UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programacion Orientada a Objetos

Sesión 6: Polimorfismo. Diseño del diagrama de clases del UML usando polimorfismo.

Recordando…

¿Tienen alguna consulta o duda sobre la clase previa?

Universidad Tecnológica del Perú

Agenda

- Introducción al Polimorfismo
- Tipos de Polimorfismo
- Ejemplos y Uso en Programación

Logro de la sesión

Al finalizar la sesión el estudiante entiende el concepto de polimorfismo y su importancia en el diseño y desarrollo de software en la programación orientada a objetos.

Utilidad

El polimorfismo es una característica poderosa de la programación orientada a objetos que promueve la flexibilidad, la reutilización del código, la abstracción y la encapsulación. Permite escribir código genérico y flexible que puede trabajar con diferentes tipos de objetos, lo que hace que el código sea más mantenible, extensible y fácil de entender.

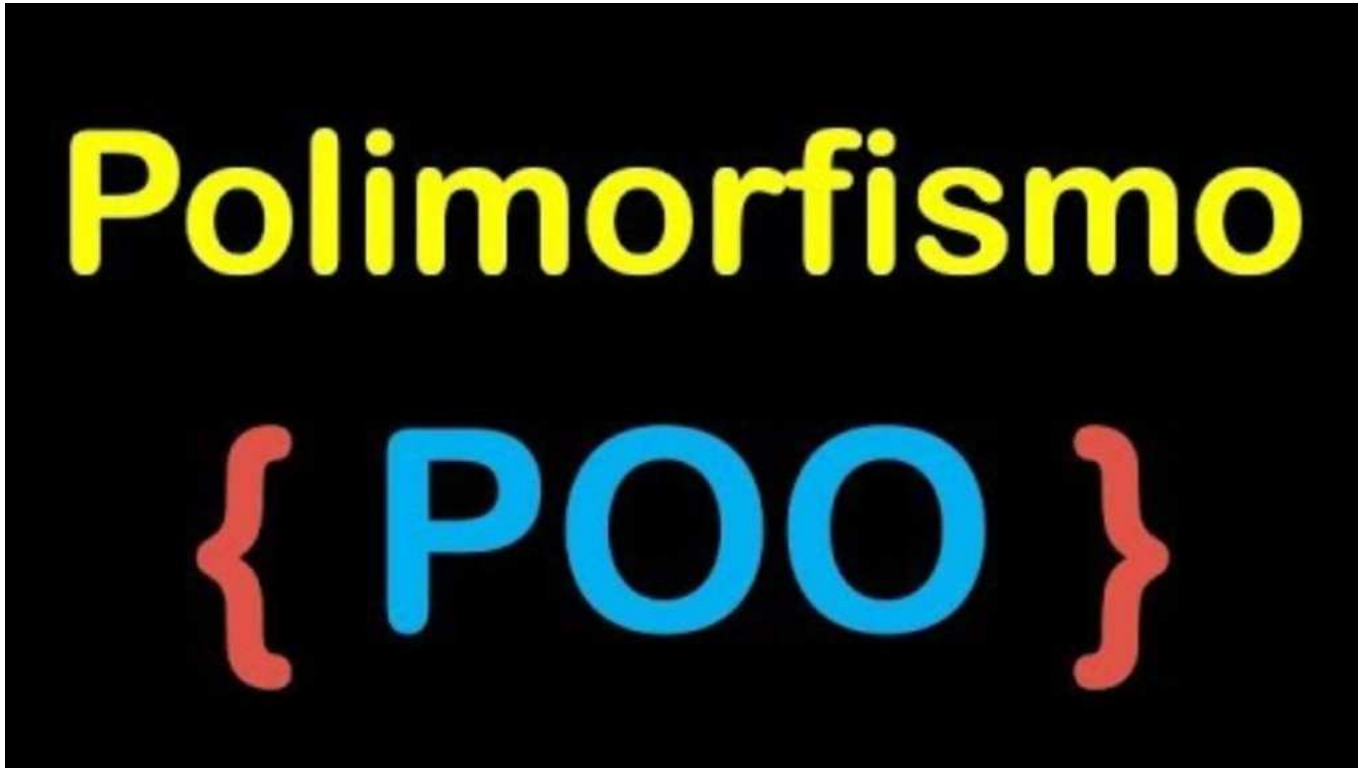
Conocimientos Previos

- Concepto básico de Programación Orientada a Objetos.
- Definición y uso de clases y objetos en Java.
- Herencia en Java.

Introducción al Polimorfismo

- **Definición**: Capacidad de un objeto para tomar muchas formas.
- Es uno de los cuatro pilares de la Programación Orientada a Objetos (junto con la encapsulación, herencia y abstracción).
- Permite que objetos de diferentes clases sean tratados como objetos de una clase común.

Polimorfismo



https://www.youtube.com/watch?v=tjjecfz9Cvk

- **Polimorfismo**

•El polimorfismo es uno de los cuatro pilares fundamentales de la programación orientada a objetos (junto con la encapsulación, la herencia y la abstracción). La palabra "polimorfismo" proviene del griego y significa "muchas formas". En programación, se refiere a la capacidad de una variable, función o método para actuar en múltiples formas.

•En Java, el polimorfismo se manifiesta principalmente de dos maneras:

•**Polimorfismo de tiempo de compilación (o sobrecarga)**: Ocurre cuando dos o más métodos en la misma clase tienen el mismo nombre pero diferentes parámetros.

•**Polimorfismo de tiempo de ejecución (o anulación)**: Ocurre cuando una subclase tiene un método específico que ya ha sido definido en su superclase (es decir, el método se sobrescribe).

- **Características clave del polimorfismo:**

•**Flexibilidad y reusabilidad**: Permite que los objetos de diferentes clases se traten como objetos de una clase común.

- **Vinculación dinámica**: En Java, gracias al polimorfismo, la decisión sobre qué versión de un método invocar se toma en tiempo de ejecución basada en el tipo de objeto, no en el tipo de referencia.

Ejemplos y Uso en Programación

Uso de la palabra clave 'override' para sobrescribir métodos en subclases.

Uso de la referencia de la clase base para referirse a objetos de subclases.

Beneficios: Flexibilidad, reutilización de código y extensibilidad en el diseño de software.

Ejercicios en Java

Ejercicio 1: Polimorfismo de tiempo de compilación (Sobrecarga)

```
class Matematicas {

   // Método para sumar dos números enteros
   int sumar(int a, int b) {
      return a + b;
   }

   // Método sobrecargado para sumar tres números enteros
   int sumar(int a, int b, int c) {
      return a + b + c;
   }

   // Método sobrecargado para sumar dos números de punto flotante
   double sumar(double a, double b) {
      return a + b;
   }
}

public class Test {
   public static void main(String[] args) {
      Matematicas mat = new Matematicas();
      System.out.println(mat.sumar(10, 20));       // 30
      System.out.println(mat.sumar(10, 20, 30)); // 60
      System.out.println(mat.sumar(10.5, 20.5));  // 31.0
   }
}
```

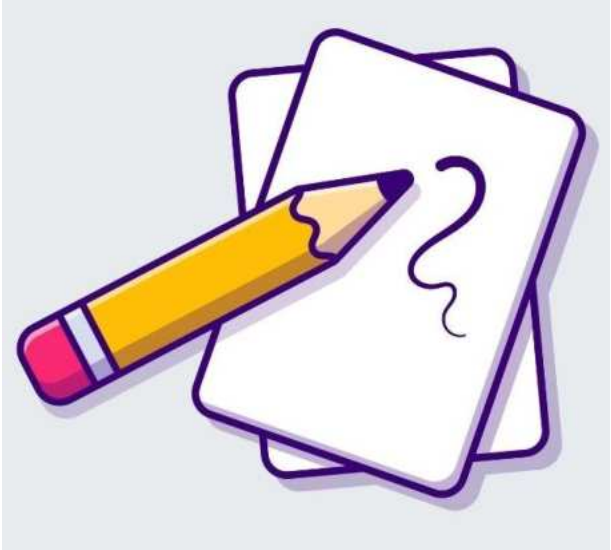
Ejercicio 2: Polimorfismo de tiempo de ejecución (Sobrescritura)

```
class Animal {
    void sonido() {
        System.out.println("El animal hace un sonido");
    }
}

class Perro extends Animal {
    @Override
    void sonido() {
        System.out.println("El perro ladra");
    }
}

class Gato extends Animal {
    @Override
    void sonido() {
        System.out.println("El gato maúlla");
    }
}

public class TestPolimorfismo {
    public static void main(String[] args) {
        Animal miAnimal = new Perro();
        miAnimal.sonido();  // El perro ladra

        miAnimal = new Gato();
        miAnimal.sonido();  // El gato maúlla
    }
}
```

Practica Guiada

Implementa los ejercicios anteriormente realizados en el programa java.

Conclusiones



- El polimorfismo es una herramienta poderosa que permite a los desarrolladores escribir código más flexible y reutilizable.
- Facilita la extensibilidad y mantenibilidad del software al permitir que los objetos de diferentes clases se traten de manera uniforme.
- Es esencial para aprovechar al máximo las ventajas de la programación orientada a objetos.

Cierre

¿Qué hemos aprendido hoy?

Bibliografía

- MORENO PÉREZ, J. Programación orientada a objetos.RA-MA Editorial. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=31933
- Vélez Serrano, José. Diseñar y programar, todo es empezar: una introducción a la Programación Orientada a Objetos usando UML y Java. Dykinson. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=36368

Universidad Tecnológica del Perú