Unit 3 code

- **Inheritance (Single, Multilevel, Hierarchical)**
- **Method Overriding**
- **Super Keyword**
- **Final Keyword**
- **Dynamic Method Dispatch (Runtime Polymorphism)**
- **Abstract Classes and Methods**
- **Interfaces**

☑ 1. Inheritance (Single, Multilevel, Hierarchical)

➤ Single Inheritance

```
java
CopyEdit
class Animal {
    void sound() {
        System.out.println("Animal makes sound");
    }
}

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.sound();  // from Animal
        d.bark();   // from Dog
    }
}
```

➤ Multilevel Inheritance

```
java
CopyEdit
class Grandparent {
    void show1() {
        System.out.println("I am Grandparent");
    }
}

class Parent extends Grandparent {
    void show2() {
```

```
        System.out.println("I am Parent");
    }
}

class Child extends Parent {
    void show3() {
        System.out.println("I am Child");
    }
}

public class MultilevelInheritance {
    public static void main(String[] args) {
        Child c = new Child();
        c.show1();
        c.show2();
        c.show3();
    }
}
```

➤ Hierarchical Inheritance

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

class Bike extends Vehicle {
    void type() {
        System.out.println("Bike");
    }
}

class Car extends Vehicle {
    void type() {
        System.out.println("Car");
    }
}

public class HierarchicalInheritance {
    public static void main(String[] args) {
        Bike b = new Bike();
        b.run();
        b.type();

        Car c = new Car();
        c.run();
        c.type();
    }
}
```

☑ 2. Method Overriding

```
java
CopyEdit
class Parent {
```

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

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

public class MethodOverriding {
    public static void main(String[] args) {
        Child c = new Child();
        c.show();  // Calls overridden method
    }
}
```

☑ 3. Super Keyword

```
java
CopyEdit
class Vehicle {
    int speed = 50;
}

class Bike extends Vehicle {
    int speed = 100;

    void displaySpeed() {
        System.out.println("Child speed: " + speed);
        System.out.println("Parent speed: " + super.speed); // refers to
parent class variable
    }
}

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

☑ 4. Final Keyword

```
java
CopyEdit
final class Shape {
    final int sides = 4;

    final void show() {
        System.out.println("Shape has " + sides + " sides.");
    }
}

public class FinalKeyword {
```

```
    public static void main(String[] args) {
        Shape s = new Shape();
        s.show();
    }
}
```

☑ 5. Dynamic Method Dispatch (Runtime Polymorphism)

```
java
CopyEdit
class Animal {
    void sound() {
        System.out.println("Animal sound");
    }
}

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

public class RuntimePolymorphism {
    public static void main(String[] args) {
        Animal a = new Dog(); // parent ref, child object
        a.sound(); // calls Dog's sound()
    }
}
```

☑ 6. Abstract Classes and Methods

```
java
CopyEdit
abstract class Animal {
    abstract void sound(); // abstract method

    void eat() {
        System.out.println("Animals eat food");
    }
}

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

public class AbstractExample {
    public static void main(String[] args) {
        Cat c = new Cat();
        c.sound();
        c.eat();
    }
}
```

☑ 7. Interfaces

```
java
CopyEdit
interface Drawable {
    void draw(); // abstract method
}

class Circle implements Drawable {
    public void draw() {
        System.out.println("Drawing Circle");
    }
}

public class InterfaceExample {
    public static void main(String[] args) {
        Circle c = new Circle();
        c.draw();
    }
}
```