# certification-70-483

Topicos para serem revisados

Tipos de referência.

Struct.

Dynamic

Methodos

Propriedades indexadas

projetos e anotações de estudo do meu preparatório para a certificação c#

Atalhos

cw tab taba -> atalho para console.wrtiLine();

Curso Alura

Aula 01

Tipo de valores

'=' -> operador de atribuição.

Tipo INT

internamente é um struct, que é uma estrutura do DOTNET

pode se utilizar System.int32

não pode ser um valor NULL

tipo inteiro anulavel, colocar interrogação a frente do tipo do valor

ex -> int? idade = null

tipo inteiro nulo, é um tipo do dotnet do system Nullable

Números inteiro

Char = faz parte dos tipos inteiro, pois ele armazena número apresentando caracter mas grava a quantidade de byte.

byte = armazena valores de 0 a 255, não permite numeros negativos.

short = um número inteiro curto, que contém apena 2 byte ou System.int16

int = um númeor inteiro representando no system o int32 contendo 32 bits, aceitando números negativo

long = inteiro longo, equivalente no dotnet com system int 64.

outros tipos que armazena valors

bool - float - int - decimal

variações dos tipos

sbyte = aceita valor negativo e seu equivalente é system.sbyte

ushort = Número inteiro que não contém sinal, valor unsigned

unit = unsigned do tipo int

Tabela de tipos inteiros

Ponto flutuante

Ao declara uma variável por tipo float deve ser colocado o sufixo no final do valor indicando que aquele valor é um ponto flutuante sufixo(f).

Tipo float é um tipo simples do dotnet, com uma simples precisão

Double - variável com dupla precisão e com capacidade maior de armazenamento

Decimal – Ao declarar uma variável de maneira literal, deve ser colocado um sulfixo no final do valor ex: 10.2m.

Structs

Automaticamente chama um construtor dessa struct que por padrão é zero.

Structs não podem ter construtores sem parâmetros

Enum

Toda enumeração começa com o valor zero por padrão, podendo atribuir um valor para cada elemento do enum

Pode se trocar o tipo do enum colocando : tipo que desejar

Atributo flags

Tipo de referencias

Quando se faz uma copia de uma variável para outra, não faz uma copia independente ele aponta o tipo de variável para a mesma memoria alocada, trabalhando como ponteiro.

Classes

Não pode utilizar uma estancia de uma classe sem antes ter uma referência para esse objeto, para criar uma nova estancia deve se atribuir o = new NomeClase();

Classe é dividida em partes, ela contém dados, propriedades, construtores e método.

método da classe podem ou não receber parâmetros.

Modificador de acessibilidade

Internal é visível a qualquer código que esteja dentro do mesmo assembly.

Private Só pode ser acessada onde ela está contida

Public deixa a classe visível para todos que tenham acesso ao assembly (Projeto).

Interface

Pode se criar uma assinatura de classes diferente passando seus methodos,valor de retorno e parametrtos.

Para adicionar mais de uma interface na mesma classes, deve separar por virgula.

Delegates

Delegates podem ser armazenados como variáveis de referência, que representam e apontam para um determinado método, e são muito usados em combinação com eventos.

Representa um método

O que interessa é o valor de retorno e o valor de entrada

Fácil troca de método quando necessário.

Pode ser iniciado com um método anônimo

Pode ser utilizado uma expressão lambida

Objetos

É o tipo mais básico do dotnet, pois todos os outros tipos derivam do OBJETO

Não pode fazer uma atribuição de uma class ao um tipo de objeto, deve ser feito uma conversão explicita(casting)

Ex: classRef = (Jogador)meuObjeto;

Dynamic

Depois de compilado ele se comparta como um objeto.

Não verifica o tipo que pode ser usado em tempo de desenvolvimento, assim retornando erro apenas em run time.

Operações que contêm expressões dynamic não são resolvidas ou verificadas pelo compilador.

O tipo dynamic existe somente em tempo de compilação e não em tempo de execução.

Methodos

Construtor é um bloco de código parecido com o methodo que contém o mesmo nome da classe e não tem retorno

Static:

Um membro estático de um tipo existe como parte do tipo, não como uma instância específica (objeto). Ele é acessado através do nome da classe. Membros estáticos são ótimos para armazenar coisas como limites (máximo e mínimo) e valores-padrão (ex.: nota aceitável para um exame escolar), uma vez que estes só precisam ser armazenados para uma classe inteira, não para cada objeto.

Modificadores do medodo

Public – Não tem restrição de uso, pode ser usado dentro ou fora do projeto

O modificador public deixa o membro visível para código externo. Além disso, uma propriedade pode adicionar regras no acessador set para validar e restringir a faixa de valores aceitos.

Internal – só pode ser acesso a partir do mesmo projeto (Assembly)

Protected (+internal)

Com protected, o método MudarMarcha só pode ser acessado pela classe veículo ou por uma classe derivada dela.

Private (+ internal) - só pode ser acessado de dentro da própria classe

Sobrecarga de método

Quando se tem uma variação de methodos, essa diferença tbm pe chamada de assinatura.

Quando passar valor para um método, isso é chamado de argumento, quando o valor chega dentro método é chamado de parâmetro.

Contrutor

Não é obrigatório. A maioria dos tipos exigirá pelo menos um construtor, para que o seu aplicativo possa garantir que o objeto seja inicializado e tenha um estado inicial válido. A construção dos objetos em sua aplicação deve ser planejada no momento em que o aplicativo é projetado.

Argumentos nomeados

Para associar os parâmetros de uma função deve se colocar o nome do argumento seguido de :, como no exemplo.

ImprimirDetalhesDoPedido(numeroPedido: 31,nomeProduto: "Caneca",vendedor: "Fatinha");

Pode ser utilizados os parâmetros de forma variada, os parâmetro não ordenados devem ficar nas posições corretas do argumentos da função.

Parametros Opcional

Inicializar a variável passada como argumento na classe

Isso pode ser útil, porque pode simplificar o uso dos métodos. Entretanto, isso causa uma forma de ocultação de informações, em que os usuários do método não sabem quais são os valores-padrão (e, nesse caso, a documentação é necessária). Os valores-padrão devem ser definidos para os parâmetros que **realmente** são opcionais, e não para os parâmetros essenciais de um método.

Método de extensão

Isso mesmo. Um método de extensão deve ser implementado numa classe pública e estática, e o parâmetro que representa a classe estendida deve ser marcado com o modificador this.

Métodos Sobrecarregados

Ela permite a existência de vários métodos de mesmo nome, porém com assinaturas que variam no número, ordem e tipo dos parâmetros.