

Polymorphism

Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word “poly” means many and “morphs” means forms, So it means many forms.

Real Life example: A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, and an employee. So the same person possesses different behavior in different situations. This is called polymorphism.

Types :

- **Compile-time Polymorphism**
 - **Runtime Polymorphism**
-
- **Compile-Time Polymorphism:** It is also known as **static polymorphism**. This type of polymorphism is achieved by **function overloading** or operator overloading. But **Java doesn't support Operator Overloading**.

Method(Function) 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 changes in the number of arguments or/and a change in the type of arguments**.

1) Method Overloading: changing no. of arguments

Main.java	Run	Output
<pre>1 //Method overloading - different no. of arguments 2 class Adder{ 3 static int add(int a, int b){ 4 return a+b; 5 } 6 7 static int add(int a,int b,int c){ 8 return a+b+c; 9 } 10 } 11 12 class Main{ 13 public static void main(String args[]){ 14 System.out.println(Adder.add(11,22)); 15 System.out.println(Adder.add(11,22,33)); 16 } 17 } 18 //End of the program</pre>		<pre>java -cp /tmp/XEfZjSZ323 Main 33 66</pre>

In this example, we have created two methods, first, add() method performs the addition of two numbers and the second add() method performs the addition of three numbers.

2) Method Overloading: changing the data type of arguments

Main.java	Run	Output
<pre>1 //Method overloading - changing data type of arguments 2 class Adder{ 3 static int add(int a, int b){ 4 return a+b; 5 } 6 static double add(double a, double b){ 7 return a+b; 8 } 9 } 10 11 class Main{ 12 public static void main(String args[]){ 13 System.out.println(Adder.add(11,11)); 14 System.out.println(Adder.add(12.3,12.6)); 15 } 16 } 17 //End of the program 18</pre>		<pre>java -cp /tmp/XEfZjSZ323 Main 22 24.9</pre>

In this example, we have created two methods that differ in data type. The first add() method receives two integer arguments and the second add() method receives two double arguments.

- **Runtime Polymorphism:** It is also known as **Dynamic Method Dispatch**. It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by **Method Overriding**.


Method overriding, on the other hand, occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be **overridden**.

If a subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

Rules for Java Method Overriding

1. The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class.
3. There must be an IS-A relationship (inheritance).

Main.java	Run	Output
<pre>1 //Java Program to illustrate the use of Java Method Overriding 2 //Creating a parent class. 3 class Vehicle{ 4 //defining a method 5 void run(){ 6 System.out.println("Vehicle is running"); 7 } 8 } 9 //Creating a child class 10 class Main extends Vehicle{ 11 //defining the same method as in the parent class 12 void run(){ 13 System.out.println("Bike is running safely"); 14 } 15 public static void main(String args[]){ 16 //creating object 17 Main obj = new Main(); 18 //calling method 19 obj.run(); 20 } 21 } 22 //End of the program</pre>		<pre>java -cp /tmp/XEfZjSZ323 Main Bike is running safely</pre>

Here in this program, When an object of a child class is created, then the method inside the child class is called. This is because The method in the parent class is overridden by the child class. Since The method is overridden, This method has more priority than the parent method inside the child class. So, the body inside the child class is executed.