Lab 26

Classroom Activity: Demonstrate inheritance in Java and Types

Inheritance is one of the four fundamental concepts of object-oriented programming (OOP) and plays a crucial role in Java. Inheritance allows a class (the subclass or derived class) to inherit properties and behaviors (fields and methods) from another class (the superclass or base class). This promotes code reuse and supports the concept of "is-a" relationships between classes. Java supports single inheritance for classes, meaning a class can extend only one superclass, but it supports multiple inheritance for interfaces.

Here's a demonstration of inheritance in Java along with some common types:

1. Single Inheritance:

class Animal {

void eat() {

System.out.println("Animal is eating...");

}

}

class Dog extends Animal {

void bark() {

System.out.println("Dog is barking...");

}

}

public class InheritanceDemo {

public static void main(String[] args) {

Dog myDog = new Dog();

myDog.eat(); // Inherited from Animal class

myDog.bark(); // Unique to Dog class

}

}

In this example, `Dog` is a subclass of `Animal`. The `Dog` class inherits the `eat` method from the `Animal` class and adds its own method `bark`.

2. Multilevel Inheritance:

class Animal {

void eat() {

System.out.println("Animal is eating...");

}

}

class Mammal extends Animal {

void sleep() {

System.out.println("Mammal is sleeping...");

}

}

class Dog extends Mammal {

void bark() {

System.out.println("Dog is barking...");

}

}

public class InheritanceDemo {

public static void main(String[] args) {

Dog myDog = new Dog();

myDog.eat(); // Inherited from Animal

myDog.sleep(); // Inherited from Mammal

myDog.bark(); // Unique to Dog

}

}

In this example, `Mammal` is a subclass of `Animal`, and `Dog` is a subclass of `Mammal`. This is an example of multilevel inheritance, where each subclass inherits from the superclass above it.

3. Interface Inheritance (Multiple Inheritance):

interface Flyable {

void fly();

}

interface Swimmable {

void swim();

}

class Bird implements Flyable {

public void fly() {

System.out.println("Bird is flying...");

}

}

class Fish implements Swimmable {

public void swim() {

System.out.println("Fish is swimming...");

}

}

public class InheritanceDemo {

public static void main(String[] args) {

Bird myBird = new Bird();

Fish myFish = new Fish();

myBird.fly();

myFish.swim();

}

}

In this example, we have two interfaces, `Flyable` and `Swimmable`, each defining a method. The `Bird` class implements `Flyable`, and the `Fish` class implements `Swimmable`. This demonstrates interface-based inheritance, allowing classes to implement multiple interfaces.

4. Hierarchical Inheritance:

class Animal {

void eat() {

System.out.println("Animal is eating...");

}

}

class Cat extends Animal {

void meow() {

System.out.println("Cat is meowing...");

}

}

class Dog extends Animal {

void bark() {

System.out.println("Dog is barking...");

}

}

public class InheritanceDemo {

public static void main(String[] args) {

Cat myCat = new Cat();

Dog myDog = new Dog();

myCat.eat(); // Inherited from Animal

myCat.meow(); // Unique to Cat

myDog.eat(); // Inherited from Animal

myDog.bark(); // Unique to Dog

}

}

In this example, both `Cat` and `Dog` classes inherit from the same `Animal` superclass. This demonstrates hierarchical inheritance, where multiple classes inherit from a single superclass.

These examples showcase various types of inheritance in Java, including single, multilevel, interface-based (multiple), and hierarchical inheritance. Each type of inheritance has its use cases and helps in building a structured and reusable codebase in object-oriented programming.