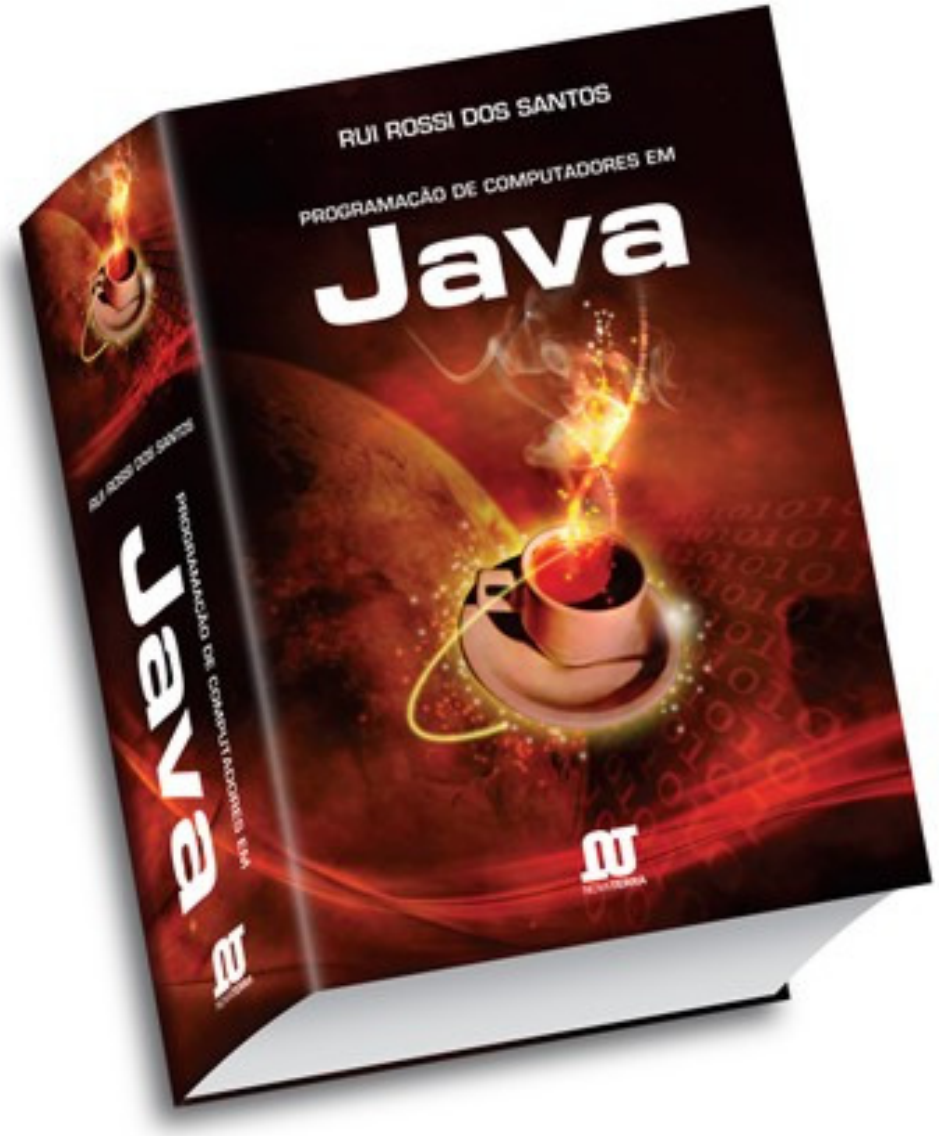


Capítulo 28

Componentes de Seleção

Objetivos do Capítulo

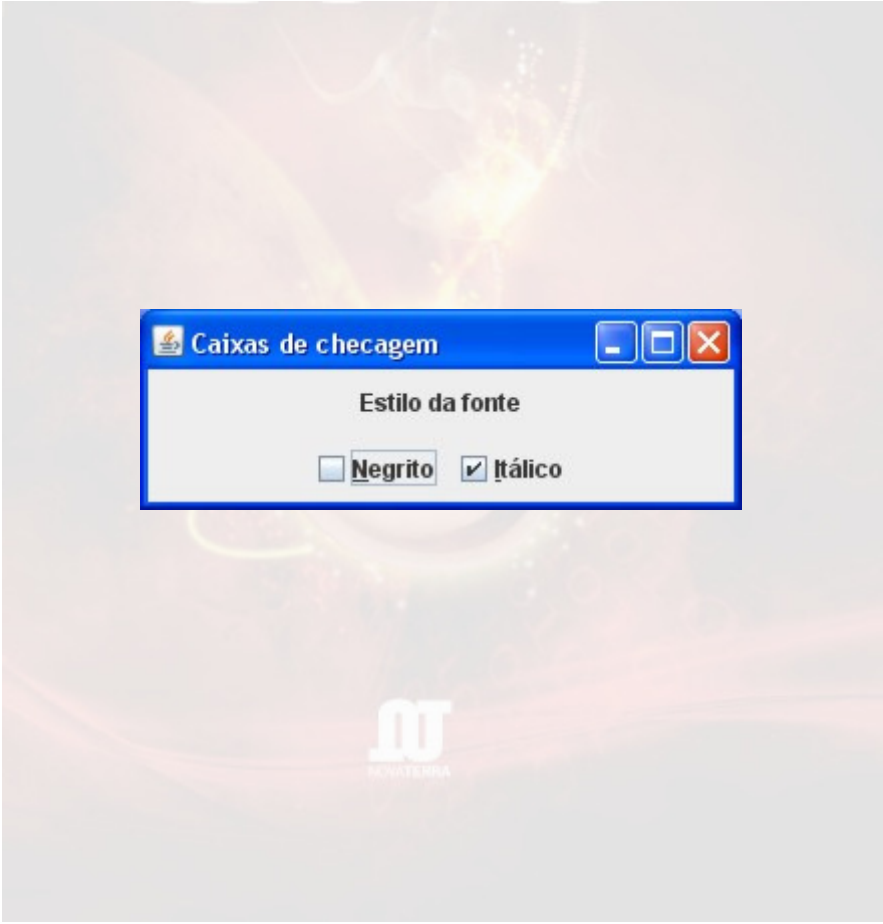
- Apresentar quatro tipos de componentes de seleção: caixas de checagem, botões de rádio, caixas de combinação e listas.
- Construir exemplos práticos que demonstram como criar, configurar e utilizar este tipo de componentes.
- Analisar os eventos que são gerados especificamente por este tipo de componentes e indicar o modo adequado de tratá-los.

Caixas de Checagem

- javax.swing.JCheckBox extends JToggleButton ...
  - Construtores:
    - JCheckBox( )
    - JCheckBox(String text)
    - JCheckBox(String text, boolean selected)
  - Métodos:
    - setMnemonic(int mnemonic)
    - setSelected(boolean b)
    - boolean isSelected( )

Caixas de Checagem

- **Código 28.1 – CaixaChecagem.java**



Botões de Rádio

- javax.swing.JRadioButton extends JToggleButton ...
  - Construtores:
    - JRadioButton( )
    - JRadioButton(String text)
    - JRadioButton(String text, boolean selected)
  - Métodos:
    - setMnemonic(int mnemonic)
    - setSelected(boolean b)
    - boolean isSelected( )
  - Importante:
    - Devem ser agrupados (adicionados a um javax.swing.ButtonGroup)

Botões de Rádio

- **Código 28.2 – BotaoRadio.java**

Caixas de Combinação

- javax.swing.JComboBox extends JComponent ...
  - Construtores:
    - JComboBox( )
    - JComboBox(ComboBoxModel aModel)
    - JComboBox(Object[] items)
    - JComboBox(Vector<?> items)
  - Métodos:
    - setMaximumRowCount(int count)
    - int getMaximumRowCount( )
    - setEditable( )
    - boolean isEditable( )
    - setSelectedIndex(int anIndex)
    - int getSelectedIndex( )
    - setModel(ComboBoxModel aModel)
    - ComboBoxModel getModel( )
    - addItem(Object anObject)
    - Object getItemAt(int index)
    - int getItemCount( )

Caixas de Combinação

- javax.swing.DefaultComboBoxModel implements ComboBoxModel
  - Construtores:
    - DefaultComboBoxModel( )
    - DefaultComboBoxModel(Object[] items)
    - DefaultComboBoxModel(Vector<?> v)
  - Métodos:
    - addElement(Object anObject)
    - Object getElementAt(int index)
    - int getIndexOf(Object anObject)
    - Object getSelectedItem( )
    - int getSize( )
    - insertElementAt(Object anObject, int index)
    - removeAllElements( )
    - removeElement(Object anObject)
    - removeElementAt(int index)
    - setSelectedItem(Object anObject)

Caixas de Combinação

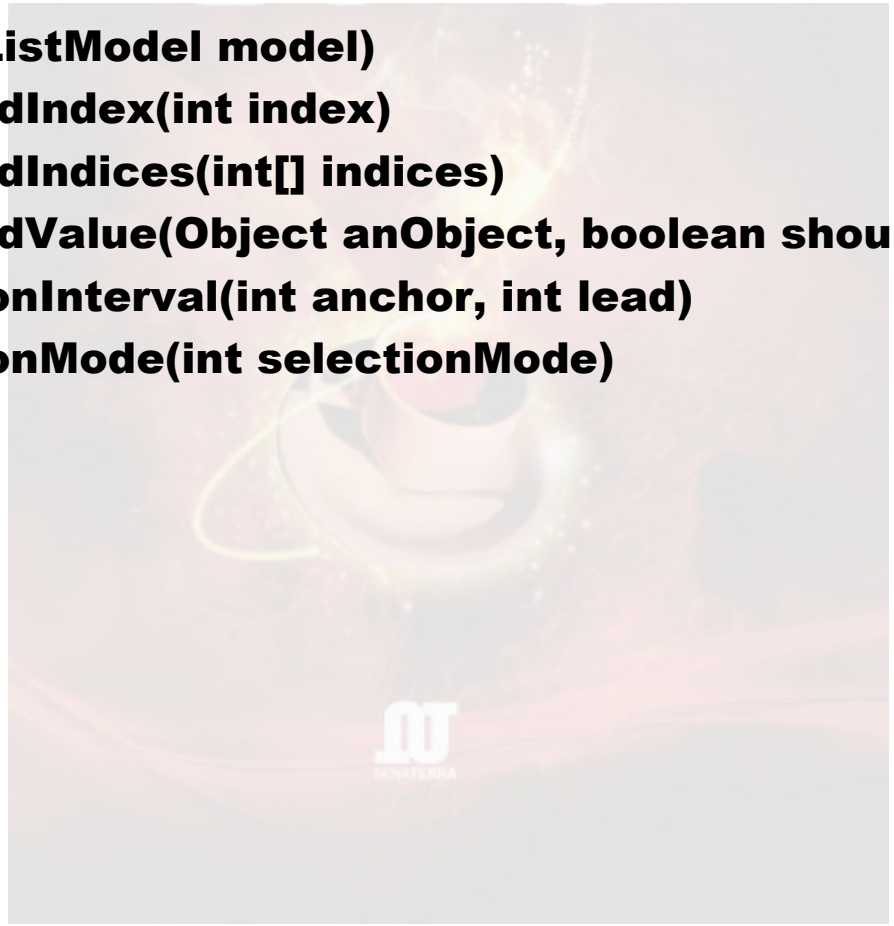
- **Código 28.3 – CaixaCombinacao.java**

Listas

- javax.swing.JList extends JComponent ...
  - Construtores:
    - JList( )
    - JList(ListModel dataModel)
    - JList(Object[] listData)
    - JList(Vector<?> listData)
  - Métodos:
    - int getMinSelectionIndex( )
    - int getMaxSelectionIndex( )
    - ListModel getModel( )
    - int getSelectedIndex( )
    - int[] getSelectedIndices( )
    - Object getSelectedValue( )
    - Object[] getSelectedValues( )
    - int getSelectionMode( )
    - ListSelectionModel getSelectionModel( )
    - boolean isSelectionEmpty( )

Listas

- interface javax.swing.List extends JComponent ...
  - Métodos:
    - setModel(ListModel model)
    - setSelectedIndex(int index)
    - setSelectedIndices(int[] indices)
    - setSelectedValue(Object anObject, boolean shouldScroll)
    - setSelectionInterval(int anchor, int lead)
    - setSelectionMode(int selectionMode)



Listas

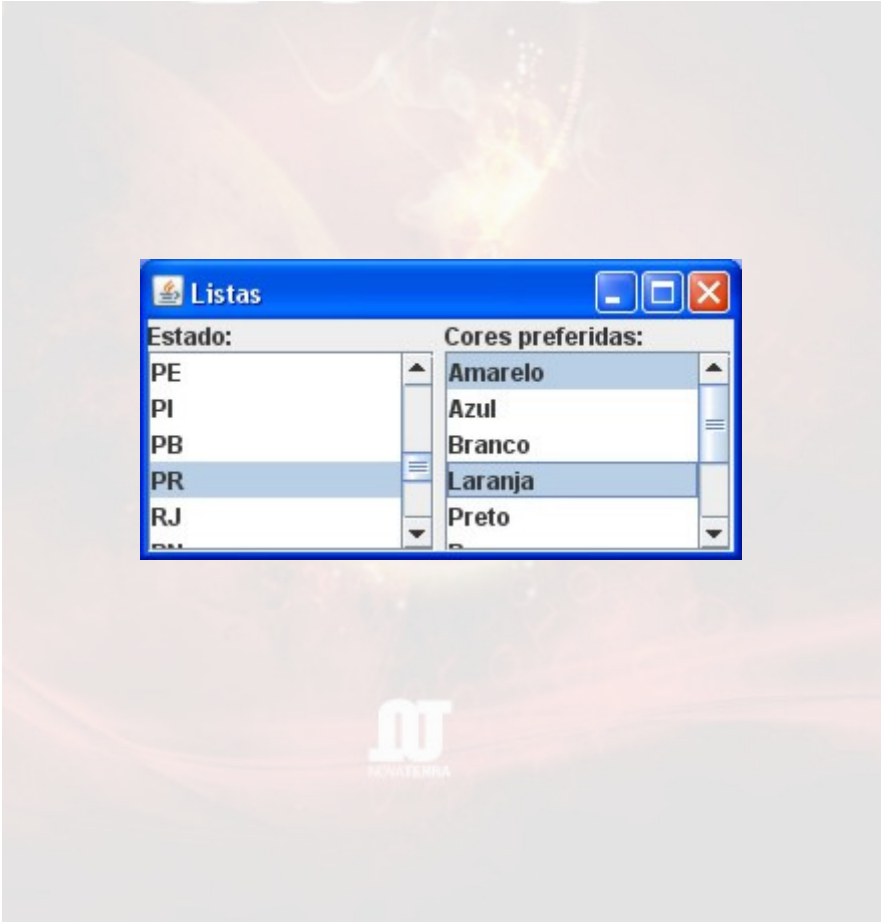
- **interface javax.swing.ListSelectionModel**
  - **Atributos:**
    - **MULTIPLE_INTERVAL_SELECTION**
    - **SINGLE_INTERVAL_SELECTION**
    - **SINGLE_SELECTION**
  - **Métodos:**
    - **int getMinSelectionIndex( )**
    - **int getMaxSelectionIndex( )**
    - **int getSelectionMode( )**
    - **boolean isSelectionEmpty( )**
    - **setSelectionInterval(int index0, int index1)**
    - **setSelectionMode(int selectionMode)**

Listas

- **javax.swing.DefaultListModel implements ListModel**
  - **Construtores:**
    - **DefaultListModel( )**
  - **Métodos:**
    - **add(int index, Object element)**
    - **addElement(Object obj)**
    - **clear( )**
    - **boolean contains(Object elem)**
    - **Object elementAt(int index)**
    - **Object get(int index)**
    - **int indexOf(Object elem)**
    - **int lastIndexOf(Object elem)**
    - **boolean isEmpty( )**
    - **removeAllElements( )**
    - **removeElement(Object obj)**
    - **removeElementAt(int index)**
    - **removeRange(int fromIndex, int toIndex)**
    - **set(int index, Object element)**
    - **setElementAt(Object obj, int index)**

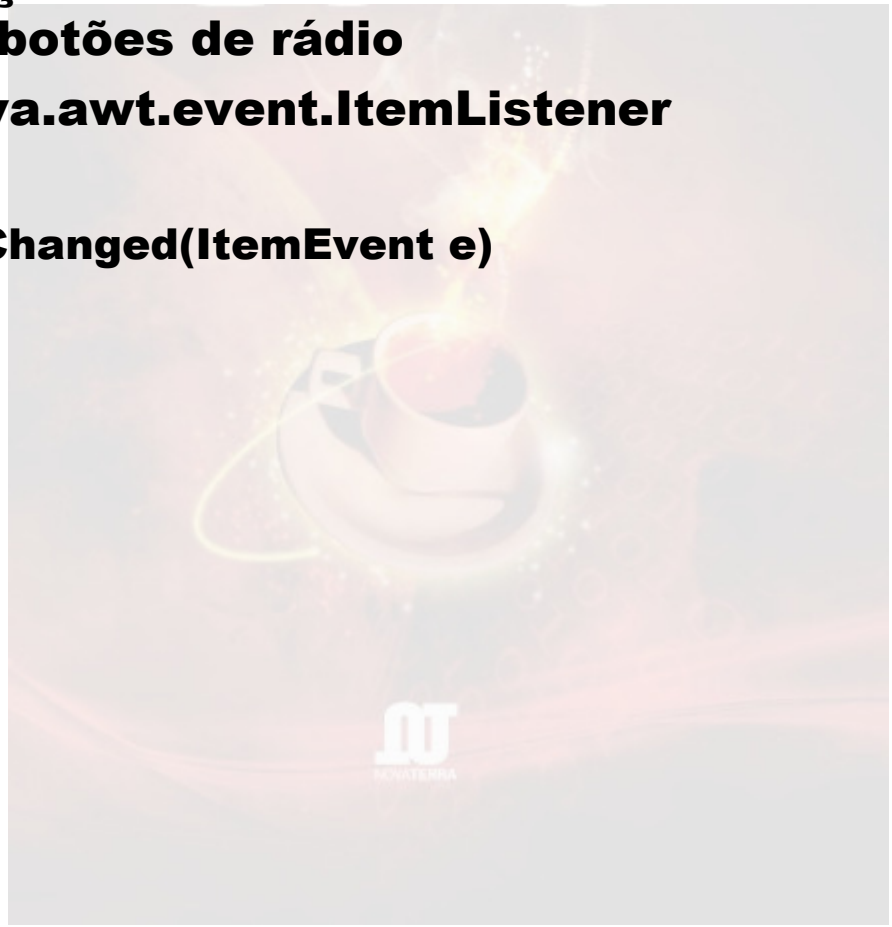
Listas

- **Código 28.4 – Lista.java**



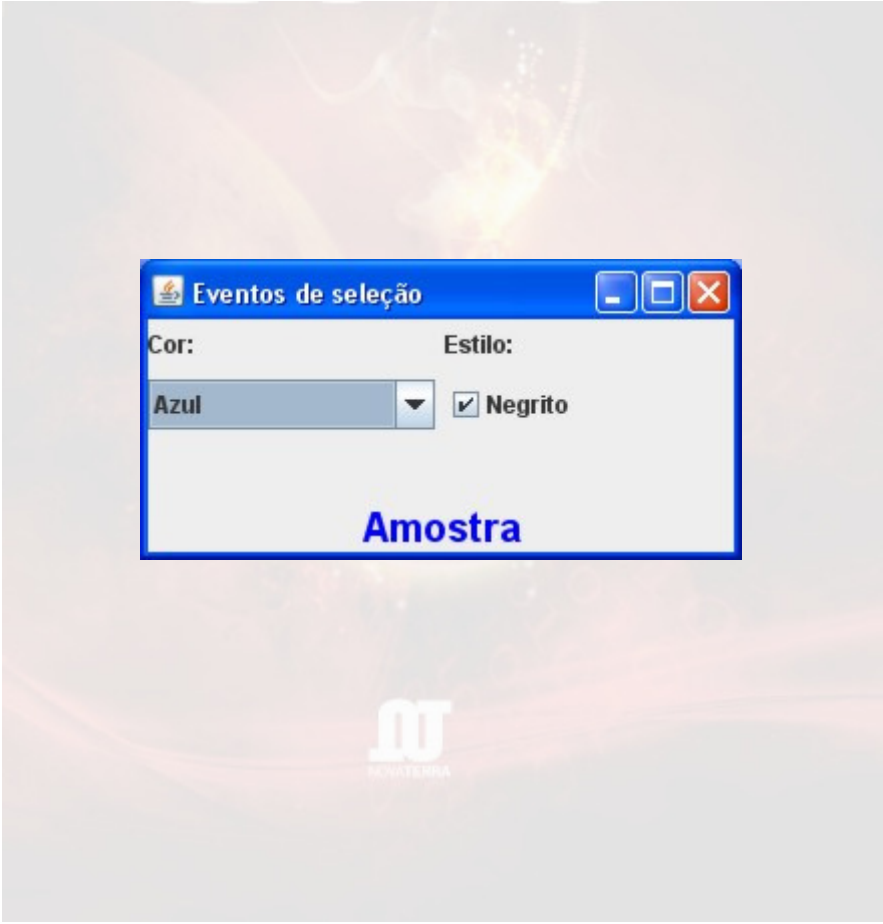
Eventos de Seleção

- **java.awt.event.ItemEvent**
  - **Gatilho: seleção de item em caixas de combinação, caixas de checagem e botões de rádio**
  - **Interface: java.awt.event.ItemListener**
  - **Métodos:**
    - **itemStateChanged(ItemEvent e)**



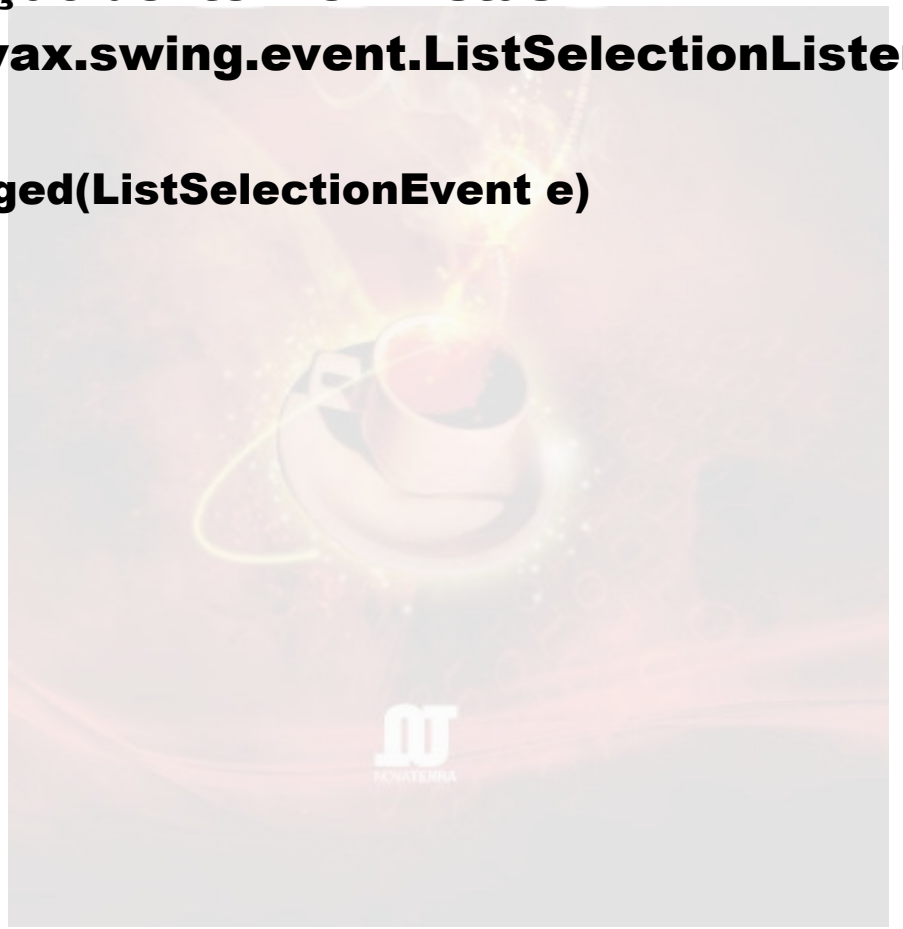
Eventos de Seleção

- **Código 28.5 – EventoItem.java**



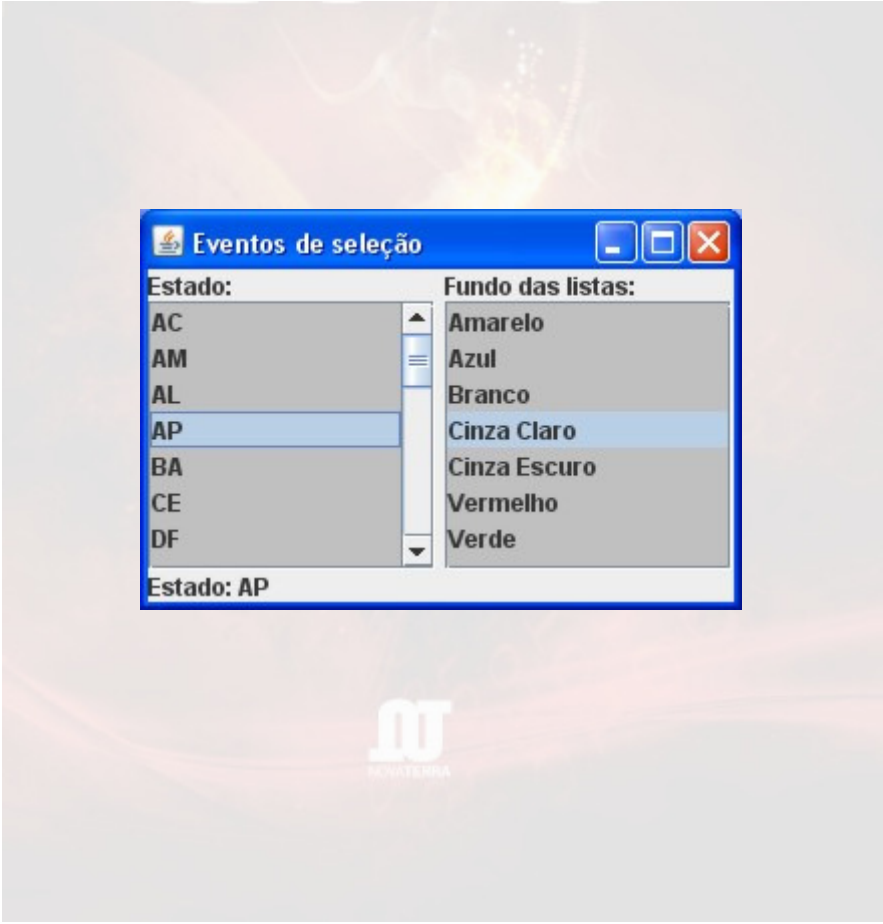
Eventos de Seleção

- **javax.swing.event.ListSelectionEvent**
  - **Gatilho: seleção de item em listas**
  - **Interface: javax.swing.event.ListSelectionListener**
  - **Métodos:**
    - **valueChanged(ListSelectionEvent e)**



Eventos de Seleção

- **Código 28.6 – EventoLista.java**



Exercício 1

- Crie uma nova janela, chamada ConfiguraFonte, de acordo com o modelo apresentado na figura abaixo.
- Esta janela deve ter duas caixas de combinação, duas caixas de checagem e um rótulo.
  - A primeira caixa de combinação deve ter o nome de fontes disponíveis no seu sistema operacional.
  - A segunda caixa de combinação deve ter diferentes tamanhos que possam ser selecionados para a fonte de um texto.
  - As duas caixas de checagem poderão ser marcadas para aplicar os estilos correspondentes a uma fonte.

Exercício 1

- Ao invés de digitar os nomes de fontes que são suportadas em seu sistema operacional para carregá-las na primeira caixa de combinação, procure recuperá-las utilizando a classe java.awt.GraphicsEnvironment. A instrução abaixo produz um vetor de strings contendo os nomes de todas as famílias de fontes suportadas. Você pode utilizar o retorno deste método para carregar as opções da primeira caixa de combinação.

```
GraphicsEnvironment.getLocalGraphicsEnvironment().
  getAvailableFontFamilyNames();
```

- Defina os tamanhos que estarão disponíveis na segunda caixa de combinação observando um intervalo de cinco unidades entre eles. Sugere-se que o menor tamanho seja 10 e que o maior seja 100.
- A formatação da fonte do rótulo deve ser atualizada sempre que houver a alteração da seleção em uma das caixas de combinação ou quando uma caixa de checagem for marcada ou desmarcada.
  - Quando a janela for apresentada pela primeira vez, o rótulo já deve estar formatado de acordo com as opções selecionadas nas caixas de combinação e de acordo com o estado das caixas de checagem.

Exercício 2

- Crie uma nova janela, chamada ListaProdutos, de acordo com o modelo apresentado na figura abaixo.
- Esta janela deve permitir que seja registrado qualquer número de produtos e deve gravar o nome de todos eles em uma lista.
- Além disso, ela também deve permitir a remoção de nomes de produtos que se encontram nesta lista.

Exercício 2

- Sugere-se que a lista seja criada utilizando uma instância da classe javax.swing.DefaultListModel como modelo. Dessa forma, você conseguirá utilizar métodos que permitem inserir e remover elementos nesta lista. Para criá-la desta forma, você pode utilizar um objeto anônimo da classe supracitada como argumento para seu construtor.
- Além disso, você deve criar esta lista de modo que ela só permita que um único elemento seja selecionado por vez. Veja como você pode criar e formatar uma lista de acordo com o que fora descrito:

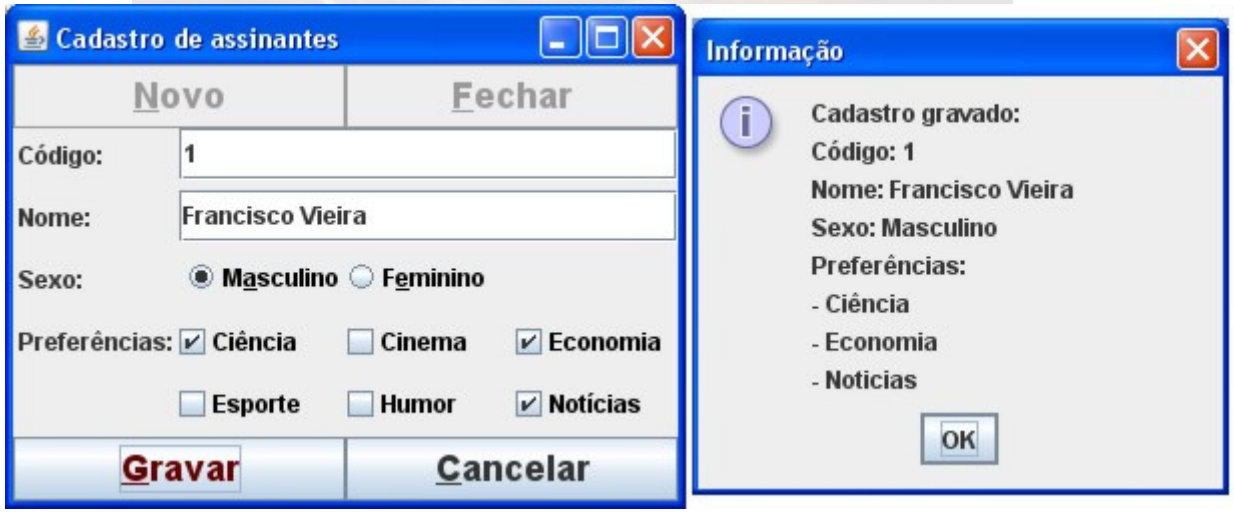
```
JList lista = new JList(new DefaultListModel());
lista.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

Exercício 2

- Quando o botão “Adicionar” for pressionado, o nome informado deve ser adicionado na lista.
  - Este nome deve ter de 5 a 50 caracteres para ser válido.
  - Se o nome for inválido, uma mensagem de erro deve ser exibida e ele não deve ser adicionado.
- Quando o botão “Remover” for pressionado, o item da lista que estiver selecionado deve ser removido.
  - Se não houver nenhum item selecionado, uma mensagem de erro deve ser exibida.

Exercício 3

- Crie uma nova janela, chamada CadastroAssinante, de acordo com o modelo apresentado na figura abaixo.
- Quando esta janela for apresentada pela primeira vez, o botão “Novo” e o botão “Fechar” devem estar habilitados e todos os outros componentes devem estar desabilitados.
- Quando o botão “Novo” for pressionado, você deve desabilitar ele e o botão “Fechar”, habilitar os demais componentes e enviar o foco para o campo “Código”.
- Se o botão “Fechar” for pressionado, a janela deve ser fechada.



Exercício 3

- Quando o botão “Gravar” for pressionado, você deve exibir uma mensagem confirmando a gravação do cadastro, deve retornar todos os componentes ao seu estado inicial e retornar o foco ao botão “Novo”.
- Se o botão “Cancelar” for pressionado, todos os componentes devem retornar ao seu estado inicial e o foco deve retornar ao botão “Novo”.
- Implemente um tratamento para os eventos de foco e de mouse para todos os botões, campos de texto e componentes de seleção.
  - Enquanto um campo de texto estiver com o foco, seu fundo deve ficar amarelo.
  - Enquanto um botão ou um componente de seleção estiver com o foco, o seu texto deve ter a cor alterada para vermelho escuro.
  - Sempre que o mouse passar sobre qualquer um destes componentes, ele deve receber imediatamente o foco.

Contato

Com o autor:

Rui Rossi dos Santos
E-mail: livros@ruirossi.pro.br
Web Site: http://www.ruirossi.pro.br

Com a editora:

Editora NovaTerra
Telefone: (21) 2218-5314
Web Site: http://www.editoranovaterra.com.br