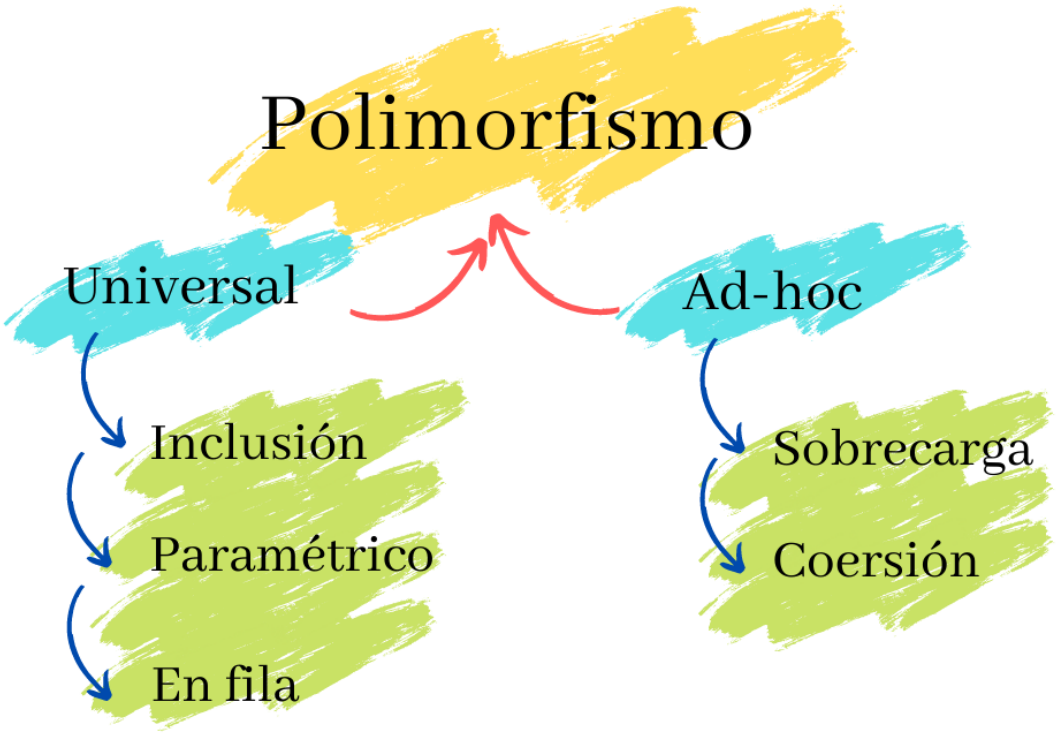
Nivel 5

Polimorfismo

¿Qué es polimorfismo?

Es la capacidad que tienen los objetos de una clase en ofrecer respuesta distinta e independiente en función de los parámetros utilizados durante su invocación. Dicho de otro modo, el objeto puede contener valores de diferentes tipos durante la ejecución del programa.



Aplicación en Java.

Para este ejemplo crearemos dos clases distintas, Gato y Perro las cuales heredarán de la superclase Animal. La clase animal tiene el método abstracto MakeSound () que se implementa de manera distinta en las subclases.

```
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");
  }
}
```

Luego hacemos lo siguiente:

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

Llamamos al método MakeSound ()

```
a.makeSound();
//Outputs "Woof"

b.makeSound();
//Outputs "Meow"
```

Entendido el concepto de polimorfismo, podemos hablar de 3 tipos de polimorfismo.

Polimorfismo paramétrico.

En este tipo de polimorfismo, existen funciones con el mismo nombre, pero se usan diferentes parámetros. Se selecciona el método dependiendo del tipo de datos que se envíe.

Aplicación en Java.

En este ejemplo, el método demo se sobrecarga 3 veces. Por lo que el método que se llamará está determinado por los argumentos que pasamos a llamar los métodos.

```
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);
    }
}
```

Y nos arrojará lo siguiente:

```
a: 10
a and b: 10,20
double a: 5.5
O/P : 30.25
```

Polimorfismo de inclusión.

Aquí podemos llamar un método sin tener que conocer su tipo, así no se tome en cuenta los detalles de las clases especializadas, utilizando una interfaz común.

```
abstract class Piece{
    public abstract void move(byte X, byte Y);
}

class Bishop extends Piece{
 @Override
  public void move(byte X, byte Y){

  }
}
```

Sobrecarga.

Se aplica cuando existen funciones con el mismo nombre en clases que son completamente independientes una de otra.

```
1
2  public class Sobrecarga {
3
4      static String s="método sobrecargado";
5
6      public void unMetodo(){
7          System.out.println("Sobrecarga");
8      }
9
10     public void unMetodo(String s) {//Método Sobrecargado
11         System.out.println(s);
12
13     }
14     public static void main(String[] args) {
15         // TODO Auto-generated method stub
16         Sobrecarga sobre=new Sobrecarga();
17         sobre.unMetodo();
18         sobre.unMetodo(s);
19     }
20
21 }
22
```

Referencias.

1. *Polimorfismo en Java: Programación orientada a objetos*. (2020, February 3). IfgeekthenEveris. https://ifgeekthen.everis.com/es/polimorfismo-en-java-programaci%C3%B3n-orientada-objetos