INHERITANCE

Simple inheritance: Doctor extends Person

Multilevel inheritance: Surgeon extends Doctor

Hierarchical inheritance: Doctor and Nurse both extend Person

Multiple inheritance (via interfaces): Surgeon implements Researcher, Teacher

Hybrid inheritance: Intern extends Student (extends Person) and also implements Worker

Abstract class: Person

Polymorphism + overriding

super keyword

final class: HospitalPolicy

Program:

```
// ============================
// ABSTRACT BASE CLASS
// ============================
abstract class Person {
    String name;
    int age;

    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    abstract void role();

    void displayDetails() {
        System.out.println("Name: " + name + ", Age: " + age);
    }
}
```

```
// ============================
// SIMPLE INHERITANCE
// ============================
class Doctor extends Person {
    String specialty;

    Doctor(String name, int age, String specialty) {
        super(name, age);
        this.specialty = specialty;
    }

    @Override
    void role() {
        System.out.println(name + " is a doctor specialized in " + specialty + ".");
    }

    void treatPatient() {
        System.out.println(name + " is treating patients.");
    }
}

// ============================
// MULTILEVEL INHERITANCE
// ============================
class Surgeon extends Doctor {
    Surgeon(String name, int age, String specialty) {
        super(name, age, specialty);
    }

    void performSurgery() {
```

```
        System.out.println(name + " is performing surgery.");
    }

    @Override
    void treatPatient() {
        super.treatPatient();
        System.out.println(name + " also performs surgical treatment.");
    }
}

// ============================
// HIERARCHICAL INHERITANCE
// ============================
class Nurse extends Person {
    Nurse(String name, int age) {
        super(name, age);
    }

    @Override
    void role() {
        System.out.println(name + " is a nurse.");
    }

    void assistDoctor() {
        System.out.println(name + " is assisting a doctor.");
    }
}

class Patient extends Person {
    String disease;
```

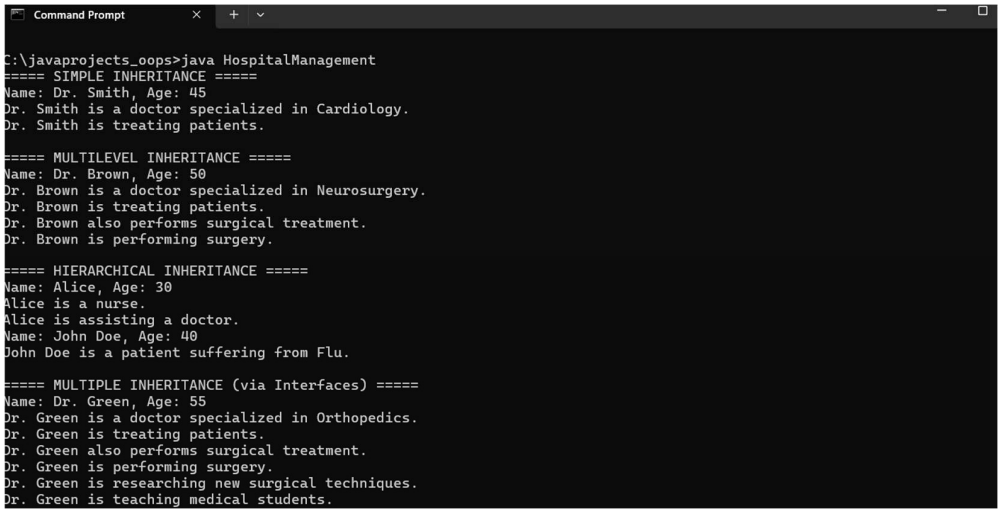
```
    Patient(String name, int age, String disease) {
        super(name, age);
        this.disease = disease;
    }

    @Override
    void role() {
        System.out.println(name + " is a patient suffering from " + disease + ".");
    }
}

// ============================
// MULTIPLE INHERITANCE (via Interfaces)
// ============================
interface Researcher {
    void research();
}

interface Teacher {
    void teach();
}

class AcademicSurgeon extends Surgeon implements Researcher, Teacher {
    AcademicSurgeon(String name, int age, String specialty) {
        super(name, age, specialty);
    }

    @Override
    public void research() {
        System.out.println(name + " is researching new surgical techniques.");
    }
```

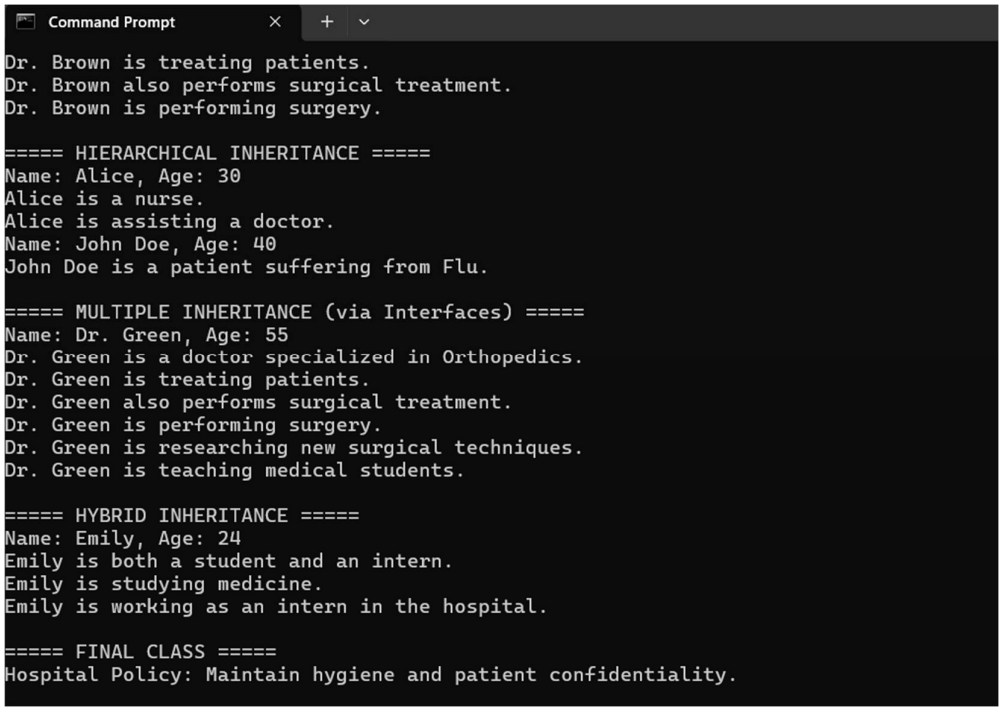
```
    @Override
    public void teach() {
        System.out.println(name + " is teaching medical students.");
    }
}

// ============================
// HYBRID INHERITANCE
// ============================
class Student extends Person {
    Student(String name, int age) {
        super(name, age);
    }

    void study() {
        System.out.println(name + " is studying medicine.");
    }

    @Override
    void role() {
        System.out.println(name + " is a medical student.");
    }
}

interface Worker {
    void work();
}

class Intern extends Student implements Worker {
    Intern(String name, int age) {
```

```
        super(name, age);
    }

    @Override
    public void work() {
        System.out.println(name + " is working as an intern in the hospital.");
    }

    @Override
    void role() {
        System.out.println(name + " is both a student and an intern.");
    }
}

// ============================
// FINAL CLASS
// ============================
final class HospitalPolicy {
    void displayPolicy() {
        System.out.println("Hospital Policy: Maintain hygiene and patient confidentiality.");
    }
}

// ============================
// MAIN APPLICATION
// ============================
public class HospitalManagement {
    public static void main(String[] args) {
        System.out.println("===== SIMPLE INHERITANCE =====");
        Doctor doc = new Doctor("Dr. Smith", 45, "Cardiology");
        doc.displayDetails();
```

```
doc.role();
doc.treatPatient();

System.out.println("\n===== MULTILEVEL INHERITANCE =====");
Surgeon surg = new Surgeon("Dr. Brown", 50, "Neurosurgery");
surg.displayDetails();
surg.role();
surg.treatPatient();
surg.performSurgery();

System.out.println("\n===== HIERARCHICAL INHERITANCE =====");
Nurse nurse = new Nurse("Alice", 30);
Patient patient = new Patient("John Doe", 40, "Flu");
nurse.displayDetails();
nurse.role();
nurse.assistDoctor();
patient.displayDetails();
patient.role();

System.out.println("\n===== MULTIPLE INHERITANCE (via Interfaces) =====");
AcademicSurgeon acadSurg = new AcademicSurgeon("Dr. Green", 55, "Orthopedics");
acadSurg.displayDetails();
acadSurg.role();
acadSurg.treatPatient();
acadSurg.performSurgery();
acadSurg.research();
acadSurg.teach();

System.out.println("\n===== HYBRID INHERITANCE =====");
Intern intern = new Intern("Emily", 24);
intern.displayDetails();
```

```
        intern.role();
        intern.study();
        intern.work();

        System.out.println("\n===== FINAL CLASS =====");
        HospitalPolicy policy = new HospitalPolicy();
        policy.displayPolicy();
    }
}
```




Command Prompt
Dr. Brown is treating patients.
Dr. Brown also performs surgical treatment.
Dr. Brown is performing surgery.
===== HIERARCHICAL INHERITANCE =====
Name: Alice, Age: 30
Alice is a nurse.
Alice is assisting a doctor.
Name: John Doe, Age: 40
John Doe is a patient suffering from Flu.
===== MULTIPLE INHERITANCE (via Interfaces) =====
Name: Dr. Green, Age: 55
Dr. Green is a doctor specialized in Orthopedics.
Dr. Green is treating patients.
Dr. Green also performs surgical treatment.
Dr. Green is performing surgery.
Dr. Green is researching new surgical techniques.
Dr. Green is teaching medical students.
===== HYBRID INHERITANCE =====
Name: Emily, Age: 24
Emily is both a student and an intern.
Emily is studying medicine.
Emily is working as an intern in the hospital.
===== FINAL CLASS =====
Hospital Policy: Maintain hygiene and patient confidentiality.

Polymorphism

Program:

Overloading → generateReceipt()

Overriding → pay()

Dynamic dispatch → array/collection of Payment

Hybrid polymorphism → RefundableCreditCardPayment (class + interface)

super keyword → LoggedCreditCardPayment

Bank class using polymorphism → accepts all payment types dynamically

```
// ================================================
// Bank Payment Processing System - Polymorphism Demo
// ================================================

import java.util.*;

// ----- Abstract Base Class -----
abstract class Payment {
    protected double amount;

    Payment(double amount) {
        this.amount = amount;
    }

    // Abstract method - must be overridden
    abstract void pay();

    // Compile-time Polymorphism (Overloading)
    void generateReceipt() {
        System.out.println("Receipt: Payment of $" + amount);
    }
```

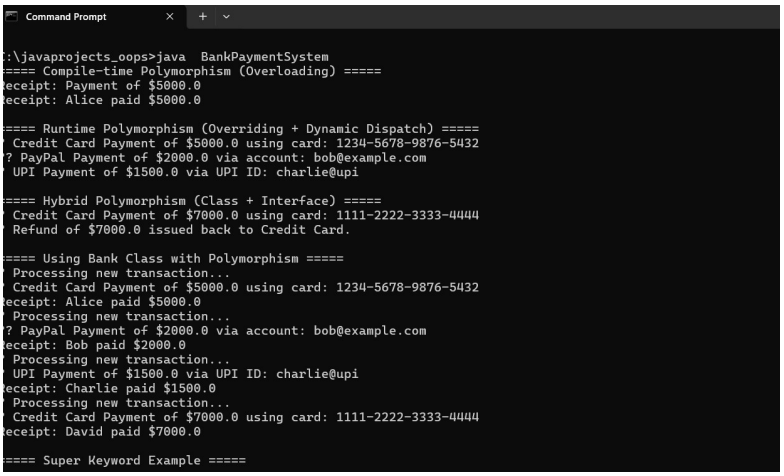
```
    void generateReceipt(String customerName) {
        System.out.println("Receipt: " + customerName + " paid $" + amount);
    }
}

// ----- Derived Class 1 -----
class CreditCardPayment extends Payment {
    private String cardNumber;

    CreditCardPayment(double amount, String cardNumber) {
        super(amount);
        this.cardNumber = cardNumber;
    }

    @Override
    void pay() {
        System.out.println("💳 Credit Card Payment of $" + amount + " using card: " + cardNumber);
    }
}

// ----- Derived Class 2 -----
class PayPalPayment extends Payment {
    private String email;

    PayPalPayment(double amount, String email) {
        super(amount);
        this.email = email;
    }

    @Override
    void pay() {
```

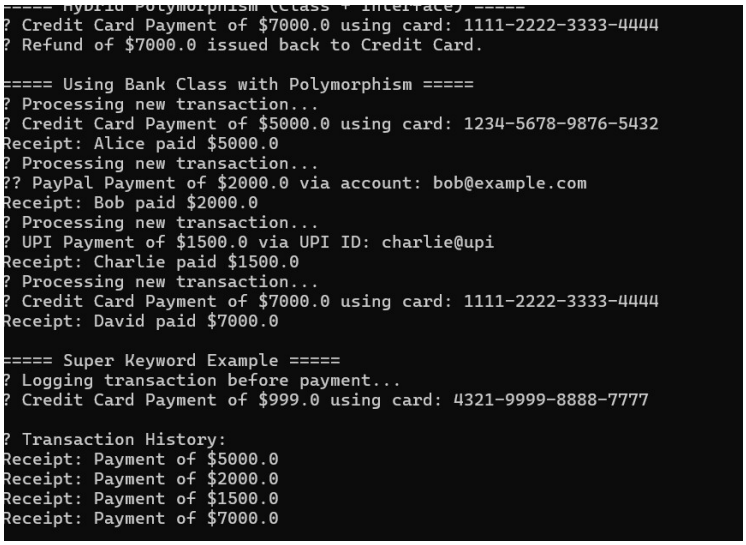
```
        System.out.println("🅿️ PayPal Payment of $" + amount + " via account: " + email);
    }
}

// ----- Derived Class 3 -----
class UpiPayment extends Payment {
    private String upiId;

    UpiPayment(double amount, String upiId) {
        super(amount);
        this.upiId = upiId;
    }

    @Override
    void pay() {
        System.out.println("📱 UPI Payment of $" + amount + " via UPI ID: " + upiId);
    }
}

// ----- Interface for Hybrid Polymorphism -----
interface Refundable {
    void refund();
}

// ----- Class implementing interface + extending class -----
class RefundableCreditCardPayment extends CreditCardPayment implements Refundable {
    RefundableCreditCardPayment(double amount, String cardNumber) {
        super(amount, cardNumber);
    }

    @Override
```

```
    public void refund() {
        System.out.println(" 💰 Refund of $" + amount + " issued back to Credit Card.");
    }
}

// ----- Bank class using Polymorphism -----
class Bank {
    private List<Payment> transactions = new ArrayList<>();

    // Accepts any payment dynamically (Polymorphism)
    void processPayment(Payment p) {
        System.out.println(" ✅ Processing new transaction...");
        p.pay();            // Dynamic dispatch
        p.generateReceipt();  // Polymorphism (overloaded)
        transactions.add(p);
    }

    void processPayment(Payment p, String customerName) {
        System.out.println(" ✅ Processing new transaction...");
        p.pay();
        p.generateReceipt(customerName);
        transactions.add(p);
    }

    void showAllTransactions() {
        System.out.println("\n 📊 Transaction History:");
        for (Payment p : transactions) {
            p.generateReceipt();
        }
    }
}
```

```
// ----- Super Keyword Demonstration -----
class LoggedCreditCardPayment extends CreditCardPayment {
    LoggedCreditCardPayment(double amount, String cardNumber) {
        super(amount, cardNumber);  // calling parent constructor
    }

    @Override
    void pay() {
        System.out.println(" 📝 Logging transaction before payment...");
        super.pay();  // calling parent method
    }
}

// ----- Main Application -----
public class BankPaymentSystem {
    public static void main(String[] args) {

        // Create Bank
        Bank bank = new Bank();

        System.out.println("===== Compile-time Polymorphism (Overloading) =====");
        Payment p1 = new CreditCardPayment(5000, "1234-5678-9876-5432");
        p1.generateReceipt();
        p1.generateReceipt("Alice");

        System.out.println("\n===== Runtime Polymorphism (Overriding + Dynamic Dispatch) =====");
        Payment p2 = new PayPalPayment(2000, "bob@example.com");
        Payment p3 = new UpiPayment(1500, "charlie@upi");

        // Store all in parent reference
```

```
        Payment[] payments = {p1, p2, p3};
        for (Payment pay : payments) {
            pay.pay();  // runtime polymorphism
        }

        System.out.println("\n===== Hybrid Polymorphism (Class + Interface) =====");
        RefundableCreditCardPayment r1 = new RefundableCreditCardPayment(7000, "1111-2222-
3333-4444");
        r1.pay();
        r1.refund();

        System.out.println("\n===== Using Bank Class with Polymorphism =====");
        bank.processPayment(p1, "Alice");
        bank.processPayment(p2, "Bob");
        bank.processPayment(p3, "Charlie");
        bank.processPayment(r1, "David");

        System.out.println("\n===== Super Keyword Example =====");
        Payment demo = new LoggedCreditCardPayment(999, "4321-9999-8888-7777");
        demo.pay();

        // Show all stored transactions
        bank.showAllTransactions();
    }
}
```


Command Prompt
C:\javaprojects_oops>java BankPaymentSystem
===== Compile-time Polymorphism (Overloading) =====
Receipt: Payment of $5000.0
Receipt: Alice paid $5000.0
===== Runtime Polymorphism (Overriding + Dynamic Dispatch) =====
? Credit Card Payment of $5000.0 using card: 1234-5678-9876-5432
?? PayPal Payment of $2000.0 via account: bob@example.com
? UPI Payment of $1500.0 via UPI ID: charlie@upi
===== Hybrid Polymorphism (Class + Interface) =====
? Credit Card Payment of $7000.0 using card: 1111-2222-3333-4444
? Refund of $7000.0 issued back to Credit Card.
===== Using Bank Class with Polymorphism =====
? Processing new transaction...
? Credit Card Payment of $5000.0 using card: 1234-5678-9876-5432
Receipt: Alice paid $5000.0
? Processing new transaction...
?? PayPal Payment of $2000.0 via account: bob@example.com
Receipt: Bob paid $2000.0
? Processing new transaction...
? UPI Payment of $1500.0 via UPI ID: charlie@upi
Receipt: Charlie paid $1500.0
? Processing new transaction...
? Credit Card Payment of $7000.0 using card: 1111-2222-3333-4444
Receipt: David paid $7000.0
===== Super Keyword Example =====


===== Hybrid Polymorphism (Class + Interface) =====
? Credit Card Payment of $7000.0 using card: 1111-2222-3333-4444
? Refund of $7000.0 issued back to Credit Card.
===== Using Bank Class with Polymorphism =====
? Processing new transaction...
? Credit Card Payment of $5000.0 using card: 1234-5678-9876-5432
Receipt: Alice paid $5000.0
? Processing new transaction...
?? PayPal Payment of $2000.0 via account: bob@example.com
Receipt: Bob paid $2000.0
? Processing new transaction...
? UPI Payment of $1500.0 via UPI ID: charlie@upi
Receipt: Charlie paid $1500.0
? Processing new transaction...
? Credit Card Payment of $7000.0 using card: 1111-2222-3333-4444
Receipt: David paid $7000.0
===== Super Keyword Example =====
? Logging transaction before payment...
? Credit Card Payment of $999.0 using card: 4321-9999-8888-7777
? Transaction History:
Receipt: Payment of $5000.0
Receipt: Payment of $2000.0
Receipt: Payment of $1500.0
Receipt: Payment of $7000.0