# Perguntas teóricas de introdução

### Em quais linguagens o C# foi inspirado?

## Foi baseada na linguagem C++ e tem muitos elementos da linguagem Pascal e Java.

### Inicialmente o C# foi criado para qual finalidade?

Inicialmente o C# buscava ser uma linguagem orientada a objeto simples, robusta, moderna e de propósito geral. Na época, ela se parecer com java mostrava que atingia estes objetivos. Além disso, era importante permitir que uma mesma aplicação pudesse ser executada em dispositivos diferentes (não necessariamente PC).

### Quais os principais motivos para a Microsoft ter migrado para o Core?

Ele foi projetado para permitir ampla adaptabilidade a novas plataformas e cargas de trabalho. Além disso é dividido em várias partes que podem se adaptar a novas plataformas em momentos diferentes.

Como foi feito a partir de uma reformulação do ASP.NET Framework, teve alterações de arquitetura que resultam em uma estrutura mais enxuta e modular tendo alguns benefícios como: maior performance, multiplataforma, open-source, injeção de dependência interna, etc.

### Cite as principais diferenças entre .Net Full Framework e .Net Core.

**Os modelos de** aplicativos -- .NET Core não suporta todos os modelos de aplicativos do .NET Framework. Ele não dá suporte especificamente a Web Forms do ASP.NET e a ASP.NET MVC, mas dá suporte ao ASP.NET Core MVC. E começando com .NET Core 3.0, o .NET Core também suporta Formulários WPF e Windows somente no Windows.

**APIs** – o .NET Core contém um grande subconjunto da biblioteca de classes base .NET Framework, com aspectos diferentes (os nomes de assembly são diferentes, os membros expostos nos tipos são diferentes em casos principais). Em alguns casos, essas diferenças exigem alterações na fonte de porta para .NET Core.

**Subsistemas** – O.NET Core implementa um subconjunto dos subsistemas no .NET Framework com o objetivo de proporcionar uma implementação e um modelo de programação mais simples. Por exemplo, o CAS (Code Access Security, segurança de acesso ao código) não é suportado, enquanto a reflexão é suportada.

**Plataformas** – O .NET Framework dá suporte a Windows e Windows Server, enquanto o .NET Core também dá suporte a macOS e Linux.

**Software Livre** – O.NET Core é um software livre, enquanto um [subconjunto somente leitura do .NET Framework](https://github.com/microsoft/referencesource) é um software livre.

# Utilizando POO - Responda e demonstre no código os itens abaixo:

### O que é POO?

A programação orientada a objetos refere-se a um tipo de programação no qual os programadores definem o tipo de dados de uma estrutura de dados e também os tipos de operações (métodos) que podem ser aplicadas à estrutura de dados.

Dessa maneira, a estrutura de dados se torna um objeto que inclui dados e funções. Além disso, os programadores podem criar relacionamentos entre um objeto e outro. Por exemplo, objetos podem herdar características de outros objetos.

Em outras palavras, é onde o desenvolvedor tem de começar a pensar fora da caixa, a imaginar uma forma aonde será preciso recorrer ao mundo real para o desenvolvimento das aplicações.

### O que é polimorfismo?

O Polimorfismo é um mecanismo por meio do qual selecionamos as funcionalidades utilizadas de forma dinâmica por um programa no decorrer de sua execução.

Com o Polimorfismo, os mesmos atributos e objetos podem ser utilizados em objetos distintos, porém, com implementações lógicas diferentes.

Exemplo simples: podemos dizer que uma classe chamada Vendedor e outra chamada Diretor podem ter como base uma classe chamada Pessoa, com um método chamado CalcularVendas. Se este método (definido na classe base) se comportar de maneira diferente para as chamadas feitas a partir de uma instância de Vendedor e para as chamadas feitas a partir de uma instância de Diretor, ele será considerado um método polimórfico, ou seja, um método de várias formas.

### O que é abstração?

Definir classes apropriadas concentrando-se no que é importante para uma determinada finalidade.

### O que é encapsulamento?

É uma maneira de encapsular a complexidade, escondendo os detalhes de implementação dentro de uma classe.

Em um processo de encapsulamento os atributos das classes são do tipo private. Para acessar esses tipos de modificadores, é necessário criar métodos setters e getters.

Por entendimento os métodos setters servem para alterar a informação de uma propriedade de um objeto. E os métodos getters para retornar o valor dessa propriedade.

### Quando usar uma classe abstrata e quando devo usar uma interface?

Uma classe abstrata pode conter lógica (código), enquanto que uma interface somente pode especificar quais os métodos ou propriedades. Uma classe que implementa a interface deve definir. Entretanto nenhuma das duas pode ser usada para construir um objeto, para isso é necessário definir uma classe que derive da abstrata (mas que não seja abstrata) ou que implemente a interface.

Normalmente utiliza-se classe abstrata quando é necessário implementar um método e reutilizar nas classes que irão implementar.

### O que faz as interfaces IDisposable, IComparable, ICloneable e IEnumerable?

IDisposable: o uso principal dessa interface é liberar recursos não gerenciados.

IComparable: é implementada por tipos cujos valores podem ser ordenados ou classificados.

ICloneable: permite que você forneça uma implementação personalizada que cria uma cópia de um objeto existente.

IEnumerable: é a interface base para todas as coleções não genéricas que podem ser enumeradas.

### Existe herança múltipla (de classes) em C#?

Não. Só é permitido herdar uma 1 classe e implementar N interfaces.

# Trabalhando com testes

### Quais os principais frameworks que podemos usar no desenvolvimento de testes?

Como Frameworks de teste temos MSTest, NUnit, XUnit, Selenium, etc.

Como frameworks auxiliares para os testes temos, Fluent Assertions, Moq, etc.