WEEK 08

What is Inheritance?

Inheritance is a feature in Java where **one class gets the properties and methods of another class.**

The new class that is created is known as subclass (child or derived class) and the existing class from where the child class is derived is known as superclass (parent or base class).

The **extends keyword** is used to perform inheritance in Java.

Syntax :

```
class derived-class extends base-class
{
//methods and fields
}
```

For example

```
import java.io.*;

// Base or Super Class
class Employee
{
         int salary = 60000;
}

// Inherited or Sub Class
class Engineer extends Employee
{
        int bonus = 10000;
        void display()
        {
                System.out.println("Salary  = " + salary);
                System.out.println("Bonus  = " + bonus);
        }
}
```

```
// Driver Class
class MainInheritance
{
	public static void main(String args[])
	{
		Engineer E1 = new Engineer();
		E1.display();
	}
}
```

is-a relationship

In Java, inheritance is an is-a relationship. That is, we use inheritance only if there exists an is-a relationship between two classes.

For example,

Car is a Vehicle
Orange is a Fruit
Surgeon is a Doctor
Dog is an Animal

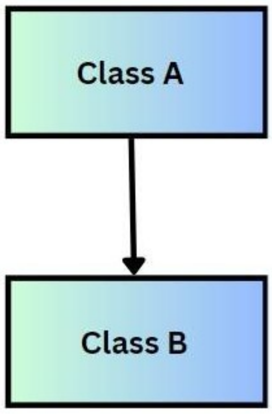
Here, Car can inherit from Vehicle; Orange can inherit from Fruit, and so on.

Types of inheritance

There are five types of inheritance.

1. Single Inheritance

In single inheritance, a single subclass extends from a single superclass.



Example: Animal class that has a method eat(). Then, we create **Dog** class that inherits from **Animal**.

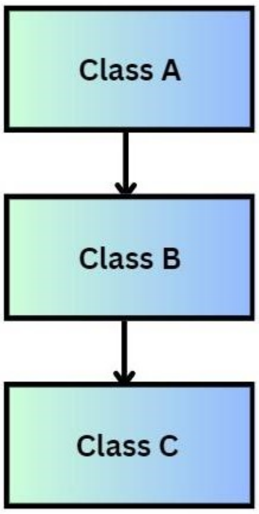
```
// Parent class
class Animal
{
	void eat()
	{
		System.out.println("This animal eats food.");
	}
}

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

// Main class
public class InheritanceExample
{
	public static void main(String[] args)
	{
		Dog myDog = new Dog();
		myDog.eat(); // Inherited from Animal
		myDog.bark(); // Own method
	}
}
```

2. Multilevel Inheritance

In multilevel inheritance, a subclass extends from a superclass and then the same subclass acts as a superclass for another class.



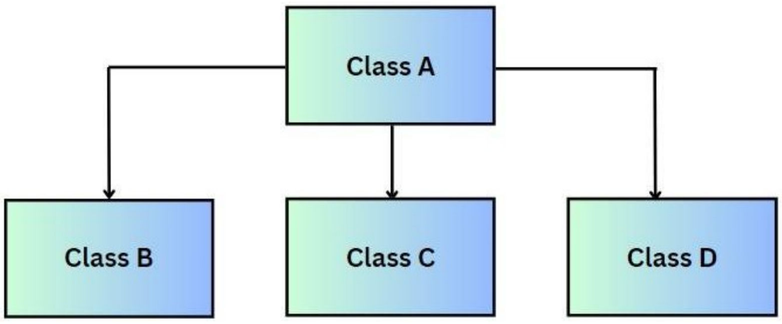
Example: Animal class that has a method eat(). Then, we create **Mammal, Dog** classes that inherits from **Animal**.

```
// Grandparent class
class Animal
{
	void eat()
	{
		System.out.println("This animal eats food.");
	}
}
// Parent class
class Mammal extends Animal
{
	void walk()
	{
		System.out.println("This mammal walks.");
	}
}
// Child class
class Dog extends Mammal
{
	void bark()
	{
		System.out.println("The dog barks.");
	}
}
```

```
// Main class
public class MultilevelExample
{
        public static void main(String[] args)
        {
                Dog myDog = new Dog();
                myDog.eat();  // Inherited from Animal
                myDog.walk(); // Inherited from Mammal
                myDog.bark(); // Own method
        }
}
```

3. Hierarchical Inheritance

In hierarchical inheritance, multiple subclasses extend from a single superclass.



```
// Parent class
class Animal
{
        void eat()
        {
                System.out.println("This animal eats food.");
        }
}
// Child classes
class Dog extends Animal
{
        void bark()
        {
                System.out.println("The dog barks.");
        }
}
```

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

// Main class
public class HierarchicalExample
{
        public static void main(String[] args)
        {
                Dog myDog = new Dog();
                myDog.eat();
                myDog.bark();
                Cat myCat = new Cat();
                myCat.eat();
                 myCat.meow();
        }
}
```

Method Overriding in Inheritance

A subclass can redefine a method of the parent class.

Example:

```
        // Parent class
        class Animal
        {
                void makeSound()
                {
                        System.out.println("The animal makes sound.");
                }
        }
```

```
// Child class (inherits from Animal)
class Dog extends Animal
{
        // Overriding the method
        void makeSound()
        {
                System.out.println("The dog barks.");
        }
}

// Main class
public class InheritanceExample
{
        public static void main(String[] args)
        {
                Dog myDog = new Dog();
                myDog. makeSound;
        }
}
```

Advantages of Inheritance

- **Code Reusability**: Avoids duplication by reusing existing class code.
- **Modularity**: Helps in organizing code in a hierarchical structure.
- **Easy to Update:** Changes in the parent class apply to all child classes.
- **Improved Code Organization** – Common properties stay in the parent class