Conceptos Básicos de la programación Orientada a Objetos (Poo)

Desarrollo y análisis de Software

Sergio Adriam Pardo Moya

Ficha:2900078

Conceptos Básicos de la programación Orientada a Objetos (Poo)

1. Abstracción:

- **Definición:** La abstracción es el proceso de enfocarse en las características esenciales de un objeto mientras se ignoran los detalles irrelevantes. En POO, se utiliza para crear modelos simplificados de objetos reales o abstractos, permitiendo a los programadores trabajar con conceptos sin preocuparse por su implementación interna.
- **Ejemplo:** Un automóvil. Al abstraer un automóvil, nos enfocamos en sus características esenciales como ruedas, motor, asientos y volante, ignorando detalles como el tipo de motor, la marca de los neumáticos o el color de la pintura. Esto nos permite hablar sobre los automóviles en general sin tener que entrar en detalles específicos de cada modelo.

2. Encapsulamiento:

- **Definición:** El encapsulamiento es la técnica de ocultar los detalles internos de un objeto y solo exponer interfaces públicas para interactuar con él. Esto promueve la modularidad y protege los datos internos de modificaciones accidentales o malintencionadas.
- **Ejemplo:** Una cápsula de medicamento. La cápsula encapsula el medicamento, ocultando su composición química y sabor al usuario. El usuario solo interactúa con la cápsula a través de la superficie externa, ingiriéndola entera.

3. Herencia:

- **Definición:** La herencia es la capacidad de una clase de heredar atributos y métodos de otra clase, creando una relación jerárquica entre ellas. Esto permite reutilizar código y crear clases más especializadas a partir de clases más generales.
- **Ejemplo:** Animales, mamíferos y perros. La clase "Perro" hereda las características de la clase "Mamífero", que a su vez hereda las características de la clase "Animal". Esto permite que todos los perros

compartan características como comer, dormir y tener pelo, sin necesidad de definirlas nuevamente en la clase "Perro".

4. Polimorfismo:

- **Definición:** El polimorfismo es la capacidad de un objeto de tomar diferentes formas o responder a la misma llamada de diferentes maneras, dependiendo de su clase o estado. Esto permite a los programadores escribir código más flexible y reutilizable.
- **Ejemplo:** Un juego de formas. Un objeto "Forma" puede ser un círculo, un cuadrado o un triángulo. Todos comparten la misma interfaz básica (dibujar()), pero cada uno la implementa de manera diferente según su forma específica.

5. Clases y Objetos:

- **Clases:** Son los planos o plantillas que definen las características y el comportamiento de los objetos. Contienen atributos (datos) y métodos (funciones) que describen cómo interactúa el objeto con el mundo exterior.
- **Objetos:** Son instancias individuales de una clase. Se crean a partir de la clase y poseen sus propios valores de atributos y pueden invocar los métodos definidos en la clase.
- **Ejemplo:** Clase "Estudiante" y objeto "Juan". La clase "Estudiante" define atributos como nombre, edad y carrera, y métodos como estudiar() y aprobar_examen(). El objeto "Juan" es una instancia de la clase "Estudiante" y tiene sus propios valores para estos atributos, como "Juan Pérez", 20 años e "Ingeniería Informática".

6. Métodos y Atributos:

- **Métodos:** Son las funciones o acciones que un objeto puede realizar. Se definen dentro de la clase y se invocan desde los objetos.
- **Atributos:** Son las variables o datos que almacena un objeto. Se definen dentro de la clase y representan las características del objeto.

- **Ejemplo:** En la clase "Estudiante", el método "estudiar()" podría permitir al objeto "Juan" aumentar su nivel de conocimiento, mientras que el atributo "edad" podría almacenar su edad actual.

7. Modularidad:

- **Definición:** La modularidad es la técnica de dividir un programa en módulos independientes y autocontenidos. Cada módulo tiene una responsabilidad específica y puede ser desarrollado, probado y mantenido de forma independiente.
- **Ejemplo:** Un sistema de gestión de biblioteca. El sistema podría dividirse en módulos para gestión de libros, gestión de usuarios, préstamos y devoluciones. Cada módulo se encarga de una tarea específica y puede ser modificado sin afectar a los demás.

8. Reusabilidad:

- **Definición:** La reusabilidad es la capacidad de utilizar código escrito para un propósito específico en otros contextos. Esto ahorra tiempo y esfuerzo a los programadores, ya que no tienen que escribir código nuevo desde cero para cada tarea.
- **Ejemplo:** Las clases y métodos bien diseñados en POO pueden ser reutilizados en diferentes proyectos. Por ejemplo, una clase para conectar a una base de datos podría ser utilizada en varias aplicaciones que necesiten acceder a datos almacenados.