Interfaces em OO

Em Orientação a Objetos (OO), uma interface é um contrato que especifica um conjunto de métodos e propriedades **que uma classe deve implementar.**

Uma interface define um conjunto de ações que uma classe deve ser capaz de executar, mas não especifica como essas ações serão implementadas. Em vez disso, cada classe que implementa a interface é responsável por fornecer sua própria implementação dos métodos e propriedades especificados na interface.

# Algumas características importantes das interfaces em OO são:

* Uma interface define um conjunto de métodos e propriedades que uma classe deve implementar, mas não especifica como eles devem ser implementados.
* As interfaces permitem que as classes sejam flexíveis e adaptem-se a diferentes contextos, sem comprometer a coesão do sistema.
* As interfaces fornecem uma maneira de abstrair as características comuns de diferentes classes, permitindo que as classes sejam tratadas de maneira polimórfica.

As interfaces são frequentemente usadas em OO quando é necessário definir um conjunto de comportamentos comuns que as classes devem suportar, mas não se importa com a implementação concreta desses comportamentos.

Por exemplo, imagine que você esteja criando um sistema de gerenciamento de estoque para uma loja e precise criar várias classes diferentes para representar os diferentes tipos de itens em estoque (por exemplo, roupas, eletrônicos, brinquedos etc.).

Embora cada uma dessas classes possa ter propriedades e comportamentos específicos, elas todas precisam ser capazes de realizar algumas ações comuns, como serem vendidas ou devolvidas. Nesse caso, você pode definir uma interface chamada **IItemEstoque** que define os métodos **Vender()** e **Devolver()**, que cada classe que representa um item em estoque deve implementar.

# 

# Algumas vantagens de usar interfaces em OO incluem:

* As interfaces promovem a coesão do sistema, permitindo que cada classe se concentre em sua própria responsabilidade e implemente apenas os métodos e propriedades relevantes para essa responsabilidade.
* As interfaces fornecem uma maneira de abstrair as características comuns de diferentes classes, permitindo que as classes sejam tratadas de maneira polimórfica.
* As interfaces fornecem um mecanismo para garantir a compatibilidade entre diferentes classes, permitindo que os desenvolvedores criem sistemas modulares e extensíveis.

| // Definindo a interface IItemEstoque public interface IItemEstoque {  void Vender();  void Devolver(); }  // Implementando a interface em uma classe public class Roupa : IItemEstoque {  public void Vender()  {  // Implementação específica para vender roupas  }   public void Devolver()  {  // Implementação específica para devolver roupas  } }  // Usando a classe Roupa com a interface IItemEstoque public class Loja {  private List<IItemEstoque> \_itensEmEstoque;   public void AdicionarItem(Roupa item) //problema de acoplamento aqui por que?  {  \_itensEmEstoque.Add(item);  }   public void VenderItem(int index)  {  \_itensEmEstoque[index].Vender();  \_itensEmEstoque.RemoveAt(index);  } }  // Exemplo de uso var loja = new Loja(); var roupa = new Roupa(); loja.AdicionarItem(roupa); loja.VenderItem(0); |
| --- |

# Exercícios para praticar e entender melhor.

1. Crie uma interface IAnimal com o método EmitirSom(). Em seguida, crie as classes Cachorro, Gato e Papagaio que implementam a interface IAnimal com sua própria implementação do método EmitirSom(). Em um programa, instancie as três classes e chame o método EmitirSom() de cada uma.
2. Crie uma interface IFormatadorTexto com o método Formatar(string texto). Em seguida, crie as classes FormatadorMaiusculo, FormatadorMinusculo e FormatadorInvertido que implementam a interface IFormatadorTexto com sua própria implementação do método Formatar(). Em um programa, instancie as três classes e use cada uma para formatar um texto diferente.
3. Crie uma interface IAnimalEstimacao com as propriedades Nome e Dono. Em seguida, crie as classes Cachorro e Gato que implementam a interface IAnimalEstimacao com suas próprias implementações das propriedades. Em um programa, instancie as duas classes e defina os valores das propriedades para cada uma.
4. Crie uma interface ICalculadora com os métodos Somar(), Subtrair(), Multiplicar() e Dividir(). Em seguida, crie uma classe Calculadora que implementa a interface ICalculadora com sua própria implementação dos métodos. Em um programa, instancie a classe Calculadora e use cada um dos métodos para realizar operações matemáticas.
5. Crie uma interface IServicoPagamento com os métodos EfetuarPagamento() e EstornarPagamento(). Em seguida, crie as classes PagamentoCartao, PagamentoBoleto e PagamentoPaypal que implementam a interface IServicoPagamento com suas próprias implementações dos métodos. Em um programa, instancie as três classes e use cada uma para efetuar e estornar pagamentos diferentes.