Graphic User Interface (GUI)

O Java possui algumas classes para a criação de uma Interface Gráfica de Usuário (*Graphic User Interface* - GUI), como a AWT (*Abstract Window Toolkit*) utilizada nas versões anteriores do Java, bem como a SWING, a qual faz uma extensão da classe AWT com diversos aprimoramentos e é utilizada na versão recente do Java.

Para a construção de uma GUI definimos o conjunto de objetos criados por diversas classes diferentes (componentes gráficos) que serão utilizados na GUI, bem como a sua distribuição para formar o visual (layout) da GUI.

A construção de uma GUI requer o uso de 3 classes externas, as quais serão associadas à nossa classe através da diretiva "**import**" no início da codificação de nossa classe. A sintaxe de utilização desta diretiva é descrita abaixo:

Sintaxe em Java

```
import <classe externa>;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

As classes do pacote AWT e suas diversas constantes numéricas para a utilização em seus métodos são definidas e disponibilizadas pela diretiva **java.awt.***.

Os eventos executados na GUI e que podem ser tratados são definidos e disponibilizados pela diretiva **java.awt.event.***.

As classes do pacote SWING e suas diversas constantes numéricas para a utilização em seus métodos são definidas e disponibilizadas pela diretiva **javax.swing.***.

Nos próximos tópicos estudaremos cada objeto (componente gráfico) que poderá ser utilizado na construção de uma GUI.

Frames

A classe **JFrame** pertence ao pacote AWT do Java e possibilita a criação de uma janela onde podemos definir a sua barra de títulos, o estilo de borda e permite adicionarmos outros objetos (componentes gráficos) em seu interior.

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

Métodos de JFrame	
Método	**Descrição**
JFrame()	Permite criar uma nova janela vazia.
JFrame("título")	Permite criar uma nova janela vazia definindo o título que será apresentado na barra de título da janela.
varString = getTitle()	Permite armazenarmos em uma variável de memória do tipo String o título definido na barra de título da janela.
varBoolean = isResizable()	Permite armazenarmos em uma variável de

	memória do tipo lógico (boolean) o valor true (v) quando a janela permitir o ajuste de suas dimensões ou o valor false (f) quando ela não permitir esse ajuste.
setResizable(true/false)	Permite definirmos que a janela poderá fazer o ajuste de suas dimensões (true) ou que não permitirá esse ajuste (false).
setTitle("título")	Permite definirmos o título que será apresentado na barra de título da janela.
setSize(<largura>, <altura>)	Permite definirmos o tamanho da janela, onde definimos a sua largura e a sua altura em pixels.
setLocation(<horizontal>, <vertical>)	Permite definirmos o posicionamento da janela na tela, onde definimos as coordenadas horizontal (eixo x) e vertical (eixo y) que define o ponto onde o canto superior esquerdo da janela estará.
setVisible(true/false)	Permite exibirmos a janela na tela do computador (true) ou ocultarmos a janela na tela do computador (false).
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)	Permite definirmos a ação do botão fechar da janela, que encerará a execução do aplicativo. Também podemos fazer isso com o tratamentos de eventos da janela, que estudaremos mais adiante.
getContentPane().setBackground(Color.<cor>) ou getContentPane().setBackground(new Color(<red>,<green>,<blue>))	Permite definirmos a cor de fundo da janela 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.
setLayout(new GridLayout(<linhas>, <colunas>))	Permite definirmos o layout de disposição dos componentes gráficos dentro da GUI. É necessário o uso de um gerenciador de layout e veremos isso mais adiante.
add(<objeto gráfico>)	Permite adicionarmos os objetos que representam os componentes gráficos da GUI, como rótulos, caixas de texto, botões, etc.. Os componentes são inseridos dentro da GUI dispostos conforme o layout utilizado, mas seguindo a ordem da esquerda para a direita, de cima para baixo (linha1-coluna1; linha1-coluna2; linha2-coluna1, linha2-coluna2, etc.).

O programa abaixo ilustra o nosso primeiro exemplo de codificação em Java, o qual constrói uma GUI com os métodos descritos anteriormente. Crie a classe **Exemplo1_Frame** dentro do NetBeans para executarmos o mesmo.

Exemplo1_Frame.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 Exemplo1_Frame herda as características da classe JFrame
class Exemplo1_Frame extends JFrame {

   // método construtor da classe que define as características da GUI
   Exemplo1_Frame()
   {
      // ajusta o título da janela
      setTitle("Minha primeira GUI em Java");
      // ajusta o tamanho da janela (largura e altura)
      setSize(300,300);
      // define a posição da janela (horizontal e vertical) do canto superior esquerdo
      setLocation(200,200);
      // define que a janela não pode ser redimensionada
      setResizable(false);
      // ajusta a cor de fundo da janela
      getContentPane().setBackground(Color.gray);

      // cria o atributo titulo com o título da janela
      String titulo = getTitle();
   } // fim do método construtor

  // método principal da classe Exemplo1_Frame
  public static void main(String args[]){

     // cria o objeto GUI que conterá uma janela
     JFrame GUI = new Exemplo1_Frame();
     // 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:

Textos e Imagens

Podemos inserir textos (etiquetas ou rótulos) e imagens em um Frame através da classe **JLabel** do Java. Essa classe disponibiliza vários métodos para a configuração de como o texto ou a imagem serão apresentados. Note que esse texto ficará fixo na interface e não será possível a sua modificação durante a execução da aplicação.

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

Sintaxe em Java
JLabel <rótulo> = new JLabel(<texto>); JLabel <rótulo> = new JLabel(<texto>, JLabel.<alinhamento>); JLabel <rótulo> = new JLabel(<imagem>); JLabel <rótulo> = new JLabel(<texto>,<imagem>, JLabel.<alinhamento>);

O alinhamento do texto poderá ser alinhado verticalmente e/ou horizontalmente. Os alinhamentos horizontais são JLabel.LEFT / SwingConstants.LEFT (à esquerda), JLabel.RIGHT / SwingConstants.RIGHT (à direita) ou JLabel.CENTER / SwingConstants.CENTER (centralizado). Os alinhamentos verticais são SwingConstants.TOP (acima), SwingConstants.CENTER (centralizado) ou SwingConstants.BOTTOM (abaixo). Quando o alinhamento não for especificado o alinhamento horizontal à esquerda e o alinhamento vertical centralizado são adotados como padrão.

Para cada nova imagem a ser exibida, 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**". Caso o arquivo de imagem se encontre dentro da pasta do projeto do NetBeans não é necessário especificar o caminho das pastas. Ex: "**tv1.gif**". Se desejar, é possível colocar a imagem dentro de uma subpasta da pasta do projeto, por exemplo, "**img**". Neste caso, o caminho seria "**img/tv1.gif**".

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

Métodos de JLabel	
Método	**Descrição**
JLabel()	Permite a criação de um rótulo vazio, sem um texto em seu interior.
JLabel(<texto>)	Permite a criação de um rótulo com o texto especificado.
JLabel(<texto>,<alinhamento>)	Permite a criação de um rótulo com o texto e alinhamentos especificados.
JLabel(<texto>,<imagem>)	Permite a criação de um rótulo com o texto e a imagem especificados.
JLabel(<texto>, <imagem>, <alinhamento>)	Permite a criação de um rótulo com o texto, a imagem e o alinhamento especificados.
varString = getText()	Permite armazenarmos em uma variável de memória do tipo String o texto definido no rótulo.
setText(<texto>)	Permite atribuir um novo texto ao rótulo.
setForeground(Color.<cor>) ou setForeground (new Color(<red>,<green>,<blue>))	Permite definirmos a cor da fonte do rótulo 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 rótulo. 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>.
setToolTipText(<texto>)	Permite atribuir um texto explicativo que será apresentado quando o ponteiro do mouse apontar no rótulo (mouse parar sobre o rótulo).
setHorizontalAlignment(<alinhamento horizontal>)	Permite definirmos o alinhamento horizontal do conteúdo do rótulo conforme a o alinhamento desejado. Os alinhamentos horizontais dever ser

	escritos com o seu nome em inglês. Exemplos: JLabel.LEFT ou SwingConstants.LEFT (à esquerda); JLabel.RIGHT ou SwingConstants.RIGHT (à direita); JLabel.CENTER / SwingConstants.CENTER (centralizado).
setVerticalAlignment(<alinhamento vertical>	Permite definirmos o alinhamento horizontal do conteúdo do rótulo conforme a o alinhamento desejado. Os alinhamentos horizontais dever ser escritos com o seu nome em inglês. Exemplos: SwingConstants.TOP (acima); SwingConstants.CENTER (centralizado); SwingConstants.BOTTOM (abaixo).
setIcon(<ImagemIcon>)	Permite atribuirmos uma imagem ao rótulo. A imagem deve ser um objeto do tipo ImageIcon.
varImageIcon = getIcon()	Permite armazenarmos em uma variável de memória do tipo ImageIcon a imagem definida no rótulo.

O programa abaixo ilustra o nosso segundo exemplo de codificação em Java, o qual constrói uma GUI com os métodos descritos anteriormente. Crie a classe **Exemplo2_Label** dentro do NetBeans para executarmos o mesmo.

Exemplo2_Label.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 Exemplo2_Label herda as características da classe JFrame
class Exemplo2_Label extends JFrame{


   // cria os atributos para os rótulos, que serão objetos no método construtor
   JLabel rotulo1, rotulo2, rotulo3, rotulo4;

   // cria o objeto com a imagem a ser apresentada na janela
   ImageIcon imagem1 = new ImageIcon("I:/PI-IHC/tv1.gif");
   ImageIcon imagem2 = new ImageIcon("I:/PI-IHC/tv2.gif");

   // método construtor da classe que define as características da GUI
   Exemplo2_Label(boolean mudaIcone)
   {
      // ajusta o título da janela
      setTitle("GUI com Rótulos e Imagens");
      // ajusta o tamanho da janela (largura e altura)
      setSize(300,300);
```

```
      // define a posição da janela (horizontal e vertical) do canto superior esquerdo
      if(!mudaIcone)
        setLocation(200,200);
      else
        setLocation(500,200);

      // define que a janela não pode ser redimensionada
      setResizable(false);
      // ajusta a cor de fundo da janela
      getContentPane().setBackground(new Color(220,220,220));
      // cria o objeto rotulo1 com uma etiqueta centralizada
      rotulo1 = new JLabel("Bem vindo à nossa segunda aplicação
GUI",JLabel.CENTER);
      // ajusta a cor da fonte do objeto rotulo1
      rotulo1.setForeground(Color.blue);
      // ajusta o texto explicativo do objeto rotulo1
      rotulo1.setToolTipText("Esse rótulo não contém imagem!!!");
      // cria o objeto rotulo2 com a imagem1
      rotulo2 = new JLabel(imagem1);
      // cria o objeto rotulo3 com uma etiqueta
      rotulo3 = new JLabel("Veja a programação de hoje!");
      // ajusta o alinhamento horizontal para “esquerda” do objeto rotulo3
      rotulo3.setHorizontalAlignment(SwingConstants.LEFT);
      // verifica o parâmetro mudaIcone para usar o método setIcon()
      if (mudaIcone)
        rotulo3.setIcon(rotulo2.getIcon());
      // ajusta a cor da fonte do objeto rotulo3
      rotulo3.setForeground(Color.darkGray);
      // verifica o parâmetro mudaIcone para usar o método getIcon()
      if (mudaIcone)
        // cria o objeto rotulo4 com uma etiqueta alinhada à direita
        rotulo4 = new JLabel("Não deixe a sua TV assim!!!", rotulo2.getIcon(),
JLabel.RIGHT);
      else
        // cria o objeto rotulo4 com uma etiqueta alinhada à direita
        rotulo4 = new JLabel("Não deixe a sua TV assim!!!", imagem2,
JLabel.RIGHT);

      // ajusta a fonte, o estilo e o tamanho da fonte do objeto rotulo4
      rotulo4.setFont(new Font("Courier",Font.BOLD,12));
      // ajusta a cor da fonte do objeto rotulo4
      rotulo4.setForeground(Color.red);
      // ajusta o alinhamento vertical para “acima” do objeto rotulo4
      rotulo4.setVerticalAlignment(SwingConstants.TOP);
      /// define o layout da janela em 4 linhas e 1 coluna
      setLayout(new GridLayout(4,1));
      // adiciona a etiqueta rotulo1 à GUI
      add(rotulo1);
      // adiciona a etiqueta rotulo2 à GUI
      add(rotulo2);
```

```
        // adiciona a etiqueta rotulo3 à GUI
        add(rotulo3);
        // adiciona a etiqueta rotulo4 à GUI
        add(rotulo4);
    } // fim do método construtor

    // método principal da classe Exemplo2_Label
    public static void main(String args[])
    {
        // cria o objeto GUI_1 que conterá uma janela, sem mudar o ícone do rótulo
        JFrame GUI_1 = new Exemplo2_Label(false);

        // cria o objeto GUI2 que conterá uma janela, mudando o ícone do rótulo
        JFrame GUI_2 = new Exemplo2_Label(true);

        // exibe a janela na tela.
        GUI_1.setVisible(true);

        // exibe a janela na tela.
        GUI_2.setVisible(true);

        // define a ação do botão fechar
        GUI_1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // define a ação do botão fechar
        GUI_2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

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

A saída do programa acima é exibida a seguir:

HTML5

A linguagem universal de apresentação de documentos na web é a HTML (*Hyper Text Markup Language*), que é composta por uma série de *tags* (indicadores) que informam ao navegador (browser) como devem ser exibidas as informações contidas no arquivo HTML.

Os documentos HTML são feitos para oferecer algum tipo de serviço aos seres humanos onde, em conjunto com outras tecnologias de web, possuem uma vasta gama de utilização, permitindo desde a simples leitura de alguma informação até a execução de aplicativos informatizados complexos de controle.

Existem diversos aplicativos (softwares) que criam documentos HTML, de forma automática, para disponibilizar o resultado de seu processamento na web. Por outro lado, também existem diversos aplicativos que usam os documentos HTML disponibilizados na web para fazer algum processamento com o conteúdo desses documentos.

O HTML5 (*Hyper Text Markup Language 5*) foi criado a partir da cooperação entre o W3C (*World Wide Web Consortium*) e o WHATGW (*Web Hypertext Application Technology Working Group*).

É a versão mais recente da HTML, a qual incorpora o HTML4 com as tecnologias XHTML e HTML DOM nível 2, bem como trabalha nativamente com gráficos, músicas, vídeos e permite a criação de aplicações web complexas.

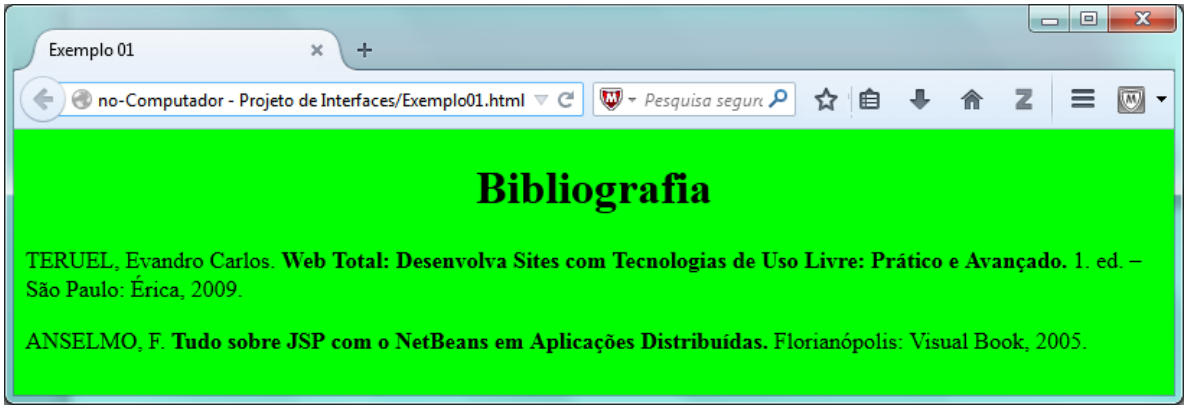
Ela é transportável e permite que o documento seja visualizado da mesma forma e com a mesma programação tanto em um PC quanto em um Tablet, Smarthpone ou SmartTV.

O programa abaixo ilustra o nosso primeiro exemplo de codificação HTML, o qual constrói uma página de web básica. Crie o arquivo **Exemplo01.html** dentro de um editor de textos sem formatação (bloco de notas), posteriormente, abra o mesmo no navegador de web.

Exemplo01.html

```
<!DOCTYPE html>
<html>
  <head>
    <title> Exemplo 01 </title>
    <meta name="author" content="Prof. Fábio">
    <meta name="copyright" content="PI/IHC">
    <meta name="keywords" content="HTML">
  </head>
  <body bgcolor="lime">
    <h1 align="center"> Bibliografia </h1>
    <p> TERUEL, Evandro Carlos. <b> Web Total: Desenvolva Sites com Tecnologias de Uso Livre: Prático e Avançado. </b>  1. ed. – São Paulo: Érica, 2009. </p>
    <p> ANSELMO, F. <b> Tudo sobre JSP com o NetBeans em Aplicações Distribuídas. </b> Florianópolis: Visual Book, 2005. </p>
  </body>
</html>
```

A saída da codificação acima é exibida a seguir:



Na listagem do **Exemplo01.html** exibido na página anterior, as *tags* HTML são identificadas pela cor vermelha, o conteúdo do documento (dados) é identificado pela cor preta, as *tags* especiais de metadados com as informações de indexação da página são identificadas pela cor verde, as propriedades das *tags*/metadados são identificadas pela cor azul e as informações contidas nas propriedades das *tags*/metadados são identificadas pela cor roxa.

Normalmente, iniciamos a *tag* HTML com o seu nome, depois colocamos uma ou mais propriedades da *tag*, posteriormente é indicado o conteúdo e finalmente finalizamos a *tag* com o mesmo nome precedido de barra.

Abaixo é ilustrada a sintaxe básica da maioria das *tags* HTML e um exemplo de utilização das *tags* e a visualização deste exemplo no navegador.

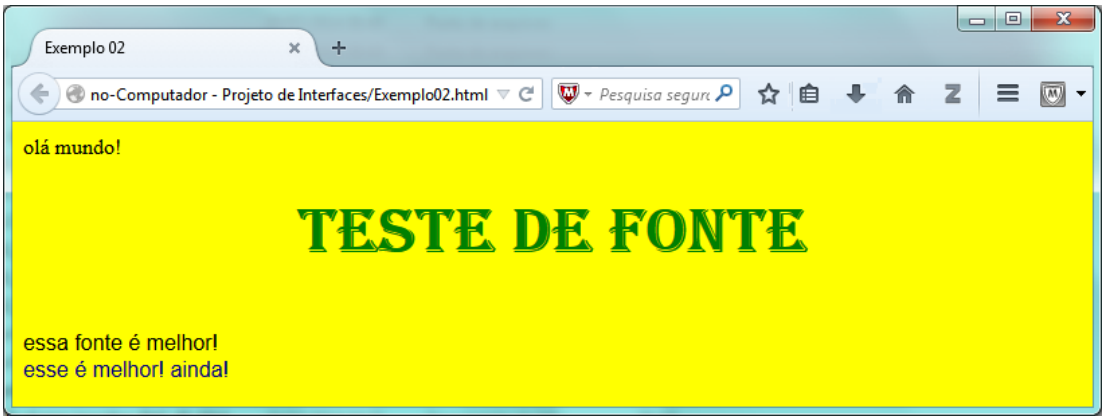
Sintaxe geral de *tags* HTML

```
<tag propriedade1="valor1" propriedadeN="valorN" > conteúdo </tag>
```

Exemplo02.html

```
<html>
  <head>
    <title> Exemplo 02 </title>
    <meta name="author" content="Prof. Fábio">
    <meta name="copyright" content="PI/IHC">
    <meta name="keywords" content="HTML">
  </head>
  <body bgcolor="yellow">
    ol&aacute mundo!
    <h1 align="center">
      <font face="algerian" size="7" color="green"> teste de fonte</font>
    </h1><br>
    <font face="arial" size="3"> essa fonte &eacute melhor!</font><br>
    <font face="helvetica, arial" color="navy"> esse &eacute melhor! ainda!</font><br>
  </body>
</html>
```

A saída da codificação acima é exibida a seguir:



A HTML oferece as seguintes vantagens:

- **Simplicidade** → Qualquer pessoa pode escrever um documento HTML.
- **Textual** → É legível e podendo ser escrita em qualquer editor.
- **Transportável** → É interpretada por qualquer navegador de qualquer plataforma, sendo independente de sistema operacional e do hardware, sendo utilizada de forma universal por qualquer navegador.
- **Aglutinador** → Faz a união ou junção dos fragmentos de informações e até mesmo outros documentos HTML com a utilização de hipertextos.

A HTML oferece as seguintes desvantagens:

- **Interpretada** → A visualização do documento HTML depende da interpretação do navegador, que pode ter uma ligeira diferenciação na forma de exibição conforme o navegador utilizado.
- **Múltiplas versões** → São necessárias versões diferentes do documento HTML conforme a mídia utilizada para a apresentação do mesmo, como navegador, webmail ou dispositivos portáteis.
- **Indexação limitada** → A indexação dos documentos HTML só podem ser feita com base em sua parte textual, sem permitir a indexação de outros elementos que possam estar associados ao documento.
- **Dados não estruturados** → A HTML não é adaptada para descrever a estrutura do documento, indicando os dados que o documento possui.
- **Não é interoperável** → Os documentos HTML não são interoperáveis e não permitem a troca de dados entre os mesmos, impossibilitando o intercâmbio de informações entre os documentos.

Imagens no HTML

Podemos inserir imagens no documento HTML com o uso da tag <img>. Nela podemos especificar até 4 propriedades para definir as características de exibição da imagem.

A propriedade "**src**" indica o arquivo de imagem a ser inserida, onde podemos especificar opcionalmente o caminho onde se encontra a imagem (hierarquia de pastas). Se desejar, é possível colocar a imagem dentro de uma subpasta da pasta do projeto, por

exemplo, "**img**". Neste caso, o caminho seria "**../img/tv1.gif**". A propriedade "**alt**" define um texto alternativo a ser exibido no lugar da imagem, caso ela não possa ser exibida por algum problema. A propriedade "**width**" define a largura de exibição da imagem. A propriedade "**height**" define a altura de exibição da imagem.

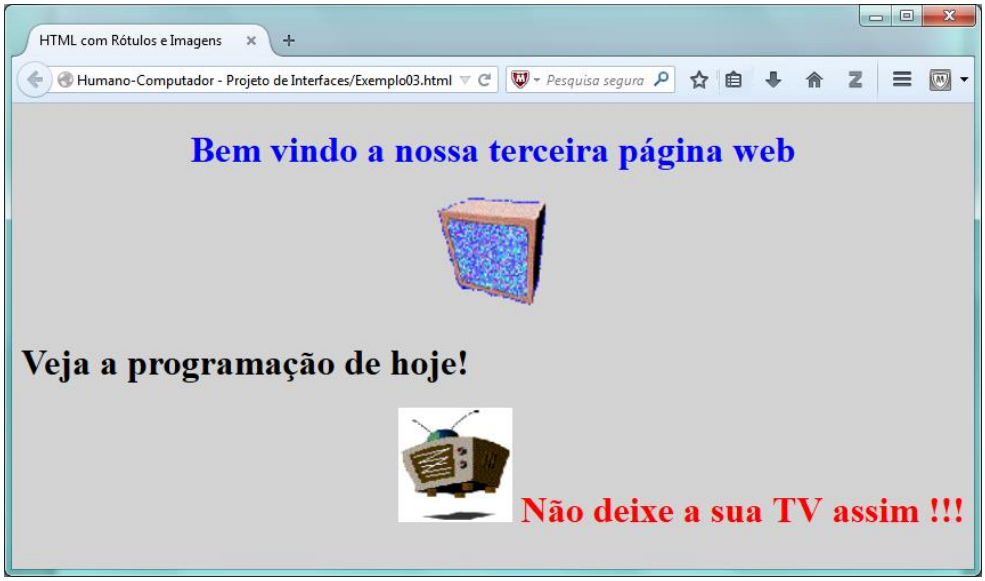
Não é necessário usar a tag de fechamento </img>, já que todas as informações da imagem são definidas nas propriedades em sua tag de abertura.

O programa abaixo ilustra o nosso terceiro exemplo de codificação HTML, o qual constrói uma página de web com rótulos e imagens. Crie o arquivo **Exemplo03.html** dentro de um editor de textos sem formatação (bloco de notas), posteriormente, abra o mesmo no navegador de web.

Exemplo03.html

```
<!DOCTYPE html>
<html>
  <head>
    <title> HTML com Rótulos e Imagens </title>
    <meta name="author" content="Prof. Fábio">
    <meta name="copyright" content="PI/IHC">
    <meta name="keywords" content="HTML">
  </head>
  <body bgcolor="lightgray">
    <h1 align="center"> <font color="blue"> Bem vindo a nossa terceira página web
</font> </h1>
    <p> <center> <img src="tv1.gif" alt="TV" width="100" height="100"> </center>
 </p>
    <h1> Veja a programação de hoje! </h1>
    <h1 align="right"> <img src="tv2.gif" alt="TV" width="100" height="100"> <font
color="red"> Não deixe a sua TV assim !!!</font> </h1>
  </body>
</html>
```

A saída da codificação acima é exibida a seguir:



Exercício (T1)

2-) Construa a GUI abaixo (Java e HTML):