Pedro Zuttro - 24101026

A programação orientada a objetos (POO) existe desde os anos 1960, mas ganhou popularidade apenas na década de 1990, coincidindo com o surgimento da internet. A POO é particularmente eficaz em ambientes de rede, o que a tornou ideal para o desenvolvimento de tecnologias web e móveis. Apesar das mudanças rápidas nas tecnologias, os conceitos fundamentais da POO permanecem estáveis, embora passem por reinterpretações ao longo do tempo.

Conceitos Fundamentais da Poo:

1- Encapsulamento: Combina dados e comportamentos em um único objeto, controlando o acesso a esses dados.

2- Herança: Permite que uma classe herde atributos e métodos de outra classe, promovendo a reutilização do código. Usa a relação "é-um" (is-a)

3- Polimorfismo: Permite que objetos respondam de maneiras diferentes à mesma mensagem, dependendo de sua classe

4- Composição: Um objeto é composto por outros objetos, estabelecendo uma relação "tem-um" (has-a) em vez de "é-um" (is-a)

Objetos e sistemas legados

A POO não substitui necessariamente os sistemas legados baseados em programação estruturada. Muitos sistemas legados funcionam bem e não precisam ser alterados. No entanto, novos desenvolvimentos geralmente justificam o uso de tecnologias OO. Uma prática comum é encapsular sistemas legados em wrappers de objetos, que permitem integrar código estruturado em um ambiente OO.

Programação Procedural vs. Orientada a objetos

. Programação Procedural: Separa dados e processos, com dados frequentemente globais, o que pode levar a problemas de integridade e dificuldades de depuração

. Programação Orientada a objetos: Combina dados e comportamentos em objetos, promovendo maior controle e encapsulamento. Isso facilita a manutenção e a segurança, pois o acesso a dados é restrito

O que é um Objeto?

Um objeto é uma entidade que contém atributos (dados) e métodos (comportamentos). Por exemplo, um objeto "Pessoa" pode ter atributos como nome e idade, e métodos como "falar" ou "andar". A comunicação entre objetos ocorre por meio de mensagens, onde um objeto invoca métodos do outro.

O que é uma Classe?

Uma classe é um "molde" para criar objetos. Ela define os atributos e métodos que os objetos terão. Por exemplo, a classe (Pessoa) pode definir atributos como (nome) e (endereço), e métodos como (getNome) e (setNome). As classes são fundamentais para a criação de objetos em POO

Encapsulamento e Ocultação de Dados

O encapsulamento é um dos pilares da POO, onde os dados e comportamentos são agrupados em um objeto. A ocultação de dados é uma parte importante do encapsulamento, onde apenas os métodos públicos de um objeto são acessíveis, enquanto os detalhes internos permanecem privados. Isso protege a integridade dos dados e facilita a manutenção.

Herança

A herança permite que uma classe herde atributos e métodos de uma classe superior (superclasse). Por exemplo, as classe (cachorro) e (gato) podem herdar atributos como (cor dos olhos) de uma classe (Mamíferos). A herança promove a reutilização de código e organização hierárquica das classes.

Polimorfismo

O polimorfismo permite que objetos de diferentes classes respondam de maneiras diferentes à mesma mensagem. Por exemplo, em um sistema de formas geométricas, tanto um (Círculo) quanto um (Retângulo) podem ter um método desenhar(), mas cada um implementa o método de forma diferente. Isso permite flexibilidade e extensibilidade no design do software

/ /

Composição

A composição é o processo de construir objetos a partir de outros objetos. Por exemplo, um carro pode ser composto por um motor, rodas e outros componentes. A relação de composição é descrita como "tem-um" (has-a), em contraste com a herança, que é uma relação "é-um" (is-a)

Resumindo, a POO é baseada em quatro conceitos principais: encapsulamento, herança, polimorfismo e composição. Esses conceitos permitem criar sistemas modulares, reutilizáveis e de fácil manutenção. A POO não substitui necessariamente a programação procedural, mas oferece uma abordagem mais estruturada e organizada para o desenvolvimento de software, especialmente em ambientes complexos e distribuídos.