1. Create Multilevel Inheritance for Vehicle

```
class Vehicle {
    void display() {
        System.out.println("This is a Vehicle");
    }
}

class Four_wheeler extends Vehicle {
    void type() {
        System.out.println("It is a Four Wheeler");
    }
}

class Petrol_Four_Wheeler extends Four_wheeler {
    void fuel() {
        System.out.println("Runs on Petrol");
    }
}

class FiveSeater_Petrol_Four_Wheeler extends Petrol_Four_Wheeler {
    void seat() {
        System.out.println("Has 5 Seats");
    }
}

class Baleno_FiveSeater_Petrol_Four_Wheeler extends FiveSeater_Petrol_Four_Wheeler {
    void model() {
        System.out.println("Model: Baleno");
    }
}

public class Main1 {
```

```
    public static void main(String[] args) {
        Baleno_FiveSeater_Petrol_Four_Wheeler
b = new Baleno_FiveSeater_Petrol_Four_Wheeler();
        b.display();
        b.type();
        b.fuel();
        b.seat();
        b.model();
    }
}
```

Output:

```
This is a Vehicle
It is a Four Wheeler
Runs on Petrol
Has 5 Seats
Model: Baleno
```

2.Demonstrate the use of the super keyword

```
class Parent {
    Parent() {
        System.out.println("Parent Constructor");
    }
    void show() {
        System.out.println("Parent Method");
    }
}
class Child extends Parent {
    Child() {
        super();
        System.out.println("Child Constructor");
    }
    void show() {
```

```
        super.show();
        System.out.println("Child Method");
    }
}
public class Main2 {
    public static void main(String[] args) {
        Child c = new Child();
        c.show();
    }
}
```

Output:

Parent Constructor

Child Constructor

Parent Method

Child Method

3.Create Hospital super class and access this class inside the patient child class and access properties from Hospital class.

```
class Hospital {
    String name = "City Hospital";
    String location = "Bengaluru";
}

class Patient extends Hospital {
    String patientName = "Likitha";
    void displayInfo() {
        System.out.println("Hospital: " + name);
        System.out.println("Location: " + location);
        System.out.println("Patient: " + patientName);
    }
}
public class Main3 {
    public static void main(String[] args) {
```

```
        Patient p = new Patient();
        p.displayInfo();
    }
}
```

Output:

Hospital: City Hospital

Location: Bengaluru

Patient: Likitha

4.Create Hierarchical inheritance

```
class Animal {
    void eat() {
        System.out.println("Animal eats food");
    }
}
class Dog extends Animal {
    void bark() {
        System.out.println("Dog barks");
    }
}
class Cat extends Animal {
    void meow() {
        System.out.println("Cat meows");
    }
}
public class Main4 {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.eat();
        d.bark();
 Cat c = new Cat();
        c.eat();
```

```
        c.meow();
    }
}
```

Output:

Animal eats food

Dog barks

Animal eats food

Cat meows

5.Create a class Calculator with the following overloaded add()

```
class Calculator {
    void add(int a, int b) {
        System.out.println("Sum: " + (a + b));
    }
    void add(int a, int b, int c) {
        System.out.println("Sum: " + (a + b + c));
    }
    void add(double a, double b) {
        System.out.println("Sum: " + (a + b));
    }
}

public class Main5 {
    public static void main(String[] args) {
        Calculator c = new Calculator();
        c.add(5, 10);
        c.add(5, 10, 15);
        c.add(5.5, 10.5);
    }
}
```

Output:

Sum: 15

Sum: 30

Sum: 16.0

6.Create a base class Shape and two subclasses Circle and Rectangle by using Extend Keyword and Print Area of Respective Type.

```
class Shape {
    void area() {
        System.out.println("Calculating area...");
    }
}

class Circle extends Shape {
    void area() {
        double r = 5;
        System.out.println("Area of Circle: " + (3.14 * r * r));
    }
}

class Rectangle extends Shape {
    void area() {
        double l = 4, b = 6;
        System.out.println("Area of Rectangle: " + (l * b));
    }
}

public class Main6 {
    public static void main(String[] args) {
        Shape s = new Circle();
        s.area();

        s = new Rectangle();
        s.area();
    }
```

```
}
```

Output:

Area of Circle: 78.5

Area of Rectangle: 24.0

---

7. Bank Interest Rates

```
class Bank {
    double getInterestRate() {
        return 0;
    }
}

class SBI extends Bank {
    double getInterestRate() {
        return 6.7;
    }
}

class ICICI extends Bank {
    double getInterestRate() {
        return 7.0;
    }
}

class HDFC extends Bank {
    double getInterestRate() {
        return 7.5;
    }
}

public class Main7 {
    public static void main(String[] args) {
```

```
        Bank b = new SBI();
        System.out.println("SBI Rate: " + b.getInterestRate() + "%");

        b = new ICICI();
        System.out.println("ICICI Rate: " + b.getInterestRate() + "%");

        b = new HDFC();
        System.out.println("HDFC Rate: " + b.getInterestRate() + "%");
    }
}
```

Output:

SBI Rate: 6.7%

ICICI Rate: 7.0%

HDFC Rate: 7.5%

---

4. Create a program for Runtime Polymorphism with Constructor Chaining

```
class Vehicle {
    Vehicle() {
        System.out.println("Vehicle Created");
    }
    void run() {
        System.out.println("Vehicle is running");
    }
}

class Bike extends Vehicle {
    Bike() {
        super();
        System.out.println("Bike Created");
    }
    void run() {
        super.run();
```

```
        System.out.println("Bike is running");
    }
}

public class Main8 {
    public static void main(String[] args) {
        Bike b = new Bike();
        b.run();
    }
}
```

Output:

Vehicle Created

Bike Created

Vehicle is running

Bike is running

---

1. Abstract Class SmartDevice with Polymorphism

```
abstract class SmartDevice {
    abstract void turnOn();
    abstract void turnOff();
    abstract void performFunction();
}

class SmartPhone extends SmartDevice {
    void turnOn() { System.out.println("SmartPhone On"); }
    void turnOff() { System.out.println("SmartPhone Off"); }
    void performFunction() { System.out.println("Calling & Browsing"); }
}

class SmartWatch extends SmartDevice {
    void turnOn() { System.out.println("SmartWatch On"); }
```

```
    void turnOff() { System.out.println("SmartWatch Off"); }
    void performFunction() { System.out.println("Tracking Fitness & Time"); }
}

class SmartSpeaker extends SmartDevice {
    void turnOn() { System.out.println("SmartSpeaker On"); }
    void turnOff() { System.out.println("SmartSpeaker Off"); }
    void performFunction() { System.out.println("Playing Music & Voice Commands"); }
}

public class Main9 {
    public static void main(String[] args) {
        SmartDevice[] devices = {new SmartPhone(), new SmartWatch(), new SmartSpeaker()};
        for (SmartDevice d : devices) {
            d.turnOn();
            d.performFunction();
            d.turnOff();
        }
    }
}
```

Output:

```
SmartPhone On
Calling & Browsing
SmartPhone Off
SmartWatch On
Tracking Fitness & Time
SmartWatch Off
SmartSpeaker On
Playing Music & Voice Commands
SmartSpeaker Off
```

---

7. Bank Interface with Savings & Current Account

```
interface Bank {
    void deposit(double amount);
    void withdraw(double amount);
    double getBalance();
}
class Account {
    double balance;
    Account(double balance) { this.balance = balance; }
}
class SavingsAccount extends Account implements Bank {
    double minBalance = 500;
    SavingsAccount(double balance) { super(balance); }
    public void deposit(double amount) { balance += amount; }
    public void withdraw(double amount) {
        if (balance - amount >= minBalance)
{ balance -= amount;}
        else {System.out.println("Minimum balance required");}
    }
    public double getBalance() { return balance; }
}

class CurrentAccount extends Account implements Bank {
    CurrentAccount(double balance) { super(balance); }
    public void deposit(double amount) { balance += amount; }
    public void withdraw(double amount) { balance -= amount; }
    public double getBalance() { return balance; }
}

public class Main10 {
    public static void main(String[] args) {
        Bank s = new SavingsAccount(1000);
        s.deposit(500);
```

```
        s.withdraw(800);
        System.out.println("Savings Balance: " + s.getBalance());

        Bank c = new CurrentAccount(2000);
        c.withdraw(500);
        System.out.println("Current Balance: " + c.getBalance());
    }
}
```

Output:

Minimum balance required

Savings Balance: 1500.0

Current Balance: 1500.0

---

8. Vehicle → Car, Bike, Truck with Static Method

```
class Vehicle1 {
    void start() {
        System.out.println("Vehicle starting");
    }
}

class Car extends Vehicle1 {
    void start() {
        System.out.println("Car starting");
    }
}
class Bike1 extends Vehicle1 {
    void start() {
        System.out.println("Bike starting");
    }
}
class Truck extends Vehicle1 {
    void start() {
```

```
        System.out.println("Truck starting");
    }
}
public class Main11 {
    static void testVehicle(Vehicle1 v) {
        v.start();
    }
    public static void main(String[] args) {
        testVehicle(new Car());
        testVehicle(new Bike1());
        testVehicle(new Truck());
    }
}
```

Output:

Car starting

Bike starting

Truck starting

---

9. a program for a Abstract Person → Student, Professor, TeachingAssistant

```
abstract class Person {
    String name;
    int age;
    Person(String name, int age) { this.name = name; this.age = age; }
    abstract void getRoleInfo();
}

class Student extends Person {
    String course;
    int roll;
    Student(String name, int age, String course, int roll) {
        super(name, age);
        this.course = course; this.roll = roll;
```

```
    }
    void getRoleInfo() {
        System.out.println("Student: " + name + ", " + age + ", " + course + ", Roll: " + roll);
    }
}

class Professor extends Person {
    String subject;
    double salary;
    Professor(String name, int age, String subject, double salary) {
        super(name, age);
        this.subject = subject; this.salary = salary;
    }
    void getRoleInfo() {
        System.out.println("Professor: " + name + ", " + age + ", " + subject + ", Salary: " + salary);
    }
}

class TeachingAssistant extends Student {
    TeachingAssistant(String name, int age, String course, int roll) {
        super(name, age, course, roll);
    }
    void getRoleInfo() {
        System.out.println("Teaching Assistant: " + name + ", " + age + ", " + course + ", Roll: " + roll);
    }
}

public class Main12 {
    public static void main(String[] args) {
        Person p1 = new Student("Likitha", 22, "ECE", 101);
        Person p2 = new Professor("Ravi", 45, "Java", 80000);
        Person p3 = new TeachingAssistant("Meera", 23, "CSE", 202);
```

```
        p1.getRoleInfo();
        p2.getRoleInfo();
        p3.getRoleInfo();
    }
}
```

Output:

Student: Likitha, 22, ECE, Roll: 101

Professor: Ravi, 45, Java, Salary: 80000.0

Teaching Assistant: Meera, 23, CSE, Roll: 202

10.Write a program for Interface Drawable Abstract Shape

```
interface Drawable {
    void draw();
}

abstract class Shape2 {
    abstract double area();
}

class Circle2 extends Shape2 implements Drawable {
    double r = 5;
    public void draw() { System.out.println("Drawing Circle"); }
    double area() { return 3.14 * r * r; }
}

class Rectangle2 extends Shape2 implements Drawable {
    double l = 4, b = 6;
    public void draw() { System.out.println("Drawing Rectangle"); }
    double area() { return l * b; }
}

class Triangle2 extends Shape2 implements Drawable {
```

```
    double base = 4, height = 3;
    public void draw() { System.out.println("Drawing Triangle"); }
    double area() { return 0.5 * base * height; }
}

public class Main13 {
    public static void main(String[] args) {
        Drawable[] shapes = {new Circle2(), new Rectangle2(), new Triangle2()};
        for (Drawable d : shapes) {
            d.draw();
            Shape2 s = (Shape2) d;
            System.out.println("Area: " + s.area());
        }
    }
}
```

Output:

```
Drawing Circle
Area: 78.5
Drawing Rectangle
Area: 24.0
Drawing Triangle
Area: 6.0
```