INTRODUÇÃO AO FRAMEWORK .NET

Cairu

Fundação Visconde de Cairu - Desde 1905

AULA 4 – ORIENTAÇÃO A OBJETOS

- Conceito
- Classes e Objetos
- Atributos e Métodos
- Construtores

APRESENTAÇÃO DIA 1 – 11/07 – PROF. ARISTÓTELES

Início	Fim	Atividade
18:30	19:20	Apresentação
19:20	20:00	Introdução ao Framework .Net e Apresentação do Visual Studio 2013 (github)
20:00	20:10	Intervalo
20:10	20:50	Introdução ao C#: tipos de dados, if, else, case, for, while, e/s console
20:50	21:30	Prática 1 – Console Application

PROGRAMAÇÃO ORIENTADA A OBJETOS

Orientação a objetos consistem em considerar os sistemas computacionais como um coleção de objetos que interagem de maneira organizada.

POO – HISTÓRICO

- A POO foi criada para tentar aproximar o mundo real do mundo virtual: a idéia fundamental é tentar simular o mundo real dentro do computador
- Na POO o programador é responsável por moldar o mundo dos objetos, e explicar para estes objetos como eles devem interagir entre si
- Os objetos "conversam" uns com os outros através do **envio de mensagens.** Junto com algumas dessas mensagens ainda é possível passar algumas informações para o objeto (parâmetros)

POO – CONCEITOS

Qualquer linguagem orientada a objetos deve oferecer suporte aos seguintes conceitos da POO:

- Abstração
- Encapsulamento
- Herança
- Polimorfismo

POO – ABSTRAÇÃO

Definição: Abstração pode ser definida como a capacidade de representar cenários complexos usando termos simples.

Cenário 1: “Vou abrir a porta daquele veículo movido a combustível, entrarei, me sentarei, darei a partida no motor, pisarei na embreagem, engatarei a primeira marcha, acelerarei, controlarei a direção em que o carro irá se mover utilizando o volante”.

Cenário 2: “vou usar o meu carro para ir ao trabalho amanhã”

POO – ABSTRAÇÃO

Correia (2006, p. 11) afirma que “pelo princípio da abstração, nós isolamos os objetos que queremos representar do ambiente complexo em que se situam, e nesses objetos representamos someta as características que são relevantes para o problema em questão”

CORREIA, Carlos; TAFNER, Malcon. Análise orientada a objetos. 2. ed. Florianópolis: Visual Books, 2006.

POO – CLASSE

Uma **classe** é uma abstração que define um tipo de objeto e o que objetos deste determinado tipo tem dentro deles (seus **atributos**) e também define que tipo de ações esse tipo de objeto é capaz de realizar (**métodos**).

POO – CLASSE (EXEMPLO)

Classe Cachorro

Características (Propriedades, Atributos)

- Raça
- Cor

Comportamentos (Ações, Métodos)

- Latir
- Farejar
- Correr

POO – CLASSE

- Uma classe representa um conjunto de objetos que possuem características e comportamentos comuns
- A POO é baseada na criação das classes, que a partir daí cria-se objetos e depois esses objetos interagem entre si

POO – CLASSE EM VB.NET

PR: indica início da classe

Nome da classe

```
class ClasseSimples
{

      public string SimplesAtributo;
      public void SimplesFunc(string SimplesParam)
      {
            // Add code here
      }

}
```

PR: indica fim da Classe

POO – ENCAPSULAMENTO

- O encapsulamento é um dos pilares da orientação a objetos sua característica é ocultar partes da implementação desta forma construir softwares que atinjam suas funcionalidades e escondam os detalhes de implementação do mundo exterior. (SINTES, 2002, p. 22 – 23)
- Correia (2006, p. 13) afirma que “as pessoas que usam os objetos não precisam se preocupar em saber como eles são constituídos internamente acelerando o tempo de desenvolvimento”.

POO – ENCAPSULAMENTO

Modificador do Visual Basic	Modificador C#	Definição
Public	Public	O tipo ou membro pode ser acessado por qualquer outro código no mesmo assembly ou em outro assembly que faça referência a ele.
Private	Private	O tipo ou membro só pode ser acessado por código na mesma classe.
Protected	Protected	O tipo ou membro só pode ser acessado por código na mesma classe ou em uma classe derivada.
Friend	Internal	O tipo ou membro pode ser acessado por qualquer código no mesmo assembly, mas não de outro assembly.
Protected Friend	Protected Internal	O tipo ou membro pode ser acessado por qualquer código no mesmo assembly ou por qualquer classe derivada em outro assembly.

POO – ATRIBUTO

- Os atributos são entidades que caracterizam um objeto gerada a partir da classe.
  - Por exemplo: A classe Carro possui o atributo **cor.** Um objeto dessa classe poderá assumir as cores amarelo, vermelho ou verde.
- Todo atributo deverá ser mapeado para um ou mais variáveis na classe
- Nem toda variável da classe é um atributo
- Pelo conceito de encapsulamento, os atributos não podem ser acessados diretamente. Portanto a classe deve possuir métodos de acesso aos atributos
- Os métodos de acesso são mais conhecidos como get/set

POO – ATRIBUTO EM C#.NET

```
class ClasseSimples
{
    private string SimplesAtributo;
    public void SimplesFunc(string SimplesParam)
    {
        // Add code here
    }
}
```

PR: Modi dor — Tipo de dado — Nome do atributo

de Acesso

POO – MÉTODO GET/SET

```
private string cor;
public string GetCor()
{
   return cor;
}

public void SetCor(string _cor)
{
    this.cor = _cor;
}
```

POO – ATRIBUTO – PROPRIEDADE

- A Propriedade é a declaração de uma variável com os métodos de acesso já implementados
- As propriedades têm procedimentos get e set, o que oferece mais controle sobre como os valores são definidos e retornados.
- Se precisar realizar algumas operações adicionais para leitura e gravação do valor da propriedade, defina um campo para armazenar o valor da propriedade e forneça a lógica básica para armazenar e recuperá-lo, ou
- A Propriedade pode ser implementada de forma automática

POO – ATRIBUTO – PROPRIEDADE

```
private string m_Sample;
public string Sample
{
    get { return m_Sample; }
    set { m_Sample = value; }
}
```

POO – ATRIBUTO – PROPRIEDADE - AUTOMÁTICA

```
public string SampleProperty { get; set; }
```

POO – MÉTODOS

- Os métodos são responsáveis por definir o comportamento dos objetos gerados a partir de uma classe
- Quando um objeto de uma classe receber uma mensagem de algum outro objeto contendo o nome de um método, a ação correspondente a este método será executada.
- Um objeto é manipulado a partir dos seus métodos
- No Visual Basic, há duas maneiras para criar um método: a declaração com **Void** é usada se o método não retorna um valor; a declaração com **tipo de dado** é usada se um método retorna um valor

POO – MÉTODOS COM RETORNO

```
class ClasseSimples
{
    public string SimplesAtributo;
    public string SimplesFunc(string SimplesParam) {
        // Add code here
    }
}
```

Tipo de Dado do Retorno

Nome do Método

Tipo de Dado do Parâmetro

Nome do Parâmetro

POO – MÉTODOS SEM RETORNO (**VOID**)

```
class ClasseSimples
{
    public string SimplesAtributo;
    public void SimplesFunc(string SimplesParam) {
        // Add code here
    }
}
```

Tipo de Dado do Retorno

me do Método

Tipo de Dado do Parâmetro

Nome do Parâmetro

POO – MÉTODOS – PARÂMETROS

- Existem duas formas de fazer a passagem de parâmetros nos métodos: passagem por valor (ByVal) e a passagem por referência (ByRef)
- Por Valor é feita uma cópia do valor passado na chamada do método para o parâmetro do método
- Por Referência é o parâmetro do método passa a apontar para a variável passada na chamada do método. Quando houver alteração no parâmetro irá refletir na variável passada na chamada no método

POO – OBJETOS

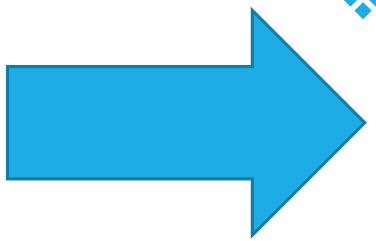
- Objetos são instancias de uma determinada classe
  - A instanciação é quando uma classe produz um objeto, como se a classe fosse uma espécie de modelo para a criação de objetos
- Para Ambler (1998, p. 5) “Um objeto é qualquer indivíduo, lugar, evento, coisa, tela, relatório ou conceito que seja aplicável ao sistema”
- Segundo Pfleeger (2004, p. 213), “cada instância tem seus próprios valores de atributos, mas compartilha o nome e os comportamentos dos atributos com a outras instancias da classe”.

AMBLER, Scott W. Análise de projeto orientado a objeto. 2. ed. Rio de Janeiro: Infobook, 1998.

POO – CLASSE X OBJETO

CLASSE

- Abstrata
- Modelo
- Escrita pela programador
- Especifica as operações



OBJETO

- Criação
- Instanciada pelo programador
- Gerada a partir de uma classe
- Executa as operações

POO – CLASSE X OBJETO

- Note que uma Classe não tem vida, é só um conceito. Mas os Objetos (animais, serem humanos, pássaros, etc) possuem vida. O seu cachorro rex é um Objeto (ou instância) da classe Cachorro. A classe Cachorro não pode latir, não pode fazer xixi no poste, ela apenas especifica e define o que é um cachorro. Mas Objetos do tipo Cachorro, estes sim podem latir, enterrar ossos, ter um nome próprio, etc.

POO – OBJETOS – CONSTRUTOR

- Objetos são instancias com o uso do operador **new**
- No exemplo abaixo, na linha 1, ListPrd é somente uma variável da classe ArrayList. Na linha 2, usando o operador new a variável Listprd passa a ser um objeto, podendo executar as operações definidas na classe ArrayList
- Seguido do operador new é invocado o método construtor da classe ArrayList

```
ArrayList ListPrd;
ListPrd = new ArrayList();
```

POO – OBJETOS – CONSTRUTOR

- Construtores são os métodos da classe que são executados automaticamente quando um objeto de um determinado tipo é criado.
- Um construtor pode executar apenas uma vez quando uma classe é criada.
- Quando um construtor é invocado reserva-se um espeço de memória para o objeto a ser criado. Uma variável recebe a referência desse espaço de memória

```
ArrayList ListPrd;
ListPrd = new ArrayList();
```

POO – OBJETOS – CONSTRUTOR

- O espaço de memória criado para o objeto pode ser referenciado por mais de uma variável.
- O exemplo abaixo ListPrd_1 possui a referência de um objeto. Depois ListPrd_2 passa a possuir a mesma referência de ListPrd_1.

```
ArrayList ListPrd_1;
ArrayList ListPrd_2;
ListPrd_1 = new ArrayList();
ListPrd_2 = ListPrd_1;
```

POO – OBJETOS – CONSTRUTOR

```
Public Class SampleClass
{
    Public SampleClass()
    {
        // Add code here
    }
}
```

POO – CLASSE EM C#.NET

Início da classe

Nome da classe

```
class ClasseSimples
{
    public string SimplesAtributo;

    public void SimplesFunc (string SimplesParam)
    {
        // Add code here
    }

}
```

Tipo de dado

Nome do atributo

Modificador de Acesso

Tipo de Retorno

Nome do Método

Tipo de Dado

Nome do Parâmetro

Início do Método

Fim do Método

Fim da Classe

POO X ESTRUTURADA

PRÁTICA 4