Polymorphism

Syeda Ajbina Nusrat, Lecturer, CSE, UITS

What is polymorphism in programming?

- **Polymorphism is the capability of a method to do different things based on the object .**
- **In other words, polymorphism allows you define one interface and have multiple implementations.**

1. It is a feature that allows one interface to be used for a general class of actions.
2. An operation may show different behavior in different instances.
3. The behavior depends on the types of data used in the operation.
4. It plays an important role in allowing objects having different internal structures to share the same external interface.
5. Polymorphism is extensively used in implementing inheritance.

Why we use polymorphism?

- Polymorphism allow you to "program in the general" rather than "program in the specific." .
- Polymorphism allow you to write programs that process objects that share the same superclass .
- With polymorphism, we can design and implement systems that are easily extensible .
- New classes can be added with little or no modification to the general portions of the program .

Types of polymorphism in java

- There are two types of polymorphism in java :
  1. Runtime polymorphism (Dynamic polymorphism)
  2. Compile time polymorphism (static polymorphism).

Runtime Polymorphism(or Dynamic polymorphism)

Method overriding is a perfect example of runtime polymorphism. In this kind of polymorphism, reference of class X can hold object of class X or an object of any sub classes of class X. For e.g. if class Y extends class X then both of the following statements are valid:

```
Y obj = new Y();
//Parent class reference can be assigned to child object
X obj = new Y();
```

Lets see the below example to understand it better :

```
public class X
{
    public void methodA() //Base class method
    {
        System.out.println ("hello, I'm methodA of class X");
    }
}

public class Y extends X
{
    public void methodA() //Derived Class method
    {
        System.out.println ("hello, I'm methodA of class Y");
    }
}
public class Z
{
   public static void main (String args []) {
       X obj1 = new X(); // Reference and object X
       X obj2 = new Y(); // X reference but Y object
       obj1.methodA();
       obj2.methodA();
   }
}
```

Output:

```
hello, I'm methodA of class X
hello, I'm methodA of class Y
```

Compile time Polymorphism(or Static polymorphism)

- Compile time polymorphism is nothing but the method overloading in java. In simple terms we can say that a class can have more than one methods with same name but with different number of arguments or different types of arguments or both.

Lets see the below example to understand it better :

```
class X
{
   void methodA(int num)
   {
       System.out.println ("methodA:" + num);
   }
   void methodA(int num1, int num2)
   {
       System.out.println ("methodA:" + num1 + "," + num2);
   }
   double methodA(double num) {
       System.out.println("methodA:" + num);
       return num;
   }
}
```

```
class Y
{
   public static void main (String args [])
   {
       X Obj = new X();
       double result;
       Obj.methodA(20);
       Obj.methodA(20, 30);
       result = Obj.methodA(5.5);
       System.out.println("Answer is:" + result);
   }
}
```

Output:

```
methodA:20
methodA:20,30
methodA:5.5
Answer is:5.5
```

When we use polymorphism ?

- You are programming in Java? Then its not a choice, you can't avoid using polymorphism. The biggest benefit of polymorphism is that it allows for extensible programs. You can have an API which deals with base types, and requires no knowledge at the time of writing what extra types might be required in the future .
- Because the API uses polymorphism it means if I create a new class by extending an existing one or implementing an existing interface I know that my new object will work seamlessly with the existing API. It also means I can keep things general in my code.

kindness always comes back

k.tolnoe