

Tópicos Especiais em Desenvolvimento de Sistemas

Conteúdos

- Linguagens Java, C# e VB.NET
- Introdução
 - Ambiente de desenvolvimento.
 - POO.
 - Apps de Console.

Java, C#, VB.NET - comparações

- Paradigma da programação orientada a objetos.
- Ambientes integrados de desenvolvimento (IDE).
 - **Java** (NetBeans, Eclipse, Android Studio, VS Code);
 - **C#.NET** e **VB.NET** (Microsoft Visual Studio . NET, VS Code).
- Analisemos hoje algumas características das linguagens Java, C#.NET, VB.NET.

Estrutura de decisão

- **Java, C#**

```
if (condição)
{
    //comandos
}
else
{
    //comandos
}
```

- **VB.NET**

```
If condição Then
    'comandos
Else
    'comandos
End If
```

Estrutura de decisão (cont.)

- **Java, C#**

```
switch (valor)
{
    case condição1 :
        //comandos
        break;
    case condição2 :
        //comandos
        break;
    ...
    default:
        //comandos
}
```

- **VB.NET**

```
Select Case valor
    Case condição
        'comandos
    ...
    Case Else
        'comandos
End Select
```

Estruturas de repetição

- **Java, C#**

```
while (condição)
{
    //comandos
}
```

- **VB.NET**

```
While condição
    'comandos
End While
```

```
Do Until condição
    'comandos
Loop
```

```
Do While condição
    'comandos
Loop
```

Estruturas de repetição (cont.)

- Java, C#

```
do  
{  
    //comandos  
} while (condição);
```

Observe que a condição se encontra no final.

- VB.NET

```
Do  
    'comandos  
Loop While condição
```

```
Do  
    'comandos  
Loop Until condição
```

Estrutura de repetição (cont.)

- **Java, C#**

```
for (inicialização; condição; incremento) {  
    //comandos  
}
```

Ex. `for(int i=0; i<N; i++) { ... }`

- **VB.NET**

```
For inicialização To valorFinal Step repetição  
    'comandos  
Next
```

Procedimentos/funções/métodos

- **Java, C#**

- **Procedimento**

```
modificador void nome(parâmetros) {  
    //comandos  
}
```

tipos: int, float, String...

modificadores: public, private...

- **Função**

```
modificador tipo nome(parâmetros) {  
    // comandos  
    return valor;  
}
```

- **VB.NET**

- **Procedimento**

```
Sub nome (parâmetros)  
    'comandos  
End Sub
```

- **Função**

```
Function nome (parâmetros) As Tipo  
    'comandos  
    Return valor  
End Function
```


- **Java**

```
class ClasseFilha extends ClasseMae {  
    private int atributo;  
  
    public ClasseFilha () {  
        super();  
    }  
  
    public ClasseFilha (int a, int b, int c)  
    {  
        super(a,b);  
        atrib = c;  
    }  
  
    public outroMetodo () {  
        super.metodoDaClasseBase();  
        //...  
    }  
}
```

- **C#**

```
class ClasseFilha : ClasseMae {  
    private int atributo;  
  
    public ClasseFilha () {  
        //chamada implícita ao construtor  
        //da ClasseMae  
    }  
  
    public ClasseFilha (int a, int b, int c)  
        : base ( a, b )  
    {  
        atrib = c;  
    }  
  
    public outroMetodo () {  
        base.metodoDaClasseBase();  
        //...  
    }  
}
```

Classe: Java x VB.Net

- Java

```
public class Pessoa {  
    private String nome;  
    private String cpf;  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
  
    //métodos get e set para atributos...  
}
```

- VB.Net

```
Public Class Pessoa  
    Private cpf As String  
    Private pnome As String  
  
    Property nome() As String  
        Get  
            Return Me.pnome  
        End Get  
        Set(ByVal valor As String)  
            Me.pnome = valor  
        End Set  
    End Property  
  
    ... Construtores e outros métodos  
End Class
```

depois
veremos
property
em C#



Herança: Java x VB.NET

- Java

```
public class Aluno extends Pessoa {  
    private String rgm;  
  
    Aluno(String nome, String rgm, String cpf){  
        super (nome,cpf);  
        this.rgm=rgm;  
    }  
  
    public String getRgm() {  
        return this.rgm;  
    }  
  
    public void setRgm(String rgm) {  
        this.rgm = rgm;  
    }  
}
```

- VB.Net

Public **Class** Aluno **Inherits** Pessoa

Private prgm As String

Sub **New**(ByVal innome As String, ByVal incpf As
String, ByVal inrgm As String)

MyBase.New (innome, incpf)

Me.Rgm = inrgm

End Sub

Property Rgm As String

Get

Return prgm

End Get

Set(ByVal valor As String)

Me.prgm = valor

End Set

End Property

End Class

Interface gráfica: código gerado (Java)

```
private void initComponents() {  
  
    jLabel1 = new javax.swing.JLabel();  
    jLabel2 = new javax.swing.JLabel();  
    ...  
    txtTitulo = new  
    javax.swing.JTextField();  
    txtRgm = new javax.swing.JTextField();  
    ...  
    bnt_Prof = new javax.swing.JButton();  
    btn_Aluno = new  
    javax.swing.JButton();  
    ...  
    jLabel1.setText("PROFESSOR");  
    getContentPane().add(jLabel1);  
    jLabel1.setBounds(20, 150, 80, 14);  
    ...  
}
```

```
    getContentPane().add(txtTitulo);  
    txtTitulo.setBounds(90, 200, 130, 20);  
  
    getContentPane().add(txtRgm);  
    txtRgm.setBounds(80, 40, 120, 20);  
  
    bnt_Prof.setText("Exibir");  
    bnt_Prof.addActionListener(new  
        java.awt.event.ActionListener() {  
        public void  
            actionPerformed(java.awt.event.ActionEvent evt) {  
                bnt_ProfActionPerformed(evt);  
            }  
        });  
    getContentPane().add(bnt_Prof);  
    bnt_Prof.setBounds(310, 200, 70, 23);  
    ...  
    ...  
}
```

Interface gráfica (código gerado VB.NET)

```
Private Sub InitializeComponent()  
    Me.Label1 = New System.Windows.Forms.Label  
    Me.Label2 = New System.Windows.Forms.Label  
    ...  
    Me.txtRgm = New System.Windows.Forms.TextBox  
    Me.txtNomeAluno = New System.Windows.Forms.TextBox  
    ...  
    Me.btnAluno = New System.Windows.Forms.Button  
    Me.btnProf = New System.Windows.Forms.Button  
    ...  
    Me.Label1.AutoSize = True  
    Me.Label1.Location = New System.Drawing.Point(30, 33)  
    Me.Label1.Name = "Label1"  
    Me.Label1.Size = New System.Drawing.Size(44, 13)  
    Me.Label1.TabIndex = 0  
    Me.Label1.Text = "ALUNO"  
    ...  
    Me.txtRgm.Location = New System.Drawing.Point(129, 56)  
    Me.txtRgm.Name = "txtRgm"  
    Me.txtRgm.Size = New System.Drawing.Size(100, 20)  
    Me.txtRgm.TabIndex = 5  
    ...  
    Me.btnAluno.Location = New System.Drawing.Point(394, 59)  
    Me.btnAluno.Name = "btnAluno"  
    Me.btnAluno.Size = New System.Drawing.Size(75, 23)  
    Me.btnAluno.TabIndex = 9  
    Me.btnAluno.Text = "Exibir"  
    Me.btnAluno.UseVisualStyleBackColor = True  
    ...  
    Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)  
    Me.AutoScaleMode =  
        System.Windows.Forms.AutoScaleMode.Font  
    Me.ClientSize = New System.Drawing.Size(491, 299)  
    Me.Controls.Add(Me.btnProf)  
    Me.Controls.Add(Me.btnAluno)  
    Me.Controls.Add(Me.txtTitulo)  
    Me.Controls.Add(Me.txtNomeProf)  
    Me.Controls.Add(Me.txtNomeAluno)  
    Me.Controls.Add(Me.txtRgm)  
    Me.Controls.Add(Me.Label6)  
    Me.Controls.Add(Me.Label5)  
    Me.Controls.Add(Me.Label4)  
    Me.Controls.Add(Me.Label3)  
    Me.Controls.Add(Me.Label2)  
    Me.Controls.Add(Me.Label1)  
    Me.Name = "Form1"  
    Me.Text = "Form1"  
    Me.ResumeLayout(False)  
    Me.PerformLayout()  
  
End Sub
```

Eventos (click em botões)

- **Java**

```
private void btn_AlunoActionPerformed(java.awt.event.ActionEvent evt) {  
    Aluno a = new Aluno("Juliana","11.111-1","111.111.111-11");  
    txtNomeAluno.setText(a.getNome());  
    txtRgm.setText(a.getRgm());  
}
```

- **VB.NET**

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)  
    Handles btnAluno.Click  
    Dim a As New Aluno("Juliana", "11.111-1", "111.111.111-11")  
    txtNomeAluno.Text = a.pnome  
    txtRgm.Text = a.prgm  
End Sub
```

Eventos (click em botões)

C# - um evento para somar dois valores digitados pelo usuário.

```
private void button1_Click (object sender, EventArgs e)
{
    double a, b, res;
    a = Convert.ToDouble(textBox1.Text);
    b = Convert.ToDouble(textBox2.Text);
    res = a + b;
    label3.Text = "Resultado (soma): " + res.ToString("0.00");

    //ou poderíamos apresentar o resultado em um quadro de diálogo simples:
    //  MessageBox.Show( "Resultado (soma): " + res.ToString("0.00") );
    //ou em um quadro de diálogo mais completo:
    //  MessageBox.Show(null,"Resultado (soma): " + res.ToString("0.00"),
    //    "Soma", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

```
public class Aluno : Pessoa {  
  
    private String rgm, cpf;  
  
    public Aluno (String nome, String umrgm, String cpf) : base (nome) {  
        this.cpf = cpf;  
        Rgm = umrgm; //usando a propriedade Rgm  
    }  
  
    public String Rgm { //esta é a propriedade Rgm  
        get {  
            return rgm;  
        }  
        set {  
            rgm = value; //palavra reservada value  
        }  
    } //fim da propriedade Rgm  
}
```

```
Aluno al =  
    new Aluno( "José Alves",  
              "12345-6",  
              "123.456.789-3");  
  
al.Rgm = "12345-8"; //propriedade
```


Vantagem de utilizar properties (exemplos C#)

```
public class Aluno : Pessoa {  
  
    private String rgm, cpf;  
  
    public Aluno (String nome, String umrgm, String cpf)  
        : base (nome) {  
        this.cpf = cpf;  
        Rgm = umrgm; //usando a propriedade Rgm  
    }  
  
    public String Rgm { //esta é a propriedade Rgm  
        get {  
            return rgm;  
        }  
        set {  
            rgm = value; //palavra reservada value  
        }  
    } //fim da propriedade Rgm  
}
```

```
public class Aluno : Pessoa {  
    private String cpf;  
    public String rgm;  
  
    public Aluno (String nome,  
        String umrgm, String cpf)  
        : base (nome) {  
        this.cpf = cpf;  
        rgm= umrgm;  
    }  
}
```

No exemplo da direita, um objeto da classe Aluno poderia modificar diretamente o atributo público **rgm**.

Uma vantagem da solução à esquerda (com *property*) é poder utilizar uma lógica de validação, formatação, conversão etc. dentro das partes get e set da propriedade!

Console apps em .NET com a linguagem C# (aplicativos de console)

Já fizemos um app de console na aula anterior.

Abrir e analisar os exemplos em:

ConsoleAppAlunoPessoa
ConsoleAppAlunoPessoaV02
ConsoleAppAlunoPessoaV03

com classes, herança, propriedades etc.

Mais sobre "propriedades", exemplo em ConsoleAppAlunoPessoaV02 com classes, herança, propriedades etc.

```
public class Pessoa {  
    //Observe que não declaramos nenhum atributo nesta classe  
  
    public Pessoa(String nome)    //método construtor  
    {  
        Nome = nome; //utilizamos a propriedade Nome  
    }  
  
    public String Nome    //esta é a propriedade Nome  
    { get; set; }    //notação para uma propriedade 'autoimplementada'  
  
    public override string ToString()  
    {  
        return "Nome: " + Nome;  
    }  
  
} //fim da classe Pessoa
```

```
public class Aluno : Pessoa { //Aluno é uma classe derivada de Pessoa

    //Observe que não declaramos nenhum atributo nesta classe

    public Aluno(String nome, String rgm, String cpf) : base(nome)
    {
        Cpf = cpf; //usando a propriedade Cpf
        Rgm = rgm; //usando a propriedade Rgm
    }

    public String Rgm //esta é a propriedade Rgm 'autoimplementada'
    { get; set; }

    public String Cpf //exemplo de uma propriedade 'autoimplementada'
    { get; set; } //se for uma propriedade de somente leitura: { get; }

    public override string ToString()
    {
        //chamamos o método ToString() da classe Pessoa e adicionamos
        //os restantes dados do aluno (propriedades Rgm e Cpf):

        return base.ToString() + ", RGM: " + Rgm + ", CPF: " + Cpf;
    }

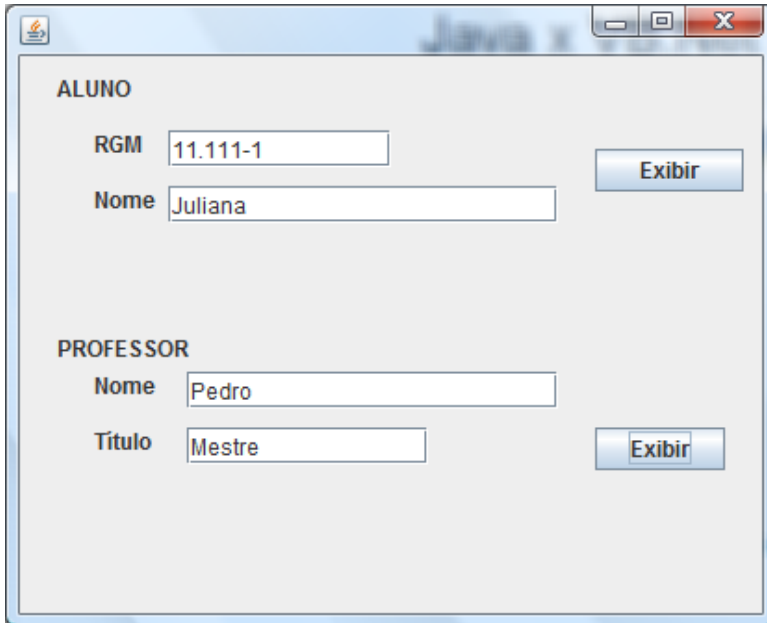
} //fim da classe Aluno
```

```
static void Main(string[] args) {
    String oRgmAluno;
    Console.WriteLine("Digite o RGM do aluno:");
    oRgmAluno = Console.ReadLine();
    Aluno al = new Aluno("José Alves", oRgmAluno, "333.444.555-6");
    Console.WriteLine("\nDados iniciais do aluno: \n" + al.ToString());

    al.Rgm = "12345-6";           //alteramos a propriedade Rgm
    al.Nome = "Ana Lopes Souza"; //alteramos a propriedade Nome
    al.Cpf = "123.456.789-10";    //alteramos a propriedade Cpf

    Console.WriteLine("\nDepois de alterar todos os dados do aluno:");
    Console.WriteLine(al.ToString());
    Console.ReadLine();
}
```

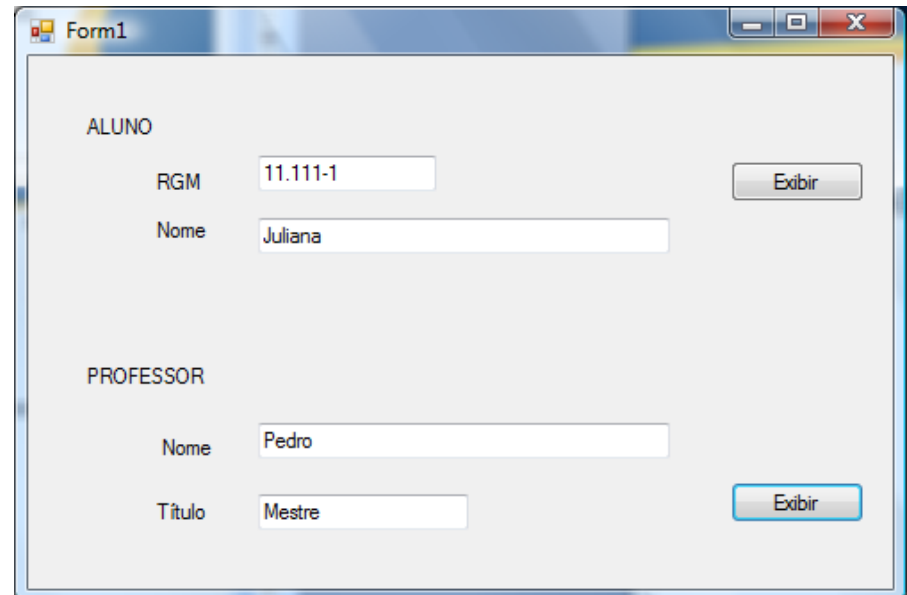
Interface gráfica em apps desktop: Java X .NET



A screenshot of a Java Swing window titled "Java x .NET". The window contains two sections: "ALUNO" and "PROFESSOR". The "ALUNO" section has a label "RGM" with a text box containing "11.111-1" and a label "Nome" with a text box containing "Juliana". To the right of these fields is a button labeled "Exibir". The "PROFESSOR" section has a label "Nome" with a text box containing "Pedro" and a label "Título" with a text box containing "Mestre". To the right of these fields is a button labeled "Exibir".

Java

.NET



A screenshot of a .NET Windows Form titled "Form1". The form contains two sections: "ALUNO" and "PROFESSOR". The "ALUNO" section has a label "RGM" with a text box containing "11.111-1" and a label "Nome" with a text box containing "Juliana". To the right of these fields is a button labeled "Exibir". The "PROFESSOR" section has a label "Nome" with a text box containing "Pedro" and a label "Título" with a text box containing "Mestre". To the right of these fields is a button labeled "Exibir".

Ambiente (IDE) Microsoft Visual Studio .NET

WindowsApplication1 - Microsoft Visual Studio

File Edit View Project Build Debug Data Format Tools Test Window Help

Form1.vb [Design]*

Form1

ALUNO

RGM

Nome

Exibir

PROFESSOR

Nome

Título

Exibir

Toolbox

All Windows Forms

Common Controls

Pointer

Button

CheckBox

CheckedListBox

ComboBox

DateTimePicker

Label

LinkLabel

ListBox

Listview

MaskedTextBox

MonthCalendar

NotifyIcon

NumericUpDown

PictureBox

ProgressBar

RadioButton

RichTextBox

TextBox

ToolTip

TreeView

Error List

0 Errors 0 Warnings 0 Messages

Description File Line Column Project

Solution Explorer

WindowsApplication1

My Project

References

bin

obj

Aluno.vb

Form1.vb

Form1.Designer.vb

Form1.resx

Pessoa.vb

Professor.vb

Solution Explorer Data Sources

Properties

btnProf System.Windows.Forms.Button

MaximumSize 0; 0

MinimumSize 0; 0

Modifiers **Friend**

Padding 0; 0; 0; 0

RightToLeft No

Size 75; 23

TabIndex 10

TabStop True

Tag

Text **Exibir**

ToolTip MiddleCenter

Text

The text associated with the control.

Toolbox (Ctrl+Alt+X)
View > Toolbox

Properties (F4)
View > Properties Windows

Solution Explorer (Ctrl+Alt+L)
View > Solution Explorer

Outros detalhes da linguagem de programação C#

A linguagem de programação C#

- Sua origem é fundamentada na linguagem C++ e também possui algumas características da linguagem Java.
- Assim como Java, a linguagem C# é orientada a objetos.
- Segue a mesma estrutura das linguagens Java e C++ e com isso facilita a migração dos desenvolvedores com experiência nestas linguagens.
- O C# possui o Garbage Collector (semelhante ao Java) que elimina da memória os dados que não estão sendo mais referenciados.
- Assim como acontece no Java, C# não aceita herança múltipla, mas essa ideia pode ser implementada parcialmente utilizando o conceito de herança de *interfaces*.

Operadores aritméticos em C#

- Os operadores utilizados em C# são os mesmos utilizados em Java.
- Operadores aritméticos. Para teste, a variável x é igual a 10.

Operador		Exemplo	Resultado
+	Adição	<code>z = x + 10;</code>	<code>z = 20</code>
-	Subtração	<code>z = x - 10;</code>	<code>z = 0</code>
*	Multiplicação	<code>z = x * 10;</code>	<code>z = 100</code>
/	Divisão	<code>z = x / 10;</code>	<code>z = 1</code>
%	Módulo (resto de uma operação de divisão)	<code>z = x % 3;</code>	<code>z = 1</code>
++	Incremento antes	<code>z = ++x;</code>	<code>z = 11</code>
	Incremento depois	<code>z = x++;</code>	<code>z = 10</code>
--	Decremento antes	<code>z = --x;</code>	<code>z = 9</code>
	Decremento depois	<code>z = x--;</code>	<code>z = 10</code>

Operadores de atribuição em C#

- Operadores de atribuição
- Para teste, as variáveis têm os valores: $x = 10$ e $y = 2$

Operador	Exemplo	Equivalência	Resultado
=	$x = y$	-	$x = 2$
+=	$x += y$	$x = x + y$	$x = 12$
-=	$x -= y$	$x = x - y$	$x = 8$
*=	$x *= y$	$x = x * y$	$x = 20$
/=	$x /= y$	$x = x / y$	$x = 5$
%=	$x \% = y$	$x = x \% y$	$x = 0$

Operadores de comparação em C#

- Operadores de comparação.
- Para teste, a variável x é igual a 10.

Operador	Descrição	Exemplo
==	Igual a	x == 20 (falso)
!=	Diferente de	x != 20 (verdadeiro)
<	Menor que	x < 20 (verdadeiro)
<=	Menor ou igual a	x <= 10 (verdadeiro)
>	Maior que	x > 20 (falso)
>=	Maior ou igual a	x >= 10 (verdadeiro)

Operadores lógicos em C#

- Operadores lógicos
- Para teste, as variáveis são: $x = 10$ e $y = 2$

Operador	Descrição	Exemplo
&&	e	$(x > y \ \&\& \ y < 5)$ (verdadeiro)
	ou	$(x > y \ \ y > 5)$ (verdadeiro)
!	Negação	$! (x != y)$ (falso)

Tabela verdade

Expressão 1	Expressão 2	&&	
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Expressão	!
true	false
false	true

Operadores lógicos em Visual Basic (VB)

Obs: os operadores lógicos em VB são palavras em inglês!

- **And, Or, Not, Xor** etc.

Veja mais em <https://learn.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/operators-and-expressions/logical-and-bitwise-operators>.

Tipos de Dados

Tipo de dados em C#	Equivalente no framework .NET
bool	System.Boolean
byte	System.Byte
char	System.Char
double	System.Double
float	System.Single
int	System.Int32
long	System.Int64
short	System.Int16
string	System.String

Declaração de uma variável

tipo variável;
ou
tipo variável = valor inicial;

Exemplos:

float peso;
int i;
int idade=29;

Tipos de dados

Tipo	Tamanho em bits	Valores
bool	8	true ou false
char	16	'\u0000' a '\uFFFF'
byte	8	0 a 255
sbyte	8	-128 a +127
short	16	-32.768 a +32.767
ushort	16	0 a 65.535
int	32	-2.147.483.648 a 2.147.483.647
uint	32	0 a 4.294.967.295
long	64	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807
ulong	64	0 a 18.446.744.073.709.551.615
decimal	128	$1,0 \times 10^{-28}$ a $7,9 \times 10^{28}$
float	32	$\pm 1,5 \times 10^{-45}$ a $\pm 3,4 \times 10^{38}$
double	64	$\pm 5,0 \times 10^{-324}$ a $\pm 1,7 \times 10^{308}$
object		
string		

Conversão entre tipos de dados: Convert

A classe Convert fornece métodos para conversão entre diferentes tipos de dados.

```
double a, b, s = 0.0;
Convert.ToDouble(|
```

▲ 13 of 18 ▼ double Convert.ToDouble(string value)

Converts the specified string representation of a number to an equivalent double-precision floating-point number.
value: A string that contains the number to convert.

Convert.

a = Conv

b = Conv

if (radi

if (radi

if (radi

if (radi

label3.T

private void

private void

private void

- ToByte
- ToChar
- ToDateTime
- ToDecimal
- ToDouble
- ToInt16
- ToInt32
- ToInt64
- ToSByte

double Convert.ToDouble(DateTime value) (+ 17 overload(s))
Calling this method always throws System.InvalidCastException.
Exceptions:
System.InvalidCastException

Ajuda no contexto

Um exemplo

```
private void button1_Click(object sender, EventArgs e) {
    double a, b, s = 0.0;
    a = Convert.ToDouble(textBox1.Text);
    b = Convert.ToDouble(textBox2.Text);
    if (radioButton1.Checked) s = a + b;
    if (radioButton2.Checked) s = a - b;
    if (radioButton3.Checked) s = a * b;
    if (radioButton4.Checked) s = a / b;
    label3.Text = "Resultado: " + s;
}
```

Estrutura (comando) if em C#

```
if (condição)
{
    // comandos
}
else
{
    // comandos
}
```

Exemplo:

```
int a = 10;
int b = 15;

if (a == b)
{
    MessageBox .Show("A = B");
}
else
{
    MessageBox .Show ("A != B");
}
```

Comando switch em C#

- Verifica o valor da expressão, não permite utilizar operações lógicas ou operadores de comparação.

```
switch(valor_ou_expressao)
{
    case 1:
        //comandos
        break;
    case 2:
        //comandos
        break;
    case n:
        //comandos
        break;
    default:
        //comandos
        break;
}
```

Exemplo:

```
int valor;
valor = Convert.ToInt32(textBox1.Text);
switch (valor)
{
    case 1:
        MessageBox.Show("Você escolheu segunda");
        break;
    case 2:
        MessageBox.Show("Você escolheu terça");
        break;
    case 3:
        MessageBox.Show("Você escolheu quarta");
        break;
    case 4:
        MessageBox.Show("Você escolheu quinta");
        break;
    case 5:
        MessageBox.Show("Você escolheu sexta");
        break;
    default:
        MessageBox.Show("Final de semana...");
        break;
}
```

Comandos while e do-while em C#

while (condição)

```
{  
    // comandos  
}
```

do

```
{  
    // comandos  
} while (condição);
```

Exemplo:

```
int valor = Convert.ToInt32(textBox1.Text);  
int i = 0;  
String conteudo = "";  
while (i <= valor)  
{  
    conteudo += "nº" + i + "\n";  
    i++;  
}  
MessageBox.Show(conteudo);
```

Comando for em C#

```
for(valor inicial; condição; incremento/decremento)
{
    // comandos
}
```

Exemplo:

```
int valor = Convert.ToInt32(textBox1.Text);
String conteudo = "";
for (int i = 0; i <= valor; i++)
{
    conteudo += "nº" + i + "\n";
}
MessageBox.Show(conteudo);
```

Namespaces (espaços de nome)

- Namespaces seguem a mesma ideia dos pacotes em Java.
- Isto ajuda no controle do escopo da classe e nomes de métodos em grandes projetos de programação.
- Para declarar nossos pacotes em C# utilizamos a palavra namespace.
Exemplo:

```
namespace Exemplo1
{
    //nossos códigos, classes, métodos etc.
}
```

- Exemplo em C#.NET:
`System.Console.WriteLine("mensagem");`
`System` é um namespace, `Console` é uma classe e `WriteLine` é um método.
- Para utilizar um namespace utilizaremos a cláusula **using** (semelhante a `import` em Java). Por exemplo:

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

Exemplos práticos resolvidos

Elaborar um programa desktop C#.NET que permita somar dois valores digitados pelo usuário e mostrar o resultado dessa soma. Resolvido como App de Console no arquivo: [AppConsoleSomarDoisValores.zip](#).

Apresentamos duas versões de um exemplo com várias operações aritméticas (soma, divisão, subtração e multiplicação) em [SomarDoisValores.zip](#).

Exercício 1 prático - **para entregar** até a próxima aula

Elaborar um programa C#.NET **do tipo "Aplicativo do Console"**, que permita calcular o valor de uma compra parcelada.

O usuário digitará o **valor da compra** e a **quantidade de parcelas** e o programa deverá calcular e visualizar o **valor das parcelas** e o **valor final da compra**.

Considere que na compra a vista (0 ou 1 parcela) daremos um desconto de 10%, que uma compra parcelada em até 5 vezes não tem juros e em compras parceladas em mais de 5 vezes haverá juros de 7% com relação ao valor inicial da compra.

Exercício 2 prático - **para entregar** até a próxima aula

Elaborar um programa C#.NET **do tipo "Windows Form"**, que permita calcular o valor de uma compra parcelada.

O usuário digitará o **valor da compra** e a **quantidade de parcelas** e o programa deverá calcular e visualizar o **valor das parcelas** e o **valor final da compra**.

Considere que na compra a vista (0 ou 1 parcela) daremos um desconto de 10%, que uma compra parcelada em até 5 vezes não tem juros e em compras parceladas em mais de 5 vezes haverá juros de 7% com relação ao valor inicial da compra.