Polymorphism

What is Polymorphism?

- The word **polymorphism** means having **many forms**.
- Polymorphism is considered as one of the important features of Object Oriented Programming. 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.

Types of Polymorphism-

- **In Java polymorphism is mainly divided into two types:**
    1. Compile time Polymorphism
    2. Runtime Polymorphism

Compile time polymorphism

- It is also known as static polymorphism. This type of polymorphism is achieved by method overloading or operator overloading.

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

Method Overloading

```
// Java program for Method overloading example 1 by using different types of arguments

class MultiplyFun {
    // Method with 2 parameter
    static int Multiply(int a, int b)    {
        return a * b;
    }

    // Method with the same name but 2 double parameter
    static double Multiply(double a, double b)    {
        return a * b;
    }
}
class Main {
    public static void main(String[] args)
    {
        System.out.println(MultiplyFun.Multiply(2, 4));
        System.out.println(MultiplyFun.Multiply(5.5, 6.3));
    }
}

--------------------------------
```

Output:

```
8
34.65
```

```
// Java program for Method overloading example 2 By using different numbers of arguments

class MultiplyFun {

    // Method with 2 parameter
    static int Multiply(int a, int b)
    {
        return a * b;
    }

    // Method with the same name but 3 parameter
    static int Multiply(int a, int b, int c)
    {
        return a * b * c;
    }
}

class Main {
    public static void main(String[] args)
    {
        System.out.println(MultiplyFun.Multiply(2, 4));

        System.out.println(MultiplyFun.Multiply(2, 7, 3));
    }
}

--------------------------------
```

Output:

```
8
42
```

Operator Overloading

- We can make the operator (‘+’) for string class to concatenate two strings. We know that this is the addition operator whose task is to add two operands.
- So a single operator ‘+’ when placed between integer operands, adds them and when placed between string operands, concatenates them.
- In java, Only “+” operator can be overloaded:
  - To add integers
  - To concatenate strings

Run time polymorphism

- It is also known as Dynamic Binding.
- 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**, occurs when a derived class has a definition for one of the member functions of the base class. That base class function is said to be overridden function and derived class function is said to be overriding function.

```java
// Java program for Method overridding

class Parent {

        void Print()
        {
                System.out.println("parent class");
        }
}

class subclass1 extends Parent {

        void Print()
        {
                System.out.println("subclass1");
        }
}

class subclass2 extends Parent {

        void Print()
        {
                System.out.println("subclass2");
        }
}
```

```java
class TestPolymorphism3 {
        public static void main(String[] args)
        {

                subclass1 a = new subclass1();
                a.Print();

                subclass2 b = new subclass2();
                b.Print();
        }
}
```

```
Output:
subclass1
subclass2
```

Compile time polymorphism vs Run time polymorphism

Compile-time polymorphism	Runtime polymorphism
Compile time polymorphism means binding is occurring at compile time.	Run time polymorphism where at run time we came to know which method is going to invoke.
It delivers quick execution since the method is known early in the compilation process.	It delivers slower execution since the method to be invoked is known at runtime.
Inheritance is not involved.	Inheritance is involved.
Method overloading is an example of compile time polymorphism.	Method overriding is an example of runtime polymorphism.