IFRN

PROGRAMAÇÃO ORIENTADA A OBJETOS EM C#

Propriedades

Prof. Gilbert Azevedo

Objetivos

- Conhecer os tipos de membros em estruturas e classes no C#
- Utilizar propriedades na escrita de classes
  - Definição e utilização de propriedades
  - Propriedades somente-leitura
  - Propriedades auto-implementadas
  - Propriedades com expressões

Tipos de Membros

- Estruturas e classes no C# podem ter os seguintes membros:
  - Campos (ou atributos): variáveis que armazenam o estado do objeto
  - Constantes: campos declarados como constantes (const)
  - Métodos: ações que podem ser executadas, definem o comportamento
  - Propriedades: características do objeto acessadas como atributos (ações)
  - Indexadores: permitem a indexação de objetos (dicionários)
  - Construtores: métodos que iniciam os dados de um objeto
  - Finalizadores: métodos que finalizam a execução de um objeto
  - Eventos: mecanismos para notificar ocorrências para outros objetos (ações do usuário)
  - Tipos aninhados: tipos declarados dentro de outro tipo (iterador)

Propriedades

- Propriedades são membros que permitem recuperar, definir ou calcular uma informação do objeto
  - Em geral, substituem os métodos de acesso (set e get) para um atributo
  - Permitem que os dados sejam acessados mais facilmente, sem comprometer a segurança
  - São métodos especiais chamados de *acessadores*
- Exemplos de Propriedades
  - Length (tamanho) de uma String ou Array – somente leitura
  - Current de um IEnumerator – somente leitura
  - Text de uma TextBox – leitura e escrita

Classe com Métodos de Acesso

- Classe Triangulo escrita com métodos de acesso

```
class Triangulo {
  private double b, h;
  public double GetBase() { return b; }
  public double GetAltura() { return h; }
  public void SetBase(double b) {
    if (b > 0) this.b = b; }
  public void SetAltura(double h) {
    if (h > 0) this.h = h; }
  public double CalcArea() {
    return b * h / 2;
  }
}
```

```
class MainClass {
  public static void Main () {
    Triangulo x = new Triangulo();
    x.SetBase(10);
    x.SetAltura(20);
    Console.WriteLine(x.GetBase());
    Console.WriteLine(x.GetAltura());
    Console.WriteLine(x.CalcArea());
  }
}
```

Classe com Propriedades

- Métodos da classe são substituídos por propriedades

```
class Triangulo {
  private double b, h;
  public double Base {
    get { return b; }
    set { if (value > 0) b = value; }
  }
  public double Altura {
    get { return h;}
    set { if (value > 0) h = value; }
  }
  public double Area {
    get { return b * h / 2; }
  }
}
```

```
class MainClass {
  public static void Main () {
    Triangulo x = new Triangulo();
    x.Base = 10;
    x.Altura = 20;
    Console.WriteLine(x.Base);
    Console.WriteLine(x.Altura);
    Console.WriteLine(x.Area);
  }
}
```

Utilizando as Propriedades

- Instanciando a classe
  - `Triangulo x = new Triangulo();`
- Escrevendo um valor na propriedade (set). O valor é copiado para o identificador *value*
  - `x.Base = 10;`
  - `x.Altura = 20;`
- Lendo o valor da propriedade (get)
  - `Console.WriteLine(x.Base);`
  - `Console.WriteLine(x.Altura);`
  - `Console.WriteLine(x.Area);`

```
set { if (value > 0) b = value; }
```

```
get { return b; }
get { return b * h / 2; }
```

Método de Acesso x Propriedades

- Métodos de Acesso

```
public double GetBase() {
  return b;
}
public void SetBase(double b) {
  if (b > 0) this.b = b;
}

Triangulo x = new Triangulo();
x.SetBase(10);
Console.WriteLine(x.GetBase());
```

- Propriedades

```
public double Base {
  get { return b; }
  set { if (value > 0) b = value; }
}

Triangulo x = new Triangulo();
x.Base = 10;
Console.WriteLine(x.Base);
```

Inicializadores de Objeto

- É possível atribuir valores a campos e propriedades públicas de um objeto sem utilizar um construtor

```
public static void Main () {
  Triangulo x = new Triangulo();
  x.Base = 10;
  x.Altura = 20;
}

public static void Main () {
  Triangulo x = new Triangulo { Base = 10, Altura = 20 };
}
```

Restrições das Propriedades

- Propriedades não armazenam valores na memória (não são atributos)
- Não podem ser usadas como argumentos *ref* ou *out* (não são alocadas na memória)
  - Metodo(ref x.Base);
- Contêm no máximo um *get* e um *set*
- Os *gets* e *sets* não podem receber parâmetros

Propriedades Auto-Implementadas

- Inseridas no C# 3, permitem a declaração concisa de propriedades, substituindo a declaração de atributos privados e propriedades get e set para cada atributo
- Valores armazenados não são validados
  - Padrão POCO (plain old C# objects)

```
class Contato {
  public string Nome { get; set; }
  public string Email { get; set; }
  public string Fone { get; set; }
  public DateTime Nascimento { get; set; }
}
```

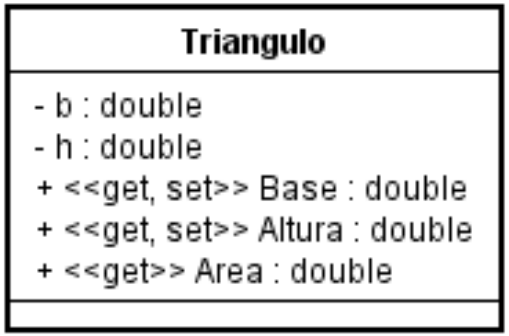
Propriedades com Expressões

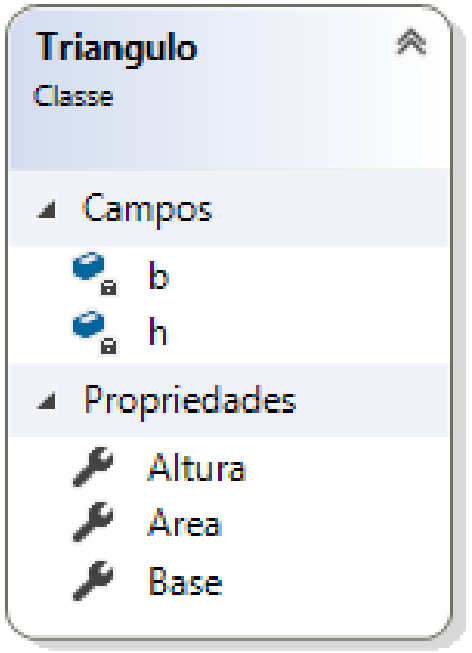
- Inseridas no C# 7, utilizam expressões (expression-body) para a definição das instruções get e set da propriedade

```
class Triangulo {
  private double b, h;
  public double Base   { get => b; set => b = value > 0 ? value: 0; }
  public double Altura { get => h; set => h = value > 0 ? value: 0; }
  public double Area   { get => b * h / 2; }
}
```

Propriedades e UML

- As propriedades serão representadas, neste curso, como atributos públicos da classe com os estereótipos *<<get, set>>*.





Referências

- Microsoft Visual C# 2010 – Passo a passo, John Sharp, Bookman, 2010
- Propriedades (Guia de Programação em C#)
  - https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/classes-and-structs/properties
- Inicializadores de Objetos
  - https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/classes-and-structs/object-and-collection-initializers