1 - Em quais linguagens o C# foi inspirado?

R: C, C++ e Java.

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

R: O C# foi criado com a finalidade de flexibilizar a criação de aplicativos, desenvolvendo os aplicativos para a plataforma .Net Framework e não para um dispositivo eletrônico específico.

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

R:  Apesar do .NET Framework ser uma plataforma interessante e possuir seu mérito é vinculado com o Windows e tem necessidade de possuir o .Net Framework instalado. Com isso é forte a tendência de perder mercado para outras plataformas como o Ruby e Node, que rodam em vários sistemas operacionais, como o Linux. Com isso a Microsoft precisou rever sua estratégia para conquistar um mercado que ainda não tem e fazer um produto melhor ainda, mais rápido e mais leve. Atendendo esta estratégia a Microsoft criou o .Net Core e vem migrando para esta nova versão do .Net, criada do zero, porém aproveitando a experiência com o .Net Framework e funcionando em outros sistemas operacionais como o Linux e Mac.

4 - Cite as principais diferenças entre .Net Full Framework e .Net Core.

.Net Full Framework

- roda somente no Windows

- uma plataforma mais madura

- está com planejamento de ser descontinuado pela Microsoft

- a grande maioria de bibliotecas de terceiros ou Nuget para C# rodam nesta plataforma

.Net Core

- roda em outros sistemas operacionais além do Windows, como o Linux e o Mac, ou seja plataforma cruzada;

- é o futuro da plataforma .Net

- algumas bibliotecas de terceiros ou Nuget podem não funcionarem nesta versão do .Net

- código aberto

- escalonável

- melhor desempenho e escalonável

- devido funcionar em Linux pode ser executado em contêineres do Docker

- mais indicado para desenvolver microserviços, devido o item anterior.

**Utilizando POO**

O que é POO?

R: É Programação Orientada a Objetos. É um paradigma de programação cujos conceitos chaves são “classe” e “objeto”, sendo que os demais conceitos são construídos em cima destes dois. Os pilares da POO são

O que é polimorfismo?

R: De uma forma genérica significa “várias formas”. Em uma linguagem de programação como o C# isso significa que pode haver várias formas de fazer coisa. Praticamos polimorfismo em C# através da sobrecarga de métodos, através da sobrescrita de métodos da classe pai na classe filha e implementação de métodos nas classes filhas dos métodos abstratos da classe pai. É um dos pilares da POO.

O que é abstração?

R: É um dos pilares da POO. Consiste em identificar as qualidades ou propriedades importantes do problema que está sendo modelado. Através desse modelo abstrato podemos nos concentrar nas características relevantes e ignorar as irrelevantes. Visa abstrair algo do mundo real e transforma-lo em um objeto na programação. Esse objeto será composto por uma identidade, propriedades e métodos.

O que é encapsulamento?

R: Também é um dos pilares da POO. Consiste em esconder os detalhes de implementação de uma classe. Todo o acesso a estes detalhes são feitos através de métodos públicos da classe. O exemplo mais clássico são propriedades privadas que somente podemos acessá-las por métodos get/set, o que a classe faz com ela não sabemos.

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

R: A interface tem por objetivo criar um “contrato” que a classe que a implementa deve obrigatoriamente obedecer ou seja garantir que os métodos e propriedades especificadas pela interface serão oferecidas pela classe, sendo indicado seu uso quando as implementações precisam compartilhar as mesmas assinaturas de métodos e possuir as mesmas propriedades.

Agora se o objetivo é criar um tipo para compartilhar comportamentos (métodos com processamento) e estados (propriedades) em comum, então o indicado é a classe abstrata.

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

R:

A interface IDisposable fornece um mecanismo para liberar recursos não gerenciados como identificadores de janela ou abrir arquivos. Esta interface possui apenas o método Dispose. Este método é usado para liberar explicitamente recursos não gerenciados em conjunto com o coletor de lixo.

A interface IComparable define um método genérico de comparação específica de tipo que um tipo de valor ou classe implementa para ordenar ou classificar suas instâncias. Esta interface possui o método CompareTo(Object), que retorna um int.

Na implementação do método CompareTo(Object) o valor int indica como dois elementos estão relacionados um ao outro:

- Menor que zero indica que as instâncias atuais precedem o objeto definido na ordem de classificação

- Igual a zero indica que a instância atual está na mesma posição de um objeto especificado na ordem de classificação.

- Maior que zero indica que a instância atual é posterior ao objeto especificado na ordem de classificação.

A interface ICloneable dá suporte à clonagem, que cria uma nova instância de uma classe com o mesmo valor de uma instância existente através da implementação do método Clone.

O C# não possui recurso nativo de clonar um objeto de forma controlada, o método MemberwiseClone somente faz o que se chama de clonagem superficial (shallow).

Devido alguns problemas existentes nesta interface a Microsoft não recomenta a utilização desta interface e, API’s públicas.

A Interface IEnumerable expõe um enumerador que dá suporte a uma iteração simples em uma coleção de um tipo especificado. Ela não permite alteração nas coleções, sendo somente leitura, somente fornece informações necessárias para percorrer a coleção e somente permite percorrer os elementos de forma sequencial e voltar para o início.

**Trabalhando com testes**

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

R: Os principais frameworks para usar no desenvolvimento de testes em .Net são xUnit, MS Test, NUnit, Moq, NSubstitute, Fluent Assertions e Selenium WebDriver;