Programação Orientada a Objetos: Conceitos, Vantagens e Adoções

A Ascensão e Importância da Programação Orientada a Objetos

- POO; embora originária de décadas atrás, ganhou destaque com a ascensão da internet.

- Seus conceitos centrais são: encapsulamento, herança, polimorfismo e composição
  - ↳ Eles permanecem cruciais no design de software, apesar dos debates contínuos sobre sua aplicação

- POO e programação estruturada são complementares; novos benefícios de desenvolvimento se beneficiam significativamente do POO, particularmente para aplicativos de internet móveis

- Este capítulo enfoca esses conceitos fundamentais, enfatizando sua importância nas decisões de design, mesmo em meio a debates sobre sua aplicação (como a herança)

POO vs. Programação Procedural e Adoção

- Os objetos definidos por atributos e comportamentos, cons. trastam com separação de dados e funções na programação procedural

- A adoção da POO, inicialmente lenta devido a sistemas legados, aumentou substancialmente.

Princípios da POO e Boas Práticas

- A ocultação de dados, restringindo o acesso a atributos e métodos, ~~de objetos internos sem afetar outros objetos que dependem deles~~ aumenta a integridade dos dados
- Os objetos interagem por passagem de mensagens, permitindo mudanças nos métodos de objetos internos sem afetar outras objetos que dependem deles
- A POO encapsula dados e seu código de manipulação dentro de objetos

Conceitos Básicos de POO

- POO aborda questões de acesso a dados descontrolado, encapsulando dados e comportamentos dentro de objetos, aumentando a integridade dos dados por meio da ocultação de dados

• Os objetos interagem através da passagem de mensagens, e as classes servem como projetos para objetos, cada um com seus próprios atributos e métodos

• Um bom design POO evita a manipulação direta de dados e favorece objetos menores e mais especializados.

Fundamentos da POO: Encapsulamento e Diagramas

• Os objetos interagem através de métodos públicos, enquanto atributos e metódos privados são encapsulados.

• Os diagramas de classe ilustram classes, atributos e métodos, com a ocultação de dados protegendo os dados

Herança, Interface e outros Conceitos

• O paradigma interface / implementação permite mudanças internas sem afetar o código externo. ~~como~~

• A herança é detalhada, mostrando como classes como Cão e Gato herdam atributos e métodos de uma classe-mãe, mamífero, evitando a duplicação de código e promovendo um melhor design

• A herança, usando uma relação "IS-A" permite que as subclasses herdem das superclasses, evitando a duplicação de código.

Polimorfismo

• O polimorfismo permite que diferentes classes respondam exclusivamente a mesma chamada de método

Herança e Composição em POO

- POO oferece herança e composição como relação "has-a" para construir classes, combinando dados e comportamento através do encapsulamento