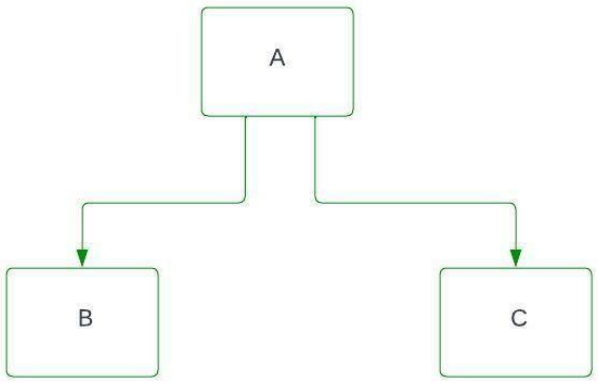
Inheritance in Java

Hierarchical inheritance in Java

Inheritance is a feature of Object-Oriented-programming in which a derived class (child class) inherits the property (data member and member functions) of the Base class (parent class). For example, a child inherits the traits of their parents.

In Hierarchical inheritance, more than one sub-class inherits the property of a single base class. There is one base class and multiple derived classes. Several other classes inherit the derived classes as well. Hierarchical structures thus form a tree-like structure. It is similar to that, mango and apple both are fruits; both inherit the property of fruit. Fruit will be the Base class, and mango and apple are sub-classes.

The below diagram shows, Class A is a Base class, B is a subclass inherited from class A, and C is a subclass it also inherits from class A.



Hierarchical inheritance in Java is a type of inheritance where multiple subclasses inherit from a single superclass. This allows for code reuse and a structured organization of classes.

Example of Hierarchical Inheritance

Let's say we have a superclass called `Animal`, and two subclasses called `Dog` and `Cat`.

PROGRAM :

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

// Subclass 1
class Dog extends Animal
{
	void bark()
	{
		System.out.println("The dog barks.");
	}
}

// Subclass 2
class Cat extends Animal
{
	void meow()
	{
		System.out.println("The cat meows.");
	}
}

// Main class to test the hierarchy
public class Animals_Test
{
	public static void main(String[] args)
	{
		Dog dog = new Dog();
		Cat cat = new Cat();

		// Calling methods from the superclass
		dog.eat();		// Output: This animal eats food.
		cat.eat();		// Output: This animal eats food.

		// Calling methods from the subclasses
		dog.bark();		// Output: The dog barks.
		cat.meow();		// Output: The cat meows.
	}
}
```

Explanation

1. **Superclass** (`Animal`): This class has a method `eat()` that can be inherited by its subclasses.

2. **Subclasses** (`Dog` and `Cat`): Both classes extend the `Animal` class and can use the `eat()` method while also having their own specific methods (`bark()` for `Dog` and `meow()` for `Cat`).

3. **Main Class**: In the `main` method, we create instances of `Dog` and `Cat`, demonstrating how both subclasses can access the `eat()` method from the `Animal` superclass, as well as their own unique methods.

This structure illustrates how hierarchical inheritance can help in organizing related classes and promoting code reuse.