



# PROGRAMAÇÃO O.O. (C#)

---

{ Entrada e Saída de Arquivos  
**Professor:** João Luiz Lagôas

}

# Classes *File* e *Directory*

Ganhando intuição



- O .NET possui duas classes nativas com vários métodos estáticos para trabalhar com arquivos e pastas. A classe *File* fornece métodos para trabalhar com arquivos e a *Directory* permite lidar com diretórios. Escreva o que você acha que fazem essas linhas de código.

# Classes *File* e *Directory*

Ganhando intuição



Código	O que o código faz
<pre>if(!Directory.Exists(@"C:\LP2")){ Directory.CreateDirectory(@"C:\LP2"); }</pre>	
<pre>if (Directory.Exists(@"C:\LP2")){     Directory.Delete(@"C:\LP2"); }</pre>	
<pre>File.Copy(@"C:\LP2\notas.txt", @"D:\LP2\nt.txt");</pre>	
<pre>DateTime myTime = Directory.GetCreationTime(@"C:\LP2\nota s.txt");</pre>	
<pre>File.Delete(@"C:\LP2\notas.txt");</pre>	
<pre>File.WriteAllText(@"C:\LP2\materia2C.txt",     @"Programação O.O., muitos controles, encapsulamento, herança avançada, polimorfismo, acesso a arquivos");</pre>	

# Classes *File* e *Directory*

## Ganhando intuição



Código	O que o código faz
<pre>if(!Directory.Exists(@"C:\LP2")){ Directory.CreateDirectory(@"C:\LP2"); }</pre>	Checa se a pasta LP2 existe. Se não, uma pasta LP2 é criada.
<pre>if (Directory.Exists(@"C:\LP2")){     Directory.Delete(@"C:\LP2"); }</pre>	Checa se a pasta LP2 existe. Se sim, ela é deletada.
<pre>File.Copy(@"C:\LP2\notas.txt", @"D:\LP2\nt.txt");</pre>	Copia o arquivo notas.txt para o arquivo nt.txt.
<pre>DateTime myTime = Directory.GetCreationTime(@"C:\LP2\nota s.txt");</pre>	Declara uma variável myTime e a atribui ao retorno do método. <small>Note que esse método retorna um objeto do tipo DateTime.</small>
<pre>File.Delete(@"C:\LP2\notas.txt");</pre>	Deleta o arquivo notas.txt.
<pre>File.WriteAllText(@"C:\LP2\materia2C.txt",     @"Programação 0.0., muitos controles, encapsulamento, herança avançada, polimorfismo, acesso a arquivos");</pre>	Cria um arquivo chamado materia2C.txt (se ele não existir) e escreve o conteúdo passado como segundo parâmetro.

# Classes *File* e *Directory*



- Assim como StreamWriter, a classe **File** cria, nos bastidores, streams para você trabalhar com arquivos. Você pode usar seus métodos para realizar ações mais comuns sem ter de criar FileStreams primeiro.
- Os objetos **Directory** permitem trabalhar com diretórios inteiros cheios de arquivos e pode-se usá-los para fazer alterações na estrutura de pastas facilmente.
- A **documentação** contendo todos os atributos/propriedades e métodos dessas classes pode ser encontrada em:
  - [https://msdn.microsoft.com/pt-br/library/system.io.file\(v=vs.110\).aspx](https://msdn.microsoft.com/pt-br/library/system.io.file(v=vs.110).aspx)
  - [https://msdn.microsoft.com/pt-br/library/system.io.directory\(v=vs.110\).aspx](https://msdn.microsoft.com/pt-br/library/system.io.directory(v=vs.110).aspx)

# Classe *File*



- Coisas que você pode fazer com a classe File
  1. Descobrir se um arquivo existe: você pode checar para ver se um arquivo existe usando método **Exists()**.
  2. Ler e escrever no arquivo: você pode usar o método **OpenRead()** para acessar dados de um arquivo, ou o método **Create()** ou **OpenWrite()** para escrever nele.
  3. Concatenar texto em um arquivo: O método **AppendAllText()** permite anexar texto a um arquivo criado. Inclusive, ele cria o arquivo se este não estiver lá quando o método executar.
  4. Obter informações:
    - o método **GetLastAccessTime()** e **GetLastWriteTime()** retornam a data e hora do último acesso e alteração do arquivo.

# Classe *Directory*



- Coisas que você pode fazer com a classe *Directory*
  1. Criar um novo diretório: Crie um diretório usando o método **CreateDirectory()**. Tudo a fazer é fornecer o caminho que o método fará todo o resto.
  2. Obter uma lista de arquivos no diretório: você pode criar uma matriz de arquivos em um diretório usando o método **GetFiles()**. Apenas informe ao método um diretório e ele se encarregará de tudo.
  3. Apagar um diretório é bem simples também. Use o método **Delete()**.

# Exemplo prático


## Entendendo o código



Gerenciador de Desculpas

Descrição:

Resultado:

Usada pela última vez:  

Pasta Seleccionada:

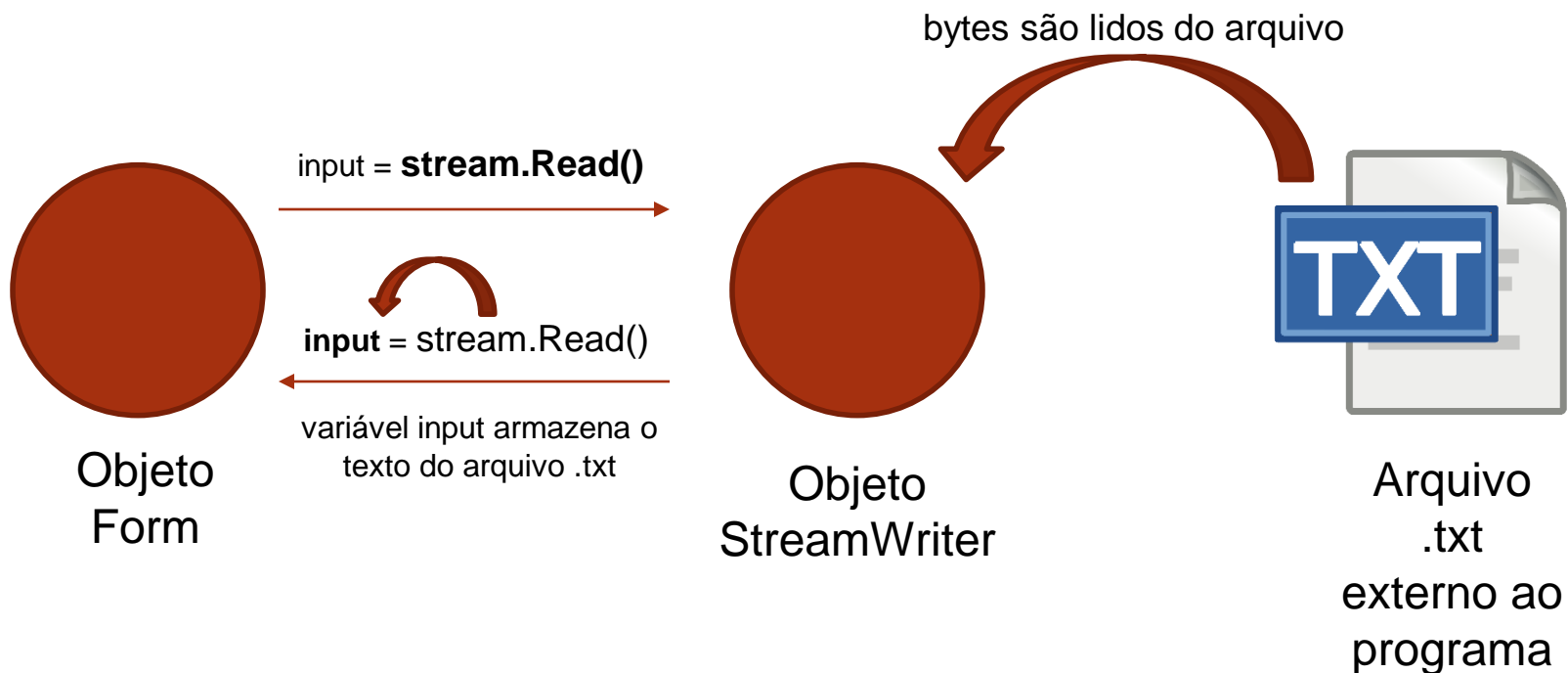


# Exemplo prático

## Entendendo o código



- Suponha que você tem um programa simples – um formulário com um tratador de eventos que precisa ler dados de um arquivo. Você usará um objeto StreamReader quando for necessário ler informações.

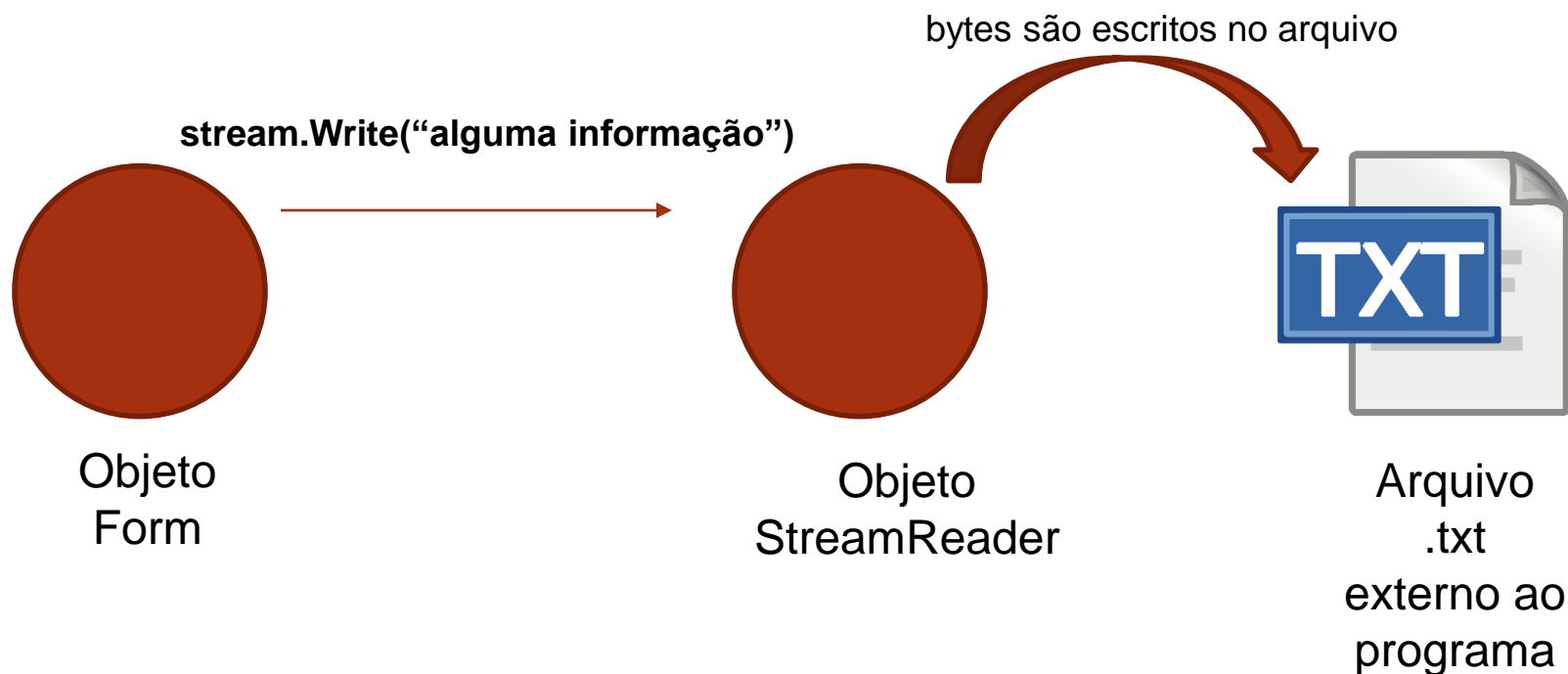


# Exemplo prático

## Entendendo o código



- E se o seu programa precisar escrever dados em um arquivo externo, ele também usará um objeto StreamWriter.

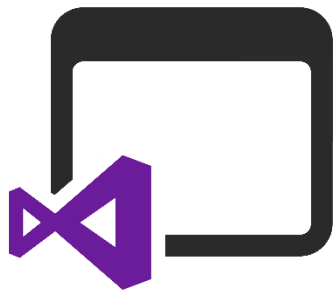


# Exemplo prático

## Entendendo o código



- O programa é composto por três classes:



### Solução/Projeto



Program.cs (onde está o método Main())



Desculpa.cs



Form1.cs

# Exemplo prático

## Entendendo o código



Desculpa.cs

```
class Desculpa
{
    public string Descricao;
    public string Resultado;
    public DateTime DataDeUltimoUso;

    public Desculpa()
    {
    }

    public Desculpa(string descricao, string resultado, DateTime
                        dataDeUltimaUso)
    {
        Descricao = descricao;
        Resultado = resultado;
        DataDeUltimoUso = dataDeUltimaUso;
    }
}
```

# Exemplo prático

## Entendendo o código



Program.cs

```
static class Program
{
    static void Main()
    {
        Application.EnableVisualStyles();

        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());
    }
}
```

# Exemplo prático

## Entendendo o código



Form1.cs

```
public partial class Form1 : Form
{
    private string selectedFolderPath;
    private Desculpa myExcuse;

    public Form1()
    {
        InitializeComponent();
        myExcuse = new Desculpa();
    }

    private void folderButton_Click(object sender, EventArgs e)
    {...}

    private void saveButton_Click(object sender, EventArgs e)
    {...}

    private void openButton_Click(object sender, EventArgs e)
    {...}

    private void randomExcuseButton_Click(object sender, EventArgs e)
    {...}

    private void saveExcuseOnFile(string filePath)
    {...}

    private void loadExcuseFromFile(string filePath)
    {...}
}
```

# Exemplo prático

## Entendendo o código



Form1.cs

```
private void folderButton_Click(object sender, EventArgs e)
{
    FolderBrowserDialog folderDialog = new
                                   FolderBrowserDialog();
    DialogResult result = folderDialog.ShowDialog();

    if (result == DialogResult.OK)
    {
        selectedFolderPath =
                               folderDialog.SelectedPath;
        folderPathTextBox.Text = selectedFolderPath;
        saveButton.Enabled = true;
        openButton.Enabled = true;
        randomExcuseButton.Enabled = true;
    }
}
```

# Exemplo prático

## Entendendo o código



Form1.cs

```
private void saveButton_Click(object sender, EventArgs e)
{
    if (String.IsNullOrEmpty(descriptionTextBox.Text) ||
        String.IsNullOrEmpty(resultTextBox.Text))
    {
        MessageBox.Show("Por favor, entre com uma descrição e resultado de desculpa");
        return;
    }

    SaveFileDialog saveDialog = new SaveFileDialog();

    saveDialog.InitialDirectory = selectedFolderPath;
    saveDialog.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";
    saveDialog.FileName = descriptionTextBox.Text + ".txt";

    DialogResult result = saveDialog.ShowDialog();

    if (result == DialogResult.OK)
    {
        saveExcuseOnFile(saveDialog.FileName);
    }
}
```



# Exemplo prático

## Entendendo o código



Form1.cs

```
private void saveExcuseOnFile(string filePath)
{
    myExcuse = new Desculpa(descriptionTextBox.Text,
                             resultTextBox.Text, dateTimePicker.Value);

    using (StreamWriter escritor = new
                                     StreamWriter(filePath))
    {
        escritor.WriteLine(myExcuse.Descricao);
        escritor.WriteLine(myExcuse.Resultado);
        escritor.WriteLine(myExcuse.DataDeUltimoUso.ToString());
    }

    MessageBox.Show("Desculpa salva com sucesso!");
}
```

# Exemplo prático

## Entendendo o código



Form1.cs

```
private void openButton_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();

    openFileDialog.InitialDirectory = selectedFolderPath;
    openFileDialog.Filter = "Text files (*.txt)|*.txt|All
                             files (*.*)|*.*";

    DialogResult result = openFileDialog.ShowDialog();

    if (result == DialogResult.OK)
    {
        loadExcuseFromFile(openFileDialog.FileName);
    }
}
```

# Exemplo prático

## Entendendo o código



Form1.cs

```
private void loadExcuseFromFile(string filePath)
{
    myExcuse = new Desculpa();

    using (StreamReader leitor = new StreamReader(filePath))
    {
        myExcuse.Descricao = leitor.ReadLine();
        myExcuse.Resultado = leitor.ReadLine();
        myExcuse.DataDeUltimoUso =
            Convert.ToDateTime(leitor.ReadLine());
    }

    descriptionTextBox.Text = myExcuse.Descricao;
    resultTextBox.Text = myExcuse.Resultado;
    dateTimePicker.Value = myExcuse.DataDeUltimoUso;

    MessageBox.Show("Desculpa carregada com sucesso!");
}
```

# Exemplo prático

## Entendendo o código



Form1.cs

```
private void randomExcuseButton_Click(object sender, EventArgs e)
{
    string[] filesPathes =
        Directory.GetFiles(selectedFolderPath, "*.txt");

    Random rand = new Random();

    loadExcuseFromFile(filesPathes[rand.Next(filesPathes.Length)]);
}
```