Botões

Podemos inserir botões em um Frame através da classe **JButton** do Java. Essa classe disponibiliza vários métodos para a configuração de como o botão será apresentado, onde podemos inserir um texto e/ou um ícone dentro do botão. Note que esse texto ficará fixo na interface e não será possível a sua modificação durante a execução da aplicação.

Para cada botão do Frame, incluiremos um novo objeto que represente a classe **JButton**. A sintaxe de criação de um objeto da classe **JButton** é ilustrada abaixo:

Sintaxe em Java
JButton <botão> = new JButton(<texto>); JButton <botão> = new JButton(<texto>, <ícone>);

Para cada novo ícone a ser exibido, incluiremos um novo objeto que represente a classe **ImageIcon** que permite o uso de imagens em GUI's. A sintaxe de criação de um objeto da classe **ImageIcon** é ilustrada abaixo:

Sintaxe em Java
ImageIcon <imagem> = new ImageIcon(<caminho e nome da imagem>);

O caminho das pastas onde se encontra o arquivo de imagem deve ser especificado a partir do disco e a cada pasta deve ser colocado a barra normal (/) e não a barra invertida (\). Ex: "**i:/PI-IHC/tv1.gif**".

A tabela abaixo demonstra os principais métodos da classe **JButton**:

Métodos de JButton	
Método	**Descrição**
JButton()	Permite a criação de um botão vazio, sem um texto em seu interior.
JButton(<texto>)	Permite a criação de um botão com o texto especificado.
JButton(<texto>,<ícone>)	Permite a criação de um botão com o texto e ícone especificados.
varString = getLabel()	Permite armazenarmos em uma variável de memória do tipo String o texto definido no botão.
setLabel(<texto>)	Permite atribuir um novo texto ao botão.
setEnabled(true/false)	Permite habilitarmos (true) ou desabilitarmos (false) o botão.
setToolTipText(<descrição>)	Permite definirmos uma mensagem de descrição da ação que o botão executará quando o mesmo for clicado. Essa descrição é apresentada quando o usuário para o ponteiro do mouse sobre o botão.

setMneumonic(KeyEvent.VK_<tecla de atalho>)	Permite definirmos uma tecla de atalho que será pressionada juntamente com a tecla ALT para possibilitar o acionamento do botão pelo teclado. A letra do texto do botão igual à tecla definida ficará sublinhada no texto do botão.
setHorizontalTextPosition(AbstractButton.<alinhamentoHorizontal>)	Permite definirmos o alinhamento horizontal que será aplicado ao texto do botão, em relação ao ícone associado ao botão. Só utilizamos esse método quando temos um botão com ícone e texto. Definimos LEFT para alinhamento do texto à esquerda do ícone do botão e RIGHT para alinhamento do texto à direita do ícone do botão.
setVerticalTextPosition(AbstractButton.<alinhamentoVertical>)	Permite definirmos o alinhamento vertical que será aplicado ao texto do botão, em relação ao ícone associado ao botão. Só utilizamos esse método quando temos um botão com ícone e texto. Definimos TOP para alinhamento do texto acima do ícone do botão e BOTTOM para alinhamento do texto abaixo do ícone do botão.
setBackground(Color.<cor> ou setBackground(new Color(<red>,<green>,<blue>)	Permite definirmos a cor de fundo do botão conforme a cor desejada. As cores disponíveis dever ser escritas com o seu nome em inglês. Exemplos: Color.white; Color.black; Color.blue; Color.cyan; Color.darkGray; Color.gray; Color.green; Color.lightGray; Color.magenta; Color.orange; Color.pink; Color.red; Color.yellow; Podemos definir uma cor personalizada indicando a composição das tonalidades de vermelho (red), verde (green) e azul (blue) com a classe Color(), onde cada tonalidade varia entre 0 e 255.
setForeground(Color.<cor> ou setForeground (new Color(<red>,<green>,<blue>)	Permite definirmos a cor da fonte do texto do botão conforme a cor desejada. As cores disponíveis dever ser escritas com o seu nome em inglês. Exemplos: Color.white; Color.black; Color.blue; Color.cyan; Color.darkGray; Color.gray; Color.green; Color.lightGray; Color.magenta; Color.orange; Color.pink; Color.red; Color.yellow; Podemos definir uma cor personalizada indicando a composição das tonalidades de vermelho (red), verde (green) e azul (blue) com a classe Color(), onde cada tonalidade varia entre 0 e 255.
setFont(new Font(<fonte>, <estilo>, <tamanho>)	Permite escolhermos a fonte a ser utilizada no texto do botão. Especificamos o nome da fonte no parâmetro <fonte>, o estilo da fonte (BOLD para negrito, ITALIC para itálico ou PLAIN

	para normal) no parâmetro <estilo> e o tamanho da fonte no parâmetro <tamanho>.
addActionListener(this)	Define que o botão terá um tratamento de eventos para possibilitar a execução de suas atividades quando o usuário clicar no mesmo, onde definimos os comandos associados ao botão no método **actionPerformed().**

O programa abaixo ilustra o nosso terceiro exemplo de codificação, o qual constrói uma GUI com os métodos descritos anteriormente. Crie a classe **Exemplo3_Botao** dentro do NetBeans para executarmos o mesmo.

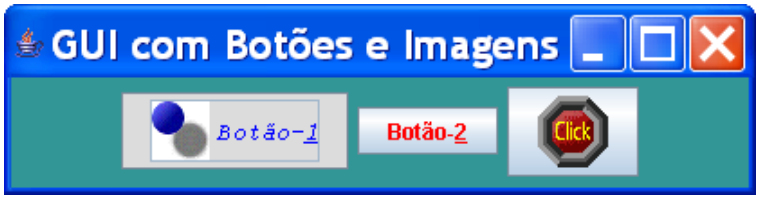
Exemplo3_Botao.java

```
// importa o pacote awt
import java.awt.*;
// importa os eventos do awt
import java.awt.event.*;
// importa o pacote swing
import javax.swing.*;

// a classe Exemplo3_Botao herda as características da classe JFrame
// e implementa uma interface para os eventos de ActionListener
// para verificar qual botão foi pressionado e para determinar
// as ações (comandos) de cada botão.
public class Exemplo3_Botao extends JFrame implements ActionListener{
   // cria os atributos para os botões, que serão objetos no método construtor
   JButton botao1,botao2, botao3;

   // cria os objetos com os ícones a serem apresentados nos botões
   ImageIcon icone1 = new ImageIcon("i:/PI-IHC/botao1.gif");
   ImageIcon icone2 = new ImageIcon("i:/PI-IHC/botao2.gif");
   ImageIcon icone3 = new ImageIcon("i:/PI-IHC/botao3.gif");

   // método construtor da classe que define as características da GUI
   public Exemplo3_Botao(){
      // ajusta o título da janela
      setTitle("GUI com Botões e Imagens");
      // ajusta o tamanho da janela (largura e altura)
      setSize(300,90);
      // define a posição da janela (horizontal e vertical) do canto superior esquerdo
      setLocation(200,200);
      // ajusta a cor de fundo da janela
      getContentPane().setBackground(new Color(50,150,150));
      // cria o objeto botao1 com um texto e um ícone
      botao1 = new JButton("Botão-1",icone1);
      // ajusta o alinhamento horizontal do texto para à direita
      botao1.setHorizontalTextPosition(AbstractButton.RIGHT);
      // ajusta o alinhamento vertical do texto para centralizado
      botao1.setVerticalTextPosition(AbstractButton.CENTER);
      // ajusta a cor de fundo do botão1
      botao1.setBackground(new Color(220,220,220));
```

```
// ajusta a cor da fonte do título do botão1
botao1.setForeground(Color.blue);
// ajusta a fonte do título do botão1
botao1.setFont(new Font("Courier",Font.ITALIC,14));
// ajusta o botão1 para ficar visível
botao1.setEnabled(true);
// adiciona um evento ao botão1, o qual será tratado nesta classe (this)
botao1.addActionListener(this);
// ajusta o texto de descrição do botão
botao1.setToolTipText("Clique aqui para executar o botão1");
// ajusta a tecla de atalho para o botão1 (ALT-1)
botao1.setMnemonic(KeyEvent.VK_1);

// cria o objeto botao2 com um texto
botao2 = new JButton("Botão-2");
// ajusta o alinhamento horizontal do texto para centralizado
botao2.setHorizontalTextPosition(AbstractButton.CENTER);
// ajusta o botão2 para ficar alinhado abaixo
botao2.setVerticalTextPosition(AbstractButton.BOTTOM);
// ajusta a cor da fonte do título do botão2
botao2.setForeground(Color.red);
// ajusta a fonte do título do botão2
botao2.setFont(new Font("Times",Font.BOLD,12));
// ajusta o botão2 para ficar visível
botao2.setEnabled(true);
// adiciona um evento ao botão2, o qual será tratado nesta classe (this)
botao2.addActionListener(this);
// ajusta o texto de descrição do botão
botao2.setToolTipText("Clique aqui para executar o botão2");
// ajusta a tecla de atalho para o botão1 (ALT-2)
botao2.setMnemonic(KeyEvent.VK_2);

// cria o objeto botao3 com um ícone
botao3 = new JButton(icone3);
// ajusta o alinhamento horizontal do texto para à esquerda
botao3.setEnabled(true);
// adiciona um evento ao botão3, o qual será tratado nesta classe (this)
botao3.addActionListener(this);
// ajusta o texto de descrição do botão
botao3.setToolTipText("Clique aqui para executar o botão3");
// ajusta a tecla de atalho para o botão1 (ALT-3)
botao3.setMnemonic(KeyEvent.VK_3);

// ajusta o layout da tela para um objeto ao lado do outro, da esquerda para
// direita e de cima para baixo
setLayout(new FlowLayout());
// adiciona o botão1 no frame
add(botao1);
// adiciona o botão2 no frame
add(botao2);
```

```
        // adiciona o botão3 no frame
        add(botao3);
    } // fim do método construtor da classe

    // método que implementa uma interface para os eventos da janela
    // para verificar qual botão foi pressionado e para determinar
    // as ações (comandos) de cada botão.
    public void actionPerformed(ActionEvent evento){

        // verifica se o botão1 foi pressionado e faz a ação correspondente
        if (evento.getSource()==botao1)
        {
            System.out.println("O botão 1 foi pressionado!");
        }
        // verifica se o botão2 foi pressionado e faz a ação correspondente
        if (evento.getSource()==botao2)
        {
            System.out.println("O botão 2 foi pressionado!");
        }
        // verifica se o botão3 foi pressionado e faz a ação correspondente
        if (evento.getSource()==botao3)
        {
            System.out.println("O botão 3 foi pressionado!");
        }
    } // fim do método que implementa a interface para eventos da janela

    // método principal da classe Exemplo3_Botao
    public static void main(String args[]){

        // cria o objeto GUI que conterá uma janela
        JFrame GUI = new Exemplo3_Botao();

        // exibe a janela na tela.
        GUI.setVisible(true);

        // define a ação do botão fechar
        GUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    } // fim do método principal
} // fim da classe
```

A saída do programa acima é exibida abaixo:



Caixas de Texto

Podemos inserir caixas de texto em um Frame através da classe **JTextField** do Java. Essa classe disponibiliza vários métodos para a configuração de como a caixa de texto será apresentada, onde podemos inserir um texto, bem como definir o tamanho da caixa de texto (em caracteres). Note que o usuário poderá digitar qualquer texto dentro da caixa de texto, sendo possível a sua modificação durante a execução da aplicação.

Para cada caixa de texto do Frame, incluiremos um novo objeto que represente a classe **JTextField**. A sintaxe de criação de um objeto da classe **JTextField** é ilustrada abaixo:

Sintaxe em Java
JTextField <caixa de texto> = new JTextField(); JTextField <caixa de texto> = new JTextField (<texto>); JTextField <caixa de texto> = new JTextField (<tamanho>); JTextField <caixa de texto> = new JTextField (<texto>,<tamanho>);

A tabela abaixo demonstra os principais métodos da classe **JTextField**:

Métodos de JTextField	
Método	**Descrição**
JTextField()	Permite a criação de uma caixa de texto vazia, sem um texto em seu interior.
JTextField(<texto>)	Permite a criação de uma caixa de texto com o com o texto especificado.
JTextField(<tamanho>)	Permite a criação de uma caixa de texto com o tamanho de caracteres especificado.
JTextField(<texto>,<tamanho>)	Permite a criação de uma caixa de texto com o texto e a caixa com o tamanho de caracteres especificados.
varString = getText()	Permite armazenarmos em uma variável de memória do tipo String o texto armazenado na caixa de texto.
setText(<texto>)	Permite atribuir um novo texto à caixa de texto.
varString = getSelectedText()	Permite armazenarmos em uma variável de memória do tipo String o texto selecionado na caixa de texto.
setEditable(true/false)	Permite definirmos se a caixa de texto permitirá a edição de seu texto (true) ou se não permitirá tal ação (false).
varBoolean = isEditable()	Permite armazenarmos em uma variável de memória do tipo boolean a informação indicando se a caixa de texto permite a edição de seu texto (true) ou se não permite tal ação (false).
selectAll()	Permite selecionarmos todo o texto contido na caixa de texto.

addActionListener(this)	Define que a caixa de texto terá um tratamento de eventos para possibilitar a execução de suas atividades quando o usuário pressionar a tecla enter na mesma, onde definimos os comandos associados à caixa de texto no método **actionPerformed().**

O programa abaixo ilustra o nosso quarto exemplo de codificação, o qual constrói uma GUI com os métodos descritos anteriormente. Crie a classe **Exemplo4_CaixadeTexto** dentro do NetBeans para executarmos o mesmo.

Exemplo4_CaixadeTexto.java

```
// importa o pacote awt
import java.awt.*;
// importa os eventos do awt
import java.awt.event.*;
// importa o pacote swing
import javax.swing.*;

// a classe Exemplo4_CaixadeTexto herda as características da classe JFrame
// e implementa uma interface para os eventos de ActionListener
// para verificar qual botão foi pressionado para determinar
// as ações (comandos) de cada botão.
class Exemplo4_CaixadeTexto extends JFrame implements ActionListener{

   // define os atributos para criar posteriormente os objetos gráficos da GUI
   JLabel LNum1,LNum2,LResultado;
   JButton BSoma, BSubtrai, BMultiplica, BDivide, BLimpa;
   JTextField TNum1,TNum2,TResultado;

   // método principal da classe Exemplo4_CaixadeTexto
   public static void main(String args[]){

      // cria o objeto GUI que conterá uma janela
      JFrame GUI = new Exemplo4_CaixadeTexto();

      // exibe a janela na tela.
      GUI.setVisible(true);

      // define a ação do botão fechar
      GUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

   } // fim do método principal

   // método construtor da classe que define as características da GUI
   public Exemplo4_CaixadeTexto(){

      // ajusta o título da janela
      setTitle("GUI de Calculadora");
      // ajusta o tamanho da janela (largura e altura)
```

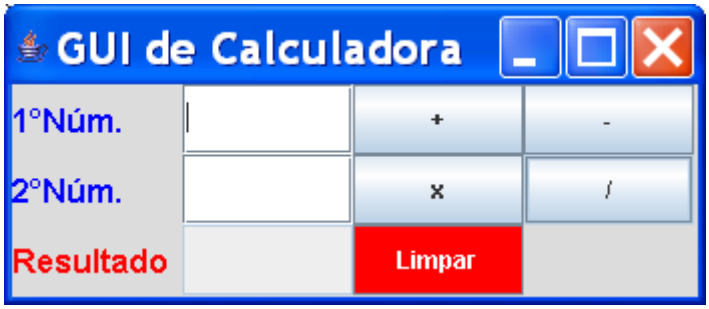
```
setSize(350,150);
// define a posição da janela (horizontal e vertical) do canto superior esquerdo
setLocation(200,200);
// ajusta a cor de fundo da janela
getContentPane().setBackground(new Color(220,220,220));
// ajusta o layout da tela para uma matriz de 3 linhas e 4 colunas
setLayout(new GridLayout(3,4));

// cria o objeto de rótulo LNum1 com o texto indicado
LNum1 = new JLabel("1°Núm.");
// ajusta a cor da fonte do rótulo LNum1
LNum1.setForeground(Color.blue);
// ajusta a fonte do rótulo LNum1
LNum1.setFont(new Font("",Font.BOLD,16));

// cria o objeto de rótulo LNum2 com o texto indicado
LNum2 = new JLabel("2°Núm.");
// ajusta a cor da fonte do rótulo LNum2
LNum2.setForeground(Color.blue);
// ajusta a fonte do rótulo LNum2
LNum2.setFont(new Font("",Font.BOLD,16));

// cria o objeto de rótulo LResultado com o texto indicado
LResultado = new JLabel("Resultado");
// ajusta a fonte do rótulo LResultado
LResultado.setFont(new Font("",Font.BOLD,16));
// ajusta a cor da fonte do rótulo LResultado
LResultado.setForeground(Color.red);

// cria o objeto de botão BSoma
BSoma = new JButton ("+");
// define que o botão BSoma terá o tratamento de eventos.
BSoma.addActionListener(this);

// cria o objeto de botão BSubtrai
BSubtrai = new JButton ("-");
// define que o botão BSubtrai terá o tratamento de eventos.
BSubtrai.addActionListener(this);

// cria o objeto de botão BMultiplica
BMultiplica = new JButton ("x");
// define que o botão BMultiplica terá o tratamento de eventos.
BMultiplica.addActionListener(this);

// cria o objeto de botão BDivide
BDivide = new JButton ("/");
// define que o botão BDivide terá o tratamento de eventos.
BDivide.addActionListener(this);

// cria o objeto de botão BLimpa
```

```
        BLimpa = new JButton ("Limpar");
        // define que o botão BLimpa terá o tratamento de eventos.
        BLimpa.addActionListener(this);
        // ajusta a cor de fundo do botão BLimpa
        BLimpa.setBackground(Color.red);
        // ajust a cor da fonte do botão BLimpa
        BLimpa.setForeground(Color.white);

        // cria o objeto de caixa de texto TNum1
        TNum1 = new JTextField();

        // cria o objeto de caixa de texto TNum2
        TNum2 = new JTextField();

        // cria o objeto de caixa de texto TResultado
        TResultado = new JTextField();
        // desabilita a edição da caixa de texto TResultado
        TResultado.setEditable(false);
        // adiciona o rótulo LNum1 ao Frame
        add(LNum1);
        // adiciona a caixa de texto TNum1 ao Frame
        add(TNum1);
        // adiciona o botão BSoma ao Frame
        add(BSoma);
        // adiciona o botâo BSubtrai ao Frame
        add(BSubtrai);
        // adiciona o rótulo LNum2 ao Frame
        add(LNum2);
        // adiciona a caixa de texto TNum2 ao Frame
        add(TNum2);
        // adiciona o botão BMultiplica ao Frame
        add(BMultiplica);
        // adiciona o botão BDivide ao Frame
        add(BDivide);
        // adiciona o rótulo LResultado ao Frame
        add(LResultado);
        // adiciona a caixa de texto TResultado ao Frame
        add(TResultado);
        // adiciona o botão BLimpa ao Frame
        add(BLimpa);
    } // fim do método construtor da classe

    // método que implementa uma interface para os eventos da janela
    // para verificar qual botão foi pressionado e para determinar
    // as ações (comandos) de cada botão.
    public void actionPerformed(ActionEvent evento){

        // verifica se o botão BLimpa foi clicado e limpa as caixas de texto
        if (evento.getSource()==BLimpa)
        {
```

```
        // limpa a caixa de textoTNum1
        TNum1.setText("");
        // limpa a caixa de textoTNum2
        TNum2.setText("");
        // limpa a caixa de textoTResultado
        TResultado.setText("");
        // retorna para o método principal (main)
        return;
    }
    // cria as variáveis internas para efetuar os cálculos
    float num1=0,num2=0,resultado=0;
    // o bloco try possibilita o tratamento de alguma exceção (erro)
    // que possa ocorrer nos comandos contidos neste bloco, como tentar
    // converter os números nas caixas de texto e não houver um valor
    // numérico preenchido.
    try
    {
        // atribui à varíavel num1 o valor armazenado na caixa de texto
        // TNum1, mas faz a conversão de String (cadeia) para float (real)
        // antes de fazer a atribuição
        num1 = Float.parseFloat(TNum1.getText());
        // atribui à varíavel num2 o valor armazenado na caixa de texto
        // TNum2, mas faz a conversão de String (cadeia) para float (real)
        // antes de fazer a atribuição
        num2 = Float.parseFloat(TNum2.getText());
    } // fim do bloco try

    // tratamento da exceção (erro) de conversão inválida de números.
    catch (NumberFormatException erro)
    {

        // apresenta na caixa de texto TResultado a mensagem de erro
        TResultado.setText("Erro");
        // retorna ao método principal (main)
        return;
    } // fim do bloco catch

    // verifica se o botão BSoma foi clicado e faz a operação da Soma
    if (evento.getSource()==BSoma)
        resultado = num1 + num2;
    // verifica se o botão BSubtrai foi clicado e faz a operação da Subtração
    if (evento.getSource()==BSubtrai)
        resultado = num1 - num2;
    // verifica se o botão BMultiplica foi clicado e faz a operação de Multiplicação
    if (evento.getSource()==BMultiplica)
        resultado = num1 * num2;
    // verifica se o botão BDivide foi clicado e faz a operação de Divisão
    if (evento.getSource()==BDivide)
        resultado = num1 / num2;
```

```
        // Apresenta o valor da variável resultado dentro da caixa de texto
        // TResultado, mas faz a conversão de float (real) para String (cadeia)
        // antes de apresentar o valor.
        TResultado.setText(String.valueOf(resultado));

    } // fim do método que implementa a interface para eventos da janela
} // fim da classe
```

A saída do programa acima é exibida abaixo:



Botões no HTML

Podemos inserir botões no documento HTML com o uso de duas *tags*, a *tag* **<input type="button">** ou a *tag* **<button>**.

As propriedades da *tag* **<input type="button">** são:

- "**name**": indica o nome do botão;
- "**value**": indica o texto de conteúdo do botão.

Não é necessário usar a *tag* de fechamento </input>, já que todas as informações do botão são definidas nas propriedades em sua *tag* de abertura.

***tag* input type="button"**

```
<input type="button" name="botaoOK" value="OK" onclick="alert('OK')">
```

As propriedades da *tag* **<button>** são:

- "**type**": pode possuir os valores "***submit***" (envio de dados do formulário), "***reset***" (limpar os campos do formulário) ou "***button***" (indica que é um botão com script);
- "**name**": indica o nome do botão;

A propriedade "**onclick**" pode ser utilizada em ambas as *tags* de botões e nela podemos especificar um script para ser executado quando o botão for clicado.

***tag* button**

```
<button type="button" name="botaoGrava" onclick="alert('Gravar')">
Gravar </button>
```

O programa abaixo ilustra o nosso quarto exemplo de codificação HTML, o qual constrói uma página de web com botões. Crie o arquivo **Exemplo04.html** dentro de um editor de textos sem formatação (bloco de notas), posteriormente, abra o mesmo no navegador de web.

Exemplo04.html

```
<html>
  <head>
    <title> Exemplo 04 </title>
    <meta name="author" content="Prof. Fábio">
    <meta name="copyright" content="PI/IHC">
    <meta name="keywords" content="HTML">
  </head>
  <body style="background-color:rgb(50,150,150)">
    <h1 align="center">
       GUI com Bot&otilde;es e Imagens
    </h1><br>
    <input type="button" name="botaoOK" value="OK" onclick="alert('OK')">
    <button type="button" name="botaoGrava" onclick="alert('Gravar')"> Gravar
</button>
    <br><br>
    <button type="button" name="botao1" onclick="alert('Botao1')"> <img
src="botao1.gif"> Bot&atilde;o-1</button>
    <button type="button" name="botao2" onclick="alert('Botao2')">Bot&atilde;o-
2</button>
    <button type="button" name="botao3" onclick="alert('Botao3')"> <img
src="botao3.gif"></button>
  </body>
</html>
```

A saída da codificação acima é exibida a seguir:

Caixas de texto e Rótulos no HTML

Podemos inserir caixa de texto no documento HTML com o uso da *tag* **<input type="text">**. As propriedades da *tag* **<input type="text">** são:

- "**name**": indica o nome da caixa de texto;
- "**value**": indica o texto de valor padrão da caixa de texto;

- "**autocomplete**": pode possuir os valores "***on***" (ativa o recurso de preenchimento automático) ou "***off***" (desativa o recurso de preenchimento automático);
- "**maxlength**": define a quantidade máxima de caracteres que será aceita no preenchimento da caixa de texto;
- "**placeholder**": define um texto de auxílio de preenchimento da caixa de texto, o qual é previamente exibido dentro da caixa de texto e que é automaticamente substituído pelo texto digitado pelo usuário;
- "**readonly**": pode possuir os valores "***true***" (desabilita a edição da caixa de texto e a deixa esmaecida) ou "***false***" (habilita a edição da caixa de texto);
- "**required**": pode possuir os valores "***true***" (a caixa de texto tem o preenchimento obrigatório) ou "***false***" (a caixa de texto tem o preenchimento facultativo);
- "**id**": define uma identificação que pode ser utilizada em outras tags ou scripts para fazer a referência à caixa de texto;
- "**size**": define a largura, em caracteres, da caixa de texto.

Não é necessário usar a *tag* de fechamento </input>, já que todas as informações do botão são definidas nas propriedades em sua *tag* de abertura.

***tag* input type="text"**

```
<label> Usu&aacute;rio <input type="text" name="usuario"
autocomplete="on" required="true"></label>
```

A *tag* **<label>** permite criarmos um rótulo para a caixa de texto. Ela tem a propriedade "**for**" que permite a vinculação do rótulo com a caixa de texto. Assim, colocamos como conteúdo de "**for**" o mesmo conteúdo da propriedade "**id**" da caixa de texto. Esse recurso permite uma melhor usabilidade, pois quando o usuário clica no rótulo ele é automaticamente posicionado na respectiva caixa de texto vinculada ao rótulo.

***tag* label**

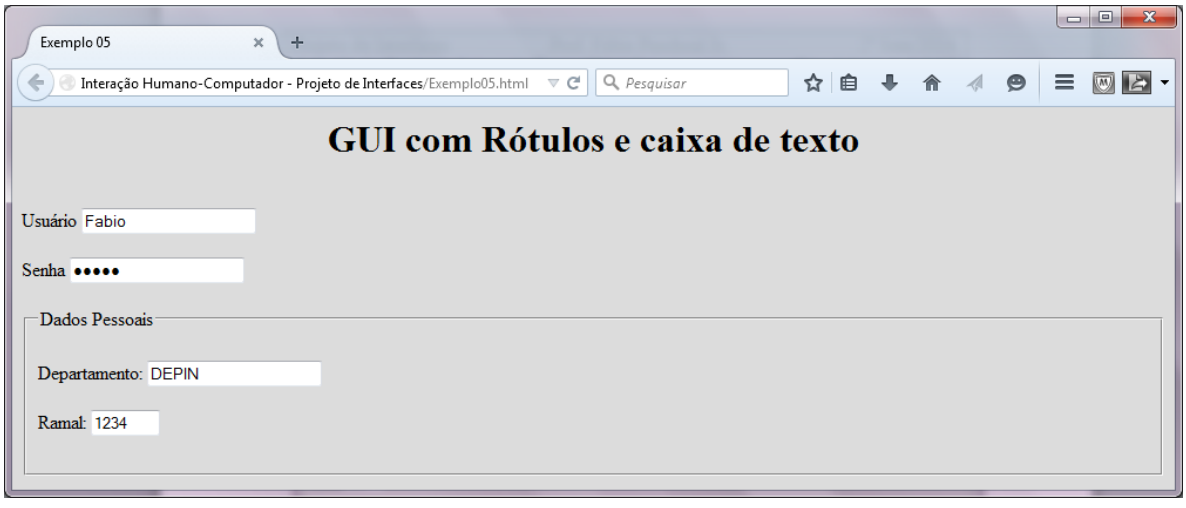
```
<label for="us"> Usu&aacute;rio <input type="text" name="usuario"
autocomplete="on" required="true" id="us" ></label>
```

O programa abaixo ilustra o nosso quinto exemplo de codificação HTML, o qual constrói uma página de web com caixas de texto. Crie o arquivo **Exemplo05.html** dentro de um editor de textos sem formatação (bloco de notas), posteriormente, abra o mesmo no navegador de web.

Exemplo05.html

```
<html>
  <head>
    <title> Exemplo 05 </title>
    <meta name="author" content="Prof. Fábio">
    <meta name="copyright" content="PI/IHC">
    <meta name="keywords" content="HTML">
  </head>
  <body style="background-color:rgb(220,220,220)">
    <h1 align="center">
       GUI com R&oacute;tulos e caixa de texto
    </h1><br>
    <label for="us"> Usu&aacute;rio <input type="text" name="usuario"
autocomplete="on" required="true" id="us"></label><br><br>
    <label for="pw">  Senha <input type="password" name="senha" autocomplete="off"
required="true" id="pw"> </label><br><br>
    <fieldset>
      <legend> Dados Pessoais </legend><br>
      <label for="dp">  Departamento: <input type="text" name="depto" id="dp">
</label><br><br>
      <label for="rm"> Ramal: <input type="text" name="ramal" size="5" id="rm">
</label><br><br>
    </fieldset>
  </body>
</html>
```

A saída da codificação acima é exibida a seguir:



O programa abaixo ilustra o nosso sexto exemplo de codificação HTML, o qual constrói uma página de web com uma calculadora simples. Crie o arquivo **Exemplo06.html** dentro de um editor de textos sem formatação (bloco de notas), posteriormente, abra o mesmo no navegador de web.

Exemplo06.html

```
<html>
  <head>
    <title>GUI de Calculadora</title>
    <meta name="author" content="Prof. Fábio">
    <meta name="copyright" content="PI/IHC">
    <meta name="keywords" content="HTML">
    <style type="text/css">
      /* Conteúdo do CSS */
      /* formatação do corpo do documento HTML */
      body {
        /* cor de fundo */
        background-color: rgb(220,220,220);
      }
      /* formatação dos elementos input do formulário com a classe=limpar */
      form input.limpar {
        /* largura em pixels */
        width: 70px;
        /* cor de fundo */
        background-color: rgb(255,0,0);
        /* cor da fonte */
        color: rgb(255,255,255);
        /* estilo da fonte como negrito */
        font-weight: bold;
      }
      /* formatação dos elementos input do formulário com a classe=oper */
      form input.oper {
        width: 70px;
        background-color: rgb(173,216,230);
        font-weight: bold;
      }
      /* formatação dos elementos label do formulário com a classe=num */
      form label.num {
        color: rgb(0,0,255);
        font-weight: bold;
      }
      /* formatação dos elementos label do formulário com a classe=res */
      form label.res {
        color: rgb(255,0,0);
        font-weight: bold;
      }
    </style>
    <script type="text/javascript">
      /* Conteúdo do JavaScript */
      // Função que limpa as caixas de texto da calculadora e todas as variáveis existentes.
      function Limpar() {
        // limpa o valor da caixa de texto num1 do formulário calculadora do documento
HTML
        document.calculadora.num1.value = '';
```

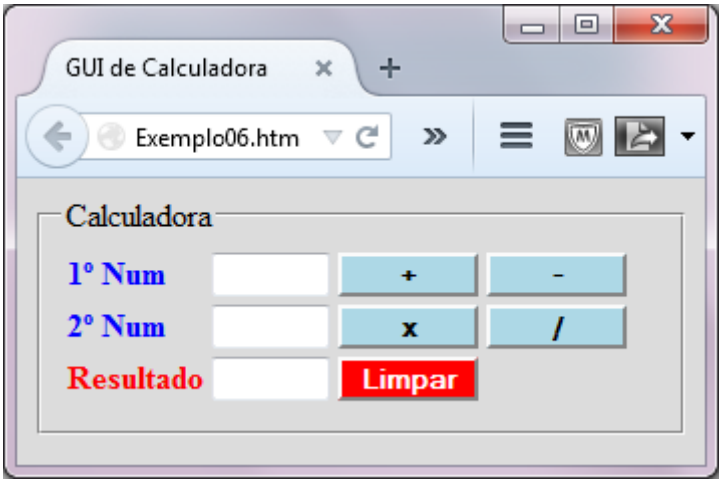
Exemplo06.html (continuação)

```
        // limpa o valor da caixa de texto num2 do formulário calculadora do documento HTML
        document.calculadora.num2.value = '';
        // limpa o valor da caixa de texto resultado do formulário calculadora do documento HTML
        document.calculadora.resultado.value = '';
        // apaga a variável de memória valor1
        delete  valor1;
        // apaga a variável de memória valor2
        delete  valor2;
        // apaga a variável de memória valor
        delete  valor;
      }
      // Função que executa as quatro operações básicas da calculadora, passando como parâmetro a operação (oper)
      function Calcula(oper) {
        // cria a variável de memória valor1 com o valor da caixa de texto num1 do formulário calculadora do documento HTML
        var valor1 = document.calculadora.num1.value;
        // cria a variável de memória valor2 com o valor da caixa de texto num2 do formulário calculadora do documento HTML
        var valor2 = document.calculadora.num2.value;
        // verifica se a operação é de soma
        if (oper == "somar") {
          // cria a variável de memória valor com o valor da soma de valor1 e valor2 convertidos para float
          var valor = parseFloat(valor1) + parseFloat(valor2);
        }
        // verifica se a operação é de subtração
        else if (oper == "subtrair") {
          // cria a variável de memória valor com o valor da subtração de valor1 e valor2 convertidos para float
          var valor = parseFloat(valor1) - parseFloat(valor2);
        }
        // verifica se a operação é de multiplicação
        else if (oper == "multiplicar") {
          // cria a variável de memória valor com o valor da multiplicação de valor1 e valor2 convertidos para float
          var valor = parseFloat(valor1) * parseFloat(valor2);
        }
        // verifica se a operação é de divisão
        else if (oper == "dividir") {
          // cria a variável de memória valor com o valor da divisão de valor1 e valor2 convertidos para float
          var valor = parseFloat(valor1) / parseFloat(valor2);
        }
        // atribui o conteúdo de valor para a caixa de texto resultado do formulário calculadora do documento HTML
        document.calculadora.resultado.value = valor;
```

Exemplo06.html (continuação)

```
      }
    </script>
  </head>
  <body>
    <!-- cria um formulário com o nome de calculadora -->
    <form name="calculadora" method="post" action="">
      <fieldset>
        <legend>Calculadora</legend>
        <!-- cria uma tabela com 3 linhas e 4 colunas -->
         <table>
          <!-- define o corpo da tabela -->
          <tbody>
            <!-- primeira linha da tabela -->
            <tr>
              <!-- coluna 1 da tabela -->
              <td><label class="num" for="num1">1º Num</label></td>
              <!-- coluna 2 da tabela -->
              <td><input name="num1" id="num1" type="text" size="5"></td>
              <!-- coluna 3 da tabela -->
              <td><input name="somar" class="oper" value="+"
onclick="Calcula('somar')" type="button"></td>
              <!-- coluna 4 da tabela -->
              <td><input name="subtrair" class="oper" value="-"
onclick="Calcula('subtrair')" type="button"></td>
            </tr>
            <!-- segunda linha da tabela -->
            <tr>
              <td><label class="num" for="num2">2º Num</label></td>
              <td><input name="num2" id="num2" type="text" size="5"></td>
              <td><input name="multiplicar" class="oper" value="x"
onclick="Calcula('multiplicar')" type="button"></td>
              <td><input name="dividir" class="oper" value="/"
onclick="Calcula('dividir')" type="button"></td>
            </tr>
            <!-- terceira linha da tabela -->
            <tr>
              <td><label class="res" for="resultado">Resultado</label></td>
              <td><input name="resultado" id="resultado" type="text" readonly="true"
size="5"></td>
              <td><input class="limpar" name="limpar" value="Limpar"
onclick="Limpar()" type="button"></td>
              <td> </td>
            </tr>
          </tbody>
        </table>
      </fieldset>
    </form>
  </body>
</html>
```

A saída do programa acima é exibida abaixo:



Exercício (T1)

7-) Insira as seguintes operações na calculadora, onde cada operação deverá ter o seu botão adequado (Java e HTML). Trate também a exceção de divisão por zero (somente em Java). Ajuste o layout da GUI conforme achar melhor.

Elevado ao quadrado → Apresentará no resultado o valor de $(1° \text{ Núm.})^2$.

Elevado à potência → Apresentará no resultado o valor de $(1° \text{ Núm.})^{2°\text{Num}}$.

Raiz Quadrada → Apresentará no resultado o valor de $\sqrt{1°Núm.}$

Para calcular a potência de qualquer número:
<resultado> = Math.pow(<base>,<expoente>);

Para calcular a raiz quadrada:
<resultado> = Math.sqrt(<número>);