JAVA

POLYMORPHISM

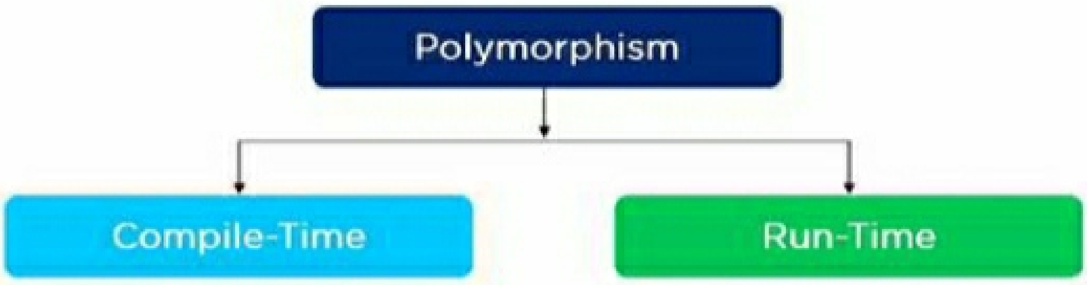
Polymorphism in Java

Polymorphism in Java is **a concept by which we can perform a single action in different ways.** Polymorphism is derived from 2 Greek words: poly and morphs. The word "**poly**" means many and "**morphs**" means forms. So polymorphism **means many forms**.

Real-life example:

- A person may have multiple characteristics at the same time.
- A woman, for example, is a sister, a mother, a wife, and an employee all at the same time. As a result, the same person behaves differently in different contexts. This is known as polymorphism.

Types of Polymorphism :



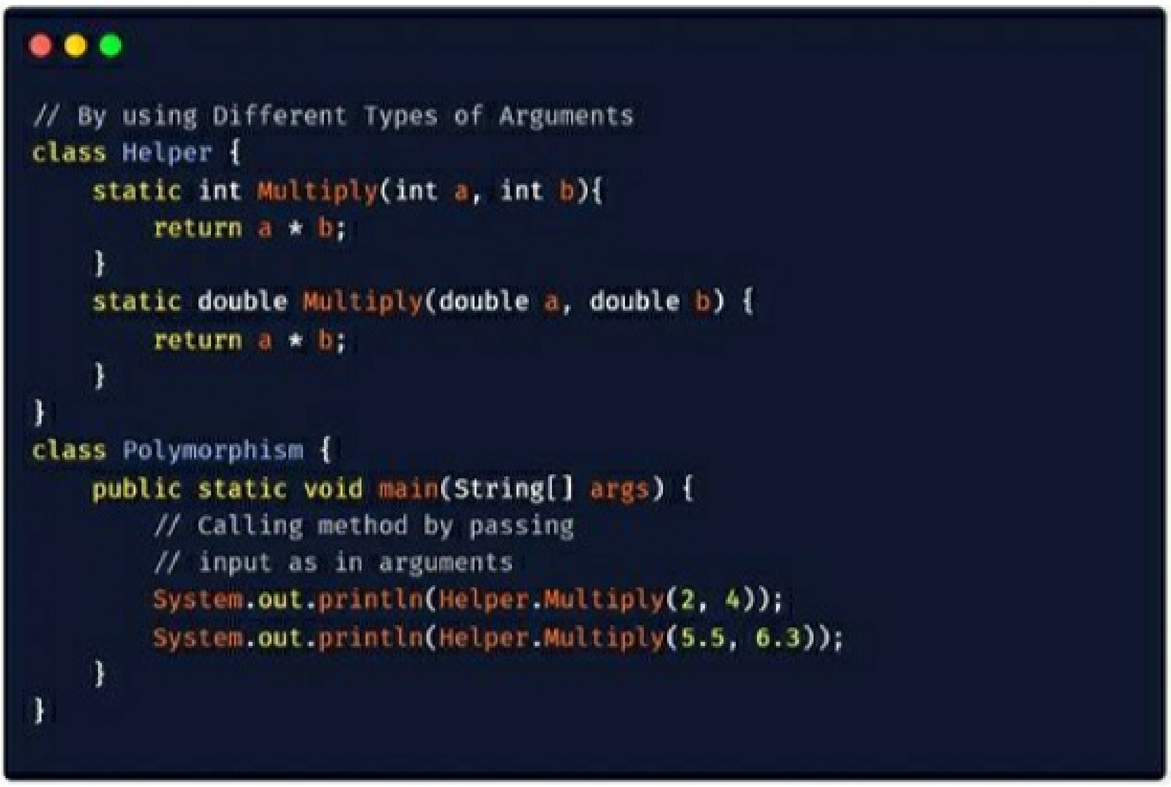
Compile-time Polymorphism

Compile-time polymorphism, also known as static polymorphism, is a type of polymorphism that occurs during the compilation process. There are two types of Compile-Time Polymorphism -

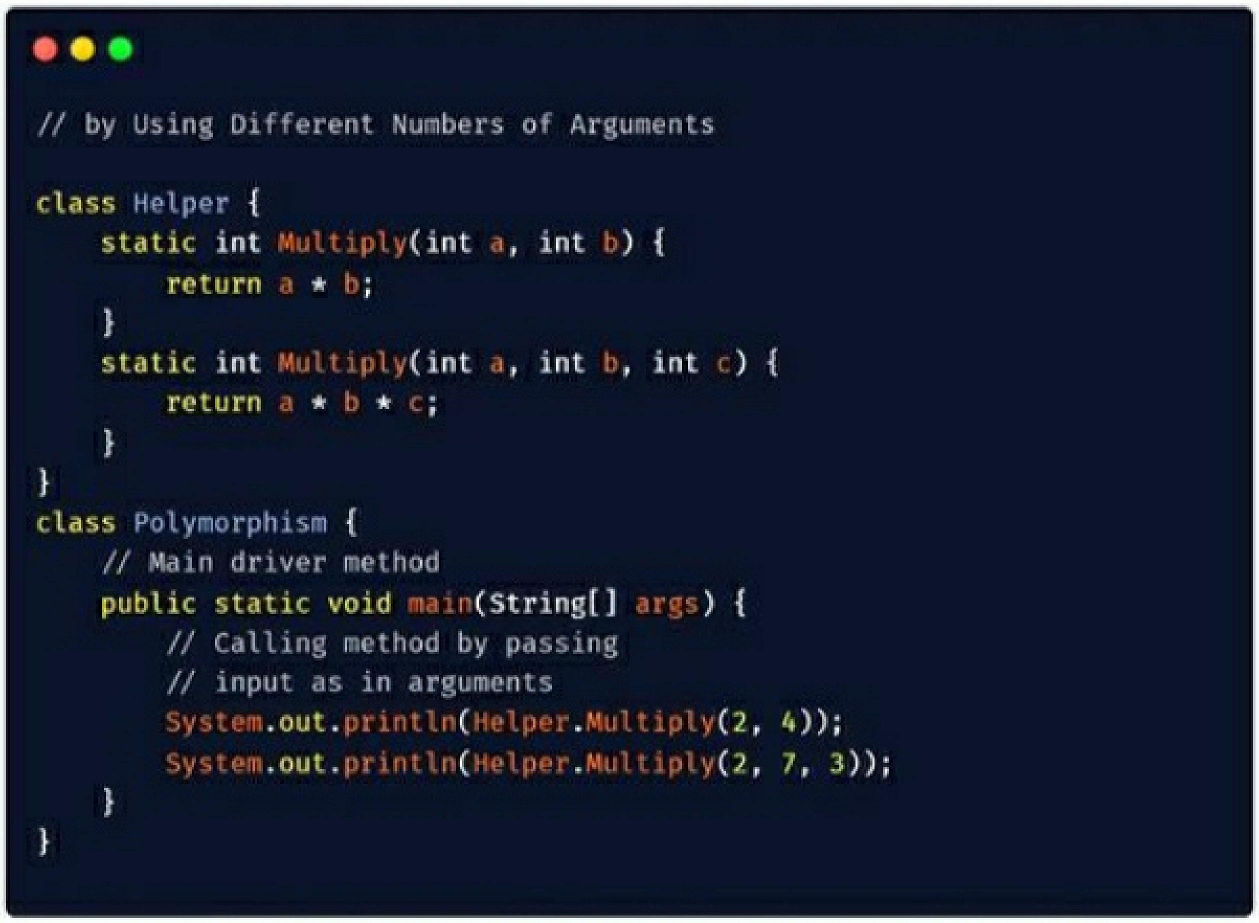
(1) Method Overloading **(2) Operator Overloading**

Method Overloading:

When there are multiple functions with the same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in the number of arguments or/and a change in the type of arguments.



```
// By using Different Types of Arguments
class Helper {
    static int Multiply(int a, int b){
        return a * b;
    }
    static double Multiply(double a, double b) {
        return a * b;
    }
}
class Polymorphism {
    public static void main(String[] args) {
        // Calling method by passing
        // input as in arguments
        System.out.println(Helper.Multiply(2, 4));
        System.out.println(Helper.Multiply(5.5, 6.3));
    }
}
```



```java
// by Using Different Numbers of Arguments

class Helper {
    static int Multiply(int a, int b) {
        return a * b;
    }
    static int Multiply(int a, int b, int c) {
        return a * b * c;
    }
}
class Polymorphism {
    // Main driver method
    public static void main(String[] args) {
        // Calling method by passing
        // input as in arguments
        System.out.println(Helper.Multiply(2, 4));
        System.out.println(Helper.Multiply(2, 7, 3));
    }
}
```

Operator Overloading :

Method Overloading is a process where a class has two or more methods with the same name. Still, the implementation of the specific method takes place according to the number of parameters in the method definition.

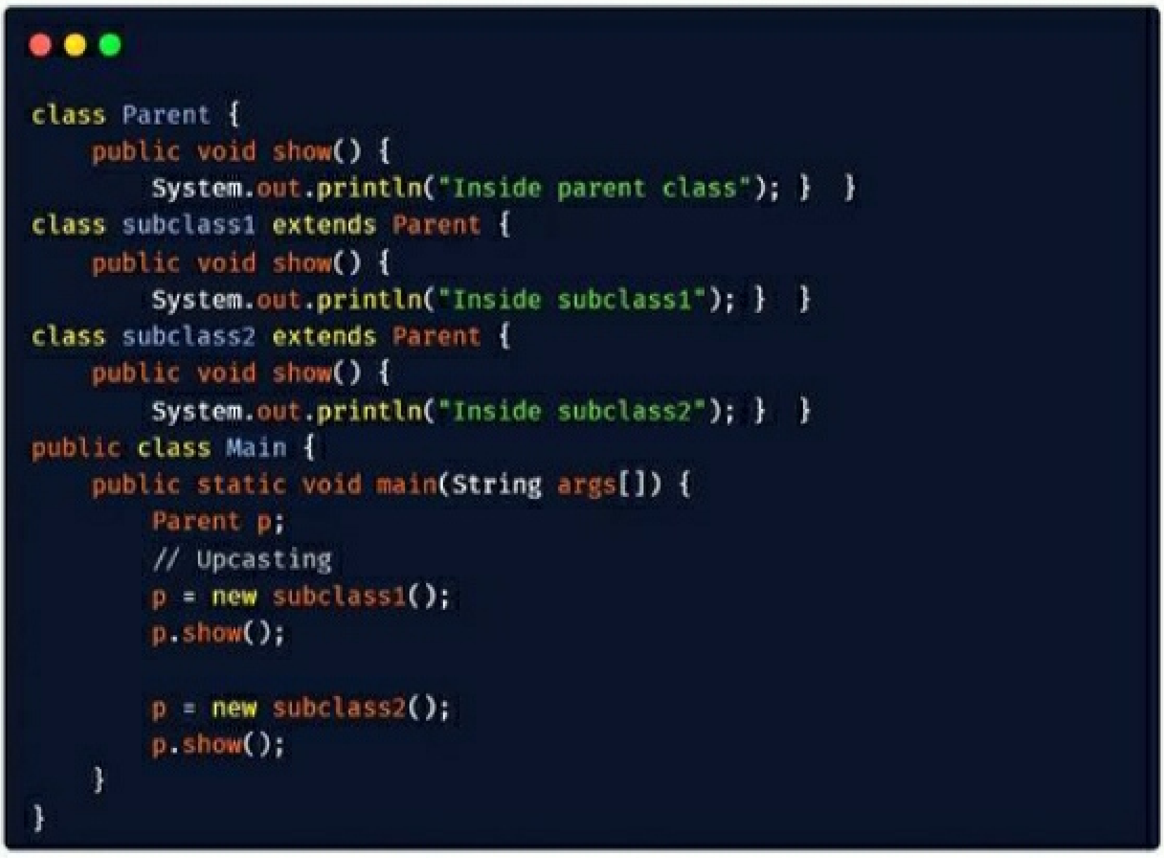
Java does not support Operator Overloading to avoid ambiguities.

Runtime Polymorphism

Dynamic polymorphism is another name for runtime polymorphism. In Java, method overriding is a technique for implementing runtime polymorphism. It is also sometimes known as Dynamic method dispatch.

Method Overriding :

Method Overriding is a procedure in which the compiler can allow a child class to implement a specific method already provided in the parent class.

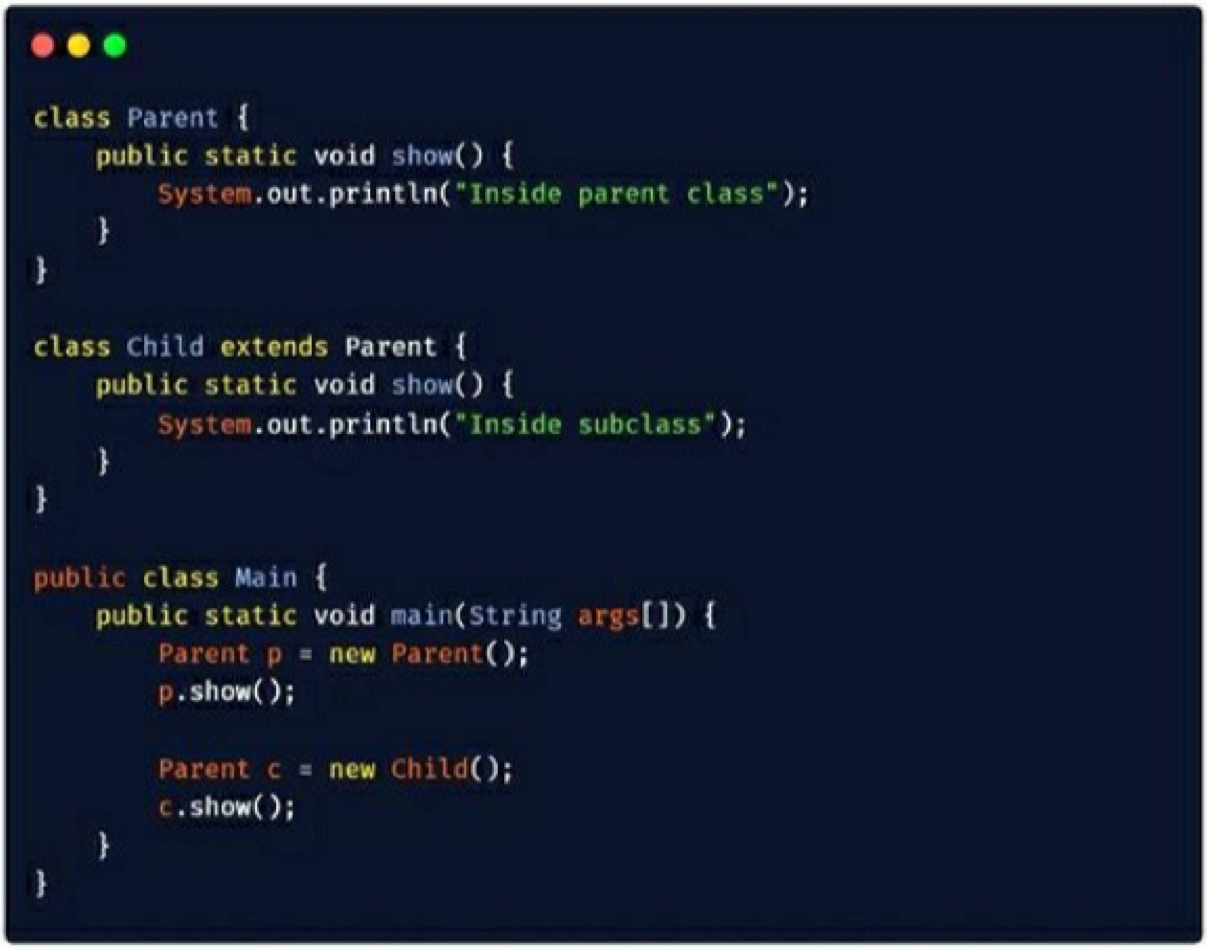


```java
class Parent {
    public void show() {
        System.out.println("Inside parent class"); }  }
class subclass1 extends Parent {
    public void show() {
        System.out.println("Inside subclass1"); }  }
class subclass2 extends Parent {
    public void show() {
        System.out.println("Inside subclass2"); }  }
public class Main {
    public static void main(String args[]) {
        Parent p;
        // Upcasting
        p = new subclass1();
        p.show();

        p = new subclass2();
        p.show();
    }
}
```

Method Hiding in Java

If a subclass defines a static method with the same signature as a static method in a superclass, the subclass method hides the superclass method. It is only possible to hide methods if both the parent and child classes have static methods.



```java
class Parent {
    public static void show() {
        System.out.println("Inside parent class");
    }
}

class Child extends Parent {
    public static void show() {
        System.out.println("Inside subclass");
    }
}

public class Main {
    public static void main(String args[]) {
        Parent p = new Parent();
        p.show();

        Parent c = new Child();
        c.show();
    }
}
```

Jayesh Deshmukh

Follow

Like Comment Share Save