HABLEMOS DE POLIMORFISMO

DYL

¿QUÉ ES?

El polimorfismo es como tener una misma acción que puede comportarse de diferentes maneras depende quien la este realizando.

¿CUANDO USARLO?

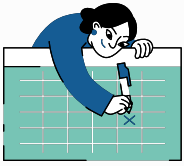
Cuando se nombran clases abstractas, cuando identificamos que varias clases usaran un mismo método pero con resultados diferentes. Va bastante unido a la herencia.

¿PORQUE USARLO?

Es una práctica simple que puede hacer mucho mas flexible nuestros programas, hacerla mas escalable, hace uso muy bueno de la abstracción, haciendo que en general sea mas claro y simple de implementar.

SOBRECARGA DE MÉTODOS

En java, generalmente se mencionan 3 tipos de polimorfismo qué se puede implementar, uno de ellos es la sobrecarga de metodos, se presenta cuando una clase tiene varios metodos que se llaman igual pero tienen parámetros distintos.



SOBREESCRITURA DE MÉTODOS

Esta se presenta cuando la clase hika redefine un método de la clase padre, y saber cual método sobreescrito se va a ejecutar se decide en ejecución dependiendo de una instancia real delobjeto.

POLIMORFISMO CON CLASES ABSTRACTAS

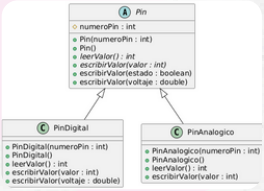
Las clases abstractas son clases que no pueden ser instanciadas, además, tienen algo especial y son los métodos abstractos, métodos que no se pueden definir pero las clases hijas obligatoriamente tienen que sobreescribir que hace el método, cuando esta clase hija se encarga de definir los métodos abstractos, se esta haciendo uso del polimorfismo.

Ejemplos de

POLIMORFISMO

Clase Pin

Digamos que hacemos la abstracción de un objeto de la vida real, un Arduino, Arduino tiene muchos pines, pero hay de dos tipos, todos los pines tienen nombre, pero los pines digitales manejan valores booleanos, en cambio, los pines analogicos manejan 1025 valores.




```
public class Pin {
protected int numeroPin;

public Pin(int numeroPin) {
    this.numeroPin = numeroPin;
}

// Polimorfismo con Sobrecarga de constructores
public Pin() {
    this.numeroPin = 0;
}

public abstract int leerValor();
public abstract void escribirValor(int valor);

// Polimorfismo con Sobrecarga de metodos de escritura
public void escribirValor(boolean estado) {
    System.out.println("Pin " + numeroPin + " escribiendo estado: " + (estado ? "HIGH" : "LOW"));
}

public void escribirValor(double voltaje) {
    System.out.println("Pin " + numeroPin + " escribiendo voltaje: " + voltaje + "V");
}
```

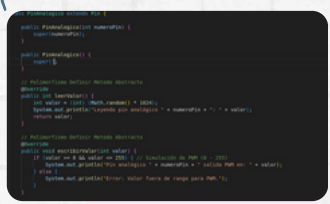
Esta es una clase abstracta, tiene sobrecarga de métodos tanto en el constructor como en el método escribirValor, además tiene dos métodos abstractos que las clases hijas tendrán que definir al ser creadas.

Clase PinDigital

ESta es una clase Hija, se llama pinDigital, y como es digital, solo puede tener valores de 1 y 0, por tanto, tiene que sobreescribir el método de escribir cuando recibe voltaje, adicionalmente los metodos que ya tenia que definir por ser abstractos y los constructores.

Clase PinAnalogico

Al contrario que su clase hermana, pin analogico si puede manejar valores de voltajes con unidades muy específicas, no necesita sobre escribir el metodo de escribir cuando recibe voltaje pero si necesita definir los métodos abstractos del padre y los constructores.



```java
abstract class Pin {
    protected int numeroPin;

    public Pin(int numeroPin) {
        this.numeroPin = numeroPin;
    }

    // Polimorfismo con Sobrecarga de constructores
    public Pin() {
        this.numeroPin = 0;
    }

    public abstract int leerValor();
    public abstract void escribirValor(int valor);

    // Polimorfismo con Sobrecarga de métodos de escritura
    public void escribirValor(boolean estado) {
        System.out.println("Pin " + numeroPin + " escribiendo estado: " + (estado ? "HIGH" : "LOW"));
    }

    public void escribirValor(double voltaje) {
        System.out.println("Pin " + numeroPin + " escribiendo voltaje: " + voltaje + "V");
    }
}

// Clase hija Pin Digital (HIGH / LOW)
class PinDigital extends Pin {

    public PinDigital(int numeroPin) {
        super(numeroPin);
    }
    // Polimorfismo sobrecarga de constructores
    public PinDigital() {
        super();
    }

    // Polimorfismo Definir Metodo Abstracto
    @Override
    public int leerValor() {
        int valor = (Math.random() < 0.5) ? 0 : 1; // Simulacion
        System.out.println("Leyendo pin digital " + numeroPin + ": " + valor);
        return valor;
    }

    // Polimorfismo Definir Metodo Abstracto
    @Override
    public void escribirValor(int valor) {
        if (valor == 0 || valor == 1) {
            System.out.println("Pin digital " + numeroPin + " set en " + (valor == 1 ? "HIGH" :
"LOW"));
        } else {
            System.out.println("Error: Valor no válido para pin digital.");
        }
    }

    // Polimorfismo Metodo Padre
    @Override
    void escribir(double voltaje) {
        int valorDigital = (voltaje >= 5.0 / 2) ? 1 : 0; // Conversion
        System.out.println("Escribiendo voltaje " + voltaje + "V como valor digital " + valorDigital +
" en el pin " + numero);
    }

}

// Clase hija  Pin Analógico (0 - 1023 en Arduino, 0 - 4095 en ESP32)
class PinAnalogico extends Pin {

    public PinAnalogico(int numeroPin) {
        super(numeroPin);
    }

    public PinAnalogico() {
        super();
    }

    // Polimorfismo Definir Metodo Abstracto
    @Override
    public int leerValor() {
        int valor = (int) (Math.random() * 1024);
        System.out.println("Leyendo pin analógico " + numeroPin + ": " + valor);
        return valor;
    }

    // Polimorfismo Definir Metodo Abstracto
    @Override
    public void escribirValor(int valor) {
        if (valor >= 0 && valor <= 255) { // Simulación de PWM (0 - 255)
            System.out.println("Pin analógico " + numeroPin + " salida PWM en: " + valor);
        } else {
            System.out.println("Error: Valor fuera de rango para PWM.");
        }
    }


    @Override
    public void escribirValor(int voltaje) {
        if (valor >= 0 && valor <= 255) {
            System.out.println("Pin analógico " + numeroPin + " salida PWM en: " + valor);
        } else {
            System.out.println("Error: Valor fuera de rango para PWM.");
        }
    }
}
```