Apresentação

Curso:

Programação .NET I

Aula 05:

Objetivos:

Classes

Campos, propriedades, métodos e construtores;

Disparo de exceção;

Estruturas de tipo (*struct*);

Campos, propriedades e métodos;

Sobrecarga de operadores;

Introdução

Até o momento, trabalhamos com tipos primitivos e já acessamos algumas classes do .NET.

Sabemos que na programação, em geral, temos a necessidade de implementarmos as nossas próprias estruturas, algoritmos reutilizáveis, tipos próprios, etc.

A partir de agora vamos aprender como trabalhar com orientação a objetos no C# e aprender as suas diferenças em relação à outras linguagens.

Classes

Classe é a palavra raiz para “Classificação”. Dependendo da capacidade de abstração do programador, essas “classificações” podem ser muito abstratas (visão macro do problema) e irem seguindo para uma representação mais especialista (visão micro do problema).

A utilização de classes se dá quando queremos representar algo do mundo real para o mundo computacional. Com isso, criamos objetos que atendem melhor a nossa necessidade para resolver um determinado problema.

Classes

A base dos objetos no .NET é a classe System.Object. Nela estão as implementações básicas de qualquer objeto implementado. Sem você saber, sua nova classe herdará as características da System.Object.

Basicamente, as classes possuem:

- Campos
- Propriedades
- Métodos
- Construtores

Classes (Escopo)

```
class NomeDaClasse
{
    tipo MetodoUm(int parametroUm)
    {
        return valorY;
    }

    tipo MetodoDois()
    {
        return valorX;
    }

    tipo campoUm;
    tipo campoDois;
}
```

PS: Se o acesso não for definido explicitamente, então automaticamente será definido como private

Classes (Escopo)

```
class NomeDaClasse
{
    acesso ipo MetodoUm(int parametroUm)
    {
        return valorY;
    }

    acesso tipo MetodoDois()
    {
        return valorX;
    }

    acesso tipo campoUm;
    acesso tipo campoDois;
}
```

PS: Iniciar nome com letras Maiúsculas quando o acesso for público e com letras minúsculas quando for privado.

Classes (Métodos)

```
class NomeDaClasse
{
     //Um metodo
    acesso tipo MetodoUm(int parametroUm)
    {
        return valorY; //precisa ser do mesmo tipo do método
    }

   //método sobrecarregado
   acesso tipo MetodoUm(int parametroUm, int parametroDois)
    {
        return parametroUm * parametroDois;
          //precisa ser do mesmo tipo do método
    }

     //Outro método
    acesso tipo MetodoDois()
    {
        return valorX; //precisa ser do mesmo tipo do método
    }

    acesso tipo campoUm;
   acesso tipo campoDois;
}
```

Classes (Construtores)

```
class NomeDaClasse
{
     //um construtor padrão
    acesso NomeDaClasse()
    {
        //BLOCO DE INSTRUÇÕES
        //um construtor não retorna um tipo
    }

    //um construtor não padrão sobrecarregado
    acesso NomeDaClasse(tipo paramUm, tipo paramDois)
    {
         //BLOCO DE INSTRUÇÕES
    }

    acesso tipo campoUm;
    acesso tipo campoDois;
}
```

Classes (Propriedades)

```
class NomeDaClasse
{
    //inicia com Maiúscula quando public e minúscula quando private
    acesso tipo NomeDaPropriedadeUm
    {
        get //Bloco de instruções para leitura
        {
            //INSTRUÇÕES PARA LEITURA DA PROPRIEDADE
        };
        set //Bloco de instruções para escrita
        {
            //INSTRUÇÕES PARA LEITURA DA PROPRIEDADE
        };
    }

    acesso tipo campoUm;
    acesso tipo campoDois;
}
```

Classes (Exemplo Acesso)

```
class Bicho
{
    Bicho()
    {
        raca = “”;
        idade = 0;
    }

    Bicho(string raca, int idade)
    {
        this.raca = nome;
        this.idade = idade;
    }

    int idade;
    string raca;
}
```

PS: Observe que esta classe é inútil, pois os métodos e os campos serão todos privados. Não haverá como acessar nada!

Classes (Exemplo Construtores)

```
class Bicho
{
    public Bicho()
    {
        raca = “”;
        idade = 0;
    }

    public Bicho(string raca, int idade)
    {
        this.raca = nome;
        this.idade = idade;
    }

    int idade;
    string raca;
}
```

Classes (Exemplo Método)

```
class Bicho
{
    public string VerificarEstagioDeVida()
    {
        if (idade <= 2)
            return "filhote";
        else if (idade <= 4)
            return "jovem";
        else if (idade <= 10)
            return "meia vida";
        else
            return “idoso";
    }

    string nome;
    string raca;
}
```

Classes (Exemplo Propriedades)

```
    class Bicho
    {
        public string Raca
        {
            set
            {
                this.raca = value;
            }

            get
            {
                return this.raca;
            }
        }

...
        string nome;
        string raca;
    }
```

Classes (Apontamentos)

- As classes no C# não permitem heranças múltiplas (para isso utiliza-se de Interfaces);
- Os campos de uma classe são inicializados automaticamente com *0(zero)*, *false* ou *null* (é boa prática explicitar a inicialização)*;*
- A regra de letras maiúsculas e minúsculas se torna exceção no caso de construtores (se o construtor for privado, ainda assim inicia-se com letra Maiúscula)
- Se o você não definir um construtor padrão, o compilador irá fazer automaticamente
- Pode-se escrever parte de um código de uma classe num arquivo e outra parte em outro arquivo (utiliza-se o modificador *partial class ...*). Esta técnica é muito utilizada em applications do tipo WindowsForm

Classes (Partial)

```
partial class NomeDaClasse
{
    acesso tipo MetodoUm(int parametroUm)
    {
        return valorY;
    }

    acesso tipo MetodoDois()
    {
        return valorX;
    }

    tipo campoUm;
    tipo campoDois;
}
```

Classes (Partial)

```
partial class NomeDaClasse
{
     //um construtor padrão
    acesso NomeDaClasse()
    {
        //BLOCO DE INSTRUÇÕES
        //um construtor não retorna um tipo
    }

    //um construtor não padrão sobrecarregado
    acesso NomeDaClasse(tipo paramUm, tipo paramDois)
    {
         //BLOCO DE INSTRUÇÕES
    }

}
```

Exercício Prático (Classes)

Exercício 1a:
Name: pessoa
Solution name: Nome_Sobrenome-Aula05

Com base no exercício 3b da Aula 04, modifique-o para utilizar um objeto do tipo Pessoa.

Implemente esta classe dividindo-a em dois arquivos diferentes. Coloque os métodos e construtores em um, e as propriedades no outro.

PS: Utilize o modificador `partial` para dividir a classe em dois arquivos.

Aplicações WindowsForm

Estudamos as classes e instanciamos um objeto para uma classificação própria para a nossa necessidade. A partir de agora, vamos trabalhar com a forma gráfica de aplicações .NET.

Para isso, devemos criar uma nova Application (File, New, Project) e escolher no template “Windows, Windows Form Application”.

Aplicações WindowsForm

Adicione a nova Application à solution já utilizada e dê o nome “hello_world_form”. Clique em “OK”.

No primeiro momento, observamos um formulário (gráfico) no lugar onde ficavam os códigos (textos). Observe que o nome da tab é “Form1.cs[Design]”, isto significa que é a visualização gráfica de uma implementação de códigos que o Visual Studio já fez para você.

Aplicações WindowsForm

Na Solution Explorer, podemos observar os arquivos: Form1.cs (que pode ser expandido) e Program.cs.

Já estamos familiarizados com o Program.cs, que é onde está o método Main da nossa aplicação. Porém, o arquivo Form1.cs é algo novo para nós. Vamos olhá-lo melhor...

Aplicações WindowsForm

Se clicarmos duas vezes no Form1.cs na janela da Solution Explorer, o Design Editor irá abrir a representação gráfica.

Clique na setinha ao lado esquerdo do item Form1.cs na Solution Explorer. Veremos que são expandidos mais dois itens...

Aplicações WindowsForm

O item Form1.Designer.cs contém os códigos necessários para gerar aquele formulário do jeito que vimos no Design Editor. Observe que o código está ocultado por um *region* com o nome “Windows Form Designer generated code”.

Aplicações WindowsForm

Olhe o método *InitializeComponent()*. Nele estarão as definições de propriedades de todos os controles que você adicionar ao formulário.

Aplicações WindowsForm

Feche os arquivos abertos e dê um duplo clique no Form1.cs. Observe o lado esquerdo do Design Editor. Temos os controles existentes do Framework .NET, clique em “Toolbox” (aproveite para fixar a janela no ‘pin’).

Os controles estão todos divididos em categorias, porém, vamos utilizar, por agora, os controles da “Commom Controls”.

Aplicações WindowsForm

Podemos ver os botões, caixas de texto, labels, radiobutton etc...

Explore os controles que estão disponíveis e lembre-se: Cada controle desse é uma classe! Possuem Métodos, Propriedades, Contrutores e, uma novidade, Eventos (veremos mais adiante).

Exercício Prático (WindowsForm)

Exercício 2:
Name: hello_world_form
Solution name: Add to solution

Crie um programa com: Um formulário; Um botão e Um TextBox.

Quando o usuário apertar o botão, o programa deve gerar uma caixa de mensagem com o conteúdo escrito no TextBox.

PS: Utilize a classe MessageBox para gerar a caixa de mensagem.

Resumo de aula

- Mandem os exercícios de aula para a tarefa no moodle.