Inheritance Problem

1. Assisted Problem :-

1. Animal Hierarchy :-

```
package InheritanceProblem.Assisted_Problems;

class Animal {
    String name;
    int age;
    Animal(String name, int age) {
        this.name = name;
        this.age = age;
    }
    void makeSound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    Dog(String name, int age) { super(name, age); }
    @Override
    void makeSound() { System.out.println("Woof Woof!"); }
}

class Cat extends Animal {
    Cat(String name, int age) { super(name, age); }
    @Override
    void makeSound() { System.out.println("Meow Meow!"); }
}

class Bird extends Animal {
    Bird(String name, int age) { super(name, age); }
    @Override
    void makeSound() { System.out.println("Tweet Tweet!"); }
}

public class AnimalHierarchy {
    public static void main(String[] args) {
        Animal a1 = new Dog("Buddy", 3);
        Animal a2 = new Cat("Kitty", 2);
        Animal a3 = new Bird("Tweety", 1);

        a1.makeSound();
        a2.makeSound();
        a3.makeSound();
    }
}
```

2. Employee Management System

```
package InheritanceProblem.Assisted_Problems;

class Employee {
    String name;
    int id;
    double salary;
    Employee(String name, int id, double salary) {
        this.name = name;
        this.id = id;
        this.salary = salary;
    }
    void displayDetails() {
        System.out.println("Name: " + name + ", ID: " + id + ", Salary: " + salary);
    }
}

class Manager extends Employee {
    int teamSize;
    Manager(String name, int id, double salary, int teamSize) {
        super(name, id, salary);
        this.teamSize = teamSize;
    }
    @Override
    void displayDetails() {
        super.displayDetails();
        System.out.println("Team Size: " + teamSize);
    }
}

class Developer extends Employee {
    String programmingLanguage;
    Developer(String name, int id, double salary, String programmingLanguage) {
        super(name, id, salary);
        this.programmingLanguage = programmingLanguage;
    }
    @Override
    void displayDetails() {
        super.displayDetails();
        System.out.println("Programming Language: " + programmingLanguage);
    }
}

class Intern extends Employee {
    int internshipDuration;
    Intern(String name, int id, double salary, int duration) {
        super(name, id, salary);
        this.internshipDuration = duration;
    }
    @Override
    void displayDetails() {
        super.displayDetails();
        System.out.println("Internship Duration: " + internshipDuration + " months");
    }
}
```

```
public class EmployeeManagement {
    public static void main(String[] args) {
        Manager m = new Manager("Alice", 101, 90000, 10);
        Developer d = new Developer("Bob", 102, 70000, "Java");
        Intern i = new Intern("Charlie", 103, 20000, 3);

        m.displayDetails();
        d.displayDetails();
        i.displayDetails();
    }
}
```

3. Vehicle and Transport System

```
package InheritanceProblem.Assisted_Problems;

class Vehicle {
    int maxSpeed;
    String fuelType;
    Vehicle(int maxSpeed, String fuelType) {
        this.maxSpeed = maxSpeed;
        this.fuelType = fuelType;
    }
    void displayInfo() {
        System.out.println("Max Speed: " + maxSpeed + " km/h, Fuel: " + fuelType);
    }
}

class Car extends Vehicle {
    int seatCapacity;
    Car(int maxSpeed, String fuelType, int seatCapacity) {
        super(maxSpeed, fuelType);
        this.seatCapacity = seatCapacity;
    }
    @Override
    void displayInfo() {
        super.displayInfo();
        System.out.println("Seat Capacity: " + seatCapacity);
    }
}

class Truck extends Vehicle {
    int loadCapacity;
    Truck(int maxSpeed, String fuelType, int loadCapacity) {
        super(maxSpeed, fuelType);
        this.loadCapacity = loadCapacity;
    }
    @Override
    void displayInfo() {
```

```
        super.displayInfo();
        System.out.println("Load Capacity: " + loadCapacity + " tons");
    }
}

class Motorcycle extends Vehicle {
    boolean hasCarrier;
    Motorcycle(int maxSpeed, String fuelType, boolean hasCarrier) {
        super(maxSpeed, fuelType);
        this.hasCarrier = hasCarrier;
    }
    @Override
    void displayInfo() {
        super.displayInfo();
        System.out.println("Has Carrier: " + hasCarrier);
    }
}

public class VehicleTransport {
    public static void main(String[] args) {
        Vehicle[] vehicles = {
                new Car(180, "Petrol", 5),
                new Truck(120, "Diesel", 10),
                new Motorcycle(150, "Petrol", true)
        };

        for (Vehicle v : vehicles) {
            v.displayInfo();
            System.out.println("------");
        }
    }
}
```

~~~~~~

2. Single Inheritance

Sample Problem 1: Library Management with Books and Authors

```
package InheritanceProblem.Single_Inheritance;

class Book {
    String title;
    int publicationYear;
    Book(String title, int publicationYear) {
        this.title = title;
        this.publicationYear = publicationYear;
    }
    void displayInfo() {
        System.out.println("Book: " + title + " (" + publicationYear + ")");
    }
```
~~~~~~

```
}

class Author extends Book {
    String authorName, bio;
    Author(String title, int publicationYear, String authorName, String bio) {
        super(title, publicationYear);
        this.authorName = authorName;
        this.bio = bio;
    }
    @Override
    void displayInfo() {
        super.displayInfo();
        System.out.println("Author: " + authorName + " | Bio: " + bio);
    }
}

public class LibraryManagement {
    public static void main(String[] args) {
        Author a = new Author("Java Basics", 2024, "John Doe", "Tech Author");
        a.displayInfo();
    }
}
```

Sample Problem 2: Smart Home Devices

```
package InheritanceProblem.Single_Inheritance;

class Device {
    String deviceId;
    String status;
    Device(String deviceId, String status) {
        this.deviceId = deviceId;
        this.status = status;
    }
    void displayStatus() {
        System.out.println("Device ID: " + deviceId + " | Status: " + status);
    }
}

class Thermostat extends Device {
    int temperatureSetting;
    Thermostat(String deviceId, String status, int temperatureSetting) {
        super(deviceId, status);
        this.temperatureSetting = temperatureSetting;
    }
    @Override
    void displayStatus() {
        super.displayStatus();
        System.out.println("Temperature: " + temperatureSetting + "°C");
    }
}
```

```java
public class SmartHome {
    public static void main(String[] args) {
        Thermostat t = new Thermostat("T001", "ON", 24);
        t.displayStatus();
    }
}
```

~~~~~~

3. Multilevel Inheritance :-

Sample Problem 1: Online Retail Order Management

```java
package InheritanceProblem.Multilevel_Inheritance;

class Order {
    int orderId;
    Order(int orderId) { this.orderId = orderId; }
    void getOrderStatus() { System.out.println("Order Placed: " + orderId); }
}

class ShippedOrder extends Order {
    String trackingNumber;
    ShippedOrder(int orderId, String trackingNumber) {
        super(orderId);
        this.trackingNumber = trackingNumber;
    }
    @Override
    void getOrderStatus() { System.out.println("Order Shipped: " + trackingNumber); }
}

class DeliveredOrder extends ShippedOrder {
    String deliveryDate;
    DeliveredOrder(int orderId, String trackingNumber, String deliveryDate) {
        super(orderId, trackingNumber);
        this.deliveryDate = deliveryDate;
    }
    @Override
    void getOrderStatus() { System.out.println("Delivered on: " + deliveryDate); }
}

public class OnlineRetailOrder {
    public static void main(String[] args) {
        DeliveredOrder d = new DeliveredOrder(101, "TRK123", "2025-09-22");
        d.getOrderStatus();
    }
}
```

Sample Problem 2: Educational Course Hierarchy
~~~~~~

```
package InheritanceProblem.Multilevel_Inheritance;

class Course {
    String courseName;
    int duration;
    Course(String courseName, int duration) {
        this.courseName = courseName;
        this.duration = duration;
    }
    void displayCourseInfo() {
        System.out.println("Course: " + courseName + " | Duration: " + duration + " weeks");
    }
}

class OnlineCourse extends Course {
    String platform;
    boolean isRecorded;
    OnlineCourse(String courseName, int duration, String platform, boolean isRecorded) {
        super(courseName, duration);
        this.platform = platform;
        this.isRecorded = isRecorded;
    }
    @Override
    void displayCourseInfo() {
        super.displayCourseInfo();
        System.out.println("Platform: " + platform + " | Recorded: " + isRecorded);
    }
}

class PaidOnlineCourse extends OnlineCourse {
    double fee, discount;
    PaidOnlineCourse(String courseName, int duration, String platform, boolean isRecorded, double fee,
double discount) {
        super(courseName, duration, platform, isRecorded);
        this.fee = fee;
        this.discount = discount;
    }
    @Override
    void displayCourseInfo() {
        super.displayCourseInfo();
        System.out.println("Fee: " + fee + " | Discount: " + discount + "%");
    }
}

public class EducationalCourses {
    public static void main(String[] args) {
        PaidOnlineCourse poc = new PaidOnlineCourse("Java Programming", 6, "Coursera", true, 2000,
10);
        poc.displayCourseInfo();
    }
}
```