## 9 – Apresentando a classe JList

A classe JList permite a criação do componente conhecido como caixa de listagem ou ListBox (nome popular entre programadores VB e C#). Uma caixa de listagem é um controle visual que exibe uma série de itens dos quais o usuário pode selecionar um ou mais itens.

A maneira mais usual de criar um caixa de listagem em Java é usar um vetor de Strings para definirmos os itens que serão exibidos na lista. O aplicativo seguinte apresenta uma janela que contém uma caixa de listagem e um botão. O usuário pode selecionar uma entre oito cidades. Ao clicar no botão exibir, o valor selecionado será exibido em um JLabel.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ExemploJList extends JFrame{
        JList lista;
        String cidades[] = {"Rio de Janeiro","São Paulo","Minas Gerais","Espírito Santo",
        "Bahia","Pernambuco","Rio Grande do Sul","Acre"};
        JButton exibir;
        JLabel rotulo;
        public ExemploJList(){
                super("Exemplo de List");
                Container tela = getContentPane();
                setLayout(null);
                exibir = new JButton("Exibir");
                rotulo = new JLabel("");
                lista = new JList(cidades);
                lista.setVisibleRowCount(5);
                JScrollPane painelRolagem = new JScrollPane(lista);
                lista.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
                painelRolagem.setBounds(40,50,150,100);
                exibir.setBounds(270,50,100,30);
                rotulo.setBounds(50,150,200,30);
                exibir.addActionListener(
                      new ActionListener(){
                              public void actionPerformed(ActionEvent e){
                              rotulo.setText("o estado é: "+lista.getSelectedValue().toString());
                      }    );
                tela.add(painelRolagem);
                tela.add(exibir);
                tela.add(rotulo);
                setSize(400, 250);
               setVisible(true);
        }
  public static void main(String args[]){
        ExemploJList app = new ExemploJList();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

Experimente selecionar um dos itens da lista e clicar no botão Exibir. Observe que o texto do item selecionado é exibido em uma caixa de mensagem. Vamos analisar o código e ver como isso foi possível.

O primeiro passo é declarar uma instância da classe JList chamada lista:

```
JList lista;
```

Em seguida declaramos e inicializamos um vetor de objetos String que contém os itens que estarão disponíveis na list:

```
String cidades[] = {"Rio de Janeiro","São Paulo","Minas Gerais","Esprito Santo", "Bahia","Pernambuco","Rio Grande do Sul","Acre"};
```

Os itens do vetor cidades são atribuídos à caixa de listagem por meio da seguinte instrução:

```
lista = new JList(cidades);
```

É importante observar que o vinculo entre a caixa de listagem e o vetor cidades não termina após essa instrução. Mais tarde, se alguma alteração for feita nos elementos do vetor, você poderá atualizar a lista com uma chamada ao método repaint da classe JComponent.

```
lista.rapaint();
```

Após a atribuição dos itens do vetor cidades à lista, temos que definir a quantidade de elementos que serão visíveis sem a necessidade de acionar as barras de rolagem isso é feito na linha:

```
lista.setVisibleRowCount(5);
```

Como queremos que o usuário seja capaz de selecionar apenas um item por vez efetuaremos uma chamada ao método setSelectionMode e usaremos uma das constantes da classe ListSelectionModel para definir a lista como sendo de seleção única:

```
lista.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

Uma lista não fornece barras de rolagem por padrão. Dessa forma, uma instância da classe JScrollPane é usada para essa finalidade:

```
JScrollPane painelRolagem = new JScrollPane(lista);
```

Quando clicamos no botão exibir, o método actionPerformed da classe publica. Chamado:

```
exibir.addActionListener(
   new ActionListener(){
       public void actionPerformed(ActionEvent e){
           rotulo.setText(“o estado é: “+lista.getSelectedValue().toString());
}     }    );
```

Nós efetuamos uma chamada ao método getSelectedValue().toString(); da classe JList para obter o valor inteiro correspondente ao item selecionados. Esse valor inicia em 0 e vai até a quantidade de itens menos 1. Obtido o valor, só precisamos fornece-lo como índice para o valor o vetor cidades exibir o resultado final na caixa de listagem.

## 9.1 – Selecionando múltiplos itens em uma caixa de listagem

No tópico anterior escrevemos um aplicativo que permita selecionar um item em uma caixa de listagem e exibir seu valor e, um JLabel. No exemplo apenas um dos itens podia ser selecionado. É possível, porém, selecionar mais de um item em uma caixa de listagem. Tudo que temos a fazer é alterar o valor da constante fornecida para o método setSelectionMode. Veja os valores possíveis:

| Valor da Constante | Resultado |
|---|---|
| SINGLE_SELECTION | Apenas um item pode ser selecionado de cada vez. |
| SINGLE_INTERVAL_SELECTION | Vários itens em um intervalo contíguo podem ser selecionados. |
| MULTIPLE_INTERVAL_SELECTION | Vários podem ser selecionados sem a restrição de intervalo. |

O exemplo seguinte é uma pequena modificação do exemplo anterior que mostra como vários itens podem ser selecionados e como você pode atribuí-los a um vetor de objetos de modo a manipulá-los mais tarde:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ExemploJList2 extends JFrame{
        JList lista;
        String cidades[] = {"Rio de Janeiro","São Paulo","Minas Gerais","Espírito Santo",
        "Bahia","Pernambuco","Rio Grande do Sul","Acre"};
        JButton exibir;
        public ExemploJList2(){
                super("Exemplo de List");
                Container tela = getContentPane();
                setLayout(null);
                exibir = new JButton("Exibir");
                lista = new JList(cidades);
                lista.setVisibleRowCount(5);
                JScrollPane painelRolagem = new  JScrollPane(lista);
        lista.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
                painelRolagem.setBounds(40,50,150,100);
                exibir.setBounds(270,50,100,30);
                exibir.addActionListener(
          new ActionListener(){
             public void actionPerformed(ActionEvent e){
                Object selecionados[] = lista.getSelectedValues();
                String resultados = "Valores selecionados:\n";
                for(int i=0;i<selecionados.length;i++)
                  resultados += selecionados[i].toString()+"\n";
                JOptionPane.showMessageDialog(null,resultados);
              }  } );
                tela.add(painelRolagem);
                tela.add(exibir);
                setSize(400, 250);
      setVisible(true);
        }
  public static void main(String args[]){
        ExemploJList2 app = new  ExemploJList2();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

Execute o código e experimente selecionar mais de um item na caixa de listagem. Lembre-se de que mais de um item pode ser selecionados mantendo a tecla Crtl pressionada enquanto clicamos nos outros itens. Clique no botão exibir, e você verá a caixa de mensagem exibindo os valores dos itens selecionados.

Este exemplo apresenta poucas modificações. A primeira delas é a modificação do valor fornecido para o método setSelectionMode:

```
lista.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
```

Veja que agora a lista permite que vários itens sejam selecionados. A outra modificação aconteceu no evento do botão exibir:

```
exibir.addActionListener(
    new ActionListener(){
            public void actionPerformed(ActionEvent e){
            Object selecionados[] = lista.getSelectedValues();
            String resultados = "Valores selecionados:\n";
            for(int i=0;i<selecionados.length;i++)
              resultados += selecionados[i].toString()+"\n";
            JOptionPane.showMessageDialog(null,resultados);
}   }   );
```

A classe JList possui um método chamado getSelectedValues que retorna um vetor de objetos da classe Object. Este valor contém todos os itens selecionados na caixa de listagem. Assim, a linha:

```
Object selecionados[ ] = lista.getSelectedValues();
```

Quando foi adicionado \n no final da frase Valores selecionados, que como visto \n pula uma linha na próxima palavra que será exibida e não ao lado.

Criar um vetor chamado selecionados e atribui a ele os itens retornados pelo método getSelectedValues. Mantenha em mente que temos um vetor objetos da classe Object. Esquecer-se deste detalhe pode ser desastroso. O laço for interage com cada um dos elementos do vetor selecionados, efetuando chamadas ao método String de cada um e concatenando o resultado obtido com o valor da variável resultados.

Cada vez que selecionados jogar um item dentro de resultado, cada item cada item vai com a opção de pular uma linha por isso os itens são exibidos um em baixo do outro.

A caixa de mensagem JOptionPane será estuda mais profundamente nos próximos capítulos da apostila. A opção showMessageDialog, que dizer uma mensagem de dialogo será exibida ao usuário.

### 9.2 – Trocando a cor da letra do JList

É possível trocar a cor da letra do componente JList abaixo do comando onde você define em qual linha e coluna vai ficar o componente você pode digitar essa linha: lista.setForeground(Color.blue);

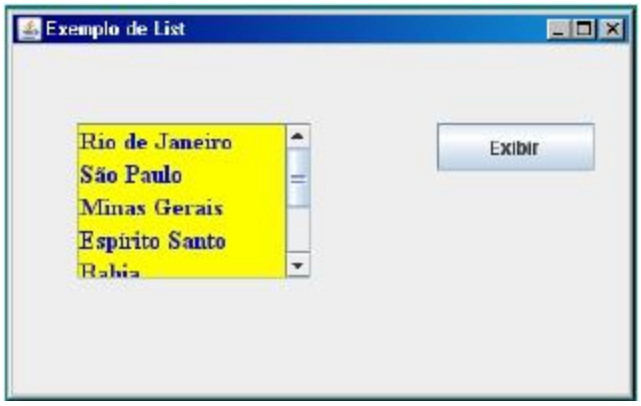
### 9.3 – Trocando a cor de fundo do JList

lista.setBackground(Color.yellow);

### 9.4 – Trocando tipo, estilo e tamanho da fonte do Jlist

lista.setFont(new Font("Times New Roman",Font.BOLD,16));

Execute o aplicativo ExemploJList3.java que você poderá conferir onde aplicar os comandos para as alterações citadas acima.




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

public class ExemploJList3 extends JFrame{
JList lista;
String cidades[] = {"Rio de Janeiro","São Paulo","Minas Gerais","Espírito Santo",
"Bahia","Pernambuco","Rio Grande do Sul","Acre"};
JButton exibir;
JLabel rotulo;
public ExemploJList3(){
super("Exemplo de List");
Container tela = getContentPane();
setLayout(null);
exibir = new JButton("Exibir");
rotulo = new JLabel("");
lista = new JList(cidades);
```

```
lista.setVisibleRowCount(5);
JScrollPane painelRolagem = new JScrollPane(lista);
lista.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
painelRolagem.setBounds(40,50,150,100);
exibir.setBounds(270,50,100,30);
rotulo.setBounds(50,150,200,30);
lista.setForeground(Color.blue);
lista.setBackground(Color.yellow);
lista.setFont(new Font("Times New Roman",Font.BOLD,16));
exibir.addActionListener(
new ActionListener(){
public void actionPerformed(ActionEvent e){
rotulo.setText("o estado é: "+lista.getSelectedValue().toString());
}
}
);
tela.add(painelRolagem);
tela.add(exibir);
tela.add(rotulo);
setSize(400, 250);
setVisible(true);
}

public static void main(String args[]){
ExemploJList3 app = new ExemploJList3();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
 }
```

## 10 – uso da classe JComboBox na criação de caixas de combinação

Objetos da classe JComboBox são usados para a criação de controles conhecidos como caixas de combinação, combo box ou lista drop-down. A diferença entre a caixa de listagem e caixa de combinação é que esta última exibe seus itens somente quando clicamos na seta (indicativo visual de que o controle possui itens entre os quais podemos escolher) ou pressionando F4 quando o foco está no componente. Além, disso, caixas de combinação são usadas quando o espaço é insuficiente para acomodar uma caixa de listagem.

A classe JcomboBox herda de JComponent, como mostra o relacionamento da aula 1.

O aplicativo seguinte apresenta uma janela que contém uma caixa de combinação e um botão. Quando selecionamos um item no JComboBox e pressionamos o botão, Exibir o valor do item selecionado é exibido em uma caixa de mensagem. Eis a listagem para o exemplo:

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

public class ExemploJComboBox extends JFrame{
        JComboBox lista;
        String cidades[] = {"Rio de Janeiro","São Paulo","Minas Gerais","Espírito
        Santo","Bahia","Pernambuco","Rio Grande do Sul","Acre"};
        JButton exibir;
        JLabel rotulo;
        public ExemploJComboBox(){
                super("Exemplo de JComboBox");
                Container tela = getContentPane();
                setLayout(null);
                exibir = new JButton("Exibir");
                rotulo = new JLabel("");
                lista = new JComboBox(cidades);
                lista.setMaximumRowCount(5);
                lista.setBounds(50,50,150,30);
                exibir.setBounds(270,50,100,30);
                rotulo.setBounds(50,150,200,30);
        exibir.addActionListener(
        new ActionListener(){
                public void actionPerformed(ActionEvent e){
                rotulo.setText("o estado é: "+lista.getSelectedItem().toString()); } }   );
                tela.add(lista);
                tela.add(exibir);
                tela.add(rotulo);
                setSize(400, 250);
                setVisible(true);
        }

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

Existem muitas similaridades na maneira de manipular as classes JList e JComboBox. Durante a análise você perceberá que muitas das técnicas que usamos anteriormente podem ser aplicadas quando estiver lidando com caixas de combinação. O primeiro passo foi declarar um objeto da classe JComboBox:

JComboBox Lista;

Em seguida temos um vetor de nomes de cidades que será usado para definimos os itens da caixa de combinação:

String cidades[] = {"Rio de Janeiro","São Paulo","Minas Gerais","Espírito Santo","Bahia","Pernambuco","Rio Grande do Sul","Acre"};

Temos um vetor de oito elementos que são atribuídos à caixa de combinação com a seguinte instrução:

lista = new JComboBox(cidades);

Um objeto da classe JComboBox já possui barras de rolagem. Tiramos proveito desse fator efetuamos uma chamada ao método setMaximumRowCount para definir a quantidade e itens visíveis na lista sem a necessidade de acionar as barras de rolagem:

lista.setMaximumRowCount(5);

O processo usado para exibir o item selecionado é o mesmo que usamos para as caixas de listagem.

```
exibir.addActionListener(
new ActionListener(){public void actionPerformed(ActionEvent e){
      rotulo.setText("o estado é: “+lista.getSelectedItem().toString());
}});
tela.add(lista);
tela.add(exibir);
tela.add(rotulo);
setSize(400, 250);
setVisible(true);
}
```

### 10.1 – Como criar uma caixa de combinação editável

Se você experimentou com o exemplo anterior deve ter percebido que a caixa de combinação é somente leitura, ou seja, estamos limitados aos itens disponíveis para a escolha.

Existe, porém, uma forma de permitir que o usuário digite o valor desejado em uma caixa de combinação. Basta efetuar uma chamada ao método setEditable fornecendo o valor true como argumento:

lista.setEditable(true);

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

public class ExemploJComboBox2 extends JFrame{
        JComboBox lista;
        String cidades[] = {"Rio de Janeiro","São Paulo","Minas Gerais","Espírito
                        Santo","Bahia","Pernambuco","Rio Grande do Sul","Acre"};
        JButton exibir;
        JLabel rotulo;
        public ExemploJComboBox2(){
                super("Exemplo de JComboBox");
                Container tela = getContentPane();
                setLayout(null);
                exibir = new JButton("Exibir");
```

```
                rotulo = new JLabel("");
                lista = new JComboBox(cidades);
                lista.setEditable(true);
                lista.setMaximumRowCount(5);
                lista.setBounds(50,50,150,30);
                exibir.setBounds(270,50,100,30);
                rotulo.setBounds(50,150,200,30);
                exibir.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                rotulo.setText("o estado é: "+lista.getSelectedItem().toString());
                }});
                tela.add(lista);
                tela.add(exibir);
                tela.add(rotulo);
                setSize(400, 250);
                setVisible(true);
        }

    public static void main(String args[]){
            ExemploJComboBox2 app = new  ExemploJComboBox2();
            app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    } }
```

Agora o usuário será capaz de escolher um dos itens disponíveis ou digitar sua escolha. É comum em campos editáveis o pressionamento da tecla enter após a entrada da informação no campo. Desse forma, é importante efetuar uma chamada ao método addActionListener da caixa de combinação e definir o procedimento a ser efetuado. O trecho do código seguinte mostra como exibir o valor que o usuário acaba de informar:

```
        exibir.addActionListener
        (
          new ActionListener()
          {
            public void actionPerformed(ActionEvent  e)
            {
             rotulo.setText("o  estado é: "+lista.getSelectedItem().toString());
             }
          }
);
```