Resumo de estudo (Conhecendo a plataforma .NET com C#

Professor: Leonardo de Lima Brito.

.Net: Plataforma de desenvolvimento unificado que permite construção de sistemas e aplicações.

C#: É uma linguagem de programação usada no .Net.

.Net: É uma plataforma de da Microsoft de desenvolvimento.

Diferença entre .Net e Framework (legado).

|  |  |
| --- | --- |
| .Net Framework | .Net |
| Wfp/Windows form/Asp.net | Multiplataforma |
| Exclusiva para o Sistema Windows | Para todos os sistemas/Windows/Linux/ios |
| Da versão 1.0 /2002 a 4.8 /2019 | .Net5 /.Net6 .... |

- Compilador .Net: (ver site: guru99.com)

* Linguagem de Alto nível: Linguagem que escrevemos e entendemos nosso código fonte.
* Linguagem de Baixo Nível: Linguagem de máquina.
* Compilador: É um programa que realiza a conversão de linguagem de alto nível para baixo nível.

compiler

Baixo Nível:

0101101110 01111000010

Liguagem de Alto Nível: C#, Python...

Fases de um Compilador:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Symbol table | | | | | Machine Code |
| Lexical Analyzer | Syntax Analyzer | Intermediate Code Generator | Code optimizer | Code Generator |

Transpilador: é uma conversão de uma linguagem ou implementação para outra. A sua saída permanece de alto nível. Ex: TypeScript -> JavaScript.

Linguagem Compilada: São linguagens onde o código fonte é traduzido para o código de máquina. Ex: Java, C#.

Linguagem Interpretada : São linguagens que fazem a leitura e interpretação diretamente do código fonte. Ex: PHP, JavaScript.

**O QUE É IDE?**

- Ambiente de Desenvolvimento Integrado. Software que tem como princípio facilitar e integrar diversas facilidades para a escrita e depuração de código.

Temos:

- Visual Studio

* Vantagens

1. Versão Gratuita (Community).
2. Debug Rico.
3. Windows e Mac.

* Desvantagens

1. Não tem para Linux.
2. Performance -> Exige muitos recursos de máquina, precisa de um pc muito bom.

- Visual Studio Code

* Vantagens

1. Totalmente Gratuito.
2. Possibilidade e Instalar Extensões.
3. Windows, Mac e Linux.
4. Muito leve em performance.

* Desvantagens

1. É necessária uma configuração inicial.
2. Não é muito intuitivo para algumas funcionalidades.

- Ride

* Vantagens

1. Rico em Funcionalidades.
2. Forte integração com o .Net.
3. Facilita o trabalho com UNITY.
4. Recomendação de refatoração de código.
5. Atalhos e comandos que aumenta produtividade.

* Desvantagens

1. Pago.
2. Performance -> Exige muitos recursos de máquina, precisa de um pc muito bom.

**Instalando o .Net SDK.**

Segue os passos abaixo para instalação do .Net SDK que deve ser o primeiro item a ser instalado no pc para iniciar o processo de preparação do ambiente de desenvolvimento.

1 Passo: Site da Microsoft : dotnet.microsoft.com/em-us/

2 Passo: Download do .Net SDK x64 ( caso seu pc seja 64bits)

3 Passo: Executar o .exe baixado no seu pc.

5 Passo: Após a instalação executar o CMD e nele digitar o seguinte: dotnet –info ( comando que mostra versão e demais detalhes do que foi instalado).

Instalando o Visual Studio Code (VS Code)

1 Passo: Site: code.visualstudio.com

2 Passo: Download for Windows.

3 Passo: Após baixar o .exe, executar o mesmo...

4 Passo: Deixar a opção sempre marcada (x) Add to path e Next/Install... até chegar a finish.

5 Passo: Abrir o Visual Studio Code e instalar as extensões:

* C# Microsoft
* C# Extensions for VsCode ( Joskreativ)
* Vscode-Icons.

Após todo o procedimento acima, vamos criar o primeiro projeto:

No terminal do Próprio Visual Studio Code pode criar as pastas, mas pode ser feito normalmente pelo Explorer do Windows.

1 Passo: Crie um pasta onde vc vai colocar o seu projeto: Ex:

C:\Projetos\ DotNet\NomeDoMeuProjeto

2 Passo: Dentro da pasta criada \NomeDoMeuProjeto -> execute o comando ->

dotnet new console

O comando acima irá criar a estrutura necessária para desenvolvimento do projeto:

**Monenclatura (Pasta de um projeto .Net).**

Após usar o comando: “dotnet new Console”

Atualizar projeto: “dotnet build”

|  |  |
| --- | --- |
| Meu Projeto | |
| bin | Binários do sistema |
| obj | objetos |
| MeuProjeto.csproj | Nome do projeto e sua extensão “.csproj” |
| Program.cs | Arquivo que vai ser executado o código que eu digitar, com o comando “dotnet run”. |

Conceito de Classe

Olá, meu nome é Jorge e tenho 20 anos.

Podemos com base acima criar a seguinte estrutura abaixo:

|  |  |
| --- | --- |
| Pessoa | Classe |
| + Nome | Atributo |
| + Idade | Atributo |
| + Apresentar() | Método |

Ver na próxima página a representação acima na forma de código no C#:

Que é representada no código c#:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ExemploFundamentos.Common.Models

{

    public class Pessoa

    {

        public string Nome { get; set; }

        public int Idade { get; set; }

        public void Apresentar()

        {

            Console.WriteLine($"OLá, meu nome é {Nome} \n e vou ter {Idade+1} anos em Novembro.");

        }

    }

}

Obs: para criar um atributo sem precisar digitar tudo, na linha do public ... basta digitar “prop “ + tab => que ele gera a linha automaticamente.

**Usando namespace => é o local que está o arquivo.**

Sempre que for usar a classe em outro local....

using ExemploFundamentos.Common.Models; (essa linha aqui é obrigatória para poder usar a classe, é o local que ela está.

Pessoa p = new Pessoa();

p.Nome = "Jorge Magnus";

p.Idade = 47;

p.Apresentar();

**Padrão a ser usado no projeto:**

camelCase - > sempre começa com a primeira letra minúscula e a cada palavra adicionada, começar com Maiúscula Ex: idadeDoAluno;

PascalCase -> sempre maiúscula, para cada nova palavra , também ser sempre maiúscula. Ex: IdadeDoAluno;

Por padrão as convenções no código seguem o camelCase e PascalCase para o seguinte esquema abaixo:

* Nome de Classe: PascalCase
* Nome de Propriedade: PascalCase
* Nome de Método: PascalCase
* Nome de variável: camelCase

Importante que o nome da classe seja também o mesmo do arquivo físico: Ex: arquivo “Pessoa.cs” = Classe -> Pessoa.

**Public class Pessoa**

**{**

**Propriedades, métodos...**

**}**

**Importante => Colocar o nome da variável sempre por extenso, completo e sem abreviações, ou seja, o mais explicado possível.**

**Tipos de dados (mais comuns).**

|  |  |
| --- | --- |
| Tipo | Ex: |
| string | String nomeAluno = “Jorge”; |
| char | String sexoPessoa = “M”; |
| bool | Bool entradaPortao = true; (true ou false) |
| byte | De 0 a 255 (integer) |
| decimal | Decimal preco = 1.35 –> 0.0M valores financeiros. |
| double | Double altura = 1.80; -> 0.0D (com casas decimais) |
| int | Int idade = 47; ( valores inteiros) |
| long | 0L -> inteiro longo. |
| float | 12.34 -> 0.0F com casas decimais. |

**O tipo datetime:**

DateTime dataAtual = DateTime.Now.AddDays(5);// dia atual adicionado mais 5 dias.Console.WriteLine("Data atual completa: " + dataAtual);Console.WriteLine("Data atual dia/mês/ano: " + dataAtual.ToString("dd/MM/yyyy"));Console.WriteLine("Data atual dia/mês/ano Hora e Minuto: " + dataAtual.ToString("dd/MM/yyyy HH:mm"));

**Operadores de Atribuição “=”**

Ex: int a = 10;

String nomePessoa = “Jorge”;

Double valorEntradaParque = 10.50;

**Observação importante: Caso queira colocar várias linhas ao mesmo tempo em comentário, seleciona CTRL + K + C. EX:**

|  |  |  |
| --- | --- | --- |
| A | CTRL + K + C | //A |
| B | //B |
| C | //C |
| D | //D |

**USANDO Parse e Convert**

Ex: int a = Convert.ToInt32(“5”); -> Aceita valor null e retorna 0.

Int a = int.Parse(“5”); -> não aceita valor null, logo se não tiver valor, da erro no sistema.

**Conversão para string**

Todos os tipos, exceto o próprio string tem o .ToString.

Ex. int.ToString

Ex:  int valor = 0;

    valor.ToString("6");

**Cast Implícito**

Ex:

int => double

Int => long;

Entenda: Por o tipo double e o tipo long serem muito maiores que o tipo int, esses podem ser atribuídos direto sem problema algum, o contrario é que não pode double => int , nesse caso é porque um tipo double tem casas decimais e não vai ser possível passar para int, já no segundo caso do long para int , pode ser que o número long seja muito maior que o limite máximo do tipo int.

**Ordem dos Operadores**

Deve seguir o padrão matemático:

1. Parênteses ( )
2. Expoente
3. Divisão /
4. Multiplicação \*
5. Adição +
6. Subtração –

**Convertendo de Forma Segura**

Importante usar o TryParse para conversão segura. Ex:

String a = “15 –“;

Int b = Convert.ToInt32(a); -> Vai da erro no programa e ele para.

Correto:

Int b = 0;

Int.TryParse(a, out b); -> Nesse caso a variável b vai ser igual a zero, pois quando tentar converte a variável a, vai da erro, então o programa passa para a variável b e retorna o resultado 0.

Pode ser feito também dessa forma:

Int.TryParse(a , out int b); onde vai ocorrer o mesmo caso acima.

**OPERADORES CONDICIONAIS**

O operador condicional possibilita mudar o fluxo de execução do código , indicando um caminho que ele deve percorrer.

São eles:

|  |  |
| --- | --- |
| Operador | Estrutura |
| if | If (condição)  {  Instrução.  } |
| If / else | If (condição)  {  Instrução.  }else  {  Instrução.  } |
| If aninhado | If (condição)  {  Instrução.  }  Else if (condição)  {  Instrução.  }  Else if (condição)  {  Instrução.  }  Else  {  Instrução final, caso as anteriores não derem certo.  } |
| Switch / case | Switch (variável)  {  Case “a”:  Case “b”:  Case “c”:  Case “d”:  Console.Writeline(“...”);  Break;  Defaut:  Console.Writeline(“..”);  Break;  } |

**Processo de depuração**

Passos para executar um processo de depuração:

1 – Passo: Clique na tecla F5 (obs: Na primeira vez que executar uma depuração , vai aparecer uma seleção na tela e clique na opção -> .NET5+ AND .NET CORE. -> Após selecionar e teclar enter vai criar um aquivo LAUNCH.JSON(Pode fexar esse arquivo). Caso não crie, vá no menu do VsCode e clique no 4 botão de cima pra baixo, “executar e depurar (ctrl + shift + D)” e na opção que aparece, clique em criar o arquivo LAUNCH.JSON. Lembrando que esse processo só e feito na primeira vez. Depois é só clicar em F5 que ele já vai direto para o depurador.

2 – Passo : Marque com o mouse onde quer começar a depurar, ao lado dos números que fica na tela do código, ao clicar vai ficar um ponto vermelho.

3 – Passo: Para fazer o passo a passo é com a tecla F10.

4 – Passo : Para sair, F5 ou no quadrado que aparece em cima , menu de botões

Ver imagem -> 

**Operadores lógicos**

|  |  |  |
| --- | --- | --- |
| Operador | Significado | Exemplo |
| == | Verifica se dois valores são iguais; se forem, retorna true, caso contrário, retorna false | (A == B) Retorna falso |
| != | Verifica se dois valores são diferentes; se forem, retorna true, caso contrário, retorna false | (A != B) Retorna true |
| > | Verifica se o primeiro valor é maior que o segundo, retornando verdadeiro em caso afirmativo | (A > B) Retorna false |
| < | Verifica se o primeiro valor é menor que o segundo, retornando verdadeiro em caso afirmativo | (A < B) Retorna true |
| >= | Verifica se o primeiro valor é maior ou igual ao segundo, retornando verdadeiro em caso afirmativo | (A >= B) Retorna false |
| <= | Verifica se o primeiro valor é menor ou igual ao segundo, retornando verdadeiro em caso afirmativo | (A <= B) Retorna true |
| || | Operador OR -> Basta uma das condições ser true para retornar true. | (condição A) || (Condição B) |
| && | Operador AND -> Para retornar true as duas condições tem que ser verdadeira, caso contrário retorna false. | (condição A) && (Condição B) |
| ! | Operador NOT -> Esse é um operador unário, o que significa que recebe apenas um valor lógico como entrada, e inverte esse valor na saída. Ou seja, True se transforma em False e vice-versa. | !variavel |

**Operadores Aritméticos**

|  |  |  |
| --- | --- | --- |
| Operador | Operação | Exemplo |
| + | Soma | x+y |
| - | Subtração | x-y |
| \* | Multiplicação | X\*y |
| / | Divisão | x/y |
| % | Modo | X%y(retorna o resto da divisão) |
| ++ | Incremento | X++ |
| -- | Decremento | x-- |

**Estrutura de repetição**

São estruturas de repetição For, While e Do While:

**Usando FOR:**

For(contador; condição; incremento contador)

{

Instrução...

}

Ex: for ( int contador = 0; contador <= 10; contador ++)

{

Console.Writeline($” O Número do contador é: {contador}”);

If (contador == 8)

{

Console.WriteLine(“O Contador chegou ao número 8”);

Break;

}

}

**Usando o While:**

While(condição)

{

Instrução...

}

ver exemplo na próxima página.

Ex: int contador = 0;

While (contador < =10)

{

If (contador < =8)

{

Console.WriteLine($”O Número do contador é: {contador}”);

Contador++;

}else

{

Console.Writeline(“Contador agora é 8”);

Break;

}

}

**Usando o Do While:**

Do

{

Instrução...

}while(condição);

Instrução encerramento...

Ex:

int soma = 0, numero = 0;

do

{

Console.WriteLine("Digite um número: (0 para parar.)");

numero = Convert.ToInt32(Console.ReadLine());

soma += numero;

}while(numero != 0);

Console.WriteLine($"Total da soma dos números digitados é: {soma}.");

**Exemplo construindo um menu interativo usando o While, Switch e case:**

string opcao;

bool exibirMenu = true;

while(exibirMenu) {

Console.Clear();

Console.WriteLine("Digite a sua opção:");

Console.WriteLine("1 - Cadastrar Cliente");

Console.WriteLine("2 - Buscar Cliente");

Console.WriteLine("3 - Apagar Cliente");

Console.WriteLine("4 - Encerrar");

opcao = Console.ReadLine();

switch(opcao)

{

case "1":

Console.WriteLine("Cadastro de Cliente");

break;

case "2":

Console.WriteLine("Busca de Cliente");

break;

case "3":

Console.WriteLine("Apagar Cliente");

break;

case "4":

Console.WriteLine("Encerrar");

exibirMenu = false;

break;

default:

Console.WriteLine("Opção Inválida!");

break;

}

}Console.WriteLine("O programa se encerrou");

**Estrutura de um Programa C#**

Arquivos de projeto:

.csproj : Contém informações referente a um projeto(build, debug, versão).

.sln : contém informações que carregam um agrupamento de projetos.

Estrutura:

|  |  |  |
| --- | --- | --- |
| **.sln** | | |
| .csproj  Projeto: Efetuar pix  🡪 | .csproj  Projeto: Common  Classes:  -ContaCorrente | .csproj  Projeto: Pagar DA  🡨 |

Para criar uma .sln :

1 – Passo: Sempre faça um backup do seu projeto.

2 – Passo: Se não tiver, instalar a extensão :

* vscode-solution-explorer
* Autor: Fernando Escola.

**Criando um projeto em .Net5:**

**Comando -> dotnet new console –framework net5.0**

***Obs: se for em .net6 o comando é -> dotnet new console***

***Caso não passe nada depois do console, o sistema instala sempre a ultima versão, se a ultima for a 6 ele cria nessa versão , se for a 7 e assim por diante.***

**Diferenças entre as versão .net5 e .net6**

***O program.cs no .Net5 fica assim:***

using System;

namespace ExemploFundamentos.NET5

{

Class Program

{

Static void Main(string[] args)

{

Console.Writeline(“Hello Word!”);

}

}

}

***O program.cs no .Net6 fica assim:***

Console.Writeline(“Hello Word!”);

***Importante, o método Main, nas versões anteriores a 6 é obrigatório. Na net6 acima já está implícito e não precisa digitar.***

**Array e Lista (Introdução Array)**

Array é uma estrutura de dados que armazena valores do mesmo tipo, com um tamanho fixo.

Ex:

1. Int[] array = new int[4]; -> Quando especifico o valor do array direto.
2. Int[] array = new int[] -> Quando vou especificar o valor do array digitando os valores {42,75,74,61} -> esse array passa a ter esses 4 valores.
3. String[] nomes = {“Jan”, “Fev”}; -> Direto, especificando os valores.

Índice: É a posição de um determinado valor de um array, sempre começando com zero.

Ex:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Valores -> | 42 | 75 | 74 | 61 |
| Posição(índice) -> | 0 | 1 | 2 | 3 |

1. Int elemento = array[0];
2. Array[0] = 42;

Ex de um código com array e for:

int[]  arrayInteiros = new int[4];

arrayInteiros[0] = 72;

arrayInteiros[1] = 64;

arrayInteiros[2] = 50;

arrayInteiros[3] = 1;

//Percorrendo o array com FOR

Console.WriteLine("Percorrando o Array com o FOR");

for (int contador = 0; contador < arrayInteiros.Length; contador++)

{

   Console.WriteLine($"Posição Nº {contador} - {arrayInteiros[contador]}");

}

Obs : se tentar acessar um array, posição que não existe, vai gerar um erro e travar o programa. Se o array é de 4 elementos, caso chame uma 5 posição o sistema gera uma exceção.

**Percorrando um array com o FOREACH**

EX:

int[]  arrayInteiros = new int[4];

arrayInteiros[0] = 72;

arrayInteiros[1] = 64;

arrayInteiros[2] = 50;

arrayInteiros[3] = 1;

//Percorrendo o array com FOREACH

Console.WriteLine("Percorrendo o array com o FOREACH");

int contadorValor = 0;

foreach(int valor in arrayInteiros)

{

   Console.WriteLine($"Posição Nº {contadorValor} - {valor}");

   contadorValor++;

}

Obs: ***O Foreach é indicado por sua facilidade de uso em relação ao for e também, quando não tiver necessidade de um contador.***

**Usando a classe Array, para mudar redimensionar o tamanho do array.**

Ex: Utilizando o código acima:

int[]  arrayInteiros = new int[4];

arrayInteiros[0] = 72;

arrayInteiros[1] = 64;

arrayInteiros[2] = 50;

arrayInteiros[3] = 1;

//Redimensionando o array.

Array.Resize(ref arrayInteiros, arrayInteiros.Length \* 2);

Console.WriteLine("Percorrando o Array com o FOR e o array redimensionado.");

for (int contador = 0; contador < arrayInteiros.Length; contador++)

{

   Console.WriteLine($"Posição Nº {contador} - {arrayInteiros[contador]}");

}

**Copiando um Array para outro**

Ex:

int[]  arrayInteiros = new int[4];

arrayInteiros[0] = 72;

arrayInteiros[1] = 64;

arrayInteiros[2] = 50;

arrayInteiros[3] = 1;

//Copiando um Array para outro:

int[] arrayInteirosDobrado = new int[arrayInteiros.Length \* 2];

Array.Copy(arrayInteiros,arrayInteirosDobrado,arrayInteiros.Length);

Console.WriteLine("Percorrando o Array com o FOR que foi copiado com os valores de um outro array.");

for (int contador = 0; contador < arrayInteiros.Length; contador++)

{

   Console.WriteLine($"Posição Nº {contador} - {arrayInteiros[contador]}");

}

**Trabalhando com listas**

Sintax de um list:

List<string> listaString = new List<string>(); Exemplo do uso de listas usando o For:

List<string> listaString  = new List<string>();

listaString.Add("SP");

listaString.Add("BA");

listaString.Add("MG");

listaString.Add("RN");

listaString.Add("RJ");

listaString.Add("PE");

listaString.Add("CE");

listaString.Add("PB");

Console.WriteLine($"Itens da minha lista: {listaString.Count} -> Capacidade: {listaString.Capacity}");

listaString.Remove("RJ");

Console.WriteLine("Percorrendo a lista com o For");

for(int contador = 0; contador < listaString.Count; contador++)

{

   Console.WriteLine($"Posição Nº {contador} - {listaString[contador]}");

}

Console.WriteLine($"Itens da minha lista(após remover o RJ): {listaString.Count} -> Capacidade: {listaString.Capacity}");

Usando a mesma lista , com o foreach:

List<string> listaString  = new List<string>();

listaString.Add("SP");

listaString.Add("BA");

listaString.Add("MG");

listaString.Add("RN");

listaString.Add("RJ");

listaString.Add("PE");

listaString.Add("CE");

listaString.Add("PB");

Console.WriteLine($"Itens da minha lista: {listaString.Count} -> Capacidade: {listaString.Capacity}");

listaString.Remove("RN");

Console.WriteLine("Percorrendo a lista com o Foreach");

int contadorForeach = 0;

foreach(string item in listaString)

{

   Console.WriteLine($"Posição Nº {contadorForeach} - {item}");

}

Console.WriteLine($"Itens da minha lista(após remover RN): {listaString.Count} -> Capacidade: {listaString.Capacity}");

**Comentários e Boas Práticas em C#**

Os comentários servem para documentar o seu código, explicando um determinado método ou execução e auxiliam outros programadores a entender o que está acontecendo.

Tipos de comentários:

|  |  |
| --- | --- |
| // | É representado em uma única linha. |
| /\* comentário \*/ | Permite escrever com várias linhas. |
| <summary> | Permite documentar classes, métodos, parâmetros, etc. |

Comentário de uma linha, exemplo:

//Exemplo de comentários de uma linha.

Comentário de várias linhas, exemplo:

/\*

  Fazendo comentários em varias linhas(texto grande.)

  Atribuir o nome e idade para pessoa

  passando o nome jorge

  passando a idade 47

  e pode escrever várias linhas aqui

  linha ....

\*/

Comentários em classes, exemplo:

{

    /// <summary>

    ///  Representa uma pessoa física.

    /// </summary>

    public class Pessoa

    {

        public string Nome { get; set; }

        public int Idade { get; set; }

        public void Apresentar()

        {

            Console.WriteLine($"OLá, meu nome é {Nome} \n e vou ter {Idade+1} anos em Novembro.");

        }

    }

}

Obs: para que seja possível utilizar o comentário <summary> ..... </summary> , antes ative uma configuração no Vscode:

1 – Passo: ícone da engrenagem.

2 – Passo: Vá até Settings (Configurações) e clico.

3 – Passo: na tela/aba que aparece, digito : editor: format on type

4 – Passo: marca essa opção.

5 – Passo: fechar a tela.

6 – Passo: agora é só digitar

///<summary>

/// descritivo da minha classe.

///<summary>

Agora quando passar o mouse em cima da classe em qualquer parte do código, vai aparecer o que foi colocado nesse comentário.

**Comentários nos métodos:**

/// <summary>

    /// Método que faz a pessoa se apresentar, dizendo seu nome e idade.

    /// </summary>

        public void Apresentar()

        {

            Console.WriteLine($"OLá, meu nome é {Nome} \n e vou ter {Idade+1} anos em Novembro.");

        }

**Outro exemplo:**

/// <summary>

      /// Método que realiza uma soma de dois números.

      /// </summary>

      /// <param name="x">O primeiro número inteiro para soma</param>

      /// <param name="y">O primeiro segunto inteiro para soma</param>

       public void Somar(int x, int y)

       {

            Console.WriteLine($"{x} + {y} = {x + y}");

       }

**Outro Exemplo:**

  /// <summary>

         /// Método que realiza a subtração de dois números:

         /// </summary>

         /// <param name="x">O Primeiro número para subtrair</param>

         /// <param name="y">O Segundo número para subtrair</param>

         /// <returns>Retorna a subtração de x e y</returns>

        public int Subtrair(int x, int y)

       {

            Console.WriteLine($"{x} - {y} = {x - y}");

            return x - y;

       }

***Obs: a palavra void é para informa que não quer retorna nada, como nesse método eu tirei e coloquei int, tenho que colocar o return, pois o método vai retorna o valor do cálculo.***

***Atenção: Não exagerar nos comentários para poluir e digitar demais, ou seja, redundante.***