



PROGRAMAÇÃO O.O. (C#)



Introdução ao Designer
Professor: João Luiz Lagôas



The .NET Framework Class Library

- Muitas classes predefinidas estão agrupadas em categorias de classes relacionadas chamadas de **namespaces**. Juntos, esses namespaces são referidos como .NET Framework Class Library (**FCL**).
- Através de diretivas precedidas pela palavra reservada **using**, especificamos qual namespace queremos que esteja presente em nosso projeto.

```
using System;
```

→ Diretiva para importar todas as classes que estão no namespace **System**

No site: [https://msdn.microsoft.com/pt-br/library/system\(v=vs.110\).aspx](https://msdn.microsoft.com/pt-br/library/system(v=vs.110).aspx)
há uma referência completa de todas as classes presentes no namespace System

namespace System



- Até agora já utilizamos algumas das classes presentes no namespace System, são elas:

	Console	Representa os fluxos de entrada, saída e erro padrão para aplicativos de console.
	Convert	Converte um tipo de dados base em outro tipo de dados base.
	Math	Fornece constantes e métodos estáticos para trigonométricas, logarítmicas e outras funções matemáticas comuns.

namespace
System

Outro namespace importante...







- Uma das grandes forças em C# é o grande número de classes nos namespaces da .NET Framework Class Library.
- Além do namespace System que já conhecemos, existe um outro muito importante que pode ser incluído pela diretiva:

```
using System.Windows.Forms;
```

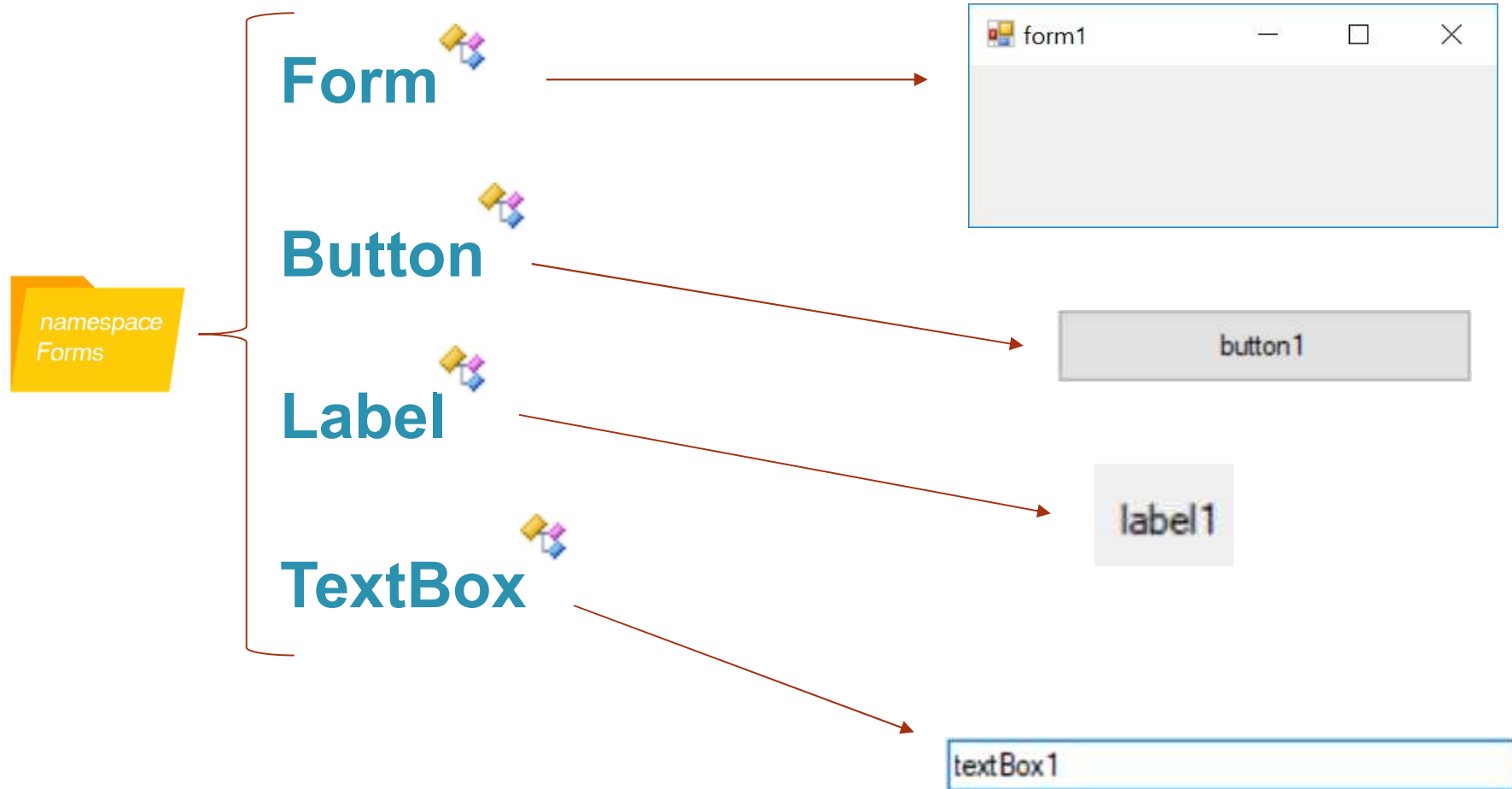
namespace System.Windows.Forms

- Algumas das classes que podem ser encontradas no namespace System.Windows.Forms são:

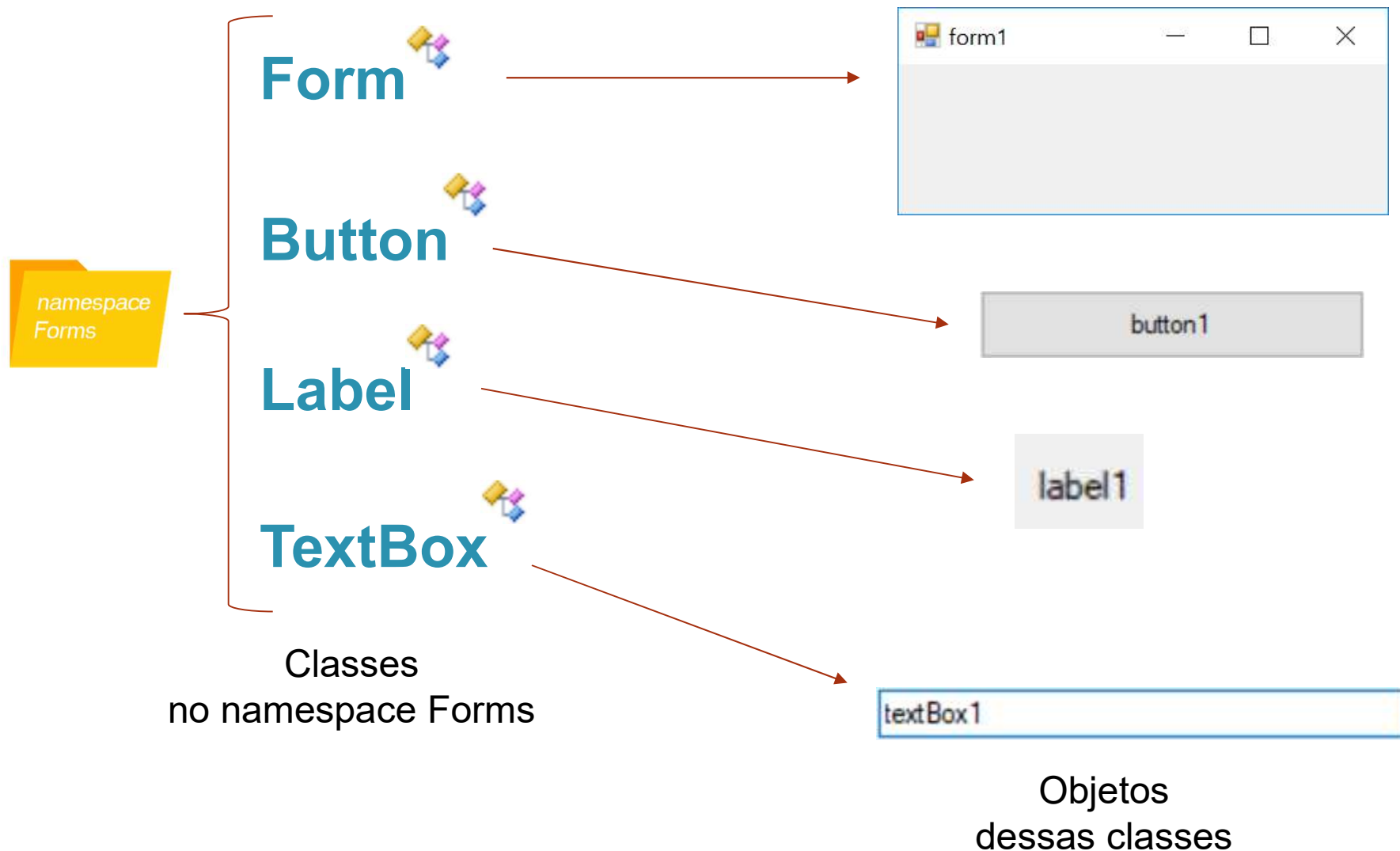
	Form	Representa uma janela ou caixa de diálogo que compõe a interface do usuário de um aplicativo.
	Button	Representa um controle de botão do Windows.
	Label	Representa um rótulo padrão do Windows.
	TextBox	Representa um controle de caixa de texto do Windows.

namespace
Forms

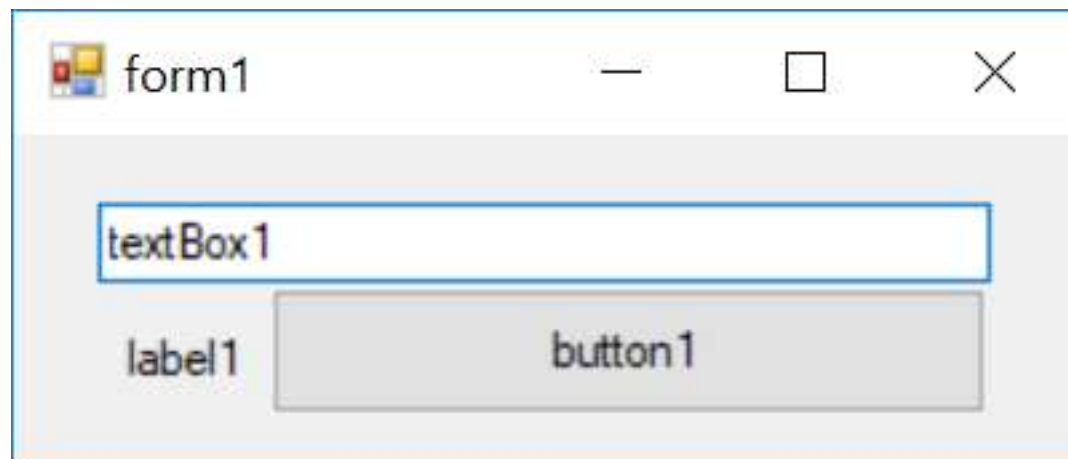
namespace System.Windows.Forms



namespace System.Windows.Forms



namespace System.Windows.Forms



namespace System.Windows.Forms

- O namespace Forms é repleto de classes prontas para se criar programas com uma boa interface gráfica.
- Vimos que para fazer uso de suas classes basta adicionar ao código a diretiva:

```
using System.Windows.Forms;
```

Mas como usar essas
classes para construir a GUI?



Designer e *GUI*



- A IDE Visual Studio oferece uma funcionalidade de **Designer** que facilita o processo de construção de aplicações com uma boa interface gráfica.
- Assim, o programador não precisa se preocupar com código fonte por trás das interfaces gráficas (instanciações de botões, labels, text boxes, etc) e pode focar apenas em suas funcionalidades.

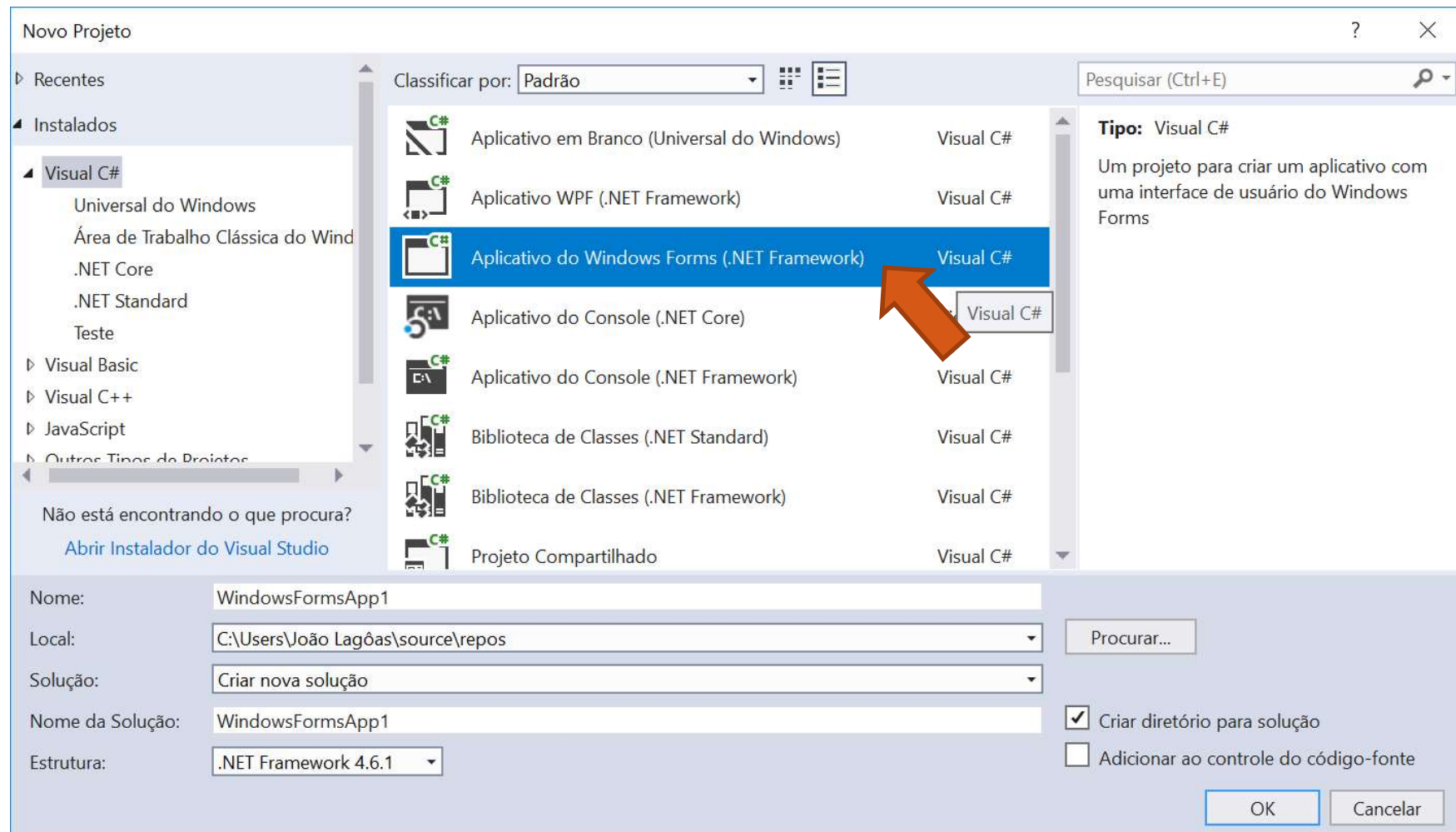


Designer e *GUI*



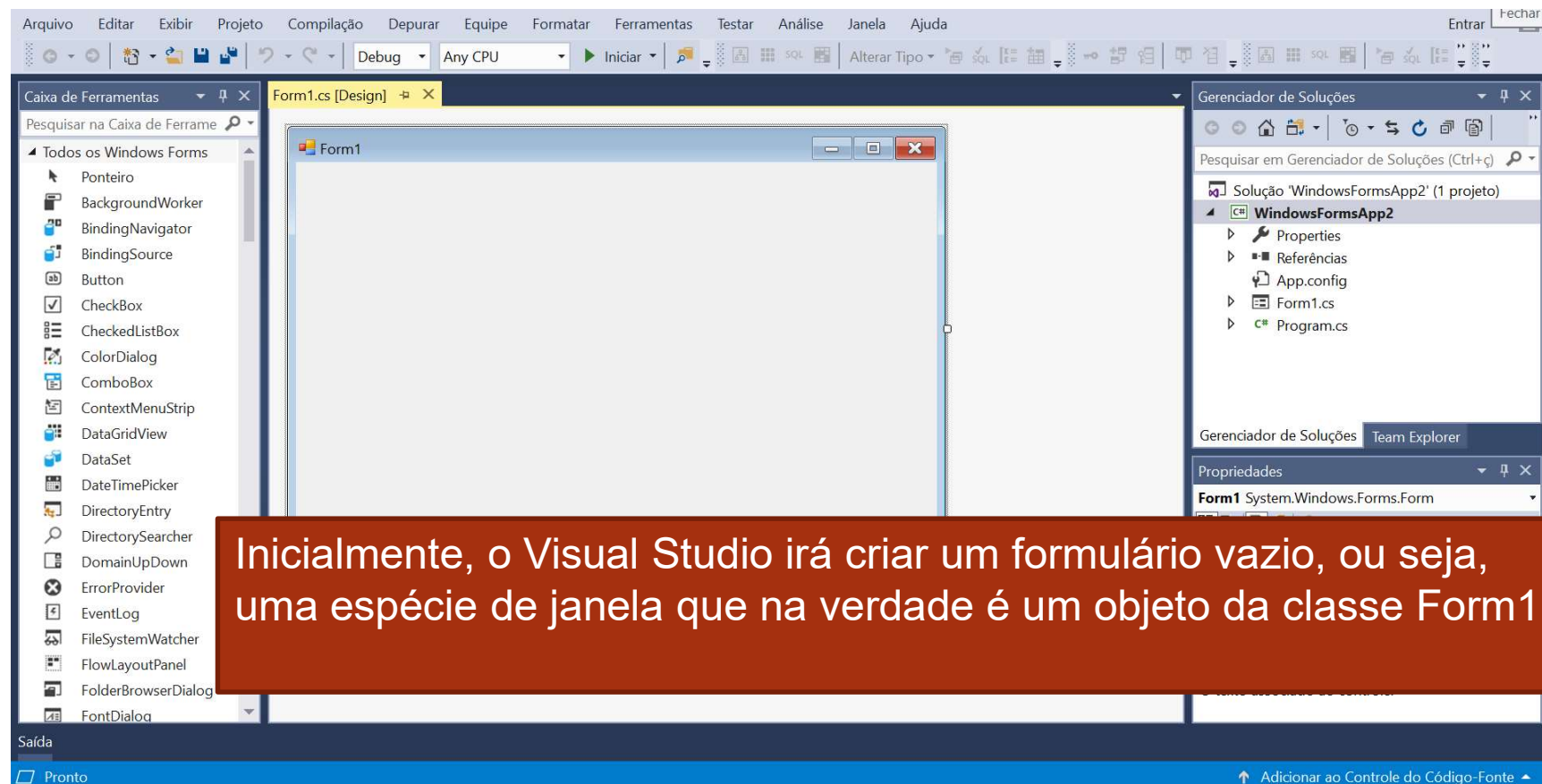
- Através do Designer, somos capazes de criar aplicações com GUI.
- GUI (*Graphical User Interfaces*) é um tipo de interface para usuários de dispositivos digitais que permite a interação com os mesmos através de elementos gráficos como ícones, botões, indicadores visuais em contraste com a interface de linha de comando.
- Para criar aplicações em C# que apresentam uma GUI, basta clicar em **Arquivo > Novo > Projeto** e então escolher **Aplicativo do Windows Forms (.NET Framework)**.

Criando uma aplicação com GUI



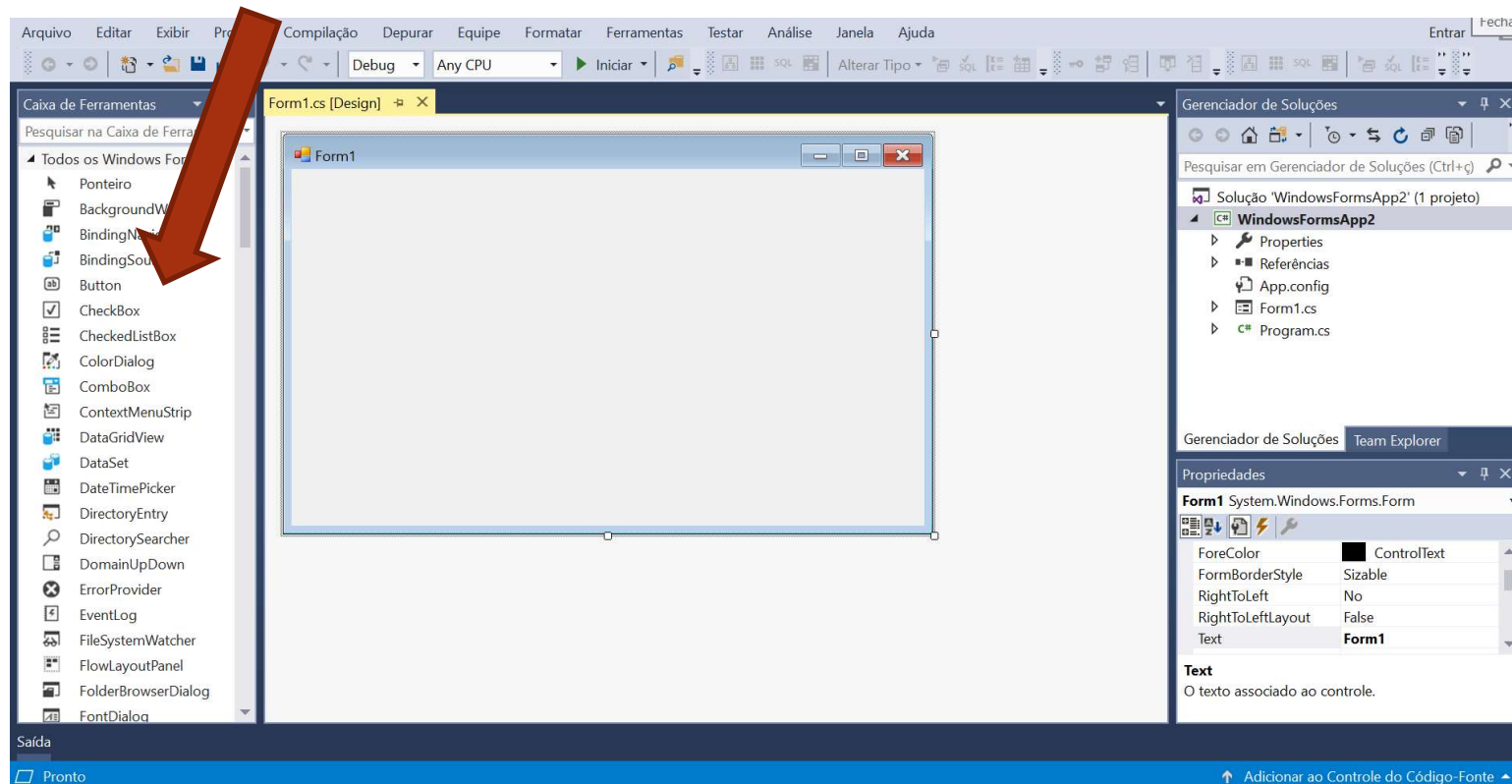
Criando uma aplicação com GUI

- Você então será direcionado para o ambiente de **Designer**.



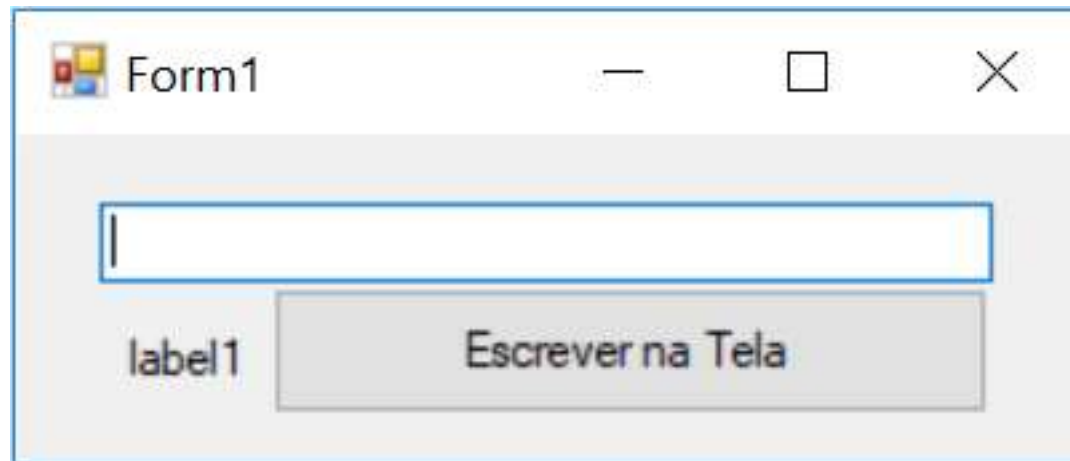
Criando uma aplicação com GUI

- No ambiente Designer, você pode “montar” o formulário da sua aplicação simplesmente arrastando elementos da **Caixa de Ferramentas** para o formulário.



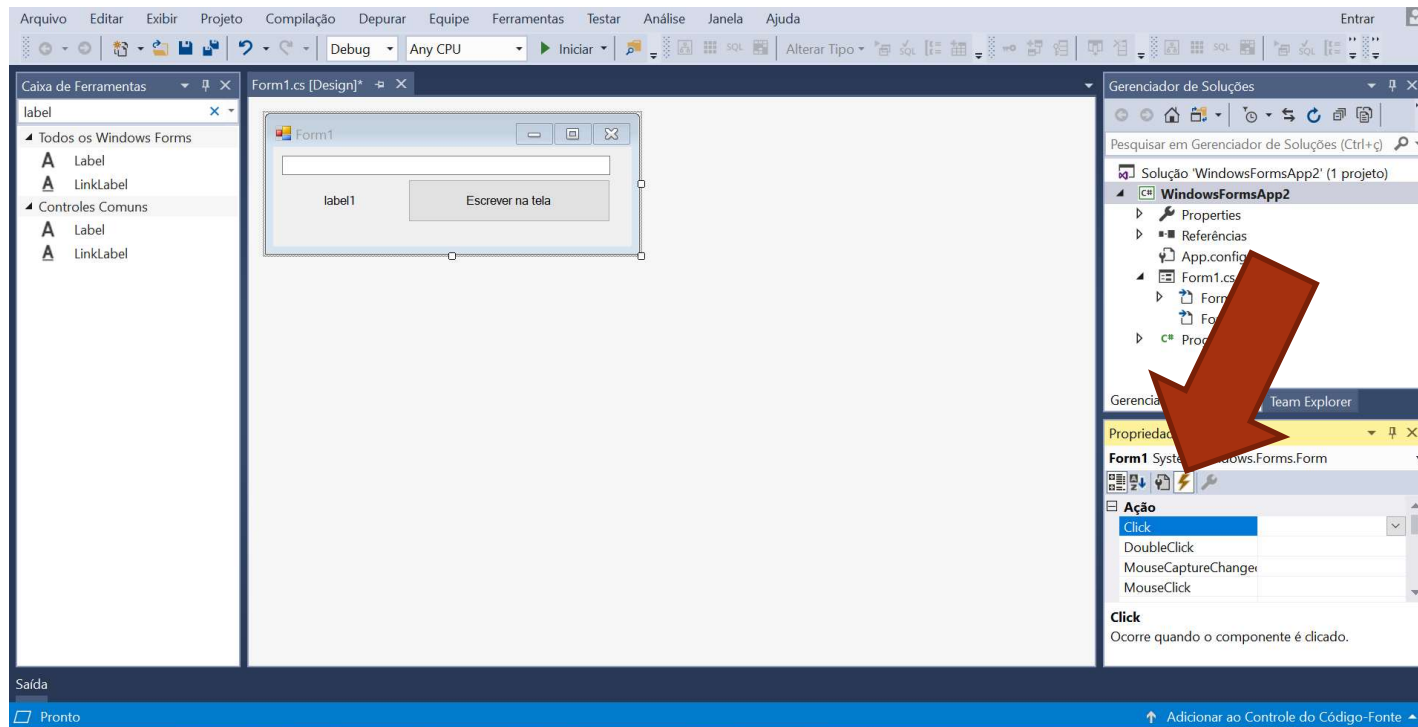
Criando uma aplicação com GUI

- Iremos arrastar três elementos da Caixa de Ferramentas: um Label (ou rótulo), um Button (ou botão) e um TextBox (ou caixa de texto) para criar o seguinte formulário abaixo:



Criando uma aplicação com GUI

- Agora iremos clicar no ícone indicado por um raio como demonstrado na figura abaixo.



- Após fazer isso, seremos redirecionados para a página de código da classe Form1. Perceba que **um novo método** foi criado automaticamente pelo programa!

Criando uma aplicação com GUI

- Esse método sempre será executado quando o botão for clicado.
- Sendo assim, o que está codificado no escopo do método button1_Click será executado a cada clique!

```
using System.Windows.Forms;

namespace Formulario
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            label1.Text = textBox1.Text;
        }
    }
}
```

Implementação do método!

Criando uma aplicação com GUI

O que o método faz?!

```
private void button1_Click(object sender,
EventArgs e)
{
    label1.Text = textBox1.Text;
}
```

Tanto a classe Label quanto a classe TextBox apresentam um atributo denominado Text.

O que o comando ao lado faz, então, é atribuir ao atributo Text do objeto label1 o valor armazenado no atributo Text do objeto textBox.

Lista de Classes que vimos hoje

- Algumas das classes e atributos que utilizamos hoje...

Classe	Atributos
Form1	Text
Button	Text
Label	Text
TextBox	Text

Entendendo por trás dos panos



- Em primeiro lugar, é importante entender que o formulário inicial que nos foi apresentado nada mais é do que um objeto da classe Form1, já presente na linguagem C#!
- Como qualquer classe, sabemos que ela contém atributos e métodos.
- Quando arrastamos elementos da Caixa de Ferramentas e os posicionamos no formulário, estamos na verdade adicionando atributos à classe Form1 que, por sua vez, são instâncias das classes Button, Label e TextBox.

Entendendo por trás dos panos



- O Designer do Visual Studio irá criar o código necessário para o funcionamento da sua aplicação gráfica. Mas no fundo, qualquer alteração que você faça no Designer será replicada em um código fonte que você não precisa se preocupar (o Designer irá organizá-lo).
- De qualquer forma, é importante entender que no fundo, estamos adicionando atributos à classe Form1 e que o método `InitializeComponent()` cria as instâncias de todos esses atributos e os posicionam corretamente no formulário de acordo como você preparou no Designer.

```
partial class Form1
{
    public TextBox textBox1;
    public Label label1;
    public Button button1;

    private System.ComponentModel.IContainer components = null;

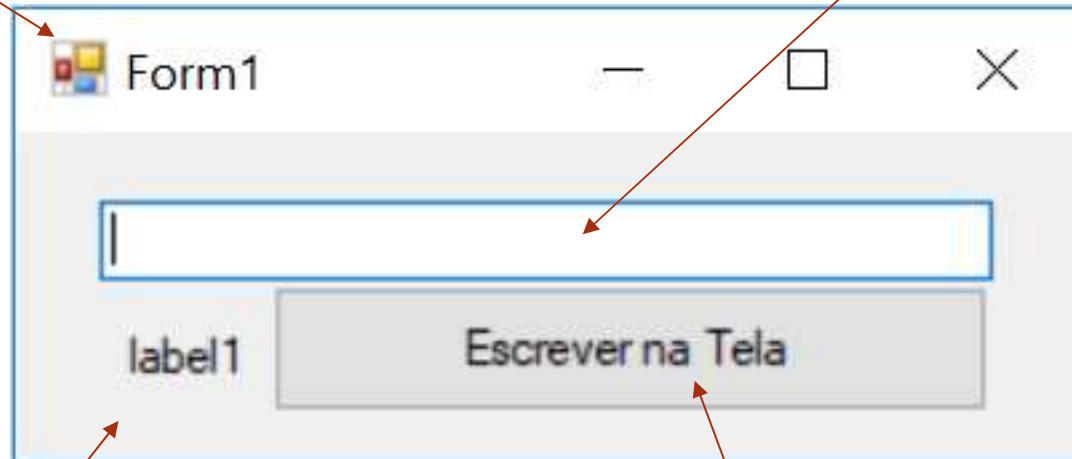
    private void InitializeComponent()
    {
        textBox1 = new TextBox();
        label1 = new Label();
        button1 = new Button();
        // etc, etc, etc...
    }
}
```

Entendendo por trás dos panos



```
Form1 f1 = new Form1();
```

```
TextBox label1 = new TextBox();
```



```
Label label1 = new Label();
```

```
Button button1 = new Button();
```