DIGITAL
INNOVATION
ONE

# Aula 3: Padrões de Codificação

## Introdução ao Ecossistema .NET & Documentação

DIGITAL
INNOVATION
ONE

# Objetivos

1. O que é escrever um bom código?
2. Entender o Clean Code
3. Convenções de nomenclatura

DIGITAL INNOVATION ONE

# Aula 3
# O que é escrever um bom código?

## Introdução ao Ecossistema .NET & Documentação

DIGITAL INNOVATION ONE

# “Bom código”

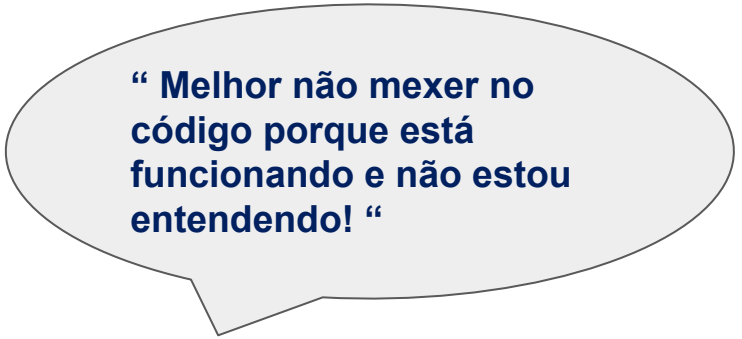
- ★ Ser confiável
- ★ Ser sustentável
- ★ Ser eficiente

Eficiência e desempenho X confiabilidade e facilidade

DIGITAL INNOVATION ONE

# Por que e como devemos padronizar?

- ➔ Melhorar comunicação entre equipe
- ➔ Facilitar manutenção de códigos
- ➔ Utilizar documentação e boas práticas de codificação, como clean code



DIGITAL INNOVATION ONE

# Aula 3
# Clean Code

## Introdução ao Ecossistema .NET & Documentação

# O que é Clean Code?

- Conjunto de boas práticas na escrita de software para obtenção de maior legibilidade e manutenibilidade de código. -> Clean Code: A Handbook of Agile Software Craftsmanship [Book] (oreilly.com)

DIGITAL INNOVATION ONE

# Regras gerais

1. Siga SEMPRE as convenções adotadas pela equipe!
2. KISS : Keep It Stupid Simple ( Matenha isto estupidamente simples)
3. Devolva o código mais limpo do que você encontrou
4. Busque sempre entender e solucionar os problemas a partir de sua raiz.

DIGITAL INNOVATION ONE

# Regras para entendimento de código

1. Seja consistente na escrita de todo o código
2. Utilize variáveis concisas e que realmente passem a informação necessária
3. Observe a necessidade de criação de objetos de valor ao invés do uso de tipos primitivos
4. Evite dependências lógicas
5. Evite condicionais negativas

**Vamos para os exemplos?**

DIGITAL INNOVATION ONE

# Regras para nomeação

1. Escolher nomes descritivos para classes, variáveis e métodos

```
var x = 10;

int tempo = 5;

int tempoEmMinutos = 30;
```

DIGITAL INNOVATION ONE

# Regras para nomeação

2. Para variáveis semelhantes, faça uma distinção identificável

```
var salario1 = 2500M;

var salario2 = 1000M;

var salarioEmReais = 5000M;

var salarioGerente = 8000M;
```

DIGITAL INNOVATION ONE

# Regras para nomeação

3. Utilizar nomes de fácil leitura e busca

```
var strTexto = "Esse texto tem uma nomeação genérica demais e de difícil pronúncia";

public void GenerateBoleto(){}
```

DIGITAL INNOVATION ONE

# Regras para nomeação

4. Utilize constantes para guardar strings a serem comparadas

```
//Evite
if(environment == "PROD"){}

//Faça
const string ENV = "PROD";

if(environment == ENV){}
```

DIGITAL INNOVATION ONE

# Regras para nomeação

5. Não use prefixos ou caracteres especiais

```
// Evite
public class clsStudent { ... }

// Evite
string strNome = "Carolina";

// Evite
var situação = "Pendente";
```

DIGITAL INNOVATION ONE

# Regras para métodos

1. Métodos não devem ser grandes e devem possuir somente um objetivo/responsabilidade

```
// Evite
public void RealizarPedido()
{
    // Cadastra o cliente
    // Aplica o desconto
    ...
}

// Fazer
public void CadastrarCliente() { ... }
public void AplicarDesconto() { ... }
...
```

DIGITAL INNOVATION ONE

# Regras para métodos

2. Métodos devem possuir nomes descritivos

```
// Evite
public void Calcular(){}

//Utilize
public void CalcularDescontoCompra(){}
```

DIGITAL INNOVATION ONE

# Regras para métodos

3. Evite a exigência de muitos parâmetros dentro do método

```
public void SalvarProduto(string nome, string tipo, string codigo, string marca, string X, string Y, ...){}

public void SalvarProduto(string nome, string tipo, string codigo = " default", int quantidade = 0){}
```

# Regras para métodos

4. Evite que uma função altere valores de outra classe sem ser a própria classe

```
// Evite
public class Produto
{
    public decimal Quantidade { get; set; }
}

var produto = new Produto();

produto.Quantidade = 10;
```

```
public class Produto
{
    public decimal Quantidade { get; private set; }

    public void CalcularQuantidade(){}
}

var produto = new Produto();

produto.Quantidade = 10; // erro, pois atributo é privado
```

DIGITAL INNOVATION ONE

# Regras para métodos

## 5. Evite utilização de flags desnecessárias

```
public class StudentRepository{

  public void CreateOrUpdate(Student Student, bool create){

    if(create){ ... }

    else{ ... }
  }
}
```

```
public class StudentRepository
{
    public void Create(Student Student) { ... }
    public void Update(Student Student) { ... }
}
```

DIGITAL INNOVATION ONE

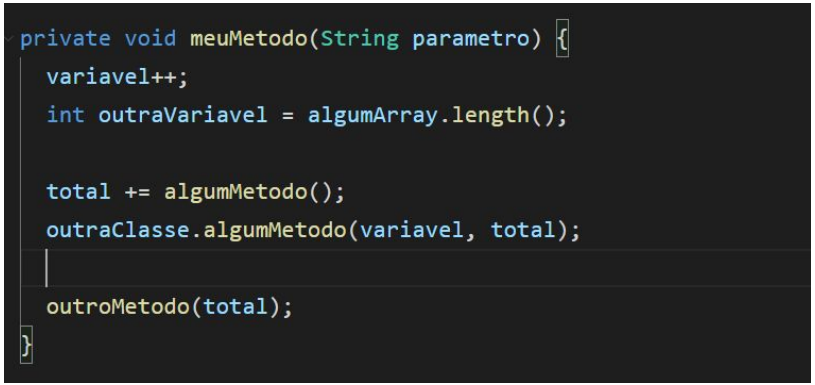
# Regras para comentários

1. Evite comentários desnecessário, torne seu código autoexplicativo
2.Não seja redundante
3. Não deixe código desnecessário comentado
4. Comentários podem ser úteis para falar sobre a intenção de uma classe ou método
5. Comentários podem explanar regras mais complexas e alertas sobre consequências mais sérias

DIGITAL INNOVATION ONE

# Regras para estruturação de código

1. Declare variáveis próximas de seu uso
2. Agrupe métodos similares
3. Declare funções de cima pra baixo
4. Mantenha poucas e curtas linhas
5. Use espaçamentos e identação corretamente



DIGITAL INNOVATION ONE

# Aula 3
# Convenções de nomenclatura

## Introdução ao Ecossistema .NET & Documentação

# Notação Húngara

➔ Facilitar o reconhecimento do tipo de variável

| Nome | Descrição |
|---|---|
| **s** | *String* |
| **sz** | Aponta o primeiro caracter da terminação zero da string |
| **st** | Ponteiro da string, o primeiro byte é contado dos caracteres |
| **h** | *handle* (título) |
| **msg** | *Message* |
| **fn** | *function* (usada com *pointer*) |
| **c** | char (8 bits) |
| **by** | unsigned char (byte or uchar - 8 bits) |
| **n** | Int |
| **b** | Boolean (verdadeiro ou falso) |
| **f** | Flag (boolean, logical) |
| **u** | integer |
| **w** | Word |
| **ch** | Char, com texto ASCII |
| **l** | long int (32 bits) |
| **dw** | unsigned long int (dword - 32 bits) |

Exemplo:

| | |
|---|---|
| **bVerdade** | **boolean** |
| **sNome** | **string** |
| **uValor** | **inteiro** |
| **msgAviso** | **message** |

DIGITAL INNOVATION ONE

# Camel Case

- Escrever palavras ou frases compostas considerando a primeira letra da primeira palavra sempre minúscula e as subsequentes maiúsculas.

Ex: valorDoDesconto, nomeCompleto, totalSalario…

DIGITAL INNOVATION ONE

# Pascal Case

➔ Escrever palavras ou frases compostas considerando a primeira letra de cada palavra maiúscula

Ex: ValorDoDesconto, NomeCompleto, TotalSalario…

DIGITAL INNOVATION ONE

# Qual o padrão para C#?

- ★ Não há uma regra obrigatória, porém grande maioria dos desenvolvedores convenciona da seguinte forma:
  - ○ Nomes de classes e métodos -> PascalCase
  - ○ Nomes de variáveis e parâmetros -> CamelCase
- ★ No caso de interfaces recomenda-se o uso do prefixo “ I “
  - ○ Ex: IEntidade, IRepositorioCliente

DIGITAL INNOVATION ONE

# Recomendações da Microsoft

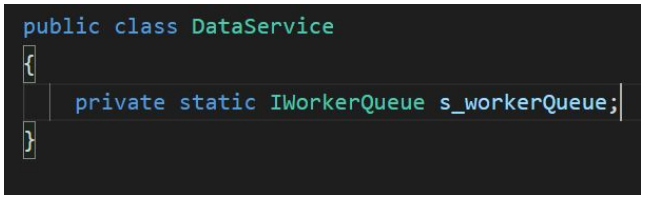
- ★ Uso do PascalCase
  - ○ Classes
  - ○ Interfaces
  - ○ Membros de tipos públicos
- ★ Uso com CamelCase
  - ○ Campos privados e internos -> deve-se ainda usá-los com prefixo “ _”.

```
public class DataService
{
    private IWorkerQueue _workerQueue;
}
```

DIGITAL INNOVATION ONE

# Recomendações da Microsoft

- ★ Uso do PascalCase
  - Classes
  - Interfaces
  - Membros de tipos públicos
- ★ Uso com CamelCase
  - Campos privados e internos
    - deve-se ainda usá-los com prefixo “ _”.
  - Campos estáticos privados ou internos
    - usar com prefixo “s_”



DIGITAL INNOVATION ONE

# Para saber mais

Tudo o que você precisa saber sobre as licenças de projetos open source | by Diego Martins de Pinho | Training Center | Medium

.NET is open source on GitHub | .NET

.NET Standard | Common APIs across all .NET implementations

Performance Improvements in .NET 5 - .NET Blog (microsoft.com)

Clean Code: A Handbook of Agile Software Craftsmanship [Book] (oreilly.com)

C# Coding Conventions | Microsoft Docs