Fundamentos de C#

Convenções e Boas Práticas

Tópicos Abordados

- Nome dos Elementos
- Properties
- Fields
- Layout
- Comentários
- Documentação

Convenções? Boas Práticas?

- Convenções de código são regras que visam facilitar o entendimento e manutenção do código
  - Não seguir convenções não implica em escrever código errado
  - Caso não deseje seguir uma convenção, tenha um bom motivo para fazer isso
- Boas práticas são ações tomadas pelo desenvolvedor consideradas corretas
  - Voltadas para a escrita de código de qualidade

Nome dos Elementos

- Nomes de variáveis, parâmetros e fields usam *camel casing*

```
int altura;
double notaProvaAluno;
bool dataVisivel;
```

A primeira palavra começa com letra minúscula. As demais com letra maiúscula.

- Tipos de dados (classes, estruturas, interfaces, enums) e métodos usam *Pascal casing*

```
class AnimalTerrestre { }
enum DiaDaSemana { }
struct PontoPlano { }
void SomarNumeros() { }
```

Todas as palavras começam com letra maiúscula

Nome dos Elementos

- Constantes usam *Pascal casing*

```
const Pi = 3.1416;
const Angulo = 90;
```

- Interfaces têm seu nome iniciado por *I*

```
interface IMotorizado { }
```

- Namespaces usam *Pascal casing* e não devem interferir em namespaces de terceiros

```
namespace Softblue.Sistemas.Abc { }
```

Properties

- Properties se assemelham mais com variáveis do que com métodos
- Evite o uso de write-only properties
- Properties devem ter acesso rápido
- O acesso a uma property não deve trazer efeitos colaterais
- Não faça com que a ordem de definição das properties seja importante
- Leituras sucessivas do valor de uma property devem retornar o mesmo resultado

Fields

- Fields devem sempre possuir visibilidade *private*
  - Com exceção de read-only fields e constantes, que podem ser ter visibilidade public
- Use properties para controlar a leitura e alteração de valores dos fields

```
class Jogo
{
    string nomeJogador;

    public string NomeJogador
    {
        get { return nomeJogador; }
        set { nomeJogador = value; }
    }
}
```

Field *private, camel case*

Property public, Pascal case

Layout

- Escreva apenas um comando por linha

```
int x = 10; Console.WriteLine(x);
```

- Faça um declaração por linha

```
int x = 5, y = 10, z = 0;
```

- Use parênteses em expressões para deixar clara a prioridade

```
if (x > 1 && z < 10 || 10 + 15 * y == 10) { }
```

- Use as chaves para delimitar blocos

```
if (x > 5)
    Console.WriteLine("Grande");
else
    Console.WriteLine("Pequeno");
```

Comentários

- Escreva o comentário sozinho em uma linha
  - Não no final da linha
- Utilize a notação //
- Deixe um espaço em branco entre os caracteres //e o início do texto
- Inicie com letra maiúscula e termine com ponto final

```
// Este comentário segue as convenções
// definidas pela Microsoft.
```

Mais Convenções e Boas Práticas

- As convenções e boas práticas mostradas aqui são apenas algumas
  - Consulte os links do módulo para saber mais

Documentação

- Documentar é importante, pois outros podem precisar usar o seu código
  - Parâmetros, retorno, exceções lançadas
- C# conta com uma notação no formato XML que permite gerar a documentação do código

```
/// <summary>
///     Classe que representa um animal.
/// </summary>
class Animal
{
    /// <summary>
    ///     Faz o animal beber água.
    /// </summary>
    /// <param name="qtde">Quantidade de água</param>
    /// <returns>True se encontrou água</returns>
    public bool BeberAgua(double qtde) { }
```

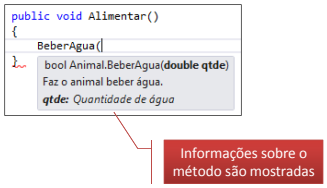
Documentação

- Ao compilar o arquivo, um arquivo XML com a documentação pode ser gerado

```
<?xml version="1.0"?>
<doc>
    <assembly>
        <name>App</name>
    </assembly>
    <members>
        <member name="T:Animal">
            <summary>
                Classe que representa um animal.
            </summary>
        </member>
        <member name="M:Animal.BeberAgua(System.Double)">
            <summary>
                Faz o animal beber água.
            </summary>
            <param name="qtde">Quantidade de água</param>
            <returns>True se encontrou água</returns>
        </member>
    </members>
</doc>
```

Documentação

- A documentação pode ser exibida pela IDE para facilitar o uso de determinado elemento



Informações sobre o método são mostradas