Renato Portilho Costa RA 24101301

Resumo Poo 24/03

A programação orientada a objetos pode parecer coisa Nova, mas Na real, existe desde os anos 60, No entanto, só lá pelos anos 90 que ela começou a bombar de Verdade, principalmente por causa da Internet. Isso aconteceu porque a Poo se encaixa muito bem No modelo de redes e aplicativos distribuídos

Diferente das tecnologias, que vivem mudando, os conceitos de Poo são mais estáveis, embora evoluam ao longo do tempo, então ao entender esses conceitos é essencial, pois eles continuam relevantes, mesmo com o avanço das tecnologias

4 Pilares

- Encapsulamento

Esconder os detalhes internos de um objeto e expõe apenas o Necessário

- Herança

Cria Novas classes com base em classes existentes, herdando seus atributos e métodos

- Polimorfismo

Um mesmo método pode se comportar de maneiras diferentes dependendo do objeto que o utiliza

- Composição

um objeto pode conter outros objetos dentro dele.

Programação estruturada x Poo

Antes da poo, a programação era mais "procedimental", ou seja, separava os Dados e as funções que os manipulavam, isso levava a problemas como código difícil de manter e debugar, já que qualquer função poderia modificar os dados.

Na Poo, os atributos e métodos ficam juntos dentro dos objetos, reduzindo problemas de acesso indevido e facilitando o reaproveitamento do código

Exemplo:

Pense em um carro, ele tem atributos (cor, modelo, ano) e comportamentos (acelerar, frear, buzinar). No modelo procedimental, os atributos ficariam separados das funções, Na Poo tudo fica dentro de um mesmo objeto "carro", deixando o código mais organizado

Objetos e classes

Um objeto é uma instância de uma classe, que funciona como um molde. Por exemplo: uma classe cachorro pode definir características como cor e raça, além de comportamentos como latir e correr. Quando criamos um cachorro específico (exemplo: um labrador chamado rex) estamos criando um objeto dessa classe.

Ainda

• Encapsulamento

Significa proteger os dados dentro do objeto. Isso é feito tornando alguns atributos privados e permitindo acesso apenas por meio de métodos públicos (getters e setters). Isso melhora a segurança e evita que partes externas do código modifiquem os dados diretamente

Ex:

Um caixa eletrônico não deixa você mexer diretamente no saldo da conta você só pode interagir por meio de funções como sacar ou depositar. Isso é encapsulamento

• Herança

Permite que uma classe aproveite atributos e métodos de outra. Isso evita a repetição de código e facilita a manutenção

Ex:

Se temos uma classe "Animal" com atributos como "idade" e "peso", podemos criar classes "cachorro" e "gato" que herdam essas características e adicionam comportamentos próprios como "latir" e "miar".

• Polimorfismo

Permite que métodos tenham implementações diferentes dependendo da classe. Isso é muito útil quando queremos tratar objetos de forma genérica, sem nos preocupar com o tipo exato

Ex: Se temos uma classe "forma" e subclasses como "círculo" e "quadrado", todos podem ter um método "desenhar()". Quando chamamos "desenhar()", cada forma vai exibir um comportamento diferente, mesmo usando o mesmo método

• Composição

Significa que um objeto pode conter outros objetos. Isso ajuda a modularizar o código e deixar o sistema mais organizado

Ex: um carro é composto por um motor, um volante e rodas. Cada um desses elementos pode ser tratado como um objeto separado, facilitando a manutenção e reutilização do código

Conclusão

A Poo trouxe uma revolução para o desenvolvimento de Software. Seus conhecimentos principais: encapsulamento, herança, polimorfismo e composição, continuam essenciais para construir sistemas robustos e flexíveis

Seja para criar um jogo, um aplicativo ou um sistema empresarial, entender bem esses fundamentos ajuda a escrever códigos mais organizados, reutilizáveis e fáceis de manter