





Alex Sander Resende de Deus

A 25 anos ensinando programação a jovens e adultos.

Apaixonado por tecnologia é atualmente coordenador de cursos na ETEC Albert Einstein. Na FIAP atua como professor da FIAP School, lecionando C#, SQLServer e Desenvolvimento Mobile

AULA 9

Sobrecarga e Sobrescrita

Sobrecarga (Overload)

- A sobrecarga de método é a possibilidade de implementar dois ou mais métodos com o mesmo nome e diferentes passagens de parâmetros.
- As diferenças nos parâmetros podem ser na quantidade e/ou no tipo.
- A sobrecarga pode ser implementada tanto em métodos da mesma classe quanto em métodos herdados de uma superclasse

Sobrecarga (Overload)

- Um exemplo bem comum (e já utilizado) é a implementação de dois construtores, um com e outro sem passagem de parâmetros.

```
public Produto() {  
    this("", "", 0, 0);  
}
```

```
public Produto(String descricao, String genero, int estoqueDisponivel, double precoCusto) {  
    this.descricao = descricao;  
    this.genero = genero;  
    this.estoqueDisponivel = estoqueDisponivel;  
    this.precoCusto = precoCusto;  
}
```

Exemplo prático

- Imagine uma empresa onde todos os funcionários ganham por hora porém, alguns ganham em reais, outros em dólar. No momento do pagamento todos devem ganhar em reais, ou seja, os que ganham em dólar tem o valor convertido em reais pela cotação do dólar no dia.

Valor da Hora:

Qtde de Horas:

Cotação do Dólar

Calcular

```
class CalculadoraSalario
{
    public double valorHora { get; set; }
    public double qtdeHora { get; set; }

    public CalculadoraSalario()
    {
        this.valorHora = 0;
        this.qtdeHora = 0;
    }

    public CalculadoraSalario(double valorHora)
    {
        this.valorHora = valorHora;
        this.qtdeHora = qtdeHora;
    }

    public double calcularSalario()
    {
        return this.valorHora * this.qtdeHora;
    }

    public double calcularSalario(double cotacaoDolar)
    {
        return (this.valorHora * this.qtdeHora) * cotacaoDolar;
    }
}
```

Classe
CalculadoraSalario

Código do Botão

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    CalculadoraSalario cSal = new CalculadoraSalario();
    double vHora, qHora, cDolar, salario;

    if(double.TryParse(txtVHora.Text, out vHora) == false)
    {
        MessageBox.Show("Digite corretamente o valor da hora");
        txtVHora.Clear();
        txtVHora.Focus();
        return;
    }

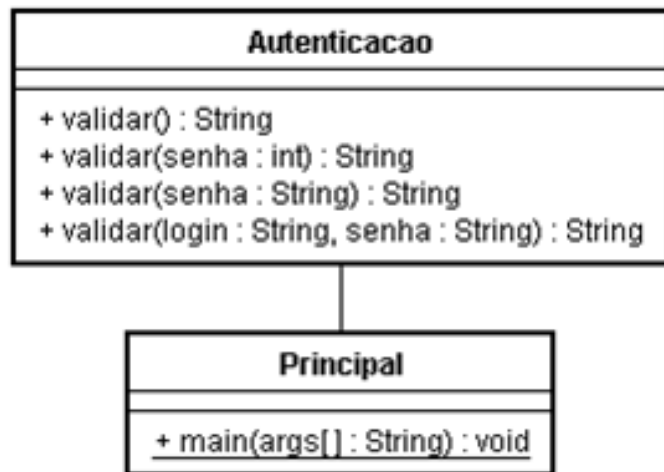
    if (double.TryParse(txtQtdeHoras.Text, out qHora) == false)
    {
        MessageBox.Show("Digite corretamente a quantidade de horas");
        txtQtdeHoras.Clear();
        txtQtdeHoras.Focus();
        return;
    }
}
```

```
if (txtDolar.Text.Length > 0)
{
    if (double.TryParse(txtDolar.Text, out cDolar) == false)
    {
        MessageBox.Show("Digite corretamente a cotação do dólar");
        txtQtdeHoras.Clear();
        txtQtdeHoras.Focus();
        return;
    }
}
else
{
    cDolar = 0;
}
cSal.qtdeHora = qHora;
cSal.valorHora = vHora;
if (cDolar == 0)
{
    salario = cSal.calcularSalario();
}
else
{
    salario = cSal.calcularSalario(cDolar);
}
MessageBox.Show("Salário: " + salario.ToString());
}
```




Momento Hands On





Classe: Autenticacao

Métodos	<ul style="list-style-type: none"> • validar (Sem parâmetros): Retorna a mensagem "<u>Bem vindo</u> visitante!"
	<ul style="list-style-type: none"> • validar (Com um parâmetro <u>int</u>): Verifica se o parâmetro informado é igual a 123, se for, retorna a mensagem "<u>Bem vindo</u> usuário teste!" senão "Usuário temporário inválido!"
	<ul style="list-style-type: none"> • validar (Com um parâmetro String): Verifica se o parâmetro é igual a "abc", se for, retorna a mensagem a String "<u>Bem vindo</u> usuário!" senão "Usuário inválido!"
	<ul style="list-style-type: none"> • validar (Com dois <u>parametros</u>): Verifica se os parâmetros informados são respectivamente "<u>adm</u>" e "master", se for, retorna a mensagem "<u>Bem vindo</u> administrador!", senão "Administrador inválido!"

Formulário

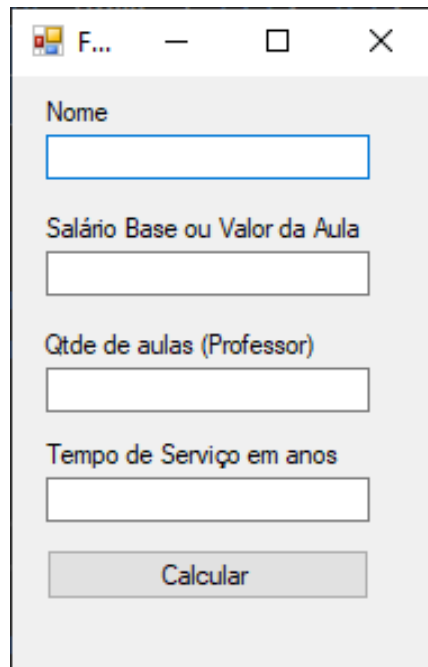
	<ul style="list-style-type: none"> • Instanciar um objeto do tipo Autenticação chamado login.
	<ul style="list-style-type: none"> • Apresentar um menu com as opções: <u>Logar como:</u> 1 – Visitante 2 – Usuário temporário 3 – Usuário 4 – Administrador 0 - Sair
	Obs.: Realizar as leituras dos dados conforme descrito no enunciado e de acordo com as estruturas dos métodos chamados.

Sobrescrita (Overwrite ou Overriding)

- A sobrescrita ou reescrita de método esta diretamente relacionada com herança e é a possibilidade de manter a mesma assinatura de um método herdado e reescrevê-lo na subclasse.
- Na chamada de um método sobrescrito o programa considera primeiro a classe a partir da qual o objeto foi instanciado, se a superclasse possuir um método com a mesma assinatura este será descartado.

Exemplo prático

- Em uma escola todos os funcionários tem um salário base e um bônus por tempo de serviço que é de 1% para cada ano trabalhado. Professores também tem este bônus, só que seu salário é calculado de forma diferente: multiplica-se o valor da aula pela quantidade de aulas semanais e o resultado por 4,5, para só depois calcular o bônus. Elabore um programa capaz de calcular o salário de qualquer funcionário.



Nome

Salário Base ou Valor da Aula

Qtde de aulas (Professor)

Tempo de Serviço em anos

Calcular

Classe Funcionario

```
class Funcionario
{
    public string nome { get; set; }
    public double tempoServico { get; set; }
    public double salarioBase { get; set; }

    public Funcionario()
    {
        this.nome = "";
        this.tempoServico = 0;
        this.salarioBase = 0;
    }

    public Funcionario(string nome, double tempoServico, double salarioBase)
    {
        this.nome = nome;
        this.tempoServico = tempoServico;
        this.salarioBase = salarioBase;
    }

    public virtual double calcularSalario()
    {
        return this.salarioBase + (this.salarioBase * (this.tempoServico / 100));
    }
}
```

Classe Professor

```
class Professor:Funcionario
{
    public int qtdeAulas { get; set; }

    public Professor()
    {
        this.qtdeAulas = 0;
    }

    public Professor(int qtdeAulas)
    {
        this.qtdeAulas = qtdeAulas;
    }

    public override double calcularSalario()
    {
        double totalAulas = qtdeAulas * salarioBase * 4.5;
        double salarioFinal = totalAulas + (totalAulas * (this.tempoServico / 100));
        return salarioFinal;
    }
}
```


Botão

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    double salarioFinal;
    int qtdeAulas;
    Professor p = new Professor();
    Funcionario f = new Funcionario();

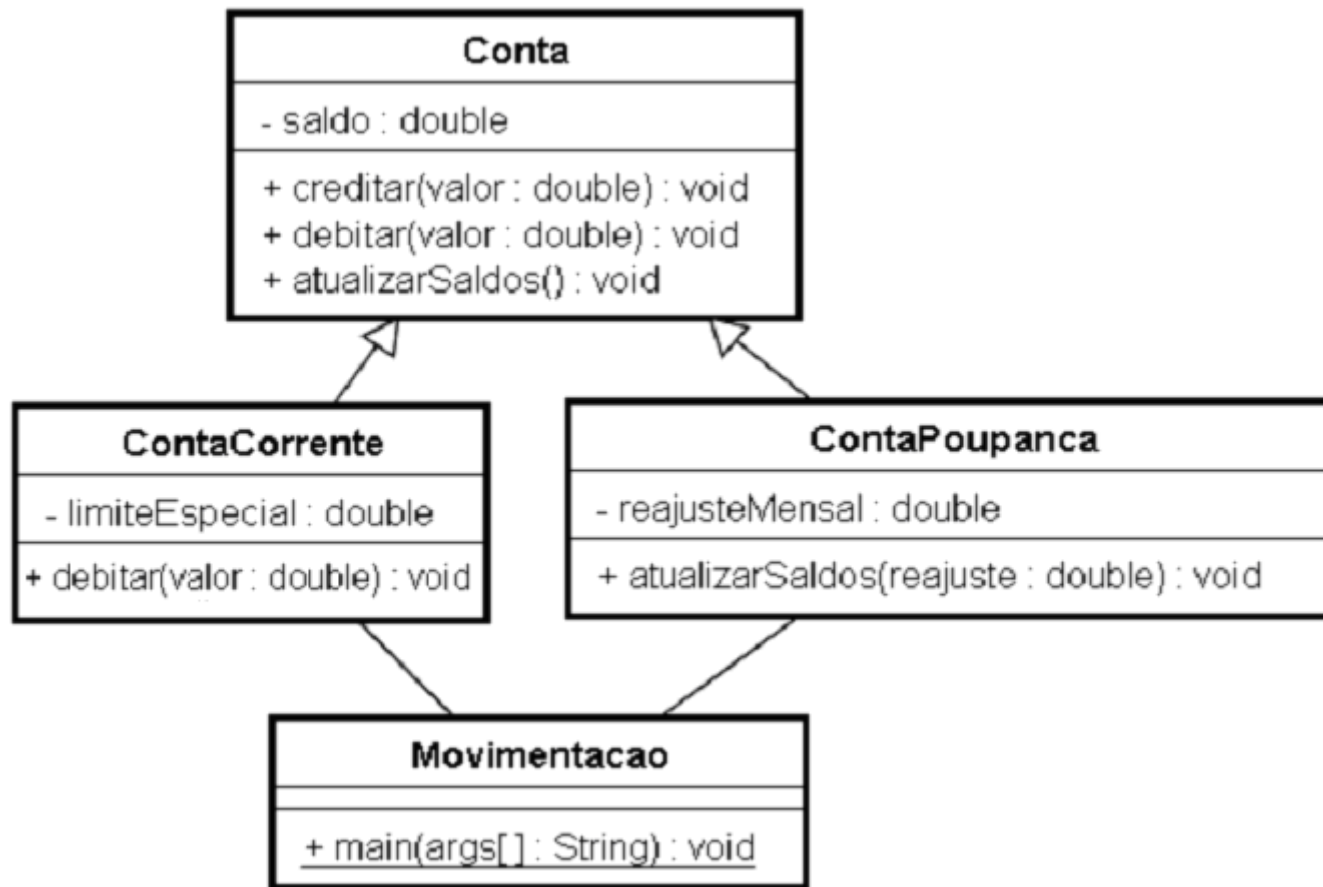
    qtdeAulas = Convert.ToInt32(txtQAulas.Text);

    if (qtdeAulas == 0)
    {
        f.nome = txtNome.Text;
        f.salarioBase = Convert.ToDouble(txtSBase.Text);
        f.tempoServico = Convert.ToDouble(txtTempo.Text);
        salarioFinal = f.calcularSalario();
    }
    else
    {
        p.nome = txtNome.Text;
        p.salarioBase = Convert.ToDouble(txtSBase.Text);
        p.tempoServico = Convert.ToDouble(txtTempo.Text);
        p.qtdeAulas = qtdeAulas;
        salarioFinal = p.calcularSalario();
    }
    MessageBox.Show("Salário Final: " + salarioFinal.ToString());
}
```



Momento Hands On





Classe: Conta	
Métodos	creditar: Recebe um valor por parâmetro e soma ao atributo saldo
	debitar: Recebe um valor por parâmetro e subtrai do atributo saldo desde que haja saldo suficiente, caso não tenha, apresenta mensagem de saldo insuficiente.
	atualizarSaldo: Verifica se o atributo saldo <u>esta</u> negativo, caso esteja, calcula 8% (0.08) sobre o valor excedente e subtrai do saldo (Cobra juros pela utilização de limite especial). Apresentar o saldo anterior e o saldo atualizado.

Classe: ContaCorrente (subclasse de Conta)	
Método	debitar: Sobrescrever o método debitar considerando o atributo <u>limiteEspecial</u> . O saldo poderá ficar negativo até o valor indicado em <u>limiteEspecial</u> .

Classe: ContaPoupança (subclasse de Conta)	
Método	atualizarSaldo: Sobrecarregar o método <u>atualizarSaldo</u> de modo que ele receba por parâmetro uma porcentagem para reajuste (um valor decimal <u>double</u>). Calcular a porcentagem informada sobre o saldo e somar ao saldo (Rendimento da poupança). Armazenar a porcentagem informada no atributo <u>reajusteMensal</u> . Apresentar o saldo anterior e o saldo atualizado.

Formulário	
	<ul style="list-style-type: none">• Instanciar um objeto do tipo <u>ContaCorrente</u> chamado cc1 com saldo inicial de 500 e limite especial de 1000.
	<ul style="list-style-type: none">• Instanciar um objeto do tipo <u>ContaPoupança</u> chamado cp1 com saldo inicial de 5000 e <u>reajusteMensal</u> de 1% (0.01)
	<ul style="list-style-type: none">• Apresentar um menu com as opções:<ul style="list-style-type: none">1 – Conta corrente2 – <u>Poupança</u>0 – Sair
	<ul style="list-style-type: none">• Apresentar outro menu com as opções:<ul style="list-style-type: none">1 – Depositar2 – Sacar3 – Consultar saldo4 – Atualizar saldo0 – Sair
	<ul style="list-style-type: none">• Realizar as chamadas aos métodos de acordo com as opções do usuário.
	Obs.: No reajuste da poupança informar a porcentagem a ser aplicada (em formato decimal)

OBRIGADO



profalex.deus@fiap.com.br



[linkedin.com/in/alexanderresende](https://www.linkedin.com/in/alexanderresende)

FIAP MBA⁺

Copyright © 2019 | Professor (a) Nome do Professor

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento, é expressamente proibido sem consentimento formal, por escrito, do professor/autor.

FIAP