Program 1 : Single Inheritance

```
class Animal {
    void eat() {
        System.out.println("Animal eats food");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println("Dog barks");
    }
}

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

Output :

```
Animal eats food
Dog barks
```

Program 2 : Multilevel Inheritance

```
class Grandparent {
    void show() {
        System.out.println("Grandparent class");
    }
}
```

```
class Parent extends Grandparent {
    void display() {
        System.out.println("Parent class");
    }
}

class Child extends Parent {
    void print() {
        System.out.println("Child class");
    }
}

public class Multilevel {
    public static void main(String[] args) {
        Child c = new Child();
        c.show();
        c.display();
        c.print();
    }
}
```

Output :

```
Grandparent class
Parent class
Child class
```

Program 3 : Hierarchical Inheritance

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

```
class Bike extends Vehicle {
    void ride() {
        System.out.println("Bike is riding");
    }
}

class Car extends Vehicle {
    void drive() {
        System.out.println("Car is driving");
    }
}

public class Hierarchical {
    public static void main(String[] args) {
        Bike b = new Bike();
        Car c = new Car();

        b.start();
        b.ride();

        c.start();
        c.drive();
    }
}
```

Output :

```
Vehicle starts
Bike is riding
Vehicle starts
Car is driving
```

Program 4 : Multiple Inheritance using Interface

```java
interface A {
    void methodA();
}

interface B {
    void methodB();
}

class C implements A, B {
    public void methodA() {
        System.out.println("Method from A");
    }

    public void methodB() {
        System.out.println("Method from B");
    }
}

public class MultipleInheritance {
    public static void main(String[] args) {
        C obj = new C();
        obj.methodA();
        obj.methodB();
    }
}
```

Output :

```
Method from A
Method from B
```

Program 5 : Method Overloading

```
class Calculator {
    int add(int a, int b) {
        return a + b;
    }

    int add(int a, int b, int c) {
        return a + b + c;
    }
}

public class Overloading {
    public static void main(String[] args) {
        Calculator c = new Calculator();
        System.out.println(c.add(10, 20));
        System.out.println(c.add(10, 20, 30));
    }
}
```

Output :

```
30
60
```

Program 6 : Method Overriding

```
class Bank {
    void interest() {
        System.out.println("Bank interest");
    }
}

class SBI extends Bank {
    void interest() {
        System.out.println("SBI interest is 6%");
    }
}

public class Overriding {
    public static void main(String[] args) {
        Bank b = new SBI();
        b.interest();
    }
}
```

Output :

```
SBI interest is 6%
```

Program 7 : Polymorphism using Interface

```
interface Animal {
    void sound();
}

class Dog implements Animal {
    public void sound() {
        System.out.println("Dog barks");
    }
}

class Cat implements Animal {
    public void sound() {
        System.out.println("Cat meows");
    }
}

public class InterfacePoly {
    public static void main(String[] args) {
        Animal a;

        a = new Dog();
        a.sound();

        a = new Cat();
        a.sound();
    }
}
```

Output :

```
Dog barks
Cat meows
```

Program 8 : Super Keyword

```
class Parent {
    void show() {
        System.out.println("Parent method");
    }
}

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

public class SuperKeyword {
    public static void main(String[] args) {
        Child c = new Child();
        c.show();
    }
}
```

Output :

```
Parent method
Child method
```

Program 9 : Final Method (Cannot Override)

```
class Parent {
    final void show() {
        System.out.println("Final method in parent");
    }
}

class Child extends Parent {
    // cannot override show()
}

public class FinalMethod {
    public static void main(String[] args) {
        Child c = new Child();
        c.show();
    }
}
```

Output :

```
Final method in parent
```

Program 10 : This Keyword

```
class Student {
    int id;
    String name;
    void setData(int id, String name) {
        this.id = id;       // refers to current object
        this.name = name;
    }

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

public class ThisKeyword {
    public static void main(String[] args) {
        Student s = new Student();
        s.setData(101, "Amit");
        s.display();
    }
}
```

Output :

```
ID: 101
Name: Amit
```

Program 11 : Print Object Address Using This

```
class Demo {
    void show() {
        System.out.println(this);
    }
}

public class ThisAddress {
    public static void main(String[] args) {
        Demo d1 = new Demo();
        Demo d2 = new Demo();

        System.out.println(d1);
        System.out.println(d2);

        d1.show();
        d2.show();
    }
}
```

Output :

```
Demo@3feba861
Demo@5b480cf9
Demo@3feba861
Demo@5b480cf9
```
