

Escola Técnica Estadual “São Paulo” - ETESP

Linguagem C# - Visual Studio 2019

Escrevendo Métodos

Até o momento, em todos os nossos projetos, as soluções foram escritas “dentro de botões”, porém, poderá ocorrer situação em que parte da solução, que é escrita em um “botão”, deverá também ser utilizada em outra parte do projeto.

Analogamente ao conceito de “refrão” utilizado nas letras de músicas e com o objetivo de evitar que “pedaços do código” sejam copiados em várias partes do projeto, gerando duplicidade de escrita, é que iniciaremos o estudo sobre Métodos.

O perfeito entendimento deste assunto será de extrema importância para a continuidade e aplicação em outra abordagem que nos acompanhará por muito tempo: a Programação Orientada a Objetos.

Um método é uma sequência nomeada de instruções (e já utilizamos métodos desde o início do curso).

Um método possui uma “assinatura” e um corpo (conteúdo do método).

A assinatura do método é composta por:

- tipo do método (no momento utilizaremos métodos “private”. Ao tratarmos de POO outros “tipos” poderão ser apresentados).
- indicativo de retorno ou não (void). Se o método não retornar nenhum resultado deverá ser utilizada a palavra “void” (vazio), caso contrário, o tipoDeRetorno poderá ser int, string, double, float...
- nome do método: O nome do método deve ser um identificador significativo que identifique sua finalidade geral (por exemplo: CalcularValor, ExibirTela...). Perceba a notação adotada: a primeira letra será maiúscula (chamamos de Pascal Case, pois esta técnica já é adotada na linguagem Pascal).
- parâmetros → Informação “opcional”, pois depende do tipo de “processamento” que o método irá executar. São “dados” fornecidos ao método. Deverão ser escritos dentro dos parênteses, como se estivesse declarando variáveis: com o nome do tipo seguido pelo nome do parâmetro. Se o método receber dois ou mais valores, separe-os com vírgula.

A sintaxe para declarar um método é:

tipo retorno NomeDoMétodo (listaDeParametros)

```
{  
    // instruções do corpo do método  
}
```

Exemplos de métodos:

a) private void LimparCampos() // o private é "OPCIONAL"!!

```
{  
    // instruções;  
}
```

b) private int SomarValores (int valor1, int valor2)

```
{  
    // instruções do método  
}
```

c) void ExibirResultado (double resultadoOperacao)

```
{  
    // instruções do método  
}
```

Responda:

- 1) O exemplo "a" recebe algum parâmetro? Devolve algum resultado? Qual é o nome do "método"?
- 2) O exemplo "b" retorna algum valor? Qual o tipo de retorno? Qual é o nome do método? O método "recebe" parâmetros? Quais?
- 3) O exemplo "c", causará erro de sintaxe, pois não foi declarado o "private"? O método retorna algum valor? Qual o tipo de retorno? Qual o nome do método? O método recebe parâmetros? Quais?

Quando escrevemos um método que retorna algum tipo de resultado (ou seja, seu tipo de retorno **NÃO É VOID**) **devemos incluir no método uma instrução de "retorno" no final do processamento do método**. Uma instrução de retorno consiste em uma palavra-chave "return" seguida por uma expressão que especifica o valor que será retornado, seguido por ponto e vírgula, exemplo:

```
private int somarValores (int valor1, int valor2)
```

```
{  
    // instruções do método  
    return valor1 + valor2;  
}
```

IMPORTANTE:

- a) O “retorno” deverá ser compatível com o tipo de retorno declarado pelo método, isto é, se declaramos que o método irá retornar um número inteiro, devemos, obrigatoriamente, retornar a resposta de um número inteiro, caso contrário, teremos um erro na programação.
- b) Todas as instruções escritas após a instrução “return” NÃO SERÃO EXECUTADAS!!!

Chamadas de Métodos

Os métodos existem para serem executados!!!

E para serem executados, precisam ser chamados!!!

Chamamos um método pelo nome dado a ele no momento de sua declaração. Se o método precisar de informações (parametros), devemos fornecê-los. Se o método retorna informações, devemos “providenciar” sua captura de alguma forma.

A sintaxe de uma chamada de método, segue conforme o modelo:

retorno = NomeDoMétodo(listaDeArgumentos)

IMPORTANTE:

- 1) “retorno” será “obrigatório” se o método foi declarado como um “método” que irá retornar algum resultado, isto é, DIFERENTE DE VOID!!! Se você não especificar a cláusula “retorno” e o método retornar um valor, o método será executado, mas o valor de retorno será perdido!!!
- 2) “NomeDoMétodo” deve ser EXATAMENTE o mesmo nome que foi atribuído em sua declaração (criação).
- 3) “listaDeArgumentos” será obrigatória se o método foi declarado com uma lista de argumentos e o valor de cada argumento deve ser compatível com o tipo de parâmetro, isto é, se o método “espera” receber um valor inteiro, DEVEMOS fornecer um valor inteiro.
- 4) Os parênteses são obrigatórios, mesmo se o método NÃO possuir argumentos.

Exemplo:

```
private int SomarValores (int valor1, int valor2)
{
    // instruções do método
    return valor1 + valor2;
}
```

Antes da chamada do “método”, o programador deverá:

```
int varValor1 = 3;          //Declaração da variável varValor1 e atribuição de conteúdo
int varValor2 = 2;          // Declaração da variável varValor2 e atribuição de conteúdo
```

```
int resultado = SomarValores(varValor1, varValor2); //chamada do método
```

Prof. Roberto de Castro

Analise os exemplos abaixo para a chamada do método “SomarValores” e depois verifique os “prováveis” problemas:

- a) SomarValores;
- b) SomarValores();
- c) SomarValores(39);
- d) SomarValores(“teste1”, “teste2”);

Análise dos problemas:

- No item “A” não definimos os parênteses.
- No item “B” não definimos os parametros.
- No item “C” não fornecemos os argumentos suficientes.
- No item “D” fornecemos argumentos com tipos incorretos.
- Em nenhuma alternativa declaramos o local em que o “resultado” do método será armazenado!

Escopo das variáveis dentro do método

Caso haja a necessidade, podemos declarar variáveis dentro dos métodos. Essas variáveis passam a existir a partir do momento em que o método é executado e “desaparecem” quando o método finaliza. Neste caso dizemos que se trata de uma “**variável local**”, isto é, existe somente dentro do método que a criou. O que significa que NÃO podemos utilizar a “variável” em outra parte do programa!!!!

Observe o exemplo:

```
private void PrimeiroMétodo()
{
    int minhaVariavel;
}

private void SegundoMetodo()
{
    minhaVariavel = 12; //gerará um erro, pois a variável NÃO existe no método!
}
```

Vamos aplicar a teoria sobre “métodos”, através de exercícios práticos?

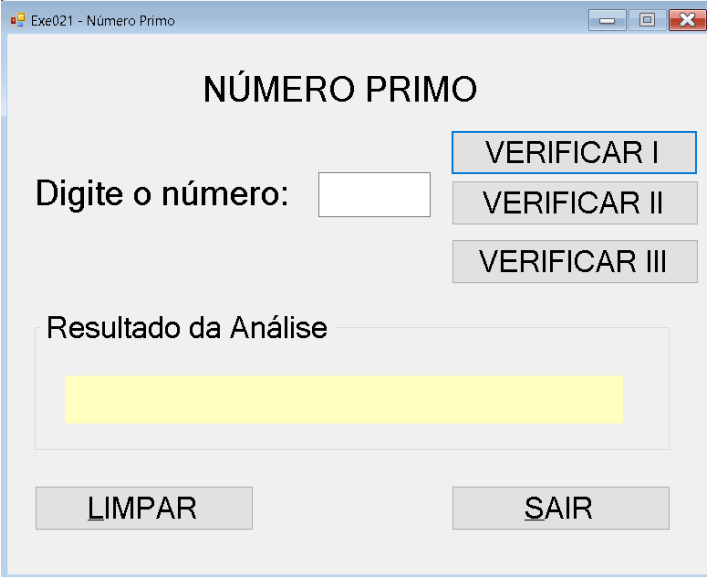
Prof. Roberto de Castro

Exercício “Modelo” (Número Primo Revisitado - com a utilização de Métodos) – Exe026A

Vamos alterar um projeto já resolvido, para ilustrar a utilização de métodos, com as seguintes variações:

- Método sem parâmetro e sem retorno.
- Método COM parâmetro e sem retorno.
- Método COM parâmetro e COM retorno.

INTERFACE



SOLUÇÃO

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

namespace Exe021
{
    public partial class FrmExe021 : Form
    {
        public FrmExe021()
        {
            InitializeComponent();
        }

        private void BtnVerificar_Click(object sender, EventArgs e)
        {
            //Chamada do método VerificarI (sem parametros e sem retorno)
            //No momento da digitação da linha abaixo, o sistema exibirá mensagem
            //de erro, informando que o método NÃO Existe, porém, exibe mensagem
            //sugerindo a criação para posterior implementação!!
            VerificarI();
        }

        private void BtnVerificarII_Click(object sender, EventArgs e)
        {
            //Chamada do método VerificarII (com parametro e sem retorno)
            VerificarII(Convert.ToInt32(TxtNumero.Text));
        }
    }
}
```

```
private void BtnVerificarIII_Click(object sender, EventArgs e)
{
    //Chamada do método VerificarIII (com parametro e com retorno).
    LblResultado.Text = VerificarIII(Convert.ToInt32(TxtNumero.Text));
}

private void BtnSair_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void TxtNumero_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsDigit(e.KeyChar) && e.KeyChar != 8)
    {
        e.Handled = true;
    }
}

private void BtnLimpar_Click(object sender, EventArgs e)
{
    LblResultado.Text = "";
    TxtNumero.Text = "";
    TxtNumero.Focus();
}

//Programação dos métodos

//Método SEM parametros e sem retorno (VOID)
private void VerificarI()
{
    int numero = Convert.ToInt16(TxtNumero.Text);
    int resto = 0;

    if (numero <= 1)
    {
        LblResultado.Text = "Número não é primo!!";
        return;
    }

    for (int divisor = 2; divisor < numero; divisor++)
    {
        //Devolve o resto da divisão
        resto = numero % divisor;

        if (resto == 0)
        {
            LblResultado.Text = numero.ToString() + " não é primo!!";
            return;
        }
    }

    LblResultado.Text = numero.ToString() + " é um número primo!!";
}

//Método que recebe um parametro e SEM retorno --> void
private void VerificarII(int numero)
{
    //O número já foi recebido pelo método por parametro
    //int numero = Convert.ToInt16(TxtNumero.Text);
}
```

```
int resto = 0;

if (numero <= 1)
{
    LblResultado.Text = "Número não é primo!!";
    return;
}

for (int divisor = 2; divisor < numero; divisor++)
{
    //Devolve o resto da divisão
    resto = numero % divisor;

    if (resto == 0)
    {
        LblResultado.Text = numero.ToString() + " não é primo!!";
        return;
    }
}

LblResultado.Text = numero.ToString() + " é um número primo!!";
}

//Método que recebe um parametro e devolve uma string
private string VerificarIII(int numero)
{
    //O número já foi recebido pelo método por parametro
    //int numero = Convert.ToInt16(TxtNumero.Text);

    int resto = 0;

    if (numero <= 1)
    {
        return "Número não é primo!!";
    }

    for (int divisor = 2; divisor < numero; divisor++)
    {
        //Devolve o resto da divisão
        resto = numero % divisor;

        if (resto == 0)
        {
            return numero.ToString() + " não é primo!!";
        }
    }

    return numero.ToString() + " é um número primo!!";
}
}
```

Prof. Roberto de Castro

Exercício proposto: Exe026B (Consulta Resultado - Componente Curricular): A partir da lista de resultados dos componentes: BD-III, Programação Web III e DTCC, elaborar projeto para receber o número de chamada do aluno (1 – 40) e exibir a menção do componente selecionado.

Considerações:

- O número de chamada deverá estar entre 1 e 10

- Tabela dos resultados:

Número de Chamada	BD III	Programação WEB III	DTCC
1	I	B	MB
2	R	B	MB
3	B	MB	B
4	I	MB	R
5	I	B	I
6	R	R	R
7	R	R	B
8	B	B	MB
9	MB	B	MB
10	B	R	B

- Para a solução poderão ser declarados três arrays com 10 elementos (um array para cada componente).

- Poderão ser declarados três métodos distintos, um para consulta BD III, outro para consulta Progr. Web III e outro para DTCC. (ou um único método

INTERFACE GRÁFICA

Exe026

08/08/2022 segunda-feira 10:41:32

CONSULTA RESULTADO

Número de Chamada

Componente Curricular

BD III
Prog. WEB III
DTCC

Consultar

Resultado

Limpar

Sair