Fundamentos de Desenvolvimento com C#

Aula 10: Classes Estáticas

Professor: Rinaldo Ferreira Junior

E-mail: rinaldo.fjunior@prof.infnet.edu.br

- **Professor:** Rinaldo Ferreira Junior
- **Graduação:** Pós-graduado em Arquitetura de Softwares
- **Atuação:** .Net | C# | SQL | NoSQL | Engenheiro de Software
- **E-mail:** rinaldo.fjunior@prof.infnet.edu.br
- **Linkedin:** https://www.linkedin.com/in/rinaldo-ferreira-junior-787326a

Aula 10: Roteiro

- Classes Estáticas
- Métodos de Extensão

Aula 10: Classes Estáticas

- São classes que não podem ser instanciadas, ou seja, não se pode usar a instrução new para criar uma instância dessa classe.
- Normalmente utilizada para uso de métodos que operam parâmetros, sem necessidade de armazenar informações.
    - Classes utilitárias
    - Como exemplo, a classe Math do .Net
- Uma classe estática permanece carregada em memória, durante todo o tempo em que o programa host estiver em execução
- Como não há instanciação, a classe e o método são chamados diretamente.

```
double value = Math.Round(100.53288, 2);
Console.Write(value);
```

```
100,53
```

Aula 10: Classes Estáticas

Instituto Infnet

- Classes estáticas são criadas da mesma forma que classes não estáticas, a diferença é que possuem o modificador de acesso static.
- Seus membros devem ser sempre, estáticos.
- O construtor, se houver, não pode receber parâmetros e não pode ser sobrecarregado.
- Classes estáticas não podem ser herdadas.

```
public static class Utils
{
    static readonly int accountDigitsSize;

    static Utils()
    {
        accountDigitsSize = 4;
    }

    public static int CreateAccountNumber()
    {
        int maxNumber = int.Parse(new String('9', accountDigitsSize));
        Random rnd = new Random();
        int accountNumber = rnd.Next(0001, maxNumber);

        return accountNumber;
    }

    public static byte CreateAccountDigit()
    {
        Random rnd = new Random();
        byte accountDigit = Convert.ToByte(rnd.Next(1, 9));

        return accountDigit;
    }

    public static bool IsValidAccount(this int account)
    {
        string sanitized = account.ToString().PadLeft(4, '0');
        return sanitized.Length == 4;
    }
}
```

Aula 10: Classes Estáticas

- Ao invocar uma classe estática, entre com o nome da classe e o método desejado.

```
static void Main(string[] args)
{

    ContaBancaria conta = new()
    {
        Agencia = "18",
        Conta = Utils.CreateAccountNumber(),
        Digito = Utils.CreateAccountDigit(),
        Instituicao = "CEF"
    };

    Console.WriteLine($"Banco: {conta.Instituicao}");
    Console.WriteLine($"Agencia: {conta.Agencia}");
    Console.WriteLine($"Conta: {conta.Conta}");
    Console.WriteLine($"Digito: {conta.Digito}");
    Console.WriteLine($"Saldo: {conta.Saldo:C}");

    Console.Read();
}
```

Aula 10: Métodos de Extensão

Instituto Infnet

- São métodos que adicionam funcionalidades à uma classe, sem a necessidade de herança ou modificação da classe original.

- São métodos estáticos, mas são chamados como se fossem métodos de instância da classe.

    - Podem receber parâmetros, mas o primeiro sempre representa o tipo de dados onde o método deve operar

```
public static class Utils
{

    public static bool IsValidAccount(this int account)
    {
        string sanitized = account.ToString().PadLeft(4, '0');
        return sanitized.Length == 4;
    }
}
```

```
static void Main(string[] args)
{
    ContaBancaria conta = new()
    {
        Agencia = "18",
        Conta = Utils.CreateAccountNumber(),
        Digito = Utils.CreateAccountDigit(),
        Instituicao = "CEF"
    };

    Console.WriteLine($"Banco: {conta.Instituicao}");
    Console.WriteLine($"Agencia: {conta.Agencia}");
    Console.WriteLine($"Conta: {conta.Conta:D4} | Válida: {conta.Conta.IsValidAccount()}");
    Console.WriteLine($"Digito: {conta.Digito}");
    Console.WriteLine($"Saldo: {conta.Saldo:C}");

    Console.Read();
}
```

Aula 10: Classes Parciais

- A instrução Partial, informa que o código de uma classe está dividido em arquivos diferentes.
- O compilador localiza e combina essas partes, gerando um único elemento compilado
- Pode ser útil em caso de acesso simultâneo ao mesmo código para modificações, em equipes grandes.

```
//Arquivo Utils.cs
namespace DadosPessoais
{

    public static partial class Utils
    {

    }
}
```

```
//Arquivo Utils1.cs
namespace DadosPessoais
{

    public static partial class Utils
    {

    }
}
```

Aula 10: Classes Parciais

- Na chamada dos membros de classes parciais, nada muda. Em design time já é possível perceber que as classes são combinadas.

```
static void Main(string[] args)
{
    ContaBancaria conta = new()
    {
        Agencia = "18",
        Conta = Utils.CreateAccountNumber(),
        Digito = Utils.CreateAccountDigit(),
        Instituicao = "CEF"
    };

    Console.WriteLine($"Banco: {conta.Instituicao}");
    Console.WriteLine($"Agencia: {conta.Agencia}");
    Console.WriteLine($"Conta: {conta.Conta:D4} | Válida: {conta.Conta.IsValidAccount()}");
    Console.WriteLine($"Digito: {conta.Digito}");
    Console.WriteLine($"Saldo: {conta.Saldo:C}");
    Console.WriteLine($"CPF Válido? {Utils.IsValidCpf("00260518778")}");

    Console.Read();
}
```