

## Caixas de Senha

Podemos inserir caixas para preenchimento de senhas em um Frame através da classe **JPasswordField** do Java. Essa classe disponibiliza os mesmos recursos da classe **JTextField**, ou seja, vários métodos para a configuração de como a caixa de senha será apresentada, onde podemos inserir um texto, bem como definir o tamanho da caixa de senha (em caracteres). Note que o usuário poderá digitar qualquer texto dentro da caixa de senha, mas cada caractere digitado será substituído por “\*” ou pelo caractere que for especificado.

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

Sintaxe em Java
<pre>JPasswordField &lt;caixa de texto&gt; = new JPasswordField(); JPasswordField &lt;caixa de texto&gt; = new JPasswordField(&lt;senha&gt;); JPasswordField &lt;caixa de texto&gt; = new JPasswordField(&lt;tamanho&gt;); JPasswordField &lt;caixa de texto&gt; = new JPasswordField(&lt;senha&gt;,&lt;tamanho&gt;);</pre>

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

Métodos de JPasswordField	
Método	Descrição
JPasswordField ()	Permite a criação de uma caixa de senha vazia, sem uma senha em seu interior.
JPasswordField (<senha>)	Permite a criação de uma caixa de senha com o com a senha especificada.
JPasswordField(<tamanho>)	Permite a criação de uma caixa de senha com o tamanho de caracteres especificado.
JPasswordField(<senha>,<tamanho>)	Permite a criação de uma caixa de senha com a senha e a caixa com o tamanho de caracteres especificados.
varchar[] = getPassword()	Permite armazenarmos em uma variável de memória do tipo vetor de char o texto armazenado na caixa de senha.
setText(<texto>)	Permite atribuir um novo texto à caixa de senha.
varString = getSelectedText()	Permite armazenarmos em uma variável de memória do tipo String o texto selecionado na caixa de senha.
setEditable(true/false)	Permite definirmos se a caixa de senha 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 senha permite a edição de seu texto (true) ou se não permite tal ação

	(false).
selectAll()	Permite selecionarmos todo o texto contido na caixa de senha.
setEchoChar(<caractere>)	Permite definirmos qual o caractere especial que será utilizado para substituir os caracteres que foram digitados na caixa de senha. Se não definirmos algum caractere especial com esse método o caractere “*” será utilizado como padrão.
addActionListener(this)	Define que a caixa de senha 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 senha no método <b>actionPerformed()</b> .

O programa abaixo ilustra o nosso quinto exemplo de codificação, o qual constrói uma GUI com os métodos descritos anteriormente. Crie a classe **Exemplo5\_CaixadeSenha** dentro do NetBeans para executarmos o mesmo.

<b>Exemplo5_CaixadeSenha.java</b>
<pre> // 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 Exemplo5_CaixadeSenha herda as características da classe JFrame // e implementa uma interface para os eventos de ActionListener // para verificar o evento ENTER e o para determinar // as ações (comandos) da caixa de senha. public class Exemplo5_CaixadeSenha extends JFrame implements ActionListener{      // define os atributos para criar posteriormente os objetos gráficos da GUI     JLabel rotulo1,rotulo2;     JTextField texto1;     JPasswordField senha1;      // método principal da classe Exemplo5_CaixadeSenha     public static void main(String args[]){          // cria o objeto GUI que conterá uma janela         JFrame GUI = new Exemplo5_CaixadeSenha();          // exibe a janela na tela.         GUI.setVisible(true);          // define a ação do botão fechar         GUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); </pre>

```
} // fim do método principal

// método construtor da classe que define as características da GUI
Exemplo5_CaixaDeSenha(){

    // ajusta o título da janela
    setTitle("Entre com a Senha");
    // ajusta o tamanho da janela (largura e altura)
    setSize(300,80);
    // ajusta a cor de fundo da janela
    getContentPane().setBackground(new Color(250,250,250));
    // ajusta o layout da tela para uma matriz de 2 linhas e 2 colunas
    setLayout(new GridLayout(2,2));
    // cria o objeto de rótulo rotulo1 com o texto indicado
    rotulo1 = new JLabel("Digite a sua senha:");
    // cria o objeto de rótulo rotulo2 com o texto indicado
    rotulo2 = new JLabel("Situação:");
    // ajusta a cor da fonte do rotulo1 com a cor especificada
    rotulo1.setForeground(Color.black);
    // ajusta a cor da fonte do rotulo2 com a cor especificada
    rotulo2.setForeground(Color.black);
    // ajusta as características da fonte do rotulo1
    rotulo1.setFont(new Font("",Font.BOLD,12));
    // ajusta as características da fonte do rotulo2
    rotulo2.setFont(new Font("",Font.ITALIC,12));
    // cria o objeto de caixa de texto texto1
    texto1 = new JTextField();
    // cria o objeto de caixa de senha senha1
    senha1 = new JPasswordField();
    // define o caractere especial a ser apresentado na caixa de senha senha1
    senha1.setEchoChar('%');
    // define que a caixa de texto senha1 terá o tratamento de eventos.
    senha1.addActionListener(this);
    // adiciona o rotulo1 na GUI (linha1, coluna1)
    add(rotulo1);
    // adiciona a senha1 na GUI (linha1, coluna2)
    add(senha1);
    // adiciona o rotulo2 na GUI (linha2, coluna1)
    add(rotulo2);
    // adiciona o texto1 na GUI (linha2, coluna2)
    add(texto1);
} // fim do método construtor

// método que implementa uma interface para os eventos da janela
// para verificar o pressionamento de enter e para determinar
// as ações (comandos) de caixa de senha.
public void actionPerformed(ActionEvent evento)
{
    // seleciona todo o texto da caixa de senha
```

```
senha1.selectAll();

// cria a variável senha2 com o texto selecionado da caixa de texto
String senha2 = senha1.getSelectedText();

// converte todo o texto para maiúscula
senha2 = senha2.toUpperCase();

// verifica se a senha está correta
if (senha2.equals("CEFET"))
{
    // ajusta a cor da fonte do texto1 com a cor especificada
    texto1.setForeground(Color.blue);
    // informa que a senha está correta
    texto1.setText("Senha correta, aguarde...");
}
// se a senha estiver incorreta
else
{
    // ajusta a cor da fonte do texto1 com a cor especificada
    texto1.setForeground(Color.red);
    // informa que a senha está inválida
    texto1.setText("Senha Inválida: " + senha2);
}

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

A saída do programa acima é exibida abaixo:



## Formatação de Caixas de Texto

Podemos formatar os números de uma caixa de texto de um Frame através da classe **NumberFormat** do Java. Essa classe disponibiliza vários métodos para a configuração de como a caixa de texto será formatada. Para utilizar essa classe, devemos importá-la com a diretiva abaixo, logo no início da classe:

```
import java.text.NumberFormat;
```

Para cada tipo de formato diferente a ser utilizado no Frame, incluiremos um novo objeto que represente a classe **NumberFormat**. A sintaxe de criação de um objeto da classe **NumberFormat** é ilustrada abaixo:

Sintaxe em Java
NumberFormat <formato> = NumberFormat.<tipo de formato>;

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

Métodos de NumberFormat	
Método	Descrição
getNumberInstance()	Formata o número com separador de grupos e de casas decimais.
getCurrencyInstance()	Formata o número com separador de grupos e de casas decimais, além do símbolo monetário.
getPercentInstance()	Formata o número com separador de grupos e de casas decimais, mas na forma percentual.
setMinimumIntegerDigits(<valor>)	Define a quantidade mínima de dígitos inteiros passado no parâmetro <valor>. Se o número tiver menos dígitos do que especificado em <valor>, serão apresentados zeros à esquerda do número.
setMaximumIntegerDigits(<valor>)	Define a quantidade máxima de dígitos inteiros passado no parâmetro <valor>. Se o número tiver mais dígitos do que especificado em <valor>, o número será truncado (cortado) até a quantidade de dígitos especificada.
setMinimumFractionDigits(<valor>)	Define a quantidade mínima de dígitos fracionários passado no parâmetro <valor>. Se o número tiver menos dígitos do que especificado em <valor>, serão apresentados zeros à direita do número.
setMaximumFractionDigits(<valor>)	Define a quantidade máxima de dígitos fracionários passado no parâmetro <valor>. Se o número tiver mais dígitos do que especificado em <valor>, o número será arredondado.

`varString = format(varFloat)`

Aplica a formatação especificada nos comandos anteriores à variável `varFloat` passada como parâmetro. Retorna a formatação no formato de `String`, que pode ser associado a uma variável `String`.

O programa abaixo ilustra o nosso sexto exemplo de codificação, o qual constrói uma GUI com os métodos descritos anteriormente. Crie a classe **Exemplo6\_Formatacao** dentro do NetBeans para executarmos o mesmo.

#### Exemplo6\_Formatacao.java

```
// importa o pacote awt
import java.awt.*;
// importa os eventos do awt
import java.awt.event.*;
// importa o pacote swing
import javax.swing.*;
// importa o pacote de formatação numérica
import java.text.NumberFormat;

// a classe Exemplo6_Formatacao herda as características da classe JFrame
public class Exemplo6_Formatacao extends JFrame
{
    // define os atributos para criar posteriormente os objetos gráficos da GUI
    JLabel rotulo1, rotulo2, rotulo3, rotulo4;
    JTextField texto1, texto2, texto3, texto4;
    // define as variáveis numéricas a serem formatadas
    double valor1 = 123456.78910111213, porcentagem = 0.123456789;

    // define os atributos para criar posteriormente os objetos de formatação
    NumberFormat formato1, formato2, formato3;

    // método principal da classe Exemplo6_Formatacao
    public static void main(String args[]){

        // cria o objeto GUI que conterá uma janela
        JFrame GUI = new Exemplo6_Formatacao();

        // 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 Exemplo6_Formatacao()
    {
```

```
// ajusta o título da janela
setTitle("Formatação de números");
// ajusta o tamanho da janela (largura e altura)
setSize(280,150);
// ajusta a cor de fundo da janela\
getContentPane().setBackground(new Color(100,150,200));
// ajusta o layout da tela para uma matriz de 4 linhas e 2 colunas
setLayout(new GridLayout(4,2));

// define o objeto formato1 para formatar números reais
formato1 = NumberFormat.getNumberInstance();
// define a quantidade mínima de casas decimais do formato1
formato1.setMinimumFractionDigits(3);

// define o objeto formato2 para formatar números monetários
formato2 = NumberFormat.getCurrencyInstance();
// define a quantidade mínima de casas decimais do formato2
formato2.setMinimumFractionDigits(2);

// define o objeto formato3 para formatar números de porcentagem
formato3 = NumberFormat.getPercentInstance();
// define a quantidade mínima de casas decimais do formato3
formato3.setMinimumFractionDigits(4);

// cria o objeto de rótulo rotulo1 com o texto indicado
rotulo1 = new JLabel("Nº sem formatação:");

// cria o objeto de caixa de texto texto1
texto1 = new JTextField();
// coloca o valor1 sem formatação
texto1.setText(""+valor1);
// desabilita a digitação na caixa de texto texto1
texto1.setEditable(false);

// cria o objeto de rótulo rotulo2 com o texto indicado
rotulo2 = new JLabel("Nº com formatação:");

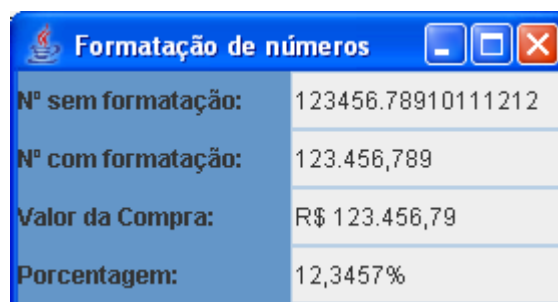
// cria o objeto de caixa de texto texto2
texto2 = new JTextField();
// coloca o valor1 com formatação de números reais
texto2.setText(""+formato1.format(valor1));
// desabilita a digitação na caixa de texto texto2
texto2.setEditable(false);

// cria o objeto de rótulo rotulo3 com o texto indicado
rotulo3 = new JLabel("Valor da Compra:");

// cria o objeto de caixa de texto texto3
texto3 = new JTextField();
// coloca o valor1 com formatação monetária
```

```
texto3.setText(""+formato2.format(valor1));  
// desabilita a digitação na caixa de texto texto3  
texto3.setEditable(false);  
  
// cria o objeto de rótulo rotulo4 com o texto indicado  
rotulo4 = new JLabel("Porcentagem:");  
  
// cria o objeto de caixa de texto texto4  
texto4 = new JTextField();  
// coloca o valor1 com formatação de números reais  
texto4.setText(""+formato3.format(valor1));  
// desabilita a digitação na caixa de texto texto4  
texto4.setEditable(false);  
  
// adiciona o rotulo1 na GUI (linha1, coluna1)  
add(rotulo1);  
// adiciona o texto1 na GUI (linha1, coluna2)  
add(texto1);  
// adiciona o rotulo2 na GUI (linha2, coluna1)  
add(rotulo2);  
// adiciona o texto2 na GUI (linha2, coluna2)  
add(texto2);  
// adiciona o rotulo3 na GUI (linha3, coluna1)  
add(rotulo3);  
// adiciona o texto3 na GUI (linha3, coluna2)  
add(texto3);  
// adiciona o rotulo4 na GUI (linha4, coluna1)  
add(rotulo4);  
// adiciona o texto4 na GUI (linha4, coluna2)  
add(texto4);  
  
} // fim do método construtor  
} // fim da classe
```

A saída do programa acima é exibida abaixo:



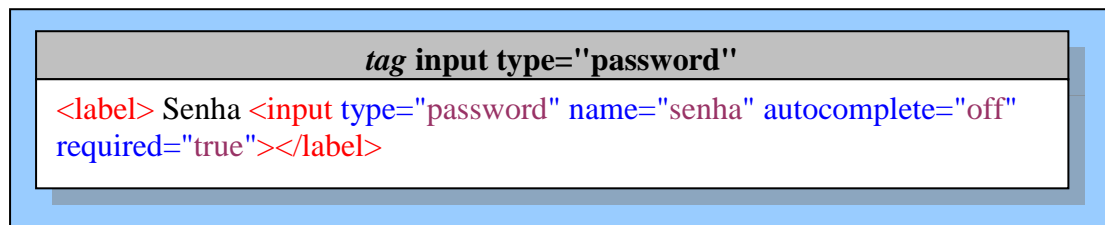
## Caixas de senha no HTML

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

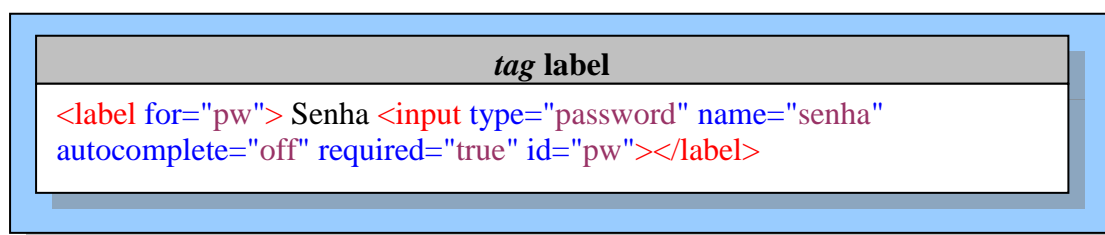


- “**name**”: indica o nome da caixa de senha;
- “**value**”: indica o texto de valor padrão da caixa de senha;
- “**autocomplete**”: pode possuir os valores “**on**” (ativa o recurso de preenchimento automático) ou “**off**” (desativa o recurso de preenchimento automático);
- “**maxlength**”: define a quantidade máxima de caracteres que será aceita no preenchimento da caixa de senha;
- “**readonly**”: pode possuir os valores “**true**” (desabilita a edição da caixa de senha e a deixa esmaecida) ou “**false**” (habilita a edição da caixa de senha);
- “**required**”: pode possuir os valores “**true**” (a caixa de senha tem o preenchimento obrigatório) ou “**false**” (a caixa de senha tem o preenchimento facultativo);
- “**id**”: define uma identificação que pode ser utilizada em outras tags ou scripts para fazer a referência à caixa de senha;
- “**size**”: define a largura, em caracteres, da caixa de senha.

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.



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



O programa abaixo ilustra o nosso sétimo exemplo de codificação HTML, o qual constrói uma página de web com caixas de senha. Crie o arquivo **Exemplo07.html** dentro de um editor de textos sem formatação (bloco de notas), posteriormente, abra o mesmo no navegador de web.

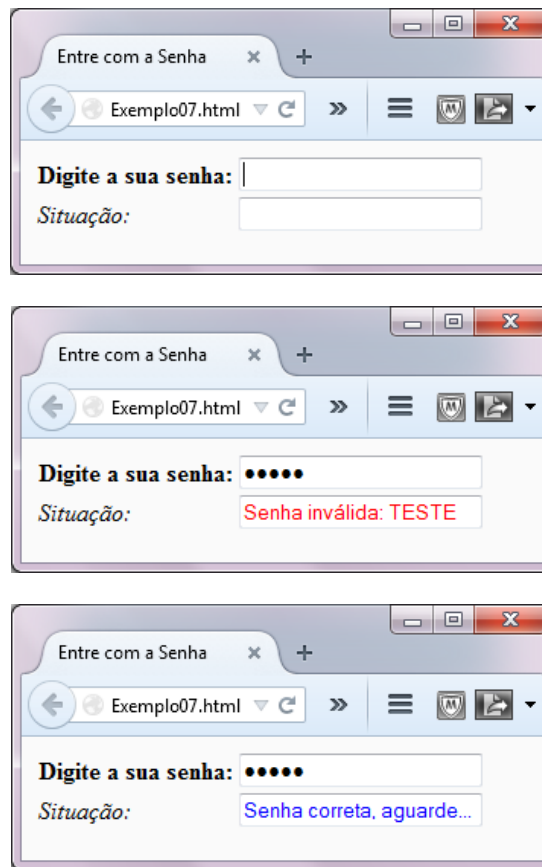
### Exemplo07.html

```
<html>
<head>
  <title>Entre com a Senha</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(250,250,250);
    }
    /* formatação dos elementos label com a classe=pw */
    label.pw {
      /* estilo da fonte como negrito */
      font-weight: bold;
      /* largura em pixels */
      width: 160px;
    }
    /* formatação dos elementos label com a classe=st */
    label.st {
      /* estilo da fonte como itálico */
      font-style: italic;
      /* largura em pixels */
      width: 160px;
    }
    /* formatação dos elementos input com a classe=tx */
    input.tx {
      /* largura em pixels */
      width: 160px;
    }
    /* formatação dos elementos input com a classe=pw */
    input.pw {
      /* largura em pixels */
      width: 160px;
    }
  </style>
  <script type="text/javascript">
    /* Conteúdo do JavaScript */
    // Função que captura o pressionamento da tecla Enter e verifica o conteúdo da
    senha digitada pelo parâmetro evento
    function teclaEnter (evento){
      // cria a variável de memória codigo_tecla para armazenar o código ASCII da tecla
      pressionada na caixa de senha pw do documento HTML
      var codigo_tecla;
```

### Exemplo07.html (continuação)

```
// verifica o parâmetro evento para obter o código ASCII da tecla pressionada
if (evento.keyCode)
    codigo_tecla = evento.keyCode;
else if (evento.charCode)
    codigo_tecla = evento.charCode;
else if (evento.which)
    codigo_tecla = evento.which;
// verifica se foi teclada a tecla ENTER
if (codigo_tecla == 13) {
    // cria a variável senha com a senha digitada convertendo para maiúscula
    var senha = document.tela.senha.value.toUpperCase();
    // verifica se a senha está correta
    if(senha == "CEFET") {
        // ajusta a cor da caixa de texto de situação para azul
        document.getElementById("st").style.color = "blue";
        // atualiza a mensagem da caixa de texto de situação
        document.tela.sit.value = "Senha correta, aguarde...";
    }
    else {
        // ajusta a cor da caixa de texto de situação para vermelho
        document.getElementById("st").style.color = "red";
        // atualiza a mensagem da caixa de texto de situação
        document.tela.sit.value = "Senha inválida: "+senha;
    }
}
}
</script>
</head>
<body>
<!-- cria um formulário com o nome de tela -->
<form name="tela" method="post" action="">
<table>
<tr>
<td> <label for="pw" class="pw">Digite a sua senha:</label></td>
<td> <input type="password" name="senha" autocomplete="off"
required="true" id="pw" class="pw" onkeypress="teclaEnter(event)"> </td>
</tr>
<tr>
<td> <label for="st" " class="st">Situa&ccedil;&atilde;o:</label></td>
<td> <input type="text" name="sit" autocomplete="off" required="false" id="st"
readonly="true" class="tx"> </td>
</tr>
</table>
</form>
</body>
</html>
```

A saída da codificação acima é exibida a seguir:



## Formatação de Caixas de Texto no HTML

Podemos formatar os números de uma caixa de texto de uma página HTML através da **API (Application Programming Interface) JavaScript Number Format v1.5.4**<sup>1</sup>. Essa API disponibiliza vários métodos para a configuração de como a caixa de texto será formatada. Para utilizar essa API, devemos fazer o download da mesma, através do arquivo de JavaScript **“NumberFormat154.js”**<sup>2</sup>, colocando no mesmo diretório/pasta do arquivo HTML. Além disso, devemos vincular o arquivo de JavaScript da API ao arquivo HTML com a diretiva abaixo, logo no início do script do documento HTML:

```
<script language="JavaScript" type="text/javascript"
      src="numberFormat154.js"></script>
```

Para cada tipo de formato diferente a ser utilizado no documento HTML, incluiremos um novo objeto que represente a API **NumberFormat**. A sintaxe de criação de um objeto da API **NumberFormat** é ilustrada abaixo:

<sup>1</sup> [http://www.mredkj.com/javascript/nf\\_api.html](http://www.mredkj.com/javascript/nf_api.html)

<sup>2</sup> <http://www.mredkj.com/javascript/NumberFormat154.js>

### Sintaxe em JavaScript

```
var <formato> = new NumberFormat();
```

A tabela abaixo demonstra os principais métodos da API **NumberFormat**:

Métodos de NumberFormat	
Método	Descrição
getOriginal()	Retorna o formato original do número, incluindo os símbolos não numéricos que ele continha.
setCommas(<ativar>)	Define se o número terá ou não a separação por vírgulas. Se <ativar> contiver true o numero terá separação por vírgulas. Se <ativar> contiver false o numero não terá separação por vírgulas.
setCurrency(<ativar>)	Define se o número terá ou não o símbolo monetário. Se <ativar> contiver true o numero terá o símbolo monetário. Se <ativar> contiver false o numero não terá o símbolo monetário.
setCurrencyPosition(<posição>)	Define a posição do símbolo monetário e do sinal negativo, conforme o valor de <posição>. As possibilidades de <posição> são as seguintes constantes: LEFT_OUTSIDE “R\$-1.00”, LEFT_INSIDE “-R\$1.00”, RIGHT_INSIDE “1.00R\$-” e RIGHT_OUTSIDE “1.00-R\$”.
setCurrencyValue(<símbolo>)	Define o símbolo monetário passado no parâmetro <símbolo>. Usualmente temos em <símbolo> o valor ‘R\$’.
setInputDecimal(<caractere>)	Define o símbolo decimal que está sendo utilizado na separação das casas decimais do número a ser formatado. Usualmente temos em <caractere> os valores ‘,’ ou ‘.’.
setNegativeFormat(<formato>)	Define o formato do número negativo, conforme o valor de <formato>. As possibilidades de <formato> são as seguintes constantes: LEFT_DASH “-1000”, RIGHT_DASH “1000-” e PARENTHESIS “(1000)”.
setNegativeRed(<ativar>)	Define o formato do número negativo com cor vermelha, conforme o valor de <ativar>. Se <ativar> contiver true o número negativo será formatado na cor vermelha e o número positivo será formatado na cor preta. Se <ativar> contiver false o número será sempre formatado na cor preta. Essa funcionalidade não se aplica em caixa de texto, e sim em outras tags HTML.
setNumber(<num>)	Define o valor numérico a ser formatado.
setPlaces(<casas decimais>, <truncar>)	Define o número de casas decimais a ser aplicado na formatação, de acordo com o valor de <casas decimais>. Se <truncar> contiver true o número será truncado conforme a quantidade

	de casas decimais. Se <truncar> contiver false (valor padrão) o número será arredondado conforme a quantidade de casas decimais.
setSeparators(<separar>, <milhares>, <decimal>)	Define se serão utilizados separadores na formatação do número. Se <separar> contiver true serão utilizados os separadores. Se <separar> contiver false não serão utilizados os separadores. Em <milhares> definimos o separador de milhar e o seu valor usual é ‘.’. Em <decimal> definimos o separador de casas decimais e o seu valor usual é ‘,’.
<string> = toFormatted()	Retorna uma String contendo o número formatado de acordo com os ajustes feitos anteriormente à execução deste método.
toPercentage()	Aplica a formatação de porcentagem ao número
toUnformatted()	Retira a formatação do número, deixando sem separadores.

O programa abaixo ilustra o nosso oitavo exemplo de codificação HTML, o qual constrói uma página de web com caixas de texto formatadas. Crie o arquivo **Exemplo08.html** dentro de um editor de textos sem formatação (bloco de notas), posteriormente, abra o mesmo no navegador de web.

### Exemplo08.html

```
<html>
<head>
  <title>Formatação de números</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(100,150,200);
    }
    /* formatação dos elementos label */
    label {
      /* estilo da fonte como negrito */
      font-weight: bold;
      /* largura em pixels */
      width: 160px;
    }
    /* formatação dos elementos input */
    input {
      /* largura em pixels */
      width: 160px;
```

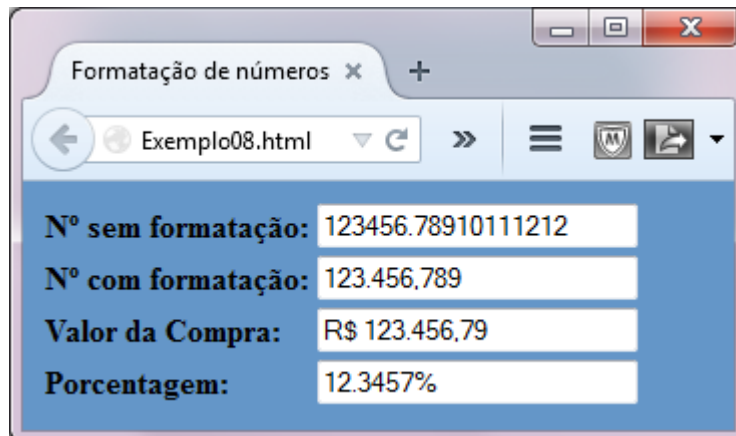
**Exemplo08.html (continuação)**

```
}
</style>
<!-- API NumberFormat154 do JavaScript -->
<script language="JavaScript" type="text/javascript" src="numberFormat154.js">
</script>
<script type="text/javascript">
  /* Conteúdo do JavaScript */
  // Função que formata os valores numéricos
  function formata (){
    // cria as variáveis com os valores a serem formatados
    var valor1 = 123456.78910111213;
    var porcentagem = 0.123456789;
    // exibe o valor sem formatação na caixa de texto com id=sf
    document.getElementById("sf").value = valor1;
    // cria a variável numReal com a formatação de número real
    var numReal = new NumberFormat();
    // cria a variável numPerc com a formatação de número em porcentagem
    var numPerc = new NumberFormat();
    // define que o número a ser formatado por numReal contém . como separador de
casas decimais
    numReal.setInputDecimal('.');
    // define que o número a ser formatado por numPerc contém . como separador de
casas decimais
    numPerc.setInputDecimal('.');
    // define que o número a ser formatado por numReal é o valor1
    numReal.setNumber(valor1);
    // define que o número a ser formatado por numPerc é a porcentagem
    numPerc.setNumber(porcentagem);
    // define a quantidade de casas decimais do número a ser formatado por numReal
    numReal.setPlaces('3', false);
    // define a quantidade de casas decimais do número a ser formatado por numPerc
    numPerc.setPlaces('4', false);
    // define o símbolo monetário do número a ser formatado por numReal
    numReal.setCurrencyValue('R$ ');
    // define a posição do símbolo monetário do número a ser formatado por numReal
    numReal.setCurrencyPosition(numReal.LEFT_OUTSIDE);
    // define os separadores do número a ser formatado por numReal
    numReal.setSeparators(true, ',', '.');
    // define os separadores do número a ser formatado por numPerc
    numPerc.setSeparators(true, ',', '.');
    // exibe o valor com formatação na caixa de texto com id=cf
    document.getElementById("cf").value = numReal.toFormatted();
    // exibe o valor com porcentagem na caixa de texto com id=pc
    document.getElementById("pc").value = numPerc.toPercentage();
    // define a quantidade de casas decimais do número a ser formatado por numReal
    numReal.setPlaces('2', false);
```

### Exemplo08.html (continuação)

```
// define que o símbolo monetário do número a ser formatado por numReal vai ser
exibido
numReal.setCurrency(true);
// exhibe o valor em moeda na caixa de texto com id=vc
document.getElementById("vc").value = numReal.toFormatted();
}
</script>
</head>
<!-- executa o script da função formata() ao carregar a página -->
<body onload="formata()">
<table>
<tr>
<td> <label for="sf">Nº sem formata&ccedil;&atilde;o:</label></td>
<td> <input type="text" name="sf" id="sf" readonly="true"></td>
</tr>
<tr>
<td> <label for="cf">Nº com formata&ccedil;&atilde;o:</label></td>
<td> <input type="text" name="cf" id="cf" readonly="true"></td>
</tr>
<tr>
<td> <label for="vc">Valor da Compra:</label></td>
<td> <input type="text" name="vc" id="vc" readonly="true"></td>
</tr>
<tr>
<td> <label for="pc">Porcentagem:</label></td>
<td> <input type="text" name="pc" id="pc" readonly="true"></td>
</tr>
</table>
</body>
</html>
```

A saída do programa acima é exibida abaixo:



Nº sem formatação:	123456.78910111212
Nº com formatação:	123.456.789
Valor da Compra:	R\$ 123.456.79
Porcentagem:	12.3457%



## Exercício (T1)

11-) Insira as seguintes formatações de números na calculadora (feita no exercício 7), onde cada formatação deverá ter o seu botão adequado. Ajuste o layout da GUI conforme achar melhor (Java e HTML).

**Aumentar casas decimais** → Aumentará o valor de Resultado em uma casa decimal.

**Diminuir casas decimais** → Diminuirá o valor de Resultado em uma casa decimal.

**Formato em Moeda** → Apresentará o valor de Resultado em formato monetário.

**Formato em Porcentagem** → Apresentará o valor de Resultado em formato porcentagem.

**Formato em Reais** → Apresentará o valor de Resultado em formato com casas decimais.

1º Núm.		+	-
2º Núm.		x	/
Resultado		Quadrado	Potência
Raiz	0.00	R\$	0.00%
	>>	<<	Limpar