Actividad 1

Introducción a la Programación Orientada a Objetos

Explicación de la filosofía de la POO

La losofía de la Programación Orientada a Objetos (POO) se basa en la conceptualización y modelado del mundo real mediante la creación y manipulación de objetos. Esta losofía proporciona un enfoque estructurado y modular para el diseño de software, permitiendo una representación más el de los elementos del mundo real y promoviendo conceptos clave. Dentro de los principios fundamentales de la losofía de la POO podemos encontrar

Abstracción **Encapsulación** **Herencia** **Polimor smo** **Identidad**

Herencia

La herencia permite la creación de nuevas clases basadas en clases existentes. Una clase derivada (o subclase) hereda atributos y métodos de una clase base (o superclase). Esto fomenta la reutilización de código y facilita la extensión y especialización de funcionalidades. Por ejemplo, si tienes una clase base "Vehículo", puedes crear clases derivadas como "Coche" y "Motocicleta" que hereden las características comunes de "Vehículo".

Cada objeto en la POO tiene una identidad única, que lo diferencia de otros objetos. La identidad se re ere al concepto de que dos objetos pueden tener la misma apariencia (mismos atributos y métodos) pero aún así son objetos distintos debido a su identidad única.

Encapsulación

La encapsulación consiste en agrupar los datos (atributos) y los métodos (funciones) que operan sobre esos datos en una unidad cohesiva llamada "objeto". Los detalles internos de un objeto están ocultos al mundo exterior, y solo se exponen las interfaces necesarias para interactuar con el objeto. Esto mejora la modularidad y facilita el mantenimiento y la reutilización del código.

Abstracción

La abstracción en la POO implica la simplicación y modelado de entidades del mundo real como objetos. Se centra en los aspectos esenciales de un objeto, ignorando detalles menos relevantes. Por ejemplo, un objeto "Coche" en un programa puede abstraerse para representar sus propiedades clave y comportamientos sin tener que preocuparse por detalles internos complejos.

Polimor smos

El polimor smo permite que objetos de diferentes clases respondan a un mismo conjunto de mensajes de manera coherente. En otras palabras, un objeto puede tomar múltiples formas dependiendo del contexto. Esto proporciona exibilidad en el diseño del software y facilita la adaptación a diferentes situaciones. Por ejemplo, varios tipos de objetos que heredan de una clase "Animal" pueden tener métodos comunes como "hacerSonido", pero cada tipo de animal implementará ese método de manera especí ca.

En conjunto, estos principios forman la base de la filosofía de la POO y proporcionan un marco conceptual sólido para el diseño de software modular, extensible y fácil de mantener. Al abstraer el mundo real en objetos y aplicar principios como encapsulación, herencia y polimor smo, la POO ayuda a crear sistemas más comprensibles, exibles y e cientes.

Concepto de objetos y clases

Conceptos fundamentales de POO: clases y objetos. Métodos y atributos en el contexto de la programación orientada a objetos. La Programación Orientada a Objetos (POO) es un paradigma de programación que se basa en el concepto de "objetos".

Estos conceptos forman la base de la Programación Orientada a Objetos y proporcionan una estructura e ciente y organizada para el desarrollo de software.

Clase

Una clase es un modelo o plano que de ne las características y comportamientos comunes de un grupo de objetos.

Ejemplo

La clase "Perro" puede tener atributos como "nombre" y métodos como "ladrar".

Objeto

Un objeto es una instancia concreta de una clase.

Ejemplo

Si tienes una clase "Perro", un objeto sería una instancia especí ca de perro, como "miPerro".

Atributos

Son las características o propiedades de un objeto que lo describen.

Ejemplo

En la clase "Persona", los atributos pueden ser "nombre", "edad" y "altura".

Métodos

Son funciones asociadas a una clase y describen el comportamiento de los objetos de esa clase.

Ejemplo

En la clase "Coche", un método puede ser "arrancar".

Polimor smo

Permite que un objeto pueda tomar múltiples formas, es decir, que un mismo método pueda comportarse de manera diferente en distintas clases.

Ejemplo

Un método "hacerSonido" que se comporte de manera diferente en las clases "Perro" y "Gato".

Herencia

Es un mecanismo que permite que una clase herede atributos y métodos de otra clase.

Ejemplo

Una clase "Estudiante" que hereda de la clase "Persona".

Encapsulamiento

Es el principio que consiste en ocultar los detalles internos de la implementación de un objeto y exponer solo lo necesario.

Ejemplo

Atributos privados que solo son accesibles mediante métodos públicos.

Comparación con la programación procedural

La programación procedural y la programación orientada a objetos (POO) son dos enfoques diferentes para diseñar y estructurar programas.

Programación Procedural	Programación Orientada a Objetos (POO)
Enfoque Centrado en Procedimientos: La programación procedural se centra en el desarrollo de procedimientos o rutinas que realizan operaciones específicas.	Enfoque Centrado en Objetos: La POO se centra en la creación y manipulación de objetos, que combinan datos y funciones.
Estructura Basada en Funciones: El código se organiza en funciones que manipulan datos y realizan operaciones.	Estructura Basada en Clases y Objetos: El código se organiza en clases que definen estructuras de datos y funciones asociadas, y los objetos son instancias de esas clases.

Comparación con la programación procedural

Programación Procedural	Programación Orientada a Objetos (POO)
Datos y Funciones Separados: Los datos y las funciones que operan sobre ellos están separados. Las funciones toman datos como entrada y devuelven resultados.	Datos y Métodos Juntos: Los datos y los métodos que operan sobre ellos están encapsulados en clases, lo que promueve la cohesión.
Menor Modularidad: Menos énfasis en la modularidad, ya que las funciones actúan de manera independiente.	Mayor Modularidad: Mayor énfasis en la modularidad debido a la encapsulación de datos y funciones en objetos.
Reutilización de Código Limitada: La reutilización de código puede ser más limitada ya que las funciones no están necesariamente encapsuladas en objetos.	Reutilización de Código Fácil: La reutilización de código se facilita mediante la creación de clases y la herencia, permitiendo la extensión y especialización.

Comparación con la programación procedural

Ejemplo comparativo:

Programación Procedural (Python):

```python
# Procedural
def calcular_area_rectangulo(base, altura):
    return base * altura

def calcular_area_circulo(radio):
    return 3.14 * radio ** 2
```

Programación Orientada a Objetos (Python):

```python
# Orientada a Objetos
class Figura:
    def __init__(self, nombre):
        self.nombre = nombre

class Rectangulo(Figura):
    def __init__(self, base, altura):
        super().__init__("Rectangulo")
        self.base = base
        self.altura = altura

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

class Circulo(Figura):
    def __init__(self, radio):
        super().__init__("Circulo")
        self.radio = radio

    def calcular_area(self):
        return 3.14 * self.radio ** 2
```

Comparación con la programación procedural

La programación procedural se centra en funciones y datos separados, mientras que la POO se centra en objetos que encapsulan datos y funciones. La POO proporciona una mayor modularidad y facilita la reutilización de código mediante el uso de clases y objetos.

En la programación orientada a objetos, los conceptos como la herencia y el polimor smo permiten una mayor exibilidad y extensibilidad en el diseño del código.

La elección entre programación procedural y orientada a objetos depende del problema a resolver y de los requisitos del proyecto, siendo la POO más adecuada para sistemas complejos y grandes, mientras que la programación procedural puede ser más apropiada para problemas más sencillos.


TIC
TALENTO TECH
UTP
faceIT