

RETO SEMANA 1: PARADIGMAS DE PROGRAMACIÓN: Programación Orientada a Objetos




INTEGRANTES:

1. NICOLE MORENO
2. VINESH VASWANI
3. CINTIA RIVERA
4. NEMORIO RIQUELME
5. KATHERINE FUENZALIDA
6. KATHERINE VENEGAS

ÍNDICE

1. ¿En qué consiste este paradigma de la programación?
2. ¿Qué problema o problemas intenta resolver?
3. Háblanos brevemente sobre su historia.
4. ¿Qué lenguajes de programación lo implementan?
5. ¿En qué escenario hipotético utilizarías este paradigma?
6. ¿Cómo se ve el futuro de la POO?
7. Conclusión.

1. ¿En qué consiste este paradigma de la programación?

Descripción	Conceptos	Lenguajes
<p>La Programación Orientada a Objetos (POO) es un paradigma de programación que se basa en la idea de estructurar el código en torno a objetos, que son entidades que combinan datos (atributos o propiedades) y comportamientos (métodos o funciones).</p>	<ul style="list-style-type: none">- Clases- Objetos- Encapsulamiento- Herencia- Polimorfismo	  

2. ¿Qué problema o problemas intenta resolver?

Este tipo de programación está orientado a **resolver problemas** relacionados a:

- a. **Reutilización del código:** promueve este tipo de práctica.
- b. **Análisis de situaciones:** ayuda a mejorar el análisis de cualquier escenario
- c. **Errores:** facilita su disminución.
- d. **Código largo:** contribuye a resolver la problemática de contar con un código muy largo que se vuelve complejo de leer, depurar y mantener.

Dentro de las **ventajas a destacar**, la POO destaca por el **ahorro de tiempo y esfuerzo**, que genera en la programación, dado que permite eliminar, añadir o modificar nuevos objetos o funciones. A esto se suma la **facilidad para distribuir el trabajo** dentro del equipo de desarrollo y sus integrantes.

3. Háblanos brevemente sobre su historia.

La idea inicial previa surge en los **años 60'** con el fin de mejorar la estructura del código, con el fin de ordenarlo y modularlo, siendo el lenguaje "**Simula**" (1967) de Ole-Johan Dahl y Kristen Nygaard el primer lenguaje en implementar el concepto de objetos. En este periodo se introdujeron también los conceptos de herencia y clases.

Este tipo de programación se consolida en **los años 70'** y se crea el término de POO, que fue acuñado por Alan Kay, creador del lenguaje de "**Smalltalk**" (1972-1980)

Sus **principales características** se basan en:

- ✓ Programas organizados como colección de objetos interconectados.
- ✓ Cada objeto tiene su propio conjunto de datos y funcionalidades.
- ✓ Las propiedades y métodos comunes a los objetos se encapsulan en clases, en donde cada clase es una plantilla para crear objetos.

Hoy en día, POO es uno de los **paradigmas más utilizados** en el desarrollo de software para crear aplicaciones de escritorio, juegos, App móviles entre otros, debido a su estructura modular y reutilizable.

4. ¿Qué lenguajes de programación lo implementan?



Los lenguajes más utilizados hoy en día son:

- ✓ **C++:** Lenguaje híbrido , combina paradigmas de programación estructura, orientada a objetos y genérica.
- ✓ **Java:** Se utiliza para App basadas en comercio electrónico, Sistema operativos de smartphones, software empresarial, entre otros.
- ✓ **Phyton:** Lenguaje de alto nivel, conocido por su sintaxis clara y facilidad de aprendizaje. Ideal para desarrollos de aplicaciones web, analisis de datos, IA y automatización de procesos.
- ✓ **PHP:** Altamente utilizado para el Desarrollo Web. Base de diversas plataformas como por ejemplo Wordpress, Drupal y Joola
- ✓ **Ruby:** Creación de lenguaje más equilibrado y agradable para el programador, centrándose en la simplicidad y eficiencia.

5. ¿En qué escenario hipotético utilizarías este paradigma?

```
python

class Libro:
    def __init__(self, titulo, autor):
        self.titulo = titulo
        self.autor = autor

class Usuario:
    def __init__(self, nombre, id_usuario):
        self.nombre = nombre
        self.id_usuario = id_usuario

class Bibliotecario:
    def prestar_libro(self, usuario, libro):
        # Lógica para gestionar el préstamo de libros

    def devolver_libro(self, usuario, libro):
        # Lógica para gestionar la devolución de libros
```

En nuestro ejemplo de biblioteca, representamos los libros y revistas como objetos, cada uno con atributos como el título, autor y género. Permittiéndonos ingresar más libros sin necesidad de escribir nuevamente todo el código.

```

class MaterialBibliografico:
    def __init__(self, titulo, autor):
        self.titulo = titulo
        self.autor = autor

    def mostrar_info(self):
        print(f"{self.titulo} - {self.autor}")

class Libro(MaterialBibliografico):
    def __init__(self, titulo, autor, genero):
        super().__init__(titulo, autor)
        self.genero = genero

    def mostrar_info(self):
        super().mostrar_info()
        print(f"Género: {self.genero}")

class Revista(MaterialBibliografico):
    def __init__(self, titulo, autor, categoria):
        super().__init__(titulo, autor)
        self.categoria = categoria

    def mostrar_info(self):
        super().mostrar_info()
        print(f"Categoría: {self.categoria}")

# Uso de las clases
libro = Libro("El Señor de los Anillos", "J.R.R. Tolkien", "Fantasía")
revista = Revista("National Geographic", "Varios", "Ciencia")

libro.mostrar_info()
revista.mostrar_info()

```

- ✓ Siguiendo con el ejemplo de la biblioteca, podemos seguir profundizando en la categorización, haciendo aún más personalizada la búsqueda y el desarrollo del objeto en cuestión.
- ✓ Además de la reutilización del código, podemos combinar entidades como clases y aprovechar la herencia y el polimorfismo.
- ✓ En este ejemplo, la herencia se relaciona a la generalidad de libro y revista, los cuales mantienen atributos comunes.

6. ¿Cómo se ve el futuro de la POO?

1. Habrá un mayor uso de la PP en **Inteligencia Artificial y Machine Learning**. Su modularidad permite construir modelos escalables y reutilizables en la IA. La combinación con paradigmas funcionales ayuda a mejorar la eficiencia en cálculos matemáticos.
2. Combinación con **Programación Funcional**: Lenguajes modernos como Python, Javascript, Kotlin y Swift integran tanto POO como Programación Funcional, ya que tiene beneficios como inmutabilidad, funciones puras y composición, las cuales se combinan con la modularidad del POO.
3. En la **Nube y Microservicios**: El POO sigue vigente, pero ahora con un enfoque más orientado a componentes.

Esto además de su gran expansión en lenguajes modernos que avanza cada día, la generación automática de lenguaje POO en la IA y la POO basada en prototipos en el desarrollo web.

El no desaparecerá, pero seguirá en constante evolución, sigue siendo una base, pero su enfoque será más modular, híbrido y flexible.

7. Conclusión

En conclusión, podemos afirmar que sin duda la Programación Orientada a Objetos (POO) se ha convertido en un aporte significativo para los programadores.

De esta manera, al poder agilizar y facilitar procesos, reducir errores y optimizar tiempo es un recurso indispensable para la ejecución de un modelo innovador, eficiente y perdurable.

Con una presentación más natural y concreta respecto a objetos del mundo real, es un buen punto de partida para sumergirse en el mundo de la programación y el código.

RETO SEMANA 1:

PARADIGMAS DE PROGRAMACIÓN: Programación Orientada a Objetos