**Resumo da Semana**

**Namespaces:**Com o passar do tempo e com o aumento do sistema, é preciso muito organização entre os arquivos e saber de qual escopo faz parte cada classe, o grande problema que surge com os sistemas grandes é a organização de todas as suas classes. Para evitar que o sistema fique caótico, podemos agrupar as classes por características comuns e dar um nome para cada um desses grupos. Isto é, agruparíamos um conjunto de classes em um espaço em comum e lhe daríamos um nome, como por exemplo StoneCo.CSharpEvolution.WeeklyChallenge. Esse espaço definido por um nome é chamado de namespace.

Segundo a convenção de nomes adotada pela Microsoft (<https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-1.1/893ke618(v=vs.71)>), os namespaces devem ter a forma: NomeDaEmpresa.NomeDoProjeto.ModuloDoSistema.

The general rule for naming namespaces is to use the company name followed by the technology name and optionally the feature and design as follows.

**Classes**Uma classe é uma forma de definir um tipo de dado, ela é formada por dados e comportamentos.

Para definir os dados, são utilizados os atributos, e para definir o comportamento são utilizados métodos. Depois que uma classe é definida podem ser criados diferentes objetos que utilizam a classe.

O código da classe Conta, por convenção, deve ficar dentro de um arquivo com o mesmo nome da classe.

**Structs**

Os structs são particularmente úteis para estruturas de dados pequenas que têm semântica de valor. Números complexos, pontos em um sistema de coordenadas ou pares chave-valor em um dicionário são exemplos de structs. O uso de structs, em vez de classes para estruturas de dados pequenas, pode fazer uma grande diferença no número de alocações de memória que um aplicativo executa.Ao contrário das classes, as estruturas são tipos de valor e não precisam de alocação de heap.

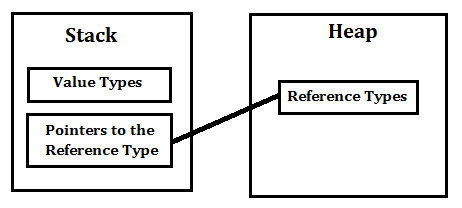
Os construtores struct são invocados com o operador new, semelhante a um construtor de classe. Porém, em vez de alocar dinamicamente um objeto no heap gerenciado e retornar uma referência a ele, um construtor de struct simplesmente retorna o valor do struct (normalmente em um local temporário na pilha), e esse valor é, então, copiado conforme a necessidade.

**Principais Tipos Primitivos**

**Principais Tipos de Referência**

Há dois tipos em C#: tipos de referência e valor. Variáveis de tipos de referência armazenam referências em seus dados (objetos) enquanto que variáveis de tipos de valor contém diretamente seus dados. Com tipos de referência, duas variáveis podem fazer referência ao mesmo objeto; portanto, operações em uma variável podem afetar o objeto referenciado pela outra variável. Com tipos de valor, cada variável tem sua própria cópia dos dados e as operações em uma variável não podem afetar a outra (exceto no caso das variáveis de parâmetros in, ref e out. Confira o modificador de parâmetro in, ref e out).

Quando você atribui um novo valor a uma variável de um tipo de valor, esse valor é copiado. Quando você atribui um novo valor a uma variável de um tipo de referência, a referência é copiada, não o objeto.



**Tipo de Acessibilidade(Classes e métodos)**.

Use os modificadores de acesso, public, protected, internal ou private, para especificar um dos níveis de acessibilidade declarada a seguir para membros.

Private  
O tipo ou membro pode ser acessado somente pelo código na mesma classe ou struct.

Protected  
O tipo ou membro pode ser acessado somente pelo código na mesma classe ou em uma classe derivada dessa classe.

Internal  
O tipo ou membro pode ser acessado por qualquer código no mesmo assembly, mas não de outro assembly.

protected internal   
O tipo ou membro pode ser acessado por qualquer código no assembly no qual ele é declarado ou de uma classe derivada em outro assembly.

private protected   
O tipo ou membro pode ser acessado somente dentro de seu assembly de declaração, por código na mesma classe ou em um tipo que é derivado dessa classe.

Os modificadores de acesso não são permitidos em namespaces. Namespaces não têm nenhuma restrição de acesso.

Dependendo do contexto no qual ocorre uma declaração de membro, apenas algumas acessibilidades declaradas são permitidas. Se não for especificado nenhum modificador de acesso em uma declaração de membro, uma acessibilidade padrão será usada(para classes será o private.

**Construtores  
Destrutores  
Condicionais**   
**Loops**   
**Interrupção de Loops**   
**Métodos**

Descreva as principais funções e o que encontramos do ponto de vista macro (Só o que contém, de forma mais abrangente) nos namespaces e destaque no mínimo 03 classes que podemos trabalhar em cada um deles:   
System.Text,   
System.Threading,   
System.Linq,   
System.IO,   
System.Data,   
System.Security,   
System.Diagnostics,   
System.Runtime,   
System.Globalization,   
System.Buffers;

**Trabalhando com Parâmetros:** de entrada, de saida, objeto parâmetro, funções como pârametros, refletindo sobre efeitos colaterais (Imutabilidade e porque é importante).

<https://pt.stackoverflow.com/questions/15510/o-que-imut%C3%A1vel-realmente-significa>

O que são e para que servem Interfaces, e o que são Generics?

**Coleções 01 (System.Collections)** –   
Arrays,   
Dictionary,   
SortedDictionary,   
HashSet,   
Listas,   
SortedList,   
Queues,   
Stack.   
Quais são as diferenças entre esses tipos, tem que métodos distintos, qual o uso indicado?

Bibliografia:

<https://docs.microsoft.com/pt-br/dotnet/csharp/language-reference/keywords/accessibility-levels>

<https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/classes-and-structs/access-modifiers>

<https://www.caelum.com.br/apostila-csharp-orientacao-objetos/>

<https://www.devmedia.com.br/principais-conceitos-da-programacao-orientada-a-objetos/32285>

<https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/classes-and-structs/structs>

<https://docs.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/structs>

<https://docs.microsoft.com/pt-br/dotnet/csharp/language-reference/keywords/reference-types>

<https://www.tutlane.com/tutorial/csharp/csharp-value-type-and-reference-type-with-examples>