this.warrantyYears = warrantyYears;
    }
```

```
    void displayProduct() {
        super.display();
        System.out.println("Warranty Years: " + warrantyYears);
        System.out.println("Final Price: " + (price + warrantyYears * 500));
    }

    public static void main(String[] args) {
        ElectronicProduct ep = new ElectronicProduct("Samsung", 20000, 2);
        ep.displayProduct();
    }
}


Output :

Brand: Samsung
Base Price: 20000
Warranty Years: 2
Final Price: 21000
```

2. MULTILEVEL INHERITANCE (Grandparent → Parent → Child)

Q1. Person → Employee → Manager

`Person`: name, age
`Employee`: empId, department
`Manager`: team size

✓ Sample Input

```
Manager
Name: Aarav
Age: 32
Emp ID: E102
Department: IT
Team Size: 12
```

✓ Sample Output

```
Name: Aarav
Age: 32
Employee ID: E102
Department: IT
Team Size: 12
```



```java
// Person -> Employee -> Manager
class Person {
    String name;
    int age;
    Person(String name, int age) { this.name = name; this.age = age; }
    void getDetails() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
```

```
class Employee extends Person {
    String empId;
    String department;
    Employee(String name, int age, String empId, String department) {
        super(name, age);
        this.empId = empId;
        this.department = department;
    }
    void getDetails() {
        super.getDetails();
        System.out.println("Employee ID: " + empId);
        System.out.println("Department: " + department);
    }
}

class ManagerML extends Employee {
    int teamSize;
    ManagerML(String name, int age, String empId, String department, int
teamSize) {
        super(name, age, empId, department);
        this.teamSize = teamSize;
    }
    void getDetails() {
        super.getDetails();
        System.out.println("Team Size: " + teamSize);
    }

    public static void main(String[] args) {
        ManagerML mgr = new ManagerML("Aarav", 32, "E102", "IT", 12);
        mgr.getDetails();
    }
}


Output :
Name: Aarav
Age: 32
Employee ID: E102
Department: IT
Team Size: 12
```

Q2. Animal → Mammal → Dog

`Animal`: eat(), sleep()
`Mammal`: warm-blooded info
`Dog`: bark()

✓ Sample Input

```
Dog
Action: Details
```

✓ Sample Output

```
Animal: Eats food
Animal: Sleeps
Mammal: Warm-blooded creature
Dog: Barks loudly
```

```java
// Animal -> Mammal -> Dog
class Animal {
  void eat() {
    System.out.println("Animal: Eats food");
  }

  void sleep() {
    System.out.println("Animal: Sleeps");
  }
}

class Mammal extends Animal {
  void mammalInfo() {
    System.out.println("Mammal: Warm-blooded creature");
  }
}

class Dog extends Mammal {
  void bark() {
    System.out.println("Dog: Barks loudly");
  }
```

```
  public static void main(String[] args) {
    Dog d = new Dog();
    d.eat();
    d.sleep();
    d.mammalInfo();
    d.bark();
  }
}


Output :
Animal: Eats food
Animal: Sleeps
Mammal: Warm-blooded creature
Dog: Barks loudly
```

Q3. Device → Computer → Laptop

`Device`: serialNo
`Computer`: processor, RAM
`Laptop`: batteryBackup

✓ Sample Input

```
Laptop
Serial: D1001
Processor: i5
RAM: 8GB
Battery: 6 Hours
```

✓ Sample Output

```
Serial No: D1001
Processor: i5
RAM: 8GB
Battery Backup: 6 Hours
```

```java
// Device -> Computer -> Laptop
class Device {
    String serialNo;
    Device(String serialNo) { this.serialNo = serialNo; }
    void show() { System.out.println("Serial No: " + serialNo); }
}

class Computer extends Device {
    String processor;
    String ram;
    Computer(String serialNo, String processor, String ram) {
        super(serialNo);
        this.processor = processor;
        this.ram = ram;
    }
    void show() {
        super.show();
        System.out.println("Processor: " + processor);
        System.out.println("RAM: " + ram);
    }
}

class Laptop extends Computer {
    String battery;
    Laptop(String serialNo, String processor, String ram, String battery) {
        super(serialNo, processor, ram);
        this.battery = battery;
    }
    void showAll() {
        super.show();
        System.out.println("Battery Backup: " + battery);
    }

    public static void main(String[] args) {
        Laptop l = new Laptop("D1001", "i5", "8GB", "6 Hours");
        l.showAll();
    }
}
```

```
Output:
Serial No: D1001
Processor: i5
RAM: 8GB
Battery Backup: 6 Hours
```

Q4. Course → OnlineCourse → SelfPacedCourse

`Course`: title, duration
`OnlineCourse`: platform
`SelfPacedCourse`: access validity

✓ Sample Input

```
SelfPacedCourse
Title: Java Basics
Duration: 30 Days
Platform: Udemy
Access Validity: 1 Year
```

✓ Sample Output

```
Course: Java Basics
Duration: 30 Days
Platform: Udemy
Access Validity: 1 Year
```



```java
// Course -> OnlineCourse -> SelfPacedCourse
class Course {
    String title;
    String duration;
    Course(String title, String duration) { this.title = title; this.duration = duration; }
    void getCourseInfo() {
        System.out.println("Course: " + title);
        System.out.println("Duration: " + duration);
    }
}
```

```
class OnlineCourse extends Course {
    String platform;
    OnlineCourse(String title, String duration, String platform) {
        super(title, duration);
        this.platform = platform;
    }
    void getCourseInfo() {
        super.getCourseInfo();
        System.out.println("Platform: " + platform);
    }
}

class SelfPacedCourse extends OnlineCourse {
    String accessValidity;
    SelfPacedCourse(String title, String duration, String platform, String
accessValidity) {
        super(title, duration, platform);
        this.accessValidity = accessValidity;
    }
    void getCourseInfoAll() {
        super.getCourseInfo();
        System.out.println("Access Validity: " + accessValidity);
    }

    public static void main(String[] args) {
        SelfPacedCourse sp = new SelfPacedCourse("Java Basics", "30 Days",
"Udemy", "1 Year");
        sp.getCourseInfoAll();
    }
}

Output:
Course: Java Basics
Duration: 30 Days
Platform: Udemy
Access Validity: 1 Year
```

3. HIERARCHICAL INHERITANCE (One Parent → Many Children)

Q1. Shape → Circle, Rectangle, Triangle

`Shape`: color
Each child writes its own draw() method to show shape-specific details.

✓ Sample Input

```
Circle
Color: Red
Radius: 5
```

✓ Sample Output

```
Drawing Circle
Color: Red
Radius: 5
Area: 78.5
```

```java
// Shape hierarchy example
class Shape {
    String color;
    Shape(String color) { this.color = color; }
    void draw() { System.out.println("Drawing Shape"); }
}

class Circle extends Shape {
    double radius;
    Circle(String color, double radius) { super(color); this.radius = radius; }
    void drawDetails() {
        System.out.println("Drawing Circle");
        System.out.println("Color: " + color);
        System.out.println("Radius: " + (int)radius);
        System.out.println("Area: " + String.format("%.1f", Math.PI * radius *
radius));
    }
```

```
    public static void main(String[] args) {
        Circle c = new Circle("Red", 5);
        c.drawDetails();
    }
}


Output :
Drawing Circle
Color: Red
Radius: 5
Area: 78.5
```

Q2. Vehicle → Car, Bike, Truck

`Vehicle`: brand
`Car`: seats
`Bike`: mileage
`Truck`: loadCapacity

✓ Sample Input

```
Truck
Brand: Ashok Leyland
Load Capacity: 12000 kg
```

✓ Sample Output

```
Vehicle Type: Truck
Brand: Ashok Leyland
Load Capacity: 12000 kg
```

```
// Vehicle -> Truck sample
class VehicleH {
    String brand;
    VehicleH(String brand) { this.brand = brand; }
    void info() { System.out.println("Brand: " + brand); }
}
```

```
class Truck extends VehicleH {
    String loadCapacity;
    Truck(String brand, String loadCapacity) { super(brand); this.loadCapacity =
loadCapacity; }
    void show() {
        System.out.println("Vehicle Type: Truck");
        System.out.println("Brand: " + brand);
        System.out.println("Load Capacity: " + loadCapacity);
    }

    public static void main(String[] args) {
        Truck t = new Truck("Ashok Leyland", "12000 kg");
        t.show();
    }
}

Output :
Vehicle Type: Truck
Brand: Ashok Leyland
Load Capacity: 12000 kg
```

Q3. Media → Book, Movie, Song

`Media`: title, year
`Book, Movie, Song`: Child adds its own fields.

✓ Sample Input

```
Movie
Title: Inception
Year: 2010
Director: Christopher Nolan
```

✓ Sample Output

```
Media Type: Movie
Title: Inception
Year: 2010
Director: Christopher Nolan
```

```java
// Media -> Movie sample
class Media {
    String title;
    int year;
    Media(String title, int year) { this.title = title; this.year = year; }
    void display() {
        System.out.println("Title: " + title);
        System.out.println("Year: " + year);
    }
}

class MovieM extends Media {
    String director;
    MovieM(String title, int year, String director) { super(title, year);
this.director = director; }
    void show() {
        System.out.println("Media Type: Movie");
        display();
        System.out.println("Director: " + director);
    }

    public static void main(String[] args) {
        MovieM m = new MovieM("Inception", 2010, "Christopher Nolan");
        m.show();
    }
}


Output :
Media Type: Movie
Title: Inception
Year: 2010
Director: Christopher Nolan
```

Q4. Payment → UPI, CreditCard, Cash

`Payment`: amount
UPI, CreditCard, Cash: Each child class writes its own processPayment() method according to its payment type.

✓ Sample Input

```
UPI
Amount: 250
UPI ID: rohan@oksbi
```

✓ Sample Output

```
Processing UPI Payment
Amount: 250
Paid via: rohan@oksbi
Payment Successful
```

```java
// Payment -> UPI example
class Payment {
    double amount;
    Payment(double amount) { this.amount = amount; }
    void processPayment() {}
}

class UPI extends Payment {
    String upiId;
    UPI(double amount, String upiId) { super(amount); this.upiId = upiId; }
    void process() {
        System.out.println("Processing UPI Payment");
        System.out.println("Amount: " + (int)amount);
        System.out.println("Paid via: " + upiId);
        System.out.println("Payment Successful");
    }

    public static void main(String[] args) {
        UPI u = new UPI(250, "rohan@oksbi");
        u.process();
    }
}
```

```
Output :
Processing UPI Payment
Amount: 250
Paid via: rohan@oksbi
Payment Successful
```

---

by Kunal Sir

Q1. University Management System (Multilevel + Hierarchical)

A university needs a class structure to manage different roles of people.

Person (base class)

- name, age

Employee extends Person (single inheritance)

- employeeId, department

TeachingStaff extends Employee (multilevel inheritance)

- subject, timetable

NonTeachingStaff extends Employee (hierarchical inheritance)

- duty, shift

Student extends Person (hierarchical inheritance)

- rollNo, course

Problem:

Design the inheritance structure so that:

- Person → Employee → TeachingStaff forms **multilevel inheritance**
- Employee → TeachingStaff and Employee → NonTeachingStaff form **hierarchical inheritance**
- Person → Student is **single inheritance**

Create a method **getDetails()** in every class and show how inheritance reduces code duplication.

```java
// University simple structure
class PersonU {
    String name; int age;
    PersonU(String n, int a) { name = n; age = a; }
    void getDetails() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

class EmployeeU extends PersonU {
    String employeeId; String department;
    EmployeeU(String n, int a, String id, String dept) { super(n,a); employeeId =
id; department = dept; }
    void getDetails() {
        super.getDetails();
        System.out.println("Employee ID: " + employeeId);
        System.out.println("Department: " + department);
    }
}


class TeachingStaff extends EmployeeU {
    String subject; String timetable;
    TeachingStaff(String n,int a,String id,String dept,String subject,String
timetable) {
        super(n,a,id,dept);
        this.subject = subject; this.timetable = timetable;
    }
    void getDetailsAll() {
        super.getDetails();
        System.out.println("Subject: " + subject);
        System.out.println("Timetable: " + timetable);
    }
}

class NonTeachingStaff extends EmployeeU {
    String duty; String shift;
    NonTeachingStaff(String n,int a,String id,String dept,String duty,String
shift) {
        super(n,a,id,dept); this.duty = duty; this.shift = shift;
```

```
    }
    void getDetailsAll() {
        super.getDetails();
        System.out.println("Duty: " + duty);
        System.out.println("Shift: " + shift);
    }
}

class StudentU extends PersonU {
    String rollNo; String course;
    StudentU(String n,int a,String rollNo,String course) { super(n,a);
this.rollNo=rollNo; this.course=course; }
    void getDetailsAll() {
        super.getDetails();
        System.out.println("Roll No: " + rollNo);
        System.out.println("Course: " + course);
    }

    public static void main(String[] args) {
        TeachingStaff t = new TeachingStaff("Dr. Rao", 45, "EMP101", "Computer
Science", "Algorithms", "Mon-Wed 10-12");
        t.getDetailsAll();
        System.out.println("----");
        NonTeachingStaff n = new NonTeachingStaff("Sunita", 38, "EMP201",
"Admin", "Registry", "Morning");
        n.getDetailsAll();
        System.out.println("----");
        StudentU s = new StudentU("Riya", 20, "S123", "B.Tech");
        s.getDetailsAll();
    }
}
```

```
Output :
Name: Dr. Rao
Age: 45
Employee ID: EMP101
Department: Computer Science
Subject: Algorithms
Timetable: Mon-Wed 10-12
----
Name: Sunita
Age: 38
Employee ID: EMP201
Department: Admin
Duty: Registry
Shift: Morning
----
Name: Riya
Age: 20
Roll No: S123
Course: B.Tech
```

Q2. Hospital Management (Single + Multilevel + Hierarchical)

A hospital wants to classify its working structure.

HospitalMember

- id, name

Doctor extends HospitalMember (single inheritance)

- specialization

Surgeon extends Doctor (multilevel inheritance)

- surgeryType

Nurse extends HospitalMember (hierarchical inheritance)

- wardAssigned

Receptionist extends HospitalMember (hierarchical inheritance)

- deskNumber

Problem:

Build the above hierarchy where:

- HospitalMember → Doctor → Surgeon forms a **multilevel** chain
- HospitalMember → Nurse and HospitalMember → Receptionist form **hierarchical inheritance**

Demonstrate method overriding of **performDuty()** for each role.

```
// Hospital structure
class HospitalMember {
    String id; String name;
    HospitalMember(String id, String name) { this.id = id; this.name = name; }
    void performDuty() { System.out.println(name + " performs general hospital
duties."); }
}

class Doctor extends HospitalMember {
    String specialization;
    Doctor(String id,String name,String specialization) { super(id,name);
this.specialization = specialization; }
    void performDuty() {
        System.out.println("Doctor " + name + " (Specialization: " +
specialization + ") consults patients.");
    }
}

class Surgeon extends Doctor {
    String surgeryType;
    Surgeon(String id,String name,String specialization,String surgeryType) {
        super(id,name,specialization); this.surgeryType = surgeryType;
    }
    void performDuty() {
        System.out.println("Surgeon " + name + " performs " + surgeryType + "
surgeries.");
    }
}

class Nurse extends HospitalMember {
    String wardAssigned;
    Nurse(String id,String name,String wardAssigned) { super(id,name);
this.wardAssigned = wardAssigned; }
    void performDuty() {
        System.out.println("Nurse " + name + " attends the " + wardAssigned + "
ward.");
    }
}
```

```
class Receptionist extends HospitalMember {
    String deskNumber;
    Receptionist(String id,String name,String deskNumber) { super(id,name);
this.deskNumber = deskNumber; }
    void performDuty() {
        System.out.println("Receptionist " + name + " manages desk " + deskNumber
+ ".");
    }

    public static void main(String[] args) {
        Surgeon s = new Surgeon("D201", "Dr. Mehta", "Surgery", "Cardiac");
        s.performDuty();
        Nurse n = new Nurse("N301", "Anita", "Pediatrics");
        n.performDuty();
        Receptionist r = new Receptionist("R401", "Vikram", "Desk-5");
        r.performDuty();
    }
}


Output :
Surgeon Dr. Mehta performs Cardiac surgeries.
Nurse Anita attends the Pediatrics ward.
Receptionist Vikram manages desk Desk-5.
```

Q3. Automobile Manufacturing (Single + Hierarchical + Multilevel)

An automobile company needs to model its vehicle categories.

Vehicle

- brand, model

Car extends Vehicle (single inheritance)

- numberOfDoors

SUV extends Car (multilevel inheritance)

- groundClearance

Truck extends Vehicle (hierarchical inheritance)

- loadCapacity

SportsCar extends Car (hierarchical inheritance)

- topSpeed

Problem:

Create the class model such that:

- Vehicle → Car → SUV forms **multilevel inheritance**
- Vehicle → Truck and Vehicle → Car form **hierarchical inheritance**
- Car extends Vehicle is **single inheritance**

Implement a method **vehicleInfo()** in all classes to show specialization increasing at each level.

```
// Automobile model
class VehicleBase {
    String brand; String model;
    VehicleBase(String brand, String model) { this.brand = brand; this.model =
model; }
```

```
    void vehicleInfo() { System.out.println("Brand: " + brand);
System.out.println("Model: " + model); }
}

class CarA extends VehicleBase {
    int numberOfDoors;
    CarA(String brand,String model,int numberOfDoors) { super(brand,model);
this.numberOfDoors = numberOfDoors; }
    void vehicleInfo() {
        super.vehicleInfo();
        System.out.println("Doors: " + numberOfDoors);
    }
}

class SUV extends CarA {
    int groundClearance;
    SUV(String brand,String model,int doors,int groundClearance) {
super(brand,model,doors); this.groundClearance = groundClearance; }
    void vehicleInfoAll() {
        super.vehicleInfo();
        System.out.println("Ground Clearance: " + groundClearance);
    }
}

class TruckA extends VehicleBase {
    String loadCapacity;
    TruckA(String brand,String model,String loadCapacity) { super(brand,model);
this.loadCapacity = loadCapacity; }
    void vehicleInfo() {
        super.vehicleInfo();
        System.out.println("Load Capacity: " + loadCapacity);
    }
}



class SportsCar extends CarA {
    int topSpeed;
    SportsCar(String brand,String model,int doors,int topSpeed) {
super(brand,model,doors); this.topSpeed = topSpeed; }
    void vehicleInfoAll() {
```

```
        super.vehicleInfo();
        System.out.println("Top Speed: " + topSpeed);
    }

    public static void main(String[] args) {
        SUV s = new SUV("Kia", "Seltos", 5, 200);
        s.vehicleInfoAll();
        System.out.println("---");
        TruckA t = new TruckA("Ashok Leyland", "Classic", "12000 kg");
        t.vehicleInfo();
    }
}

Output :
Brand: Kia
Model: Seltos
Doors: 5
Ground Clearance: 200
---
Brand: Ashok Leyland
Model: Classic
Load Capacity: 12000 kg
```

Q4. E-Commerce Product Catalog (Multilevel + Hierarchical + Single)

An e-commerce platform wants to organize products.

Product

- id, name

Electronics extends Product (single inheritance)

- warranty

MobilePhone extends Electronics (multilevel inheritance)

- cameraQuality

Laptop extends Electronics (hierarchical inheritance)

- processor

Clothing extends Product (hierarchical inheritance)

- size, fabric

Problem:

Model the platform so that:

- Product → Electronics → MobilePhone forms **multilevel inheritance**
- Product → Electronics and Product → Clothing form **hierarchical inheritance**
- Electronics extends Product is **single inheritance**

Create a method `display()` in every class.
Each subclass should show its own extra information along with the details it inherits from its parent class.

```
// E-commerce product catalog
class ProductE {
    String id; String name;
    ProductE(String id, String name) { this.id = id; this.name = name; }
    void display() {
        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
```

```
    }
}

class ElectronicsE extends ProductE {
    int warranty; // years
    ElectronicsE(String id, String name, int warranty) { super(id,name);
this.warranty = warranty; }
    void display() {
        super.display();
        System.out.println("Warranty: " + warranty + " years");
    }
}

class MobilePhone extends ElectronicsE {
    String cameraQuality;
    MobilePhone(String id,String name,int warranty,String cameraQuality) {
        super(id,name,warranty); this.cameraQuality = cameraQuality;
    }
    void displayAll() {
        super.display();
        System.out.println("Camera Quality: " + cameraQuality);
    }
}

class LaptopE extends ElectronicsE {
    String processor;
    LaptopE(String id,String name,int warranty,String processor) {
super(id,name,warranty); this.processor = processor; }
    void displayAll() {
        super.display();
        System.out.println("Processor: " + processor);
    }
}

class ClothingE extends ProductE {
    String size; String fabric;
    ClothingE(String id,String name,String size,String fabric) { super(id,name);
this.size = size; this.fabric = fabric; }
    void displayAll() {
        super.display();
        System.out.println("Size: " + size);
```

```
        System.out.println("Fabric: " + fabric);
    }

    public static void main(String[] args) {
        MobilePhone mp = new MobilePhone("P100", "Galaxy S", 1, "108MP");
        mp.displayAll();
        System.out.println("---");
        LaptopE lap = new LaptopE("L200", "ThinkPad", 2, "i7");
        lap.displayAll();
        System.out.println("---");
        ClothingE cloth = new ClothingE("C300", "Denim Jacket", "M", "Denim");
        cloth.displayAll();
    }
}

Output:
ID: P100
Name: Galaxy S
Warranty: 1 years
Camera Quality: 108MP
---
ID: L200
Name: ThinkPad
Warranty: 2 years
Processor: i7
---
ID: C300
Name: Denim Jacket
Size: M
Fabric: Denim
```
