

Tópicos Abordados



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

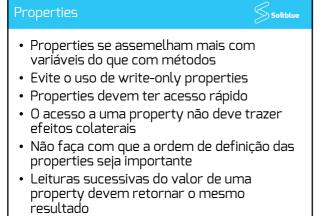
Convenções? Boas Práticas?



- 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

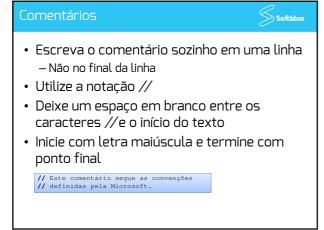
Nome dos Elementos Nomes de variáveis, parâmetros e fields usam camel casing int altura; double notaProvaAluno; bool dataVisivel; A primeira palavra começa com letra minúscula. As demais com letra maiúscula. As demais com letra maiúscula. Tipos de dados (classes, estruturas, interfaces, enums) e métodos usam Pascal casing class AnimalTerrestre { } enum DiaDaSemana { } struct PontoPlano { } void SomarNumeros() { }

Constantes usam Pascal casing Constantes usam Pascal casing Const Pi = 3.1416; const Angulo = 90; Interfaces têm seu nome iniciado por / interface IMotorizado { } Namespaces usam Pascal casing e não devem interferir em namespaces de terceiros namespace Softblue.Sistemas.Abc { } Interface IMOTORIZADO { } Namespace Softblue.Sistemas.Abc { } Namespace Softblue.Sistemas.Abc { } Softblue



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; }

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 & 2 < 10 || 10 + 15 * y == 10) () Use as chaves para delimitar blocos if (x > 5) Console.WriteLine("Grande"); else Console.WriteLine("Pequeno");



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.
/// 
/// 
/// // 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

