2 – Adicionando componentes JLabel ou rótulos na sua janela

Freqüentemente chamado de rótulo, esse componente raramente tem seu conteúdo alterado e, quando usado corretamente, possibilita manipulações bem interessantes, como veremos a segui. Vamos começar escrevendo um aplicativo que permite instanciar as classe JLabel de maneiras diferentes.
Esse aplicativo que adiciona JLabel, posiciona na janela, altera a cor e altera a fonte desse componente:

```
import javax.swing.*;
import java.awt.*;
public class ExemploLabel extends JFrame{
  JLabel rotulo1,rotulo2,rotulo3,rotulo4;
  public ExemploLabel(){
    super("Exemplo com Label");
    Container tela = getContentPane();
    setLayout(null);
        rotulo1 = new JLabel ("Nome");
        rotulo2 = new JLabel ("Idade");
        rotulo3 = new JLabel ("Telefone");
        rotulo4 = new JLabel ("Celular");
        rotulo1.setBounds(50,20,80,20);
        rotulo2.setBounds(50,60,80,20);
        rotulo3.setBounds(50,100,80,20);
        rotulo4.setBounds(50,140,80,20);
        rotulo1.setForeground(Color.red);
        rotulo2.setForeground(Color.blue);
        rotulo3.setForeground(new Color(190,152,142));
        rotulo4.setForeground(new Color(201,200,100));
        rotulo1.setFont(new Font("Arial",Font.BOLD,14));
        rotulo2.setFont(new Font("Comic Sans MS",Font.BOLD,16));
        rotulo3.setFont(new Font("Courier New",Font.BOLD,18));
        rotulo4.setFont(new Font("Times New Roman",Font.BOLD,20));
        tela.add(rotulo1);
        tela.add(rotulo2);
        tela.add(rotulo3);
        tela.add(rotulo4);
        setSize(400, 250);
        setVisible(true);
        setLocationRelativeTo(null);
  }
  public static void main(String args[]){
        ExemploLabel app = new ExemploLabel();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

Veja o programa depois de executado:

Para entender melhor como funciona o programa destacamos alguns trechos do programa, sempre lembrando que temos que no começo do programa importar esses pacotes que adicionam conteúdos a janela:

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

2.1 - Apos fazermos as importações vamos criar os objetos do tipo JLabel:

```
JLabel rotulo1,rotulo2,rotulo3,rotulo4;
```

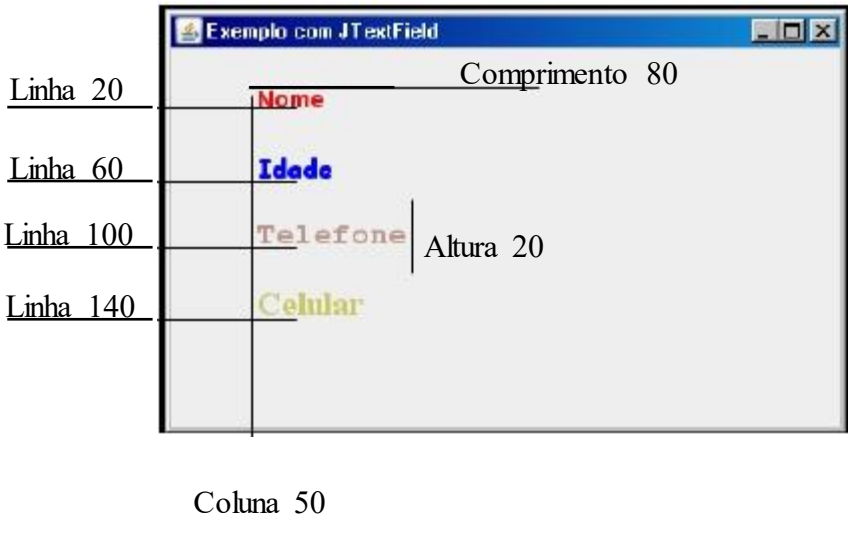
2.2 - Em seguida atribuir o conteúdo ao JLabel:

```
rotulo1 = new JLabel ("Nome");
rotulo2 = new JLabel ("Idade");
rotulo3 = new JLabel ("Telefone");
rotulo4 = new JLabel ("Celular");
```

2.3 - Próximo passo definir o largura e altura do JLabel e a coluna e a linha que ele irá ocupar na janela.

```
rotulo1.setBounds(50,20,80,20);
rotulo2.setBounds(50,60,80,20);
rotulo3.setBounds(50,100,80,20);
rotulo4.setBounds(50,140,80,20);
```

50 – Coluna, 20 – linha, 80 – largura, 20 - comprimento



2.4 - Próximo passo definir a cor da letra dos componentes JLabel, lembrando que se esses objetos não forem adicionados ao aplicativo, a cor default é preto.

```
rotulo1.setForeground(Color.red);
rotulo2.setForeground(Color.blue);
```

2.5 - Esses dois trechos abaixo especifica como criar cores personalizadas para o componente JLabel.

```
rotulo3.setForeground(new Color(190,152,142));
rotulo4.setForeground(new Color(201,200,100));
```

2.6 - E ainda podemos definir a fonte, o estilo e o tamanho da letra do componente JLabel.

```
rotulo1.setFont(new  Font("Arial",Font.BOLD,14));
rotulo2.setFont(new  Font("Comic Sans MS",Font.BOLD,16));
rotulo3.setFont(new  Font("Courier New",Font.BOLD,18));
rotulo4.setFont(new  Font("Times New Roman",Font.BOLD,20));
```

Lembrando que o nome da fonte tem que ser definida como esta na opção de fonte do sistema.

Exemplo: Comic Sans MS,
Courier New,
Arial, Lucida Sans,
Tahoma,
Times New Roman

Já o estilo pode ser definido como:

Font.BOLD	fonte em negrito
Font.ITALIC	fonte em itálico
Font.PLAIN	fonte normal
Font.BOLD+Font.ITALIC	negrito e itálico

E a numeração corresponde o tamanho da fonte.

```
rotulo1.setFont(new  Font("Arial",Font.BOLD,14));
```

Você pode definer 14, 16, 18, 20 entre outros tamanhos que fica a escolha do usuário.

E finalmente o método tela que representa a janela deverá ser chamado, onde vai exibir o rótulo o conteúdo do JLabel na janela.

```
tela.add(rotulo1);
tela.add(rotulo2);
tela.add(rotulo3);
tela.add(rotulo4);
```

2.7 – Adicionando imagem ao componente JLabel

É Possível exibir imagem em rótulos como instâncias da classe JLabel. Tais rótulos podem conter apenas imagens ou imagens e texto. O aplicativo seguinte mostra apenas uma imagem adicionada com um JLabel na janela.
Exemplo:

```
import javax.swing.*;
import java.awt.*;
public class LabelImagem extends JFrame{
  JLabel imagem;
  public LabelImagem(){
    super("Uso da classe JLabel com Imagem");
    Container tela = getContentPane();
    ImageIcon icone = new ImageIcon("sapo.jpeg");
    imagem = new JLabel(icone);
    tela.add(imagem);
    setSize(500, 460);
    setVisible(true);
  }
  public static void main(String args[]){
    LabelImagem app = new LabelImagem();
    app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

3 – Adicionando componentes JTextField ou caixa de texto na sua janela

A classe JTextField possibilita a criação de caixas de texto de uma única linha. Os usos mais freqüentes desse controle são para receber e validar dados informados pelo usuário do aplicativo. Essa classe herda a maioria de seus atributos, eventos e métodos da classe JTextComponent.

```
import javax.swing.*;
import java.awt.*;
public class ExemploJTextField extends JFrame{
  JLabel rotulo1,rotulo2,rotulo3,rotulo4;
  JTextField texto1,texto2,texto3,texto4;
  public ExemploJTextField (){
    super("Exemplo com JTextField");
    Container tela = getContentPane();
    setLayout(null);
    rotulo1 = new JLabel ("Nome");
    rotulo2 = new JLabel ("Idade");
    rotulo3 = new JLabel ("Telefone");
    rotulo4 = new JLabel ("Celular");
    texto1  = new JTextField(50);
    texto2  = new JTextField(3);
    texto3  = new JTextField(10);
    texto4  = new JTextField(10);
    rotulo1.setBounds(50,20,80,20);
    rotulo2.setBounds(50,60,80,20);
    rotulo3.setBounds(50,100,80,20);
    rotulo4.setBounds(50,140,80,20);
    texto1.setBounds(110,20,200,20);
    texto2.setBounds(110,60,20,20);
    texto3.setBounds(110,100,80,20);
    texto4.setBounds(110,140,80,20);
    tela.add(rotulo1);
    tela.add(rotulo2);
    tela.add(rotulo3);
    tela.add(rotulo4);
    tela.add(texto1);
    tela.add(texto2);
    tela.add(texto3);
    tela.add(texto4);
    setSize(400, 250);
    setVisible(true);
    setLocationRelativeTo(null);
  }

  public static void main(String args[]){
        ExemploJTextField app = new ExemploJTextField();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

Vejamos agora os trechos principais que fazem com que as caixas de texto apareçam na janela.Declarar os objetos: TextField texto1,texto2,texto3,texto4;

Estipular a quantidade de caracteres para as caixas de texto:

```
texto1 = new JTextField(50);   texto2 = new JTextField(3);
texto3 = new JTextField(10);   texto4 = new JTextField(10);
```

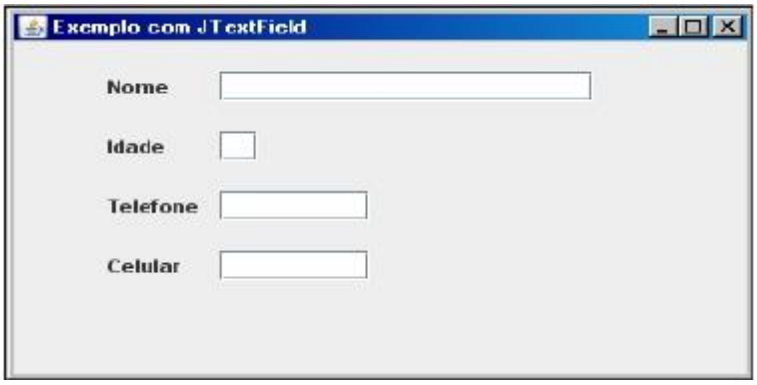
Especificar posicionamento das caixas:

```
texto1.setBounds(110,20,200,20);
texto2.setBounds(110,60,20,20);
texto3.setBounds(110,100,80,20);
texto4.setBounds(110,140,80,20);
```

E chamar o método tela para exibir as caixas na janela:

```
tela.add(texto1);
tela.add(texto2);
tela.add(texto3);
tela.add(texto4);
```

Estamos vendo agora a janela com os componentes acionados.



3.1 – Foco para uma caixa de texto

Podemos também utilizar o comando requestFocus(); para apontar o foco para uma determinada caixa de texto: texto1. requestFocus();

3.2 – Cor da fonte para a caixa de texto

Podemos também utilizar o comando setForeground(Color.cor); para alterar a cor da fonte da caixa de texto: texto1. setForeground(Color.cor);

3.3 – Trocando a fonte, estilo e tamanho da fonte do JtextField:

```
Texto1.setFont(new Font("NomeFonte",Font.ESTILO,TAMANHO));
```

3.4 – Alinhamento do texto de uma caixa de texto

O comando setHorizontalAlignment() define o alinhamento dentro da caixa de texto: texto1. setHorizontalAlignment(JTextField.LEFT);

setHorizontalAlignment (JTextField.RIGHT); posiciona o texto dentro da caixa de compra da direita pra esquerda,

setHorizontalAlignment(JTextField.LEFT); posiciona o texto dentro da caixa da esquerda pra direita.

setHorizontalAlignment(JTextField.CENTER); centraliza o texto dentro da caixa de texto.

3.5 – Alteração da cor de fundo da caixa de texto

Como visto anteriormente podemos efetuar essa alteração chamando o comando: texto1.setBackground(Color.blue);

4 – Controle JFormattedTextField para data, telefone, CEP, CPF

O componente JFormattedTextField, similar a uma caixa de texto JTextField, fornece a possibilidade da validação e exibição de dados formatados em tempo real, ao seja, à medida que seu conteúdo é inserido. Esse componente é conhecido nas linguagens Visual Basic, Delphi, C++ como MaskEdit, ou simplesmente caixa de texto tipo máscara.

A vantagem de usar uma caixa de texto tipo máscara é que o usuário é forçado a digitar os dados no formato especificado. O controle não aceita conteúdo que não siga os padrões predefinidos.

Antes de usar o componente JFormattedTextField, é preciso entender o funcionamento da classe MaskFormatter (do pacote javax.swing.text). Essa classe é usada para especificar a máscara que será usada na caixa de texto.

Para definir os caracteres que serão aceitos, você deve usar a seguinte tabela:

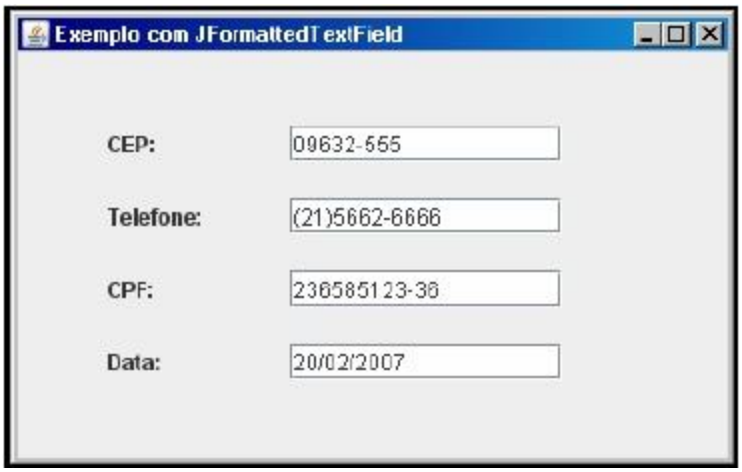
Símbolo	Valor Aceito
#	Um número
?	Uma letra
A	Uma letra ou um número
*	Qualquer conteúdo será aceito pelo controle
U	Uma letra convertida em maiúsculo
L	Uma letra convertida em minúsculos

Obtida a tabela vamos ver essa classe em ação. O aplicativo seguinte exibe quatro caixas de texto na qual deve ser digitado o CEP, TELEFONE, CPF e uma DATA qualquer.

import javax.swing.*;

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

public class ExemploJFormattedTextField extends JFrame{
 JLabel rotulocep,rotulotel,rotulocpf,rotulodata;
 JFormattedTextField  cep,tel, cpf, data;
 MaskFormatter  mascaracep,mascaratel,  mascaracpf, mascaradata;

  public ExemploJFormattedTextField(){
   super("Exemplo com JFormattedTextField");
   Container tela = getContentPane();
   setLayout(null);

   rotulocep = new JLabel("CEP: ");
   rotulotel = new JLabel("Telefone: ");
   rotulocpf = new JLabel("CPF: ");
   rotulodata = new JLabel("Data: ");
   rotulocep.setBounds(50,40,100,20);
   rotulotel.setBounds(50,80,100,20);
   rotulocpf.setBounds(50,120,100,20);
   rotulodata.setBounds(50,160,100,20);

  try{
        mascaracep = new MaskFormatter("#####-###");
        mascaratel  = new MaskFormatter("(##)#####-#####");
        mascaracpf = new MaskFormatter("##########-##");
        mascaradata = new MaskFormatter("##/##/####");
        mascaracep.setPlaceholderCharacter('_');
        mascaratel.setPlaceholderCharacter('_');
        mascaracpf.setPlaceholderCharacter('_');
        mascaradata.setPlaceholderCharacter('_');
     }
    catch(ParseException excp){}

   cep = new JFormattedTextField(mascaracep);
   tel = new JFormattedTextField(mascaratel);
   cpf = new JFormattedTextField(mascaracpf);
   data = new JFormattedTextField(mascaradata);
   cep.setBounds(150,40,100,20);
   tel.setBounds(150,80,100,20);
   cpf.setBounds(150,120,100,20);
   data.setBounds(150,160,100,20);
   tela.add(rotulocep);
   tela.add(rotulotel);
   tela.add(rotulocpf);
   tela.add(rotulodata);
   tela.add(cep);
   tela.add(tel);
   tela.add(cpf);
```

```
    tela.add(data);

   setSize(400, 250);
    setVisible(true);
  }
  public static void main(String args[]){
        ExemploJFormattedTextField app = new ExemploJFormattedTextField();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```



Para construir o exemplo, usamos as variáveis globais:

```
    FormattedTextField  cep,tel,  cpf, data;
MaskFormatter  mascaracep,mascaratel,  mascaracpf,  mascaradata;
```

O código que inicializa a máscara foi inserido em um bloco try{ } de modo a evitar uma exceção:

```
   try
   {
        mascaracep  = new MaskFormatter("#####-###");
        mascaratel   = new MaskFormatter("(##)####-####");
        mascaracpf = new MaskFormatter("#########-##");
        mascaradata  = new MaskFormatter("##/##/####");
        mascaracep.setPlaceholderCharacter('_');
        mascaratel.setPlaceholderCharacter('_');
        mascaracpf.setPlaceholderCharacter('_');
        mascaradata.setPlaceholderCharacter('_');
     }
   catch(ParseException  excp)
   {}
```

Observe o uso do símbolo # para permitir somente valores numéricos e uma chamada ao método setPlaceholderCharacter para definir o caractere que será exibido como marcador para a máscara, geralmente o caractere de sublinhado. Essa máscara é fornecida como argumento para o construtor da classe JFormattedTextField:

```
cep  = new JFormattedTextField(mascaracep);
tel  = new JFormattedTextField(mascaratel);
cpf  = new JFormattedTextField(mascaracpf);
data = new JFormattedTextField(mascaradata);
```

4.1 – Adicionando cores na letra do JFormattedTextField

Essas linhas podem ser adicionadas no programa anterior acima do objeto tela.

```
cep.setForeground(Color.blue);
tel.setForeground(Color.red);
cpf.setForeground(Color.green);
data.setForeground(new  Color(255,128,128));
```

4.2 – Adicionando cores no fundo do JFormattedTextField

Já esses códigos podem ser adicionados logo abaixo ao anterior.

```
cep.setBackground(Color.yellow);
tel.setBackground(Color.yellow);
cpf.setBackground(new Color(255,255,204));
data.setBackground(new  Color(255,255,204));
```

4.3 – Alterando o tipo, estilo, tamanho da fonte do JFormattedTextField

Já esses códigos podem ser adicionados logo abaixo ao anterior.

```
cep.setFont(new  Font("Times New Roman",Font.BOLD,14));
tel.setFont(new  Font("Comic Sans MS",Font.PLAIN,14));
cpf.setFont(new  Font("Arial",Font.BOLD,14));
data.setFont(new  Font("Tahoma",Font.BOLD,14));
```

Aplicativo sendo executado depois de todos os códigos Java, estarem adicionados no aplicativo, veja os efeitos com as cores:

5. JTextArea

5.1– Entendimento da classe JTextArea

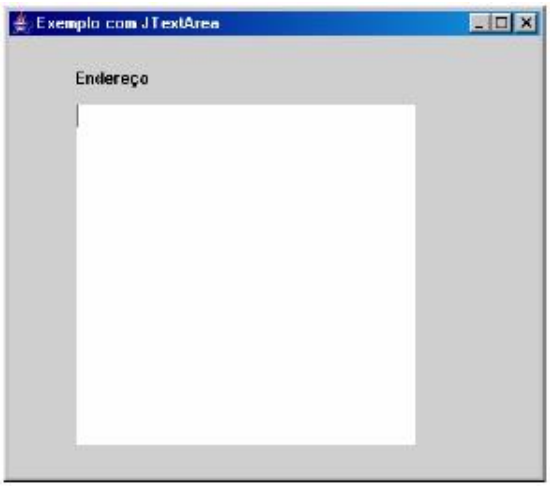
A classe JTextField permite a criação de caixas de textos que aceitam a entrada de apenas uma linha de texto. A classe JTextArea permite a entrada e manipulação de múltiplas linhas de texto. Emobra a classe JTextArea forneça uma série de métodos úteis para a manipulação de seu conteúdo, algumas operações mais avançadas, tais como formatar e aplicar estilos diversos ao texto de controle, são possíveis apenas com o uso da classes JTextPane e JeditorPane. Esta apostila não aborda estas duas classes mas, você poderá pesquisa-las na documentação Java.

A classe JTextArea fornece seis construtores. Dentre eles, o mais comumente usado é aquele que cria uma área de texto em branco com um determinado número de linhas e colunas. Veja a sintaxe para esse construtor:

JTextArea(int linhas, int colunas).

A listagem seguinte apresenta um aplicativo que contém um rótulo, uma área de texto. Somente para o usuário visualizar como funciona o aplicativo:

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

public class ExemploJTextArea extends JFrame
{
 JLabel rotulo;
 JTextArea texto;
 public ExemploJTextArea(){
   super("Exemplo com JTextArea");
   Container tela = getContentPane();
   tela.setLayout(null);
   rotulo = new JLabel ("Endereço");
   texto = new JTextArea(20,30);
   rotulo.setBounds(50,20,100,20);
   texto.setBounds(50,50,250,250);
   tela.add(rotulo);
   tela.add(texto);
   setSize(400, 350);
   setVisible(true);
   setLocationRelativeTo(null);
 }

 public static void main(String args[]){
        ExemploJTextArea app = new ExemploJTextArea();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }
}
```



Este aplicativo apresenta poucas novidades. Começamos definindo uma variável chamada texto que servirá para instanciarmos a área de texto:
JTextArea texto;

A área de texto é criada com a seguinte instrução:
texto = new JTextArea(20,30);

Nosso controle vai exibir 20 linhas de texto e 30 colunas. É importante observar que esses valores definem apenas o conteúdo que será visível no controle e não a quantidade de texto que poderá ser inserida.

5.2– Definição das barras de rolagem para os controles JTextArea

O aplicativo que digitamos no tópico anterior apresentou o uso da classe JTextArea para construir áreas de texto que aceitem mais de uma linha de conteúdo. O leitor deve ter percebido que, se a quantidade de texto exceder o espaço reservado para o controle, parte de seu conteúdo não poderá ser visualizado devido à ausência das barras de rolagens. A listagem seguinte é uma pequena modificação do exemplo anterior que demostra como a classe JScrollPane pode ser usada para adicionar barras de rolagem a objetos da classe JTextArea:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ExemploJTextArea3 extends JFrame{
  JLabel rotulo;
  JTextArea texto;
  JScrollPane painelrolagem;
  JPanel painel;
  public ExemploJTextArea3(){
    super("Exemplo com JTextField");
    Container tela = getContentPane();
    tela.setLayout(null);
    rotulo = new JLabel ("Endereço");
    rotulo.setBounds(50,20,100,20);
    texto = new JTextArea(10,20);
    painelrolagem  = new JScrollPane(texto);
    painelrolagem.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
```

```
  painelrolagem.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
   painel = new JPanel();
   painel.add(painelrolagem);
   painel.setBounds(40,40,250,250);
   tela.add(rotulo);
   tela.add(painel);
   setSize(300, 280);
   setVisible(true);
   setLocationRelativeTo(null);
 }
 public static void main(String args[]){
         ExemploJTextArea3 app = new ExemploJTextArea3();
         app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }
}
```

Como este aplicativo é bem semelhante ao do tópico anterior, vamos nos concentrar apenas nos trechos de códigos relevantes à adição das barras de rolagem.

Definindo as variáveis: JLabel rotulo; JTextArea texto;
JScrollPane painelrolagem; JPanel painel;

A área de texto é criada com a seguinte linha: texto = new JTextArea(10,20);

Veja que criamos uma instância da classe JTextArea da forma usual. Em seguida a linha: painelrolagem = new JScrollPane(texto);

Cria uma instância da classe JScrollPane fornecendo a referência texto como argumento. É importante observar que, ao usar esse construtor, as dimensões do painel de rolagem serão definidas de modo acomodar a área de texto criada previamente.

Por padrão, as barras de rolagem definidas com o construtor anterior serão exibidas somente quando o conteúdo da área de texto exceder sua capacidade de exibição. Neste exemplo optamos por sempre mostrar as barras de rolagem. Isso foi conseguido com as duas instruções seguintes:

painelrolagem.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROL
LBAR_ALWAYS);

painelrolagem.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_S
CROLLBAR_ALWAYS);

O aparecimento ou não das barras de rolagem vertical e horizontal é controlado pelos métodos **setVerticalScrollBarPolicy** e **setHorizontalScrollBarPolicy** Esses dois métodos aceitam os valores de constantes na classe JScrollPane.

Para que seus aplicativos sigam os padrões predefinidos, use as constantes que exibirão as barras de rolagem somente quando forem necessárias.

Neste aplicativo fizemos uso da classe **JPanel** para incluir as barras de rolagem. Com isso temos mais controle sobre a disposição dos componentes que estamos usando. Ela se destaca apenas um truque que usamos para definir o espaçamento de todos os controles corretamente.

painel = new JPanel();

Para finalizar a análise, é adicionado no painel as barras de rolagem (painelderolagem), pois as barras de rolagem quando estavam sendo instanciadas pelo JScrollPane, recebeu o a área de texto (texto).

painel.add(painelrolagem);

- posiciona o painel na janela: painel.setBounds(40,40,250,250);

- adiciona o painel na janela: tela.add(painel);

A tabela seguinte relaciona essas constantes e descreve seu uso:

Constante	Uso
HORIZONTAL_SCROLL_AS_NEEDED	Exibe a barra de rolagem horizontal somente quando for necessário.
HORIZONTAL_SCROLL_NEVER	A barra de rolagem horizontal nunca é exibida.
HORIZONTAL_SCROLL_ALWAYS	Define que a barra de rolagem horizontal deve ser exibida sempre.
VERTICAL_SCROLL_AS_NEEDED	Exibe a barra de rolagem vertical somente quando for necessário.
VERTICAL _SCROLL_NEVER	A barra de rolagem vertical nunca é exibida.
VERTICAL_SCROLL_ALWAYS	Define que a barra de rolagem vertical deve ser exibida sempre.

5.3 – Controlando a quebra automática de linhas

Por padrão, controles criados como instância da classe JTextArea não oferecem a funcionalidade de quebra automática de linhas. Isso é incoveniente em algumas situações em que o usuário digita longos trechos de texto no

componente. Imagine algo em torno de 50 a 100 palavras sendo exibidas em apenas uma linha. Nada elegante. É possível forçar a quebra de linha automática nos objetos JTextArea com uma chamada ao método setLineWrap. Este método aceita os argumentos true ou false. Você tem que usar juntamente com o setLineWrap o método setWrapStyleWord para que haja a quebra de linha acontença entre palavras, se a próxima palavra não couber na linha atual, deve ser deslocada para a próxima linha. Este método aceita os argumentos true ou false.

Veja a sintaxe:

```
texto.setLineWrap(true);
texto.setWrapStyleWord(true);
```

Exemplo :

```
import javax.swing.*;
import java.awt.*;
public class ExemploJTextArea2 extends JFrame{
JLabel rotulo;
JTextArea texto;
public ExemploJTextArea2(){
super("Exemplo com JTextField");
Container tela = getContentPane();
setLayout(null);
rotulo = new JLabel ("Endereço");
texto = new JTextArea(20,30);
rotulo.setBounds(50,20,100,20);
texto.setBounds(50,50,300,250);
texto.setLineWrap(true);
texto.setWrapStyleWord(true);
tela.add(rotulo);
tela.add(texto);
setSize(400, 350);
setVisible(true);
setLocationRelativeTo(null);
}

public static void main(String args[]){
ExemploJTextArea2 app = new ExemploJTextArea2();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
 }
```

5.4 – Cor da fonte, Cor de fundo

Nada como alterar a cor da fonte do JTextArea, argumento igual ao JTextField

```
texto.setForeground(Color.blue);
texto.setBackground(Color.yellow);
```

5.5 – Tipo, estilo e tamanho de fonte

Nada como alterar a cor de fundo do JTextArea, argumento igual ao JTextField:

texto.setFont(new Font("Comic Sans MS",Font.BOLD+Font.ITALIC,20));

Exemplo :

```
import javax.swing.*;
import java.awt.*;
public class ExemploJTextArea4 extends JFrame{
JLabel rotulo;
JTextArea texto;
public ExemploJTextArea4(){
super("Exemplo com JTextArea");
Container tela = getContentPane();
tela.setLayout(null);
rotulo = new JLabel ("Endereço");
texto = new JTextArea(20,30);
rotulo.setBounds(50,20,100,20);
texto.setBounds(50,50,250,250);
texto.setForeground(Color.blue);
texto.setBackground(Color.yellow);
texto.setFont(new Font("Comic Sans MS",Font.BOLD+Font.ITALIC,20));
tela.add(rotulo);
tela.add(texto);
setSize(400, 350);
setVisible(true);
setLocationRelativeTo(null);
}

public static void main(String args[])
{
ExemploJTextArea4 app = new ExemploJTextArea4();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
 }
```