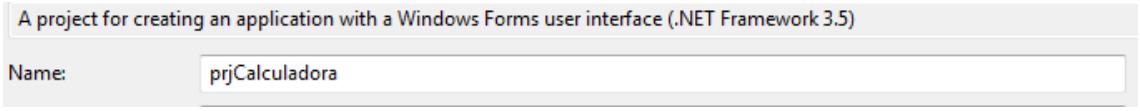
PROJETO DE CALCULADORA EM C#

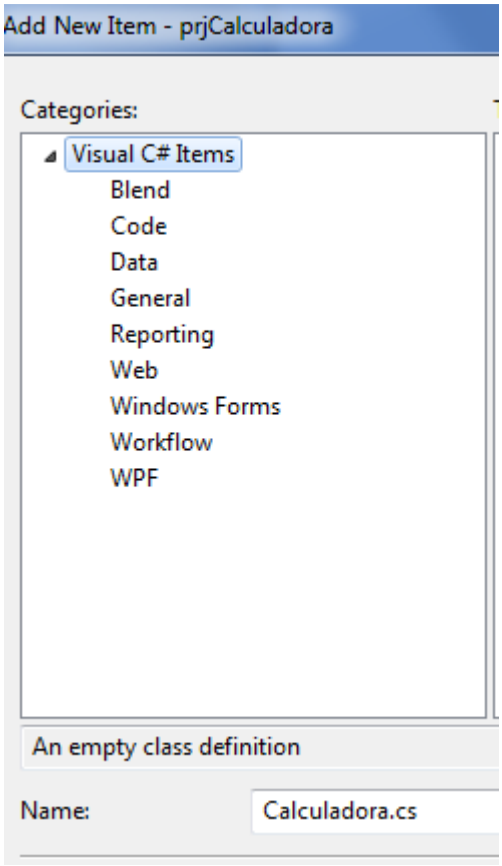
Crie um projeto em C# chamado $prjCalculadora$.



Crie no projeto uma classe chamada "Calculadora", ela seguirá o seguinte desenho de uml:

$Calculadora$
$- visor : String$ $- aux : double$ $- op : String$
$+ \mathbf{setDigito}(bt: Button): void$ $+ \mathbf{setOperador}(o: Button): void$ $+ \mathbf{calcular}(): void$

Obs.: o Diagrama não prevê o construtor e os getters e setters, já que são métodos padrões.



Vamos codificar a classe definindo as suas propriedades:

```
class Calculadora
{
    private String visor;
    private double aux;
    private String op;
}
```

Defina os encapsulamentos para cada propriedade, usando o *encapsulate fields*:

```
class Calculadora
{
    private String visor;

    public String Visor
    {
        get { return visor; }
        set { visor = value; }
    }
    private double aux;

    public double Aux
    {
        get { return aux; }
        set { aux = value; }
    }
    private String op;

    public String Op
    {
        get { return op; }
        set { op = value; }
    }
}
```

Crie o construtor abaixo do último encapsulamento, no caso o operador *Op*:

```
        public String Op
        {
            get { return op; }
            set { op = value; }
        }

        public Calculadora(String visor, double aux, String op) {
            this.visor = visor;
            this.aux = aux;
            this.op = op;

        }
```

Crie agora os seguintes métodos, logo abaixo do construtor. Eles serão responsáveis por preencher o visor, definir a operação a ser realizada e calcular o resultado ao ser pressionado o botão de igual.

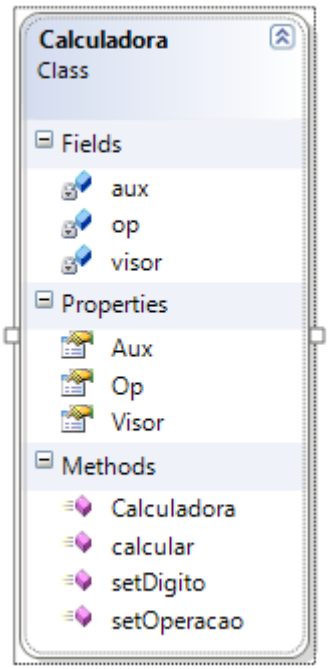
Como deveremos passar para a classe objetos da classe botão que pertencem a biblioteca do C# inclua a mesma no início das diretivas ***using***:

```
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace prjCalculadora
{
    class Calculadora
```

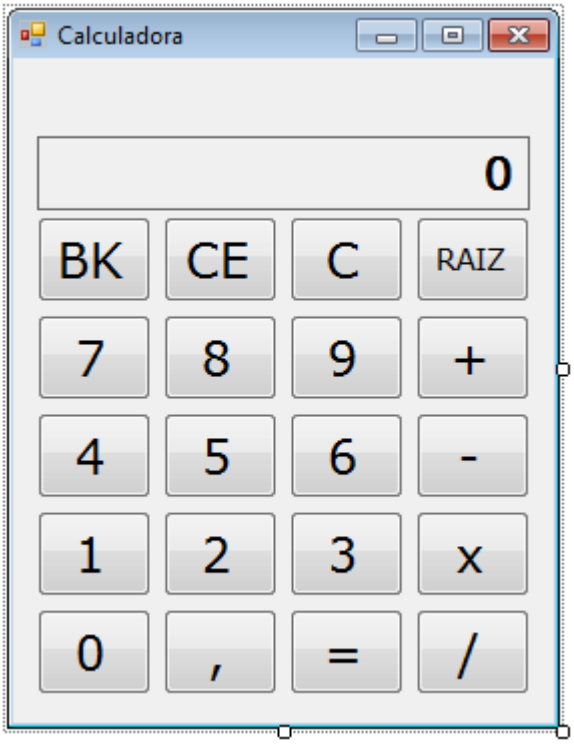
Agora codifique os demais métodos:

```
public void setDigito(Button bt) {
if (Visor.Equals("0"))Visor=bt.Text; // se visor vazio substitui o numero pelo digitado
else Visor+=bt.Text; // acrescenta digito a direita
}

public void setOperacao(Button btOperador) {
    Op = btOperador.Text; // joga  a operacao matematica
    aux = Convert.ToDouble(Visor); // salva o visor na variavel auxiliar
    Visor = "0"; // zera o visor
}

public void calcular() {

    double v = Convert.ToDouble(Visor); // pega o visor e converte em numero
    if (Op.Equals("+")) Visor = Convert.ToString(aux + v); // somar
    if (Op.Equals("-")) Visor = Convert.ToString(aux - v); // subtrair
    if (Op.Equals("x")) Visor = Convert.ToString(aux * v); // multiplicar
    if (Op.Equals("/")) Visor = Convert.ToString(v / aux); // dividir
}
```

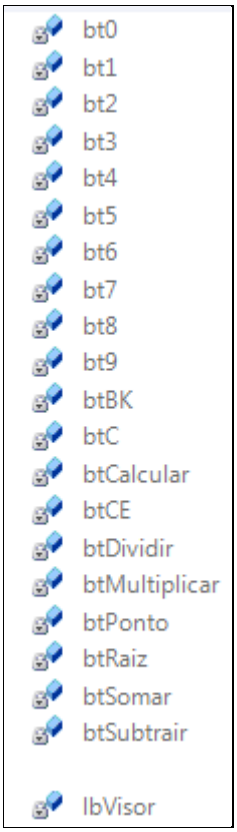
A estrutura final da nossa classe será conforme o diagrama abaixo:



Agora desenhe no formulário uma calculadora, conforme o layout abaixo:

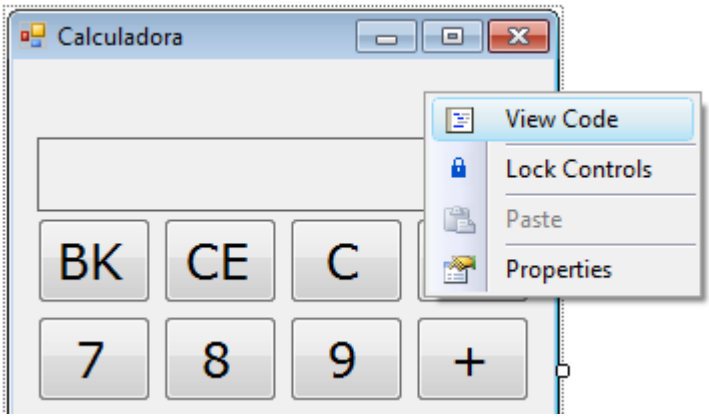


Os nomes dos componentes estão listados abaixo:



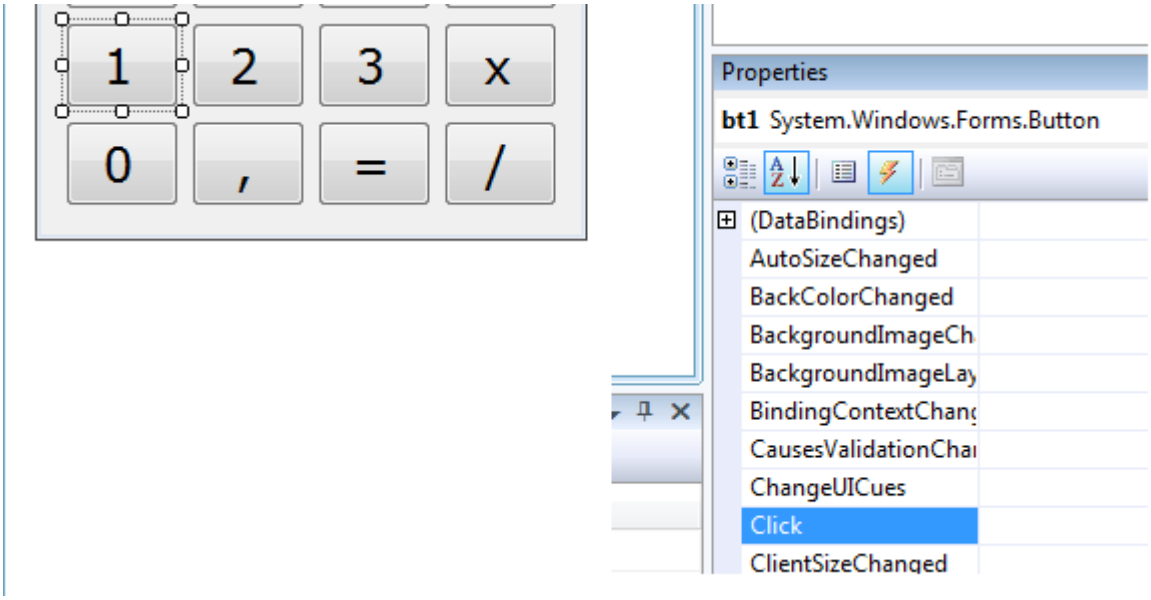
Nota: a inicial ***bt*** significa tipo **BUTTON** e inicial ***lb*** identifica tipo **LABEL**.

Crie um objeto da classe calculadora no formulário conforme abaixo, acessando o ***view code*** do formulário:




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace prjCalculadora
{
    public partial class Form1 : Form
    {
        // a linha abaixo inicial un objeto
        //calculadora no formulário com o nome de c
        Calculadora c = new Calculadora("0", 0, "");
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

Volte ao modo de design do formulário e crie os eventos de click para os botões numéricos, conforme o exemplo abaixo para o botão com o número “1”.



Ao gerar o evento codifique conforme o código abaixo:

```
namespace prjCalculadora
{
    public partial class Form1 : Form
    {
        // a linha abaixo inicial un objeto
        //calculadora no formulário com o nome de c
        Calculadora c = new Calculadora("0", 0, "");
        public Form1()
        {
            InitializeComponent();
        }

        private void bt1_Click(object sender, EventArgs e)
        {
            c.setDigito(bt1);
            lbVisor.Text = c.Visor;
        }
    }
}
```

Repita este processo para todos os botões numéricos, de 2 a 5:

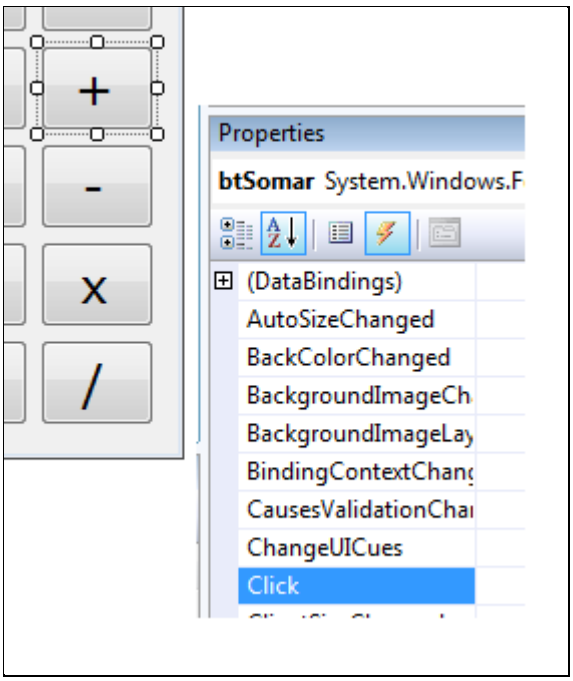
```
private void bt2_Click(object sender, EventArgs e)
{
    c.setDigito(bt2);
    lbVisor.Text = c.Visor;
}

private void bt3_Click(object sender, EventArgs e)
{
    c.setDigito(bt3);
    lbVisor.Text = c.Visor;
}

private void bt4_Click(object sender, EventArgs e)
{
    c.setDigito(bt4);
    lbVisor.Text = c.Visor;
}

private void bt5_Click(object sender, EventArgs e)
{
    c.setDigito(bt5);
    lbVisor.Text = c.Visor;
}
```

E também para os botões “0” e de 6 a 9:

```
private void bt0_Click(object sender, EventArgs e)
{
    c.setDigito(bt0);
    lbVisor.Text = c.Visor;
}
```

```
private void bt6_Click(object sender, EventArgs e)
{
    c.setDigito(bt6);
    lbVisor.Text = c.Visor;
}

private void bt7_Click(object sender, EventArgs e)
{
    c.setDigito(bt7);
    lbVisor.Text = c.Visor;
}

private void bt8_Click(object sender, EventArgs e)
{
    c.setDigito(bt8);
    lbVisor.Text = c.Visor;
}

private void bt9_Click(object sender, EventArgs e)
{
    c.setDigito(bt9);
    lbVisor.Text = c.Visor;
}
```

Agora codificaremos os botões de operação, começaremos com o botão de somar:



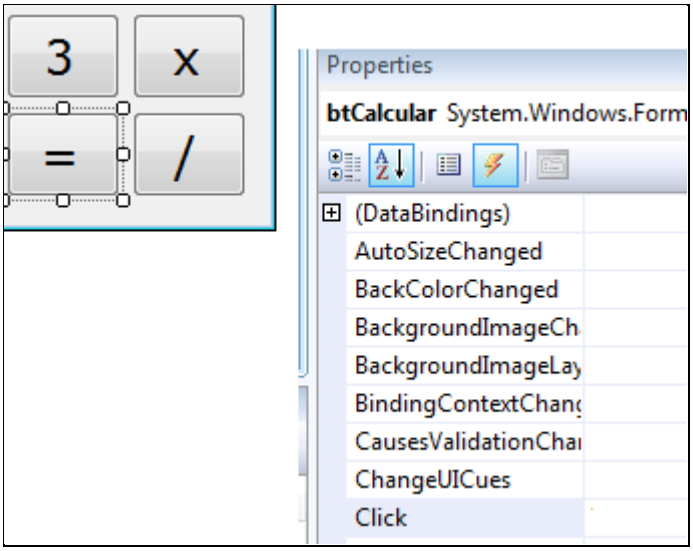
Codifique conforme as linhas abaixo:

```
private void btSomar_Click(object sender, EventArgs e)
{
    c.setOperacao(btSomar);
}
```

Faça o mesmo para os botões de subtrair, multiplicar e dividir.

```
private void btSubtrair_Click(object sender, EventArgs e)
{
    c.setOperacao(btSubtrair);
}

private void btMultiplicar_Click(object sender, EventArgs e)
{
    c.setOperacao(btMultiplicar);
}

private void btDividir_Click(object sender, EventArgs e)
{
    c.setOperacao(btDividir);
}
```

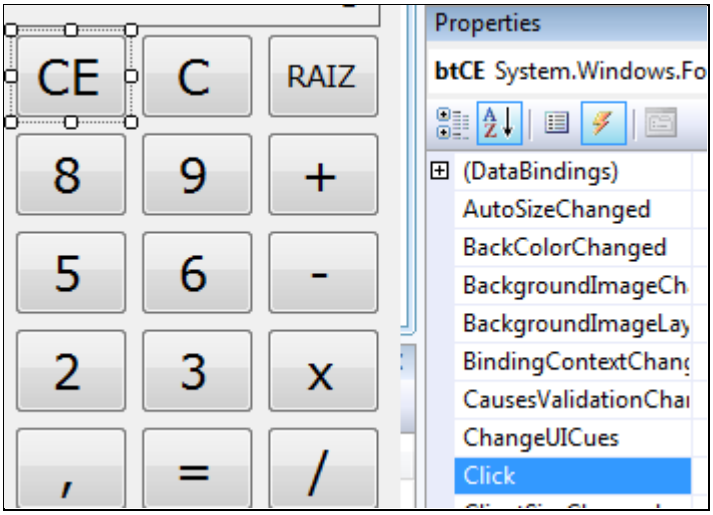
Programaremos agora o botão de calcular, ou seja, o botão com o símbolo de igual:



Insira o código abaixo:

```
private void btCalcular_Click(object sender, EventArgs e)
{
    c.calcular();
    lbVisor.Text = c.Visor;

}
```

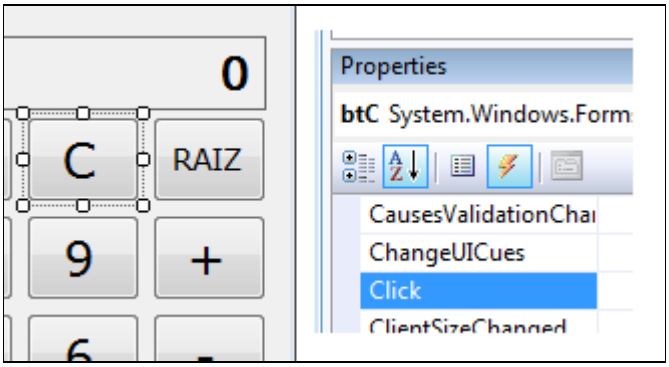
Codifique agora o botão CE, para a calculadora poder iniciar um novo cálculo sem a necessidade de fechar o formulário.



Codifique o botão conforme a seguir:

```
private void btCE_Click(object sender, EventArgs e)
{
    // recria o objeto pelo construtor
    c = new Calculadora("0", 0, "");
    // ajusta o visor com as informações
    lbVisor.Text = c.Visor;

}
```

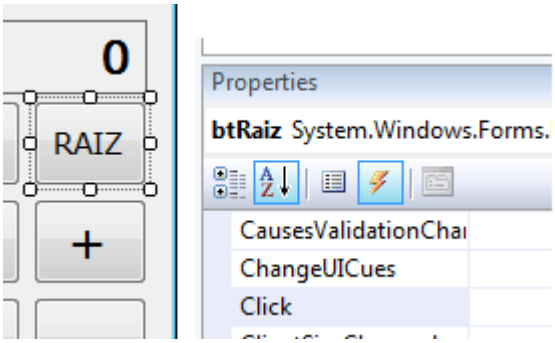
Faça o mesmo processo com o botão C, que ajustará apenas o valor do visor:



Codifique conforme a seguir:

```
private void btC_Click(object sender, EventArgs e)
{
    // ajusta o visor para conter 0
    c.Visor = "0";
    // ajusta o visor com as informações atualizadas
    lbVisor.Text = c.Visor;

}
```

O botão da raiz quadrada é o nosso próximo passo a codificar, programe o evento clique do botão conforme a seguir:



Codifique conforme a seguir:

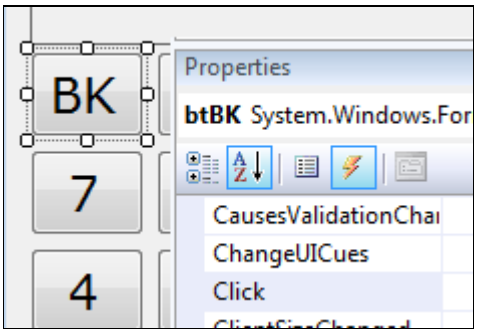
```
private void btRaiz_Click(object sender, EventArgs e)
{
    // cria uma variável e calcula a raiz usando
    // a biblioteca matemática
    double raiz = Math.Sqrt(Convert.ToDouble(c.Visor));
    // atualiza o visor com a resposta
    c.Visor = Convert.ToString(raiz);
    // ajusta o visor com as informações atualizadas
    lbVisor.Text = c.Visor;
}
```

O botão de *backspace* é mais complexo, para programa-lo iremos criar um novo método na **classe Calculadora**, conforme a seguir, logo após o método calcular.

```
public void calcular() {

    double v = Convert.ToDouble(Visor); // pega o visor e converte em numero
    if (Op.Equals("+")) Visor = Convert.ToString(aux + v); // somar
    if (Op.Equals("-")) Visor = Convert.ToString(aux - v); // subtrair
    if (Op.Equals("x")) Visor = Convert.ToString(aux * v); // multiplicar
    if (Op.Equals("/")) Visor = Convert.ToString(v / aux); // dividir
}
```

```
public void backspace() {
    // copia todos os caracteres do visor a partir da posição zero
    // menos o último caracater a direita
    String novo = Visor.Substring(0, Visor.Length - 1);
    // se novo ficar vazio preencha com zero
    if (novo.Equals("")) novo = "0";
    // ajusta o visor
    Visor = novo;
}
```

Programe agora no formulário o clique do botão *backspace*:



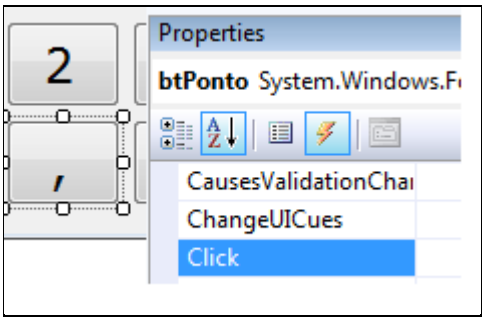
Codifique conforme a seguir:

```
private void btBK_Click(object sender, EventArgs e)
{
    c.backspace();
    lbVisor.Text = c.Visor;
}
```

Agora o botão ponto, que controla a digitação de uma decimal no visor. Também escreveremos um método para isso na classe Calculadora, abaixo do método ***backspace***:

```
public void backspace() {
    // copia todos os caracteres do visor a partir da posição zero
    // menos o último caracater a direita
    String novo = Visor.Substring(0, Visor.Length - 1);
    // se novo ficar vazio preencha com zero
    if (novo.Equals("")) novo = "0";
    // ajusta o visor
    Visor = novo;
}
public void pontoDecimal(){
    // se o visor não contém virgula decimal
    // então acrescente-a a direita no visor
    if (!Visor.Contains(",")) Visor += ",";
}
```

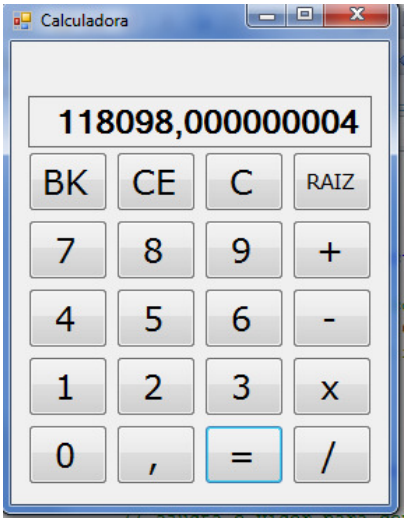
Programe agora o clique do botão de ponto:



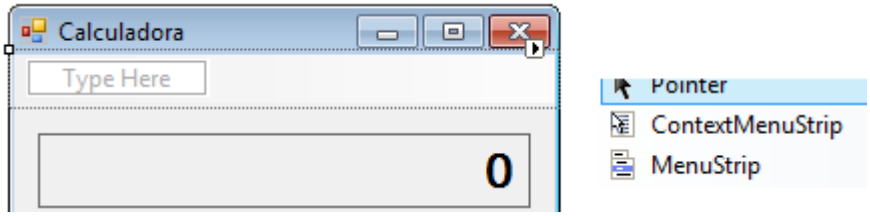
Codifique conforme a seguir:

```
private void btPonto_Click(object sender, EventArgs e)
{
    c.pontoDecimal();
    lbVisor.Text = c.Visor;
}
```

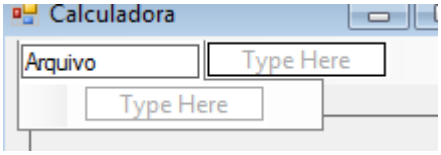
Agora faça um **build** do projeto e execute o projeto: Você terá um resultado conforme a seguir:



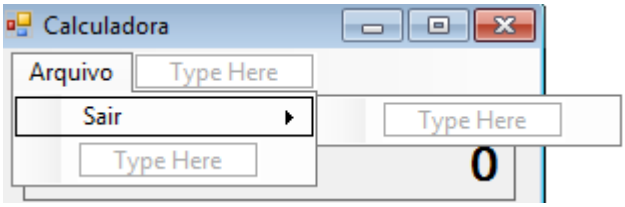
Vamos acrescentar um ***menu*** agora no sistema, arraste o componente *menuStrip* para o formulário:



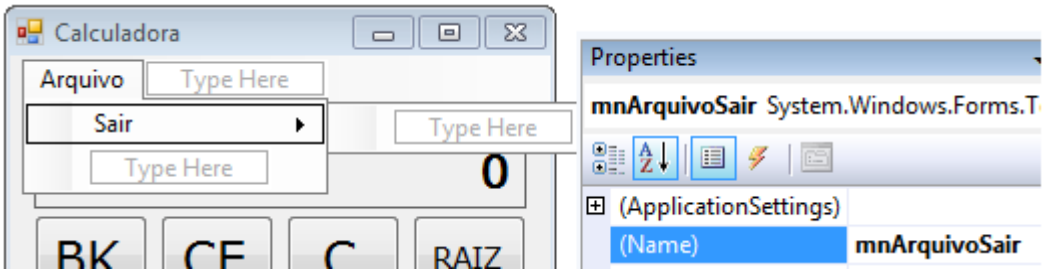
Digite o ***menu*** "Arquivo":



Abaixo a opção Sair:



Mantenha a opção "Sair" selecionada e em propriedades mude o nome do componente para "*mnArquivoSair*":



Programe o evento clique deste componente para fechar o programa ao sair:

```
private void mnArquivoSair_Click(object sender, EventArgs e)
{
    //fecha o formulário
    this.Close();
}
```

O resultado final será:

Para controlar o número de dígitos a serem exibidos na calculadora usaremos agora o método estático da classe String $Format$ no método calcular da classe Calculadora:

```
public void calcular() {

    double v = Convert.ToDouble(Visor); // pega o visor e converte em numero
    if (Op.Equals("+")) Visor = String.Format("{0:0.0000}",aux + v); // somar
    if (Op.Equals("-")) Visor = String.Format("{0:0.0000}",aux - v); // subtrair
    if (Op.Equals("x")) Visor = String.Format("{0:0.0000}", aux * v); // multiplicar
    if (Op.Equals("/")) Visor = String.Format("{0:0.0000}", v / aux); // dividir
}
```

No exemplo acima estamos forçando a exibir exatamente 04 casas decimais. A sintaxe é um pouco longa, mas de simples entendimento. Se você quiser flutuar o número de casas decimais até certo número de casas decimais então modifique o zero para #.

No Exemplo abaixo só serão exibido no máximo duas casas decimais.

```
String.Format("{0:0.##}", 123.4567);      // "123.46"
String.Format("{0:0.##}", 123.4);         // "123.4"
String.Format("{0:0.##}", 123.0);         // "123"
```

Se quiser controlar o número de casas decimais antes da vírgula (esquerda), acrescente n zeros que achar necessário como no exemplo a seguir:

```
String.Format("{0:00.0}", 123.4567);      // "123.5"
String.Format("{0:00.0}", 23.4567);       // "23.5"
String.Format("{0:00.0}", 3.4567);        // "03.5"
String.Format("{0:00.0}", -3.4567);       // "-03.5"
```

Caso ache interessante exibir o separador de milhar utilize então a sintaxe conforme mostrado no próximo exemplo:

```
String.Format("{0:0,0.0}", 12345.67);     // "12,345.7"
String.Format("{0:0,0}", 12345.67);       // "12,346"
```

Note que o formatador usa a notação inglesa, porém o resultado visual na caixa de texto obedecerá à língua definida na configuração do Windows.

Para alinhar números use número negativo de espaços para alinhar o lado esquerdo e positivo para alinhar a direita, conforme o exemplo abaixo:

```
String.Format("{0,10:0.0}", 123.4567);    // "     123.5"
String.Format("{0,-10:0.0}", 123.4567);   // "123.5     "
String.Format("{0,10:0.0}", -123.4567);   // "    -123.5"
String.Format("{0,-10:0.0}", -123.4567);  // "-123.5    "
```

A seguir formatadores que podem ser usados em datas:

```
DateTime dt = new DateTime(2008, 3, 9, 16, 5, 7, 123);

String.Format("{0:y yy yyy yyyy}", dt);  // "8 08 008 2008"   year
String.Format("{0:M MM MMM MMMM}", dt);  // "3 03 Mar March"  month
String.Format("{0:d dd ddd dddd}", dt);  // "9 09 Sun Sunday" day
String.Format("{0:h hh H HH}",     dt);  // "4 04 16 16"      hour 12/24
String.Format("{0:m mm}",          dt);  // "5 05"            minute
String.Format("{0:s ss}",          dt);  // "7 07"            second
String.Format("{0:f ff fff ffff}", dt);  // "1 12 123 1230"   sec.fraction
String.Format("{0:F FF FFF FFFF}", dt);  // "1 12 123 123"    without zeroes
String.Format("{0:t tt}",          dt);  // "P PM"            A.M. or P.M.
String.Format("{0:z zz zzz}",      dt);  // "-6 -06 -06:00"   time zone
```

```
// month/day numbers without/with leading zeroes
String.Format("{0:M/d/yyyy}", dt);            // "3/9/2008"
String.Format("{0:MM/dd/yyyy}", dt);          // "03/09/2008"

// day/month names
String.Format("{0:ddd, MMM d, yyyy}", dt);    // "Sun, Mar 9, 2008"
String.Format("{0:dddd, MMMM d, yyyy}", dt);  // "Sunday, March 9, 2008"

// two/four digit year
String.Format("{0:MM/dd/yy}", dt);            // "03/09/08"
String.Format("{0:MM/dd/yyyy}", dt);          // "03/09/2008"
```

```
String.Format("{0:t}", dt);  // "4:05 PM"                         ShortTime
String.Format("{0:d}", dt);  // "3/9/2008"                        ShortDate
String.Format("{0:T}", dt);  // "4:05:07 PM"                      LongTime
String.Format("{0:D}", dt);  // "Sunday, March 09, 2008"          LongDate
String.Format("{0:f}", dt);  // "Sunday, March 09, 2008 4:05 PM"  LongDate+ShortTime
String.Format("{0:F}", dt);  // "Sunday, March 09, 2008 4:05:07 PM" FullDateTime
String.Format("{0:g}", dt);  // "3/9/2008 4:05 PM"                ShortDate+ShortTime
String.Format("{0:G}", dt);  // "3/9/2008 4:05:07 PM"             ShortDate+LongTime
String.Format("{0:m}", dt);  // "March 09"                        MonthDay
String.Format("{0:y}", dt);  // "March, 2008"                     YearMonth
String.Format("{0:r}", dt);  // "Sun, 09 Mar 2008 16:05:07 GMT"   RFC1123
String.Format("{0:s}", dt);  // "2008-03-09T16:05:07"             SortableDateTime
String.Format("{0:u}", dt);  // "2008-03-09 16:05:07Z"            UniversalSortableDat
```

Métodos da classe String – detalhes de sintaxe e uso

O tipo string apesar de se comportar como tipo primitivo é na verdade uma classe em C#. A classe string possui uma série de métodos estáticos e não estáticos, que são utilizamos para formatação, concatenação, desmembramento, ***substring***, etc. Vamos analisar alguns destes métodos da classe string.

Substring

objeto.Substring(startIndex, Length);

Parâmetros:

startIndex : A posição inicial na string, lembrando que a primeira letra começa na posição zero.

Length: O número de letras a serem copiadas na string.

O valor de retorno é vazio caso a String tenha posição maior que o tamanho da String ou tamanho seja zero.

O método *substring* tem como parâmetro a posição inicial que queremos obter e quantos caracteres devem ser extraídos. Caso não seja informado o número de caracteres a ser extraído, a função retornará o restante da string a partir da posição inicial informada.

IndexOf

O método *IndexOf* é utilizado para localizar uma determinada palavra dentro da string. Este método retornará a posição da string desejada. Caso a string não seja encontrada, será retornado o valor -1.

```
string nome = "Testando da Silva";
int pos = nome.IndexOf("Silva");
//A partir do índice 5
int pos2 = nome.IndexOf("Silva", 5);
```

ToUpper e ToLower

As funções ToUpper e ToLower permitem colocar uma string em letra minúsculas ou maiúsculas, conforme o exemplo a seguir.

```
string nome = "Maurício";

nome = nome.ToUpper();
nome = nome.ToLower();
```

TrimStart, TrimEnd e Trim

As funções de Trim servem para remover espaços em branco das strings. A função TrimStart remove os espaços em branco do início da string, já a função TrimEnd remove os espaços em branco do final da string. A função Trim remove os espaços em branco do inicio e do fim da string.

```
string nome = "   MAURICIO    ";

nome = nome.TrimEnd();
nome = nome.TrimStart();

nome = nome.Trim();
```

PadLeft e PadRight

As funções PadLeft e PadRight servem para preencher uma string a esquerda ou a direita com um caracter especificado. Os exemplos a seguir mostra o uso das funções PadLeft e PadRight.

```
string nome = "Mauricio";

nome = nome.PadRight(10, ' ');
// "Mauricio  "

string codigo = "123";
codigo = codigo.PadLeft(6, '0');
// "000123"
```

Join e Split- método estático

A função Split serve para quebrar uma string em um array de strings de acordo com um separador. Ao contrário da função split, a função Join concatena um array de string inserindo um separador.

```
string linha = "Teste, 10, 20, 10/06/2007";

string[] campos = linha.Split(',');

string linhaNova = String.Join(';', campos);
```

Retorna o tamanho da String.

Exercícios

1. Codifique e crie os seguintes botões na calculadora:
    a. 1/x

b. +/-
c. Porcentagem (%)
d. Seno – considerar valor sempre em angulos
e. Cosseno - considerar valor sempre em angulos
f. Tangente - considerar valor sempre em angulos

Nota: escrever métodos na classe calculadora para realizar as ações de botões, reescreva o método da raiz quadrada para que a mesma seja feita também pela classe.

2. Bloquear a digitação de mais de 10 digitos decimais na calculadora e alterar a exbição do calcular para usar também 10 dígitos, para isso será necessário adicionar código ao método **setDigito da classe Calculadora.**