





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 6

Programação Orientada a Objetos

Orientação a Objetos

É uma técnica de desenvolvimento de softwares que consiste em representar os elementos do mundo real (que pertencem ao escopo da aplicação) dentro do software.

Em tese, é uma forma mais natural de informatização, já que leva em consideração os elementos como eles realmente existem.

Conceitos da orientação a objetos

Entenda “conceitos” como uma série de regras e convenções que padronizam as aplicações orientadas a objetos e que possibilitam o uso de todos os recursos inerentes a essa técnica.

Abstração

Abstração é a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais.

Habilidade mental que permite aos seres humanos visualizar os problemas do mundo real com vários graus de detalhe, dependendo do contexto corrente do problema.



Elementos básicos da programação orientada a objetos



Classe de modelagem

Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de software.

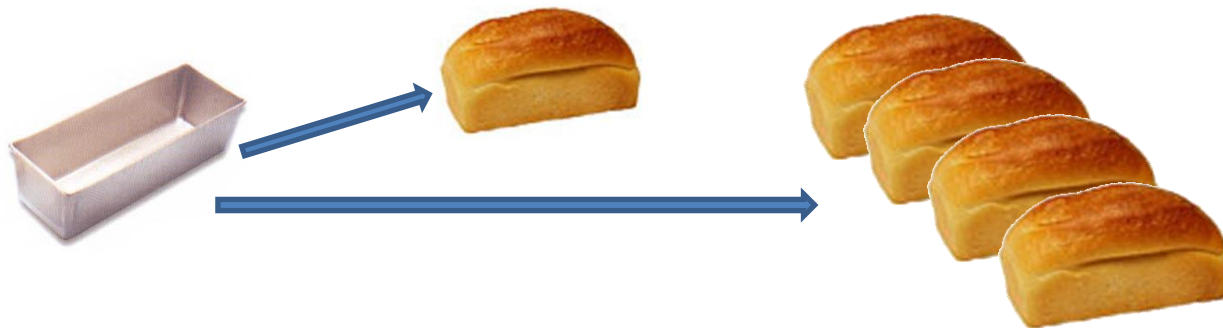
A classe de modelagem pode ser entendida como um molde, uma forma, um modelo que define as características e as funcionalidades dos futuros objetos que serão criados a partir dela.

Objeto

- Em modelagem orientada a objetos, um objeto é uma instância de uma classe existente no sistema de software.
- Um objeto representa uma entidade do mundo real dentro da aplicação de forma individual, possuindo todas as informações e funcionalidades abstraídas na concepção da classe.

- — □ •
- • • + • □
- • +
- - Um objeto é criado (instanciado) a partir de uma classe e, portanto, possui todos os elementos definidos na classe.
 - É possível criar quantos objetos forem necessários a partir de uma classe e todos terão as mesmas características e funcionalidades.

Analogias de Classe X Objeto



Estrutura básica e uma classe de modelagem

- Atributos ou variáveis de instância

São as informações, os dados, que serão armazenados nos objetos.

- Métodos

São as ações, as regras, as funcionalidades que serão executadas pelos objetos.

UML



UML

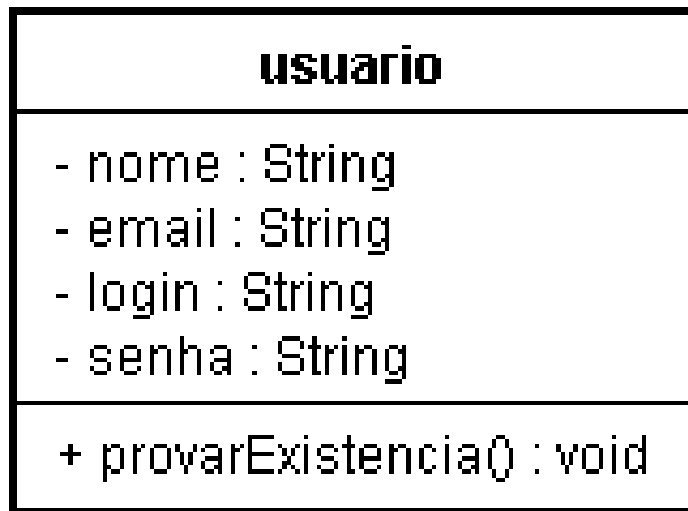
- A partir do momento em que os elementos básicos da orientação a objetos são assimilados, podemos modelar classes nas especificações corretas utilizando a principal ferramenta de modelagem e documentação de aplicações orientadas a objeto existente no mercado:
- A UML (Unified Modeling Language ou Linguagem unificada de modelagem).

- A UML não é uma metodologia de desenvolvimento, o que significa que ela não diz para você o que fazer primeiro e em seguida ou como projetar seu sistema, mas ela lhe auxilia a visualizar seu desenho e a comunicação entre objetos.
- Permite que desenvolvedores visualizem os produtos de seu trabalho em diagramas padronizados.
- Os objetivos da UML são: especificação, documentação, e estruturação para sub-visualização e maior visualização lógica de um total desenvolvimento de um sistema de informação.

Classe de modelagem: Usuario

- Abstração e modelagem da classe de usuário de uma aplicação.
- Informações armazenadas em cada usuário (objeto).
 - nome
 - email
 - login
 - senha
- Ações exercidas pelos (objetos do tipo) usuários na aplicação.
 - Provar existência.

Classe de modelagem: Usuario



Modificadores de acesso

- Private

Só fica visível dentro da classe em que foi implementado

- Public

Fica visível em toda a aplicação

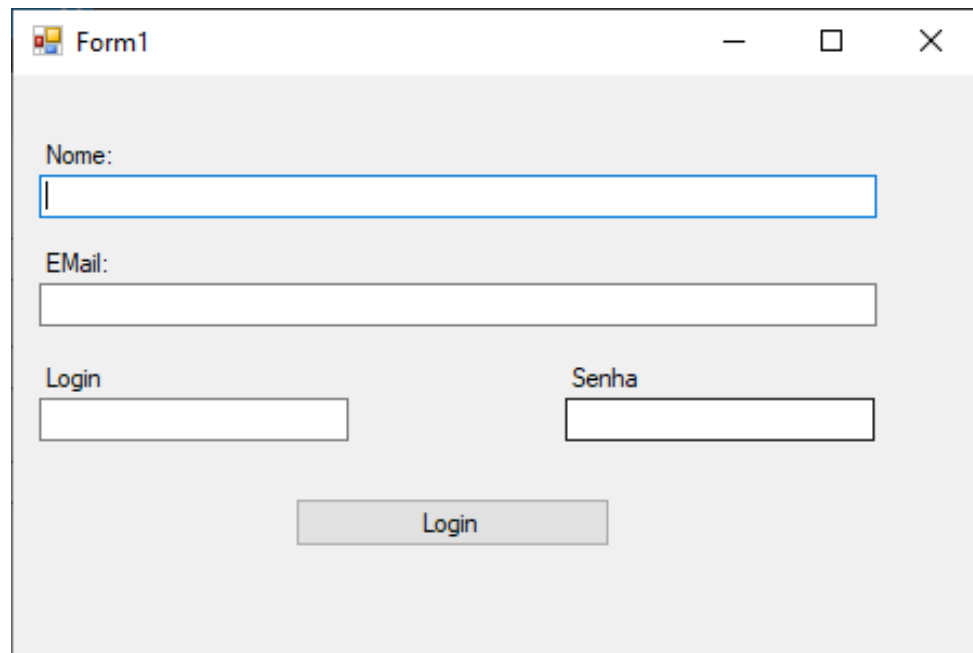
- Protected

Fica visível na classe em que foi implementado e em suas sub-classe (veremos em breve).



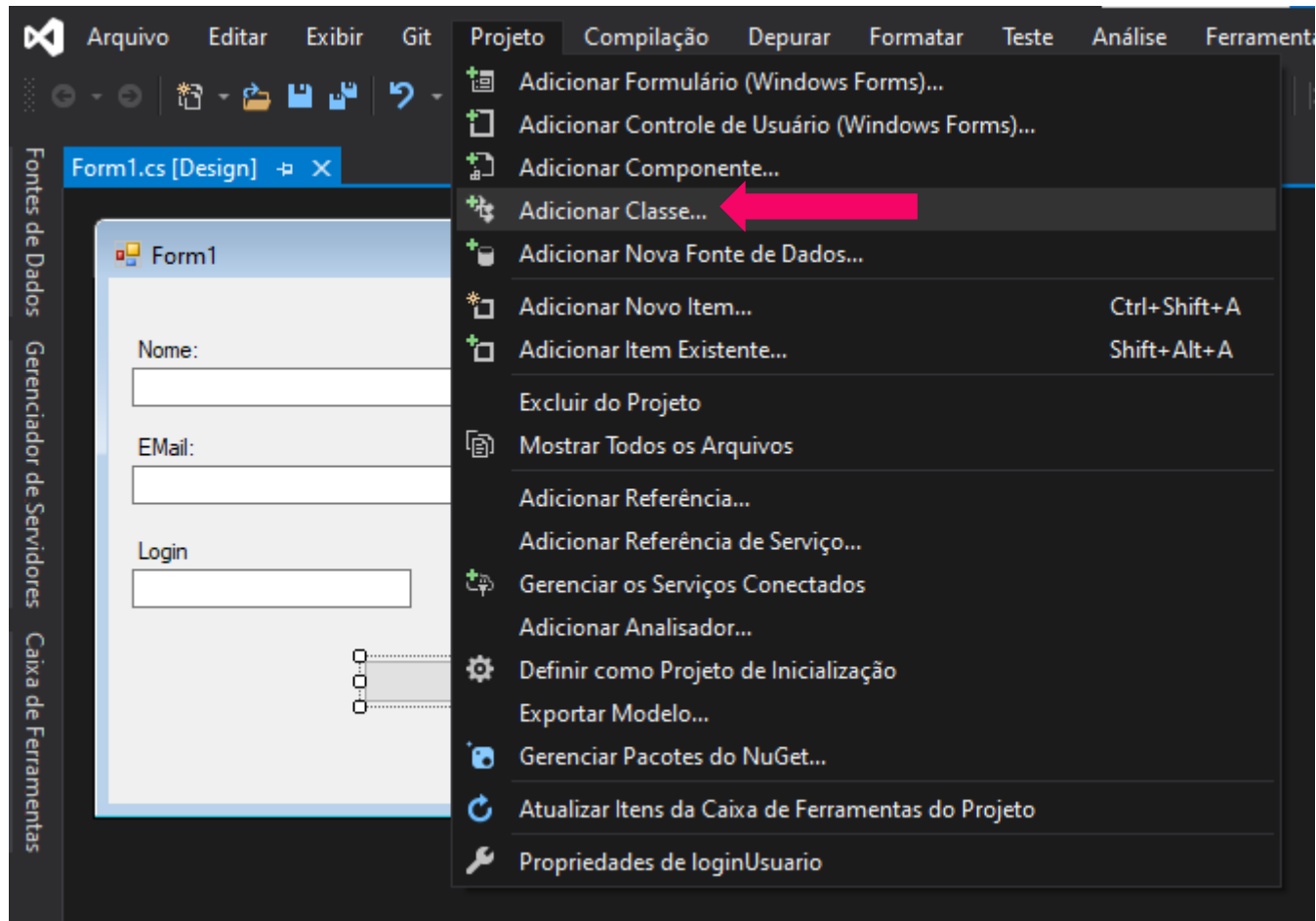
Criação do Primeiro Projeto

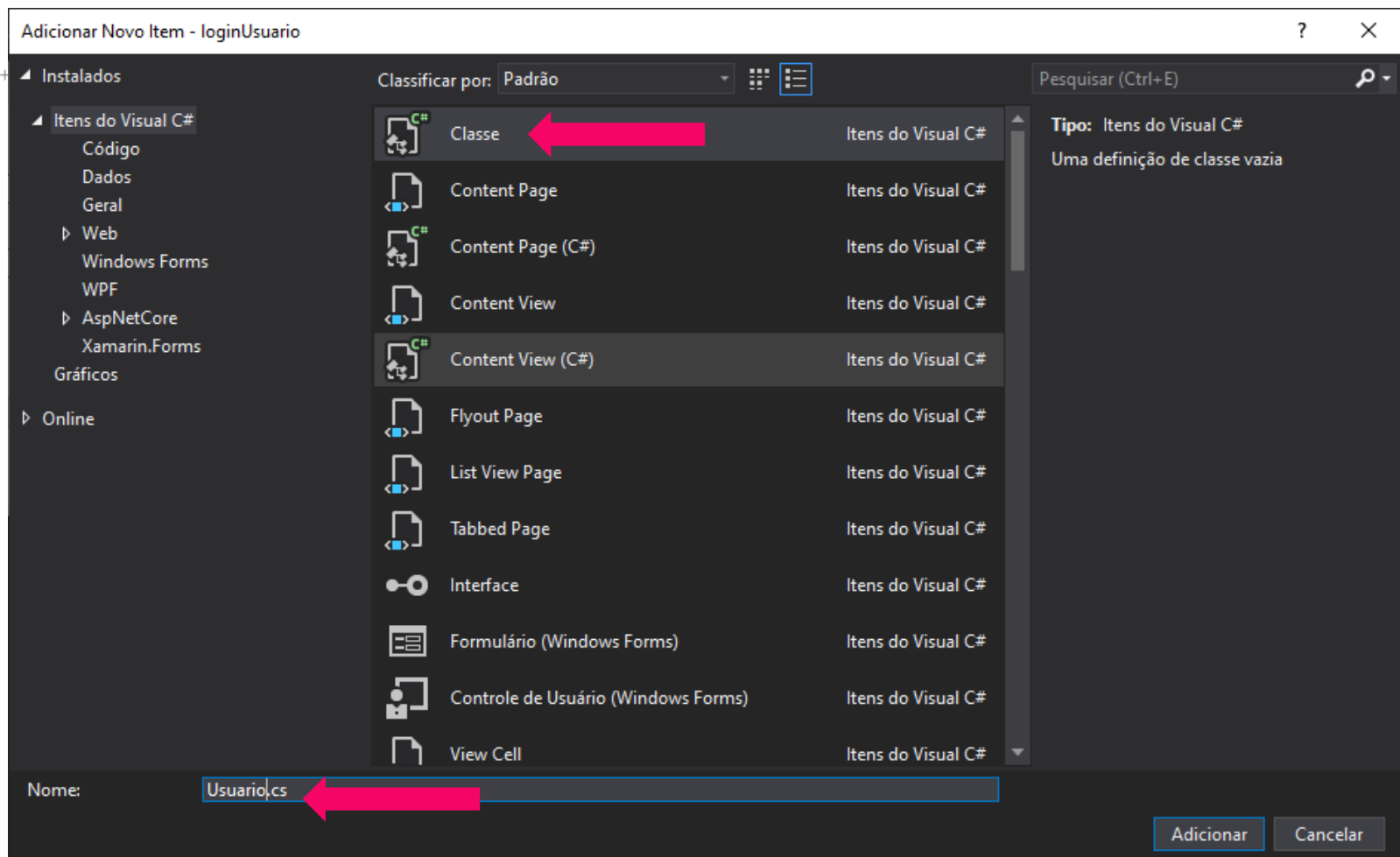




A screenshot of a Windows application window titled "Form1". The window contains a login form with the following elements:

- A label "Nome:" followed by a text input field.
- A label "EMail:" followed by a text input field.
- A label "Login" followed by a text input field.
- A label "Senha" followed by a text input field.
- A "Login" button centered below the input fields.





```
class Usuario
{
    //Declarando os atributos
    public string nome {get;set;}
    public string email { get; set; }
    public string login { get; set; }
    public string senha { get; set; }

    public void provarExistencia()
    {
        MessageBox.Show("Olá " + this.nome + "!!!!");
    }
}
```

```
private void btnLogin_Click(object sender, EventArgs e)
{
    //Declarando um objeto da classe
    Usuario user = new Usuario();

    //Enviando dados aos atributos
    user.nome = txtNome.Text;
    user.email = txtEmail.Text;
    user.login = txtLogin.Text;
    user.senha = txtSenha.Text;

    //Chamando o método
    user.provarExistencia();
}
```


Como instanciar um objeto

Nome da Classe

Nome da Classe

```
Usuario user = new Usuario();
```

Nome do Objeto

Como realizar uma chamada a método

Nome do Objeto

```
user.provarExistencia();
```

Nome do Método

Recapitulando nosso “HelloWorld” orientado a objeto

- Em uma aplicação orientada a objetos quem manipula dados e ações são os objetos.
- Através de análise e abstração definimos as entidades a serem representadas na aplicação e criamos uma classe (um tipo) com todos os dados e ações especificados (classe de modelagem).
- Definido o tipo, criamos (instanciamos) objetos a partir dos tipos pré-existentes (por enquanto, sempre no método main).



Momento Hands On





Elaborar um programa que efetue as quatro operações básicas da matemática. Os cálculos devem ficar em métodos de uma classe de modelagem



Form1

Primeiro número

Segundo Número

Somar

Subtrair

Multiplicar

Dividir

Resposta

```
class Calculadora
{
    public double n1 { get; set; }
    public double n2 { get; set; }
    public double res { get; set; }

    public void somar()
    {
        res = n1 + n2;
    }

    public void subtrair()
    {
        res = n1 - n2;
    }

    public void multiplicar()
    {
        res = n1 * n2;
    }

    public void dividir()
    {
        res = n1 / n2;
    }
}
```

```
private void btnSomar_Click(object sender, EventArgs e)
{
    Calculadora calc = new Calculadora();
    calc.n1 = Convert.ToDouble(txtN1.Text);
    calc.n2 = Convert.ToDouble(txtN2.Text);
    calc.somar();
    lblResp.Text = calc.res.ToString();
}

private void btnSubtrair_Click(object sender, EventArgs e)
{
    Calculadora calc = new Calculadora();
    calc.n1 = Convert.ToDouble(txtN1.Text);
    calc.n2 = Convert.ToDouble(txtN2.Text);
    calc.subtrair();
    lblResp.Text = calc.res.ToString();
}
```

```
private void btnMultiplicar_Click(object sender, EventArgs e)
{
    Calculadora calc = new Calculadora();
    calc.n1 = Convert.ToDouble(txtN1.Text);
    calc.n2 = Convert.ToDouble(txtN2.Text);
    calc.multiplicar();
    lblResp.Text = calc.res.ToString();
}

private void btnDividir_Click(object sender, EventArgs e)
{
    Calculadora calc = new Calculadora();
    calc.n1 = Convert.ToDouble(txtN1.Text);
    calc.n2 = Convert.ToDouble(txtN2.Text);
    calc.dividir();
    lblResp.Text = calc.res.ToString();
}
```

CONSTRUTORES





São **métodos** especiais que servem para construir e inicializar instâncias de uma classe (novos objetos). Em geral os construtores são responsáveis por **atribuir valores iniciais para os atributos** do novo objeto que está sendo criado, mas eventualmente pode ser necessário algum processamento adicional que vai além da inicialização de atributos.



```
class Usuario
{
    //Método Construtor
    public Usuario()
    {
    }

    //Declarando os atributos
    public string nome {get;set;}
    public string email { get; set; }
    public string login { get; set; }
    public string senha { get; set; }

    public void provarExistencia()
    {
        MessageBox.Show("Olá " + this.nome + "!!!!");
    }
}
```

O método construtor é um método público que leva o mesmo nome da classe.

Neste exemplo, nenhuma ação foi definida no método.

```
//Método Construtor  
public Usuario()  
{  
    this.nome = "";  
    this.email = "";  
    this.login = "";  
    this.senha = "";  
}
```

Já neste exemplo o método construtor inicializa os atributos como vazios ("")

```
public Usuario(string nome, string email, string login, string senha)
{
    this.nome = nome;
    this.email = email;
    this.login = login;
    this.senha = senha;
}
```

E os parâmetros vindos
de fora são inseridos nos
atributos

Aqui, o método construtor
recebe parâmetros
vindos de fora da classe

```
private void btnLogin_Click(object sender, EventArgs e)
{
    //Instanciando um objeto e passando valores para os atributos pelo construtor
    Usuario user = new Usuario(txtNome.Text,txtEMail.Text,txtLogin.Text,txtSenha.Text);

    //Chamando o método
    user.provarExistencia();
}
```

Veja o código do Button.
O Objeto é criado e os
valores passados para os
atributos através do
construtor

```
public Usuario()  
{  
    ...  
}  
  
public Usuario(string nome, string email, string login, string senha)  
{  
    this.nome = nome;  
    this.email = email;  
    this.login = login;  
    this.senha = senha;  
}
```

Uma classe pode ter
quantos métodos
construtores forem
necessários!!!!



Momento Hands On





Refazer a calculadora de forma que o construtor receba o N1 e o N2 por parâmetro.




```
public Calculadora(double n1, double n2)
{
    this.n1 = n1;
    this.n2 = n2;
}
```

Veja o código do Classe.
O Objeto é criado e os
valores passados para os
atributos através do
construtor

```
private void btnSomar_Click(object sender, EventArgs e)
{
    Calculadora calc = new Calculadora(Convert.ToDouble(txtN1.Text), Convert.ToDouble(txtN2.Text));
    calc.somar();
    lblResp.Text = calc.res.ToString();
}
```

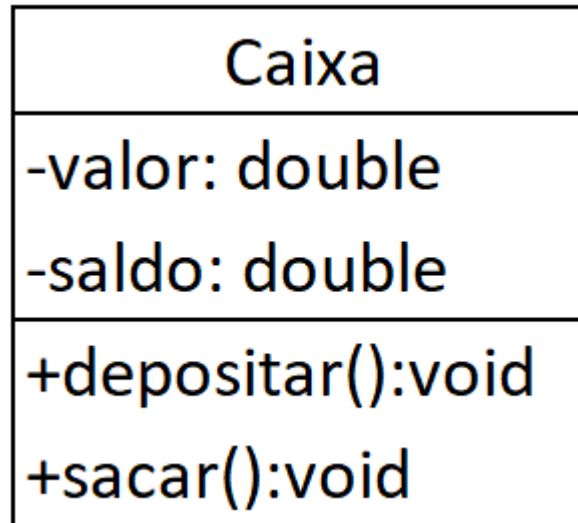
Em todos os
BUTTONS



Para treinar mais



Seguindo o diagrama de classe UML abaixo, crie um programa capaz de controlar o saldo de um caixa



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