Programação Orientada a Objetos (POO) - Aula 08

Professor: Jerffeson Gomes Dutra

Relacionamentos entre Objetos

Na vida real os objetos interagem entre si e uma das formas mais comuns de relacionamento é um objeto fazer parte de outro ou ter outro.

Exemplos:

- Jerffeson **TEM UMA** carteira;
- Um personagem **TEM VÁRIOS** itens no inventário;
- Um motor **É PARTE DE** um carro;

Encapsulamento

O encapsulamento é um dos princípios fundamentais da programação orientada a objetos. Ele envolve ajudar a controlar o acesso aos membros de uma classe (atributos e métodos). O encapsulamento tem três principais objetivos:

Abstração: Ocultar os detalhes de implementação dos métodos e atributos, fornecendo uma interface simples e clara para interagir com o objeto.

Proteção: Controlar o acesso aos membros da classe, permitindo que apenas certos métodos acessem ou modifiquem os atributos, garantindo que os dados sejam manipulados de forma segura e consistente.

O encapsulamento é frequentemente implementado utilizando modificadores de acesso.

Modificadores de Acesso

Os modificadores de acesso são palavras-chave que controlam a visibilidade dos membros de uma classe. Existem três principais modificadores de acesso em TypeScript:

public: Membros marcados como `public` são acessíveis de qualquer lugar, tanto dentro como fora da classe. Por padrão, os membros de uma classe são `public`.

private: Membros marcados como `private` só podem ser acessados dentro da própria classe onde foram definidos. Eles não podem ser acessados de fora da classe.

protected: Membros marcados como `protected` podem ser acessados dentro da própria classe e pelas subclasses (herança), mas não podem ser acessados de fora da classe ou de subclasses fora do escopo.

Exercício 01

1. Cria uma classe chamada ContaBancaria.
   a. As contas bancárias possuem um número, saldo e titular;
2. Um titular (pessoa dona da conta) tem um nome;
3. O saldo é uma atributo muito sensível e não deve ser acessado diretamente, ele deve ser **private;**
4. Criar 3 métodos, um método para retorne o saldo da conta, um método para a conta receber transferência e outro para transferir dinheiro para outra conta.
5. Crie uma classe transferência, nessa classe deve existir um método que receber o valor que será transferido e as duas contas envolvidas.

Exercício 02

- Altere o exemplo anterior para que não seja possível informar valores negativos nos métodos de transferência.
- Altere o exemplo anterior para caso tente transferir um valor maior do que existe na conta, exibir o texto: “Saldo insuficiente”.

Exercício 3

- Crie a classe Guerreiro e a classe Equipamento.
- A classe Guerreiro tem 3 atributos, todos privados: vida, mana e ataque.
- O guerreiro pode equipar 3 equipamentos: na cabeça, corpo e arma
- Todos os equipamentos oferecem bônus de status, com mais ataque, vida ou mana.
- Crie um método na classe Guerreiro que retorne o ataque total do Guerreiro:

ataqueTotal = ataque do personagem + bônus de ataque dos itens