Polimorfismo

August 30, 2023

1 Polimorfismo

El polimorfismo es un concepto fundamental en la programación orientada a objetos que permite que objetos de diferentes clases puedan ser tratados de manera uniforme a través de una interfaz común. Esto significa que puedes usar la misma función o método en diferentes tipos de objetos, independientemente de su clase subyacente, siempre y cuando sigan la misma interfaz.

En el contexto de la programación orientada a objetos, el polimorfismo se logra mediante el uso de herencia y la implementación de métodos con la misma firma (nombre y parámetros) en diferentes clases. Los objetos de estas clases pueden comportarse de manera diferente según su implementación concreta de esos métodos, pero desde el punto de vista del código que los utiliza, pueden ser tratados de manera uniforme.

Veamos un ejemplo en Python para entenderlo mejor:

Supongamos que tienes una jerarquía de clases que representan diferentes formas geométricas, y todas estas clases tienen un método llamado calcular_area():

[ ]:
```python
class Forma:
    def calcular_area(self):
        pass

class Cuadrado(Forma):
    def __init__(self, lado):
        self.lado = lado

    def calcular_area(self):
        return self.lado ** 2

class Circulo(Forma):
    def __init__(self, radio):
        self.radio = radio

    def calcular_area(self):
        return 3.14159 * self.radio ** 2

class Triangulo(Forma):
    def __init__(self, base, altura):
        self.base = base
```

```
        self.altura = altura

    def calcular_area(self):
        return 0.5 * self.base * self.altura
```

En este ejemplo, tenemos una clase base Forma y tres clases derivadas Cuadrado, Circulo y Triangulo, cada una de las cuales tiene su propia implementación del método calcular_area().

Ahora, puedes crear instancias de estas clases y tratarlas de manera uniforme utilizando polimorfismo:

[ ]:
```
def imprimir_area(forma):
    print(f"El área de la forma es: {forma.calcular_area():.2f}")

cuadrado = Cuadrado(5)
circulo = Circulo(3)
triangulo = Triangulo(4,6)

imprimir_area(cuadrado)   # Imprimirá el área del cuadrado
imprimir_area(circulo)    # Imprimirá el área del círculo
imprimir_area(triangulo)  # Imprimirá el área del triángulo
```

```
El área de la forma es: 25.00
El área de la forma es: 28.27
El área de la forma es: 12.00
```

En este ejemplo, la función imprimir_area() puede recibir diferentes tipos de objetos (cuadrados, círculos, triángulos) y calcular el área de cada uno sin saber los detalles de implementación de cada clase específica. Esto es posible debido al polimorfismo.

En resumen, el polimorfismo permite tratar objetos de diferentes clases de manera uniforme a través de una interfaz común, lo que hace que el código sea más flexible, reutilizable y fácil de mantener.