Fundamentos de Desenvolvimento com C#

Aula 08: Sobrecargas e Construtores

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 08: Roteiro

- Sobrecargas (Overloading)
- Construtores

Aula 08: Sobrecargas

Instituto Infnet

- Permite a criação de múltiplos métodos com o mesmo nome, mas com parâmetros diferentes.
- É uma técnica para criar métodos com comportamentos semelhantes sem a necessidade de criar métodos com nomes diferentes.
- Vantagens:
  - Melhor legibilidade do código
  - Consistência (operações smelhantes mantém o mesmo nome)
  - Reutilização, pois sobrecargas costumam compartilhar código
  - Flexibilidade na escolha do método mais apropriado

Aula 08: Sobrecargas

- Há diferentes meios de se criar uma sobrecarga para um método. Todos envolvem mudanças na assinatura dos métodos sobrecarregados.
  - Quantidade de parâmetros diferentes entre os métodos
  - Tipos de dado diferentes entre os métodos
  - O posicionamento dos parâmetros nos métodos

Aula 08: Sobrecargas

- Diferença no tipo de dados dos parâmetros.

- O método Debit está sobrecarregado, tendo como diferença, apenas o tipo de dado do parâmetro value.

```
public class ContaBancaria
{
    2 references
    public string Instituicao { get; set; } = string.Empty;

    2 references
    public string Agencia { get; set; } = string.Empty;

    2 references
    public int Conta { get; set; }

    2 references
    public byte Digito { get; set; }

    0 references
    public bool Poupanca { get; set; } = false;

    2 references
    public DateTime DataOperacao { get; private set; }

    4 references
    public double Saldo { get; private set; }

    0 references
    public void Debit(double value)
    {
        this.Saldo -= value;
    }

    0 references
    public void Debit(int value)
    {
        this.Saldo -= value;
    }
```

Aula 08: Sobrecargas

- Diferença na quantidade de parâmetros.

- Agora o método Debit possui uma segunda sobrecarga contendo um parâmetro a mais, para a data da operação.

```
public class ContaBancaria
{
    2 references
    public string Instituicao { get; set; } = string.Empty;

    2 references
    public string Agencia { get; set; } = string.Empty;

    2 references
    public int Conta { get; set; }

    2 references
    public byte Digito { get; set; }

    0 references
    public bool Poupanca { get; set; } = false;

    2 references
    public DateTime DataOperacao { get; private set; }

    4 references
    public double Saldo { get; private set; }

    0 references
    public void Debit(double value)
    {
        this.Saldo -= value;
    }

    0 references
    public void Debit(int value)
    {
        this.Saldo -= value;
    }

    1 reference
    public void Debit(double value, DateTime dataOperacao)
    {
        this.Saldo -= value;
        this.DataOperacao = dataOperacao;
    }
}
```

Aula 08: Uso de sobrecargas

- Mesmo método, valores de tipos diferentes

```
static void Main(string[] args)
{
    ContaBancaria conta = new()
    {
        Agencia = "18",
        Conta = 1212,
        Digito = 1,
        Instituicao = "CEF"
    };

    Console.Write("Informe o valor a ser debitado: ");
    string? valor = Console.ReadLine();
    bool valido = double.TryParse(valor, out double valorValido);
    if (valido)
    {
        conta.Debit(valorValido);
    }

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

    Console.Read();
}
```

```
static void Main(string[] args)
{
    ContaBancaria conta = new()
    {
        Agencia = "18",
        Conta = 1212,
        Digito = 1,
        Instituicao = "CEF"
    };

    Console.Write("Informe o valor a ser debitado: ");
    string? valor = Console.ReadLine();
    bool valido = int.TryParse(valor, out int valorValido);
    if (valido)
    {
        conta.Debit(valorValido);
    }

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

    Console.Read();
}
```

Aula 08: Uso de sobrecargas

- Mesmo método, quantidade diferente de parâmetros

```
static void Main(string[] args)
{
    ContaBancaria conta = new()
    {
        Agencia = "18",
        Conta = 1212,
        Digito = 1,
        Instituicao = "CEF"
    };

    Console.Write("Informe o valor a ser debitado: ");
    string? valor = Console.ReadLine();
    bool valido = double.TryParse(valor, out double valorValido);
    if (valido)
    {
        conta.Debit(valorValido);
    }

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

    Console.Read();
}
```

```
static void Main(string[] args)
{
    ContaBancaria conta = new()
    {
        Agencia = "18",
        Conta = 1212,
        Digito = 1,
        Instituicao = "CEF"
    };

    Console.Write("Informe o valor a ser debitado: ");
    string? valor = Console.ReadLine();
    bool valido = double.TryParse(valor, out double valorValido);
    if (valido)
    {
        conta.Debit(valorValido, DateTime.Now);
    }

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

    Console.Read();
}
```

Aula 08: Construtores

- Construtores são métodos executados automaticamente pelo runtime, na instanciação da classe.
- Embora seja um método, seu nome deve ser o mesmo nome da classe.
- Como método, pode ser sobrecarregado.
  - Um construtor sem parâmetros, é o construtor default da classe
  - Construtores com parâmetros, são construtores alternativos
- Se você não criar construtores para a sua classe, o compilador cria um construtor default automaticamente

Aula 08: Construtores

- Classe instanciada com o construtor alternativo.

```
public class ContaBancaria
{
    1 reference
    public ContaBancaria()
    {

    }

    0 references
    public ContaBancaria(string instituicao, double saldoInicial)
    {
        this.Instituicao = instituicao;
        this.Saldo = saldoInicial;
    }

     ...
}
```

```
static void Main(string[] args)
{
    ContaBancaria conta = new("CEF", 500)
    {
        Agencia = "18",
        Conta = 1212,
        Digito = 1
    };

    Console.Write("Informe o valor a ser debitado: ");
    string? valor = Console.ReadLine();
    bool valido = double.TryParse(valor, out double valorValido);
    if (valido)
    {
        conta.Debit(valorValido, DateTime.Now);
    }

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

    Console.Read();
}
```