INHERITANCE

Inheritance is one of the four fundamental principles of Object-Oriented Programming (OOP).

It allows a class to inherit properties and behaviors (fields and methods) from another class. The class that inherits the properties is called the subclass (or derived class), and the class from which properties are inherited is called the superclass

(or base class).

Inheritance promotes code reusability and establishes a natural

hierarchical relationship between classes.

1. What is Inheritance?

Inheritance is a mechanism wherein a new class is derived from

an existing class.

The derived class (child class) inherits the attributes and methods

of the base class (parent class),allowing code reuse and the

creation of a natural hierarchy.

2. Benefits of Inheritance

Code Reusability: Inheritance allows a class to reuse the fields and methods of another class.

Method Overriding: Subclasses can provide specific implementations for methods that are defined in the parent class.

Polymorphism: Inheritance supports polymorphism, allowing objects to be treated as instances of their parent class.

3. Types of Inheritance

Single Inheritance: A class inherits from one superclass.

Multilevel Inheritance: A class inherits from a superclass, and another class inherits from this derived class.

Hierarchical Inheritance: Multiple classes inherit from a single superclass.

Multiple Inheritance: A class inherits from more than one superclass.

Java does not support multiple inheritance directly to avoid complexity

and ambiguity. However, it can be achieved using interfaces.

==========Examples==========

Example: Single Inheritance

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

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

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat();  // Output: This animal eats food.
        dog.bark(); // Output: The dog barks.
    }
}
```

Example: Multilevel Inheritance

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

class Dog extends Animal {
```

```
    void bark() {
        System.out.println("The dog barks.");
    }
}

class Puppy extends Dog {
    void weep() {
        System.out.println("The puppy weeps.");
    }
}

public class Main {
    public static void main(String[] args) {
        Puppy puppy = new Puppy();
        puppy.eat();  // Output: This animal eats food.
        puppy.bark(); // Output: The dog barks.
        puppy.weep(); // Output: The puppy weeps.
    }
}
```

Example: Hierarchical Inheritance

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

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

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

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat();  // Output: This animal eats food.
        dog.bark(); // Output: The dog barks.

        Cat cat = new Cat();
```

```
        cat.eat();  // Output: This animal eats food.
        cat.meow(); // Output: The cat meows.
    }
}
```