



Polimorfismo


Galilea Nazareth Esparza Mtz

Fundamentos de Bases de Datos

Eduardo Flores Gallegos




Que es un polimorfismo

- En programación orientada a objetos, polimorfismo es la capacidad que tienen los objetos de una clase en ofrecer respuesta distinta e independiente en función de los parámetros (diferentes implementaciones) utilizados durante su invocación. Dicho de otro modo el objeto como entidad puede contener valores de diferentes tipos durante la ejecución del programa.
- 

Ejemplo

Un ejemplo clásico de poliformismo es el siguiente. Podemos crear dos clases distintas: Gato y Perro, que heredan de la superclase Animal. La clase Animal tiene el método abstracto makesound() que se implementa de forma distinta en cada una de las subclases (gatos y perros suenan de forma distinta). Entonces, un tercer objeto puede enviar el mensaje de hacer sonido a un grupo de objetos Gato y Perro por medio de una variable de referencia de clase Animal, haciendo así un uso polimórfico de dichos objetos respecto del mensaje mover

```
class Animal {  
    public void makeSound() {  
        System.out.println("Grr...");  
    }  
}  
class Cat extends Animal {  
    public void makeSound() {  
        System.out.println("Meow");  
    }  
}  
class Dog extends Animal {  
    public void makeSound() {  
        System.out.println("Woof");  
    }  
}
```



Como todos los objetos Gato y Perro son objetos Animales, podemos hacer lo siguiente

```
public static void main(String[ ] args) {  
    Animal a = new Dog();  
    Animal b = new Cat();  
}
```



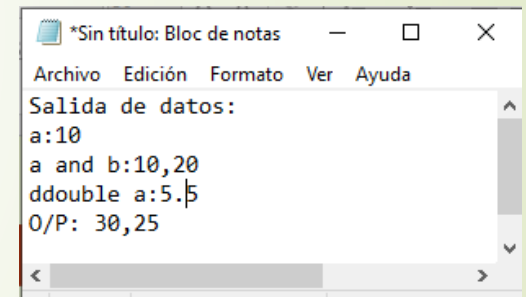
Tipos de polimorfismo

- **Sobrecarga:** El más conocido y se aplica cuando existen funciones con el mismo nombre en clases que son completamente independientes una de la otra.
- **Paramétrico:** Existen funciones con el mismo nombre pero se usan diferentes parámetros (nombre o tipo). Se selecciona el método dependiendo del tipo de datos que se envíe.
- **Inclusión:** Es cuando se puede llamar a un método sin tener que conocer su tipo, así no se toma en cuenta los detalles de las clases especializadas, utilizando una interfaz común.

Paramétrico

➡ Ejemplo:

```
Class Overload
{
    Void demo (int a)
    {
        System.out.println ("a: " + a);
    }
    Void demo (int a, int b)
    {
        System.out.println ("a and b: " + a + "," + b);
    }
    Double demo(double a) {
        System.out.println("double a: " + a);
        Return a*a;
    }
}
Class MethodOverloading
{
    Public static void main (String args [])
    {
        Overload Obj = new Overload();
        Double result;
        Obj .demo(10);
        Obj .demo(10, 20);
        Result = Obj .demo(5.5);
        System.out.println("O/P : " + result);
    }
}
```



Inclusión

➡ Ejemplo:

```
Abstract class Piece{  
    Public abstract void move(byte X, byte Y);  
}  
  
Class Bishop extends Piece{  
    @Override  
    Public void move(byte X, byte Y){  
  
    }  
}
```




Diferencia entre polimorfismo y sobrecarga

- El polimorfismo presenta unas claras ventajas aplicado desde las interfaces, ya que nos permite crear nuevos tipos sin necesidad de modificar las clases ya existentes.
 - Por el contrario, un método está sobrecargado si dentro de una clase existen dos o más declaraciones de dicho método con el mismo nombre pero con parámetros distintos, por lo que no hay que confundirlo con polimorfismo.
- 