Aula 3

- JMenu
- JPopupMenu
- JInternalFrame
- Aparências
- Interface de Múltiplos Documentos

© 2005 by Pearson Education do Brasil

22.4 Utilizando menus com frames

- **Menus:**
  - **Permitem que o usuário realize ações sem poluir desnecessariamente uma GUI com componentes extras.**
  - **Só podem ser anexados a objetos das classes que fornecem membros `setMenuBar`, como `JFrame` e `Japplet`.**
  - **Classe `MenuBar`:**
    - **Contém os métodos necessários para gerenciar uma barra de menus.**
  - **Classe `Jmenu`:**
    - **Contém os métodos necessários para gerenciar menus.**
  - **Classe `JMenuItem`:**
    - **Contém os métodos necessários para gerenciar itens de menu.**
      - **Pode ser utilizado para iniciar uma ação ou pode ser um submenu.**

© 2005 by Pearson Education do Brasil

22.4 Utilizando menus com frames (*Continuação*)

- Classe `JCheckBoxMenuItem`:
  - Contém os métodos necessários para gerenciar itens de menu que podem ser ativados ou desativados.
- Classe `JRadioButtonMenuItem`.
  - Contém os métodos necessários para gerenciar itens de menu que podem ser ativados ou desativados como `JCheckBoxMenuItem`s.
  - Quando múltiplos `JRadioButtonMenuItem`s são mantidos como parte de um `ButtonGroup`, apenas um item no grupo pode ser selecionado de cada vez.
- Mnemônicos:
  - Caracteres especiais que fornecem acesso rápido a um menu ou item do menu a partir do teclado.

© 2005 by Pearson Education do Brasil

Resumo

MenuFrame.java

(1 de 8)

```
1  // Fig. 22.5: MenuFrame.java
2  // Demonstrando menus.
3  import java.awt.Color;
4  import java.awt.Font;
5  import java.awt.BorderLayout;
6  import java.awt.event.ActionListener;
7  import java.awt.event.ActionEvent;
8  import java.awt.event.ItemListener;
9  import java.awt.event.ItemEvent;
10 import javax.swing.JFrame;
11 import javax.swing.JRadioButtonMenuItem;
12 import javax.swing.JCheckBoxMenuItem;
13 import javax.swing.JOptionPane;
14 import javax.swing.JLabel;
15 import javax.swing.SwingConstants;
16 import javax.swing.ButtonGroup;
17 import javax.swing.JMenu;
18 import javax.swing.JMenuItem;
19 import javax.swing.JMenuBar;
20
```

© 2005 by Pearson Education do Brasil

Resumo

MenuFrame.java

(2 de 8)

```
21 public class MenuFrame extends JFrame
22 {
23    private final Color colorValues[] =
24       { Color.BLACK, Color.BLUE, Color.RED, Color.GREEN };
25    private JRadioButtonMenuItem colorItems[]; // itens do menu Color
26    private JRadioButtonMenuItem fonts[]; // itens do menu Font
27    private JCheckBoxMenuItem styleItems[]; // itens do menu Font Style
28    private JLabel displayJLabel; // exibe o texto de exemplo
29    private ButtonGroup fontButtonGroup; // gerencia itens do menu Font
30    private ButtonGroup colorButtonGroup; // gerencia itens do menu Color
31    private int style; // utilizado para criar estilo para fonte
32
33    // construtor sem argumentos configura a GUI
34    public MenuFrame()
35    {
36       super( "Using JMenus" );
37
38       JMenu fileMenu = new JMenu( "File" ); // cria o menu File
39       fileMenu.setMnemonic( 'F' ); // configura o mnemônico como F
40
41       // create About... menu item
42       JMenuItem aboutItem = new JMenuItem( "About..." );
43       aboutItem.setMnemonic( 'A' ); // configura o mnemônico com
44       fileMenu.add( aboutItem ); // adiciona o item about ao men
45       aboutItem.addActionListener(
46
```

Cria um **JMenu**

Chama o método **JMenu setMnemonic**

Adiciona o **JMenuItem** ‘**About…**’ a **fileMenu**

© 2005 by Pearson Education do Brasil

```
47          new ActionListener() // classe interna anônima
48          {
49             // exibe um diálogo de mensagem quando o usuário sel
50             public void actionPerformed( ActionEvent event )
51             {
52                JOptionPane.showMessageDialog( MenuFrame.this,
53                   "This is an example\nof using menus",
54                   "About", JOptionPane.PLAIN_MESSAGE );
55             } // fim do método actionPerformed
56          } // fim da classe interna anônima
57       ); // fim da chamada para addActionListener
58
59       JMenuItem exitItem = new JMenuItem( "Exit" ); // cria o item exit
60       exitItem.setMnemonic( 'x' ); // configura o mnemônico como x
61       fileMenu.add( exitItem ); // adiciona o item exit ao menu File
62       exitItem.addActionListener(
63
64          new ActionListener() // classe interna anônima
65          {
66             // termina o aplicativo quando o usuário clicar em exitItem
67             public void actionPerformed( ActionEvent event )
68             {
69                System.exit( 0 ); // encerra o aplicativo
70             } // fim do método actionPerformed
71          } // fim da classe interna anônima
72       ); // fim da chamada para addActionListener
73
```

Cria um **ActionListener** para processar o evento de ação de **aboutItem**

Exibe uma caixa de diálogo de mensagem.

Cria e adiciona o item do menu **exitItem**

Registra um **ActionListener** que termina a aplicação

© 2005 by Pearson Education do Brasil

```
74       JMenuBar bar = new JMenuBar(); // cria a barra de menu
75       setJMenuBar( bar ); // adiciona a barra de menu ao aplicat
76       bar.add( fileMenu ); // adiciona o menu File à barra de me
77
78       JMenu formatMenu = new JMenu( "Format" ); // cria o menu Format
79       formatMenu.setMnemonic( 'r' ); // configura mnemônico como r
80
81       // array listando cores de string
82       String colors[] = { "Black", "Blue", "Red", "Green" };
83
84       JMenu colorMenu = new JMenu( "Color" ); // cria o menu Color
85       colorMenu.setMnemonic( 'C' ); // configura mnemônico c
86
87       // cria itens do menu Color com botões de opção
88       colorItems = new JRadioButtonMenuItem[ colors.length ];
89       colorButtonGroup = new ButtonGroup(); // gerencia cores
90       ItemHandler itemHandler = new ItemHandler(); // handler para
91
92       // cria itens do menu Color com botões de opção
93       for ( int count = 0; count < colors.length; count++ )
94       {
95          colorItems[ count ] =
96             new JRadioButtonMenuItem( colors[ count ] ); // cria o
97          colorMenu.add( colorItems[ count ] ); // adiciona o item a
98          colorButtonGroup.add( colorItems[ count ] ); // adiciona ao grupo
99          colorItems[ count ].addActionListener( itemHandler );
100      } // fim de for
101
```

Adiciona **fileMenu** a um **JMenuBar** e anexa o **JMenuBar** à janela da aplicação

MenuFrame.java

Cria um menu **formatMenu**

Cria um submenu **colorMenu**

Cria um array **colorItems** para **JRadioButtonMenuItem**

Cria um **ButtonGroup** a fim de assegurar que somente um dos itens de menu seja selecionado em determinado momento

Adiciona **JRadioButtonMenuItem**s a **colorMenu** e registra **ActionListener**s

© 2005 by Pearson Education do Brasil

```
102      colorItems[ 0 ].setSelected( true ); // seleciona primeiro item Color
103
104      formatMenu.add( colorMenu ); // adiciona o menu Color ao menu F
105      formatMenu.addSeparator(); // adiciona o separador no menu
106
107      // array listing font names
108      String fontNames[] = { "Serif", "Monospaced", "SansSerif" };
109      JMenu fontMenu = new JMenu( "Font" ); // cria o menu Font
110      fontMenu.setMnemonic( 'n' ); // configura o mnemônico como n
111
112      // cria itens do menu radiobutton para nomes de fonte
113      fonts = new JRadioButtonMenuItem[ fontNames.length ];
114      fontButtonGroup = new ButtonGroup(); // gerencia nomes de fo
115
116      // cria itens do menu Font com botões de opção
117      for ( int count = 0; count < fonts.length; count++ )
118      {
119         fonts[ count ] = new JRadioButtonMenuItem( fontNames[ count
120         fontMenu.add( fonts[ count ] ); // adiciona fonte ao menu Fo
121         fontButtonGroup.add( fonts[ count ] ); // adiciona ao grupo
122         fonts[ count ].addActionListener( itemHandler ); // adiciona handler
123      } // fim do for
124
125      fonts[ 0 ].setSelected( true ); // seleciona primeiro item de menu Font
126      fontMenu.addSeparator(); // adiciona barra separador ao menu Font
127
```

Invoca o método **AbstractButton setSelected**

MenuFrame.java

Adiciona **colorMenu** ao **formatMenu** e adiciona uma linha separadora horizontal

Cria um array **JRadioButtonMenuItem fonts**

Cria um **ButtonGroup** a fim de assegurar que somente um dos itens de menu seja selecionado em determinado momento

Adiciona **JRadioButtonMenuItem**s a **colorMenu** e registra **ActionListener**s

Configura a seleção-padrão e adiciona um separador horizontal

© 2005 by Pearson Education do Brasil

Resumo

MenuFrame.java

```
128     String styleNames[] = { "Bold", "Italic" }; // nomes de estilos
129     styleItems = new JCheckBoxMenuItem[ styleNames.length ];
130     StyleHandler styleHandler = new StyleHandler(); // handler de estilos
131
132     // criar itens do menu Style com caixas de seleção
133     for ( int count = 0; count < styleNames.length; count++ )
134     {
135        styleItems[ count ] =
136           new JCheckBoxMenuItem( styleNames[ count ] ); // para estilo
137        fontMenu.add( styleItems[ count ] ); // adicina ao menu Font
138        styleItems[ count ].addItemListener( styleHandler ); // handler
139     } // fim do for
140
141     formatMenu.add( fontMenu ); // adiciona menu Font ao m
142     bar.add( formatMenu ); // adiciona menu Format à barra de menu
143
144     // set up label to display text
145     displayJLabel = new JLabel( "Sample Text", SwingConstants.CENTER );
146     displayJLabel.setForeground( colorValues[ 0 ] );
147     displayJLabel.setFont( new Font( "Serif", Font.PLAIN, 72 ) );
148
149     getContentPane().setBackground( Color.CYAN ); // configura o fundo
150     add( displayJLabel, BorderLayout.CENTER ); // adiciona displayJLabel
151  } // fim de construtor MenuFrame
152
```

Cria **JCheckBoxMenuItem**s

Adiciona **fontMenu** a **formatMenu** e **formatMenu** a **JMenuBar**

© 2005 by Pearson Education do Brasil

Resumo

MenuFrame.java

```
153   // classe interna para tratar eventos de ação dos itens de menu
154   private class ItemHandler implements ActionListener
155   {
156      // processa seleções de cor e fonte
157      public void actionPerformed( ActionEvent event )
158      {
159         // processa seleções de cor
160         for ( int count = 0; count < colorItems.length; count++ )
161         {
162            if ( colorItems[ count ].isSelected() )
163            {
164               displayJLabel.setForeground( colorValues[ count ] );
165               break;
166            } // fim do if
167         } // fim do for
168
169         // processa seleção de fonte
170         for ( int count = 0; count < fonts.length; count++ )
171         {
172            if ( event.getSource() == fonts[ count ] )
173            {
174               displayJLabel.setFont(
175                  new Font( fonts[ count ].getText(), style,
176            } // fim do if
177         } // fim do for
178
```

Determina o **JRadioButtonMenuItem** selecionado

O método **getSource** retorna uma referência ao **JRadioButtonMenuItem** que gerou o evento

© 2005 by Pearson Education do Brasil

Resumo

MenuFrame.java

```
179         repaint(); // desenha novamente o aplicativo
180      } // fim do método actionPerformed
181   } // fim da classe ItemHandler
182
183   // classe interna para tratar eventos dos itens de menu com caixa de seleção
184   private class StyleHandler implements ItemListener
185   {
186      // processa seleções de estilo da fonte
187      public void itemStateChanged( ItemEvent e )
188      {
189         style = 0; // inicializa estilo
190
191         // verifica seleção de negrito
192         if ( styleItems[ 0 ].isSelected() )
193            style += Font.BOLD; // adiciona negrito ao estilo
194
195         // verifica seleção de itálico
196         if ( styleItems[ 1 ].isSelected() )
197            style += Font.ITALIC; // adiciona itálico ao estilo
198
199         displayJLabel.setFont(
200            new Font( displayJLabel.getFont().getName(), style, 72 ) );
201         repaint(); // desenha novamente o aplicativo
202      } // fim do método itemStateChanged
203   } // fim da classe StyleHandler
204} // fim da classe MenuFrame
```

Chamado se o usuário selecionar um **JCheckBoxMenuItem** no **fontMenu**

Determina se um ou ambos os **JCheckBoxMenuItem**s estão selecionados

© 2005 by Pearson Education do Brasil

Observação sobre aparência e comportamento 22.3

Os mnemônicos fornecem acesso rápido a comandos de menu e comandos de botão pelo teclado.

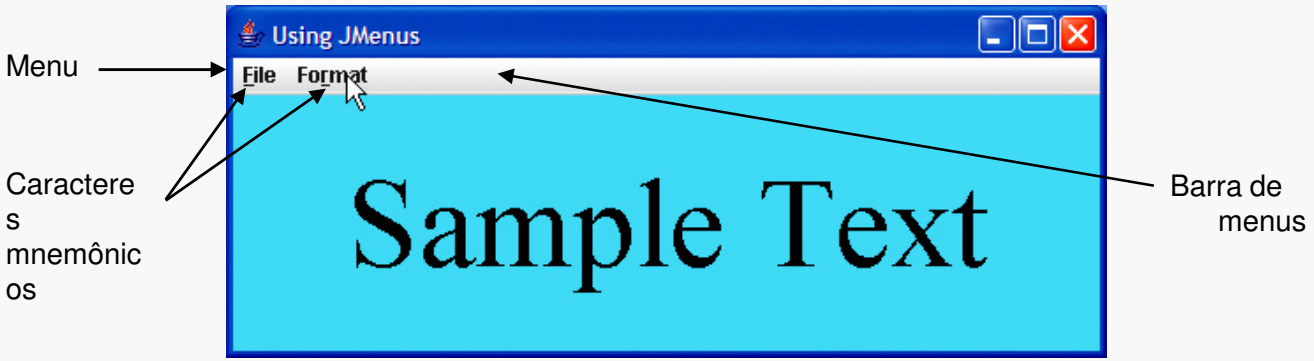
© 2005 by Pearson Education do Brasil

Resumo

MenuTest.java

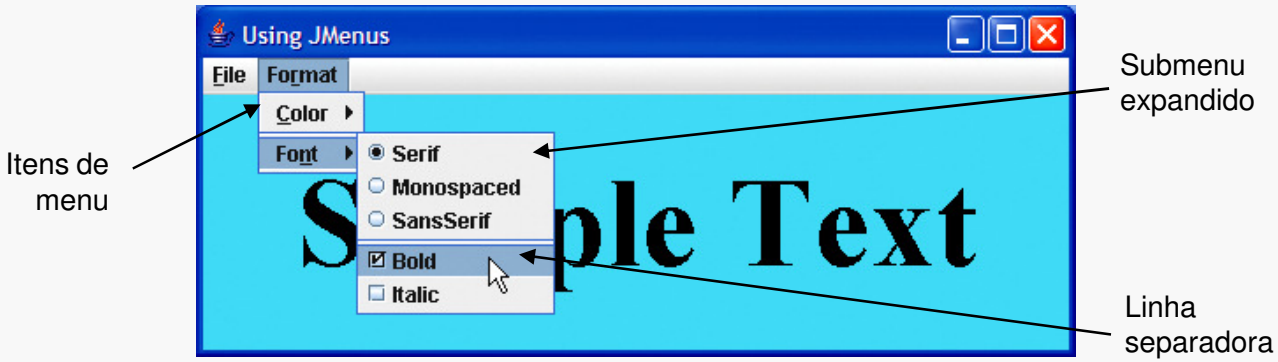
(1 de 2)

```
1  // Fig. 22.6: MenuTest.java
2  // Testando MenuFrame.
3  import javax.swing.JFrame;
4
5  public class MenuTest
6  {
7     public static void main( String args[] )
8     {
9        MenuFrame menuFrame = new MenuFrame(); // cria MenuFrame
10       menuFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
11       menuFrame.setSize( 500, 200 ); // configura o tamanho do frame
12       menuFrame.setVisible( true ); // exibe o frame
13    } // fim do main
14 } // fim da classe MenuTest
```



© 2005 by Pearson Education do Brasil

Resumo



`MenuTest.java`

(2 de 2)

© 2005 by Pearson Education do Brasil

Observação sobre aparência e comportamento 22.4

Diferentes mnemônicos devem ser utilizados para cada botão ou item de menu. Normalmente, a primeira letra do rótulo no item de menu ou botão é utilizada como o mnemônico. Se diversos botões ou itens de menu iniciam com a mesma letra, escolha a próxima letra mais significativa no nome (por exemplo, `X` comumente é escolhida para um botão ou item de menu chamado `Exit`).

© 2005 by Pearson Education do Brasil

Erro comum de programação 22.3

Esquecer de configurar a barra de menus com o método `JFrame` `setJMenuBar` resulta na barra de menus não sendo exibido no `JFrame`.

© 2005 by Pearson Education do Brasil

Observação sobre aparência e comportamento 22.5

Os menus aparecem da esquerda para a direita na ordem em que eles são adicionados a um `JMenuBar`.

© 2005 by Pearson Education do Brasil

Observação sobre aparência e comportamento 22.6

Um submenu é criado adicionando-se um menu como um item de menu a um outro menu. Quando o mouse é posicionado sobre um submenu (ou o mnemônico do submenu é pressionado), o submenu expande para mostrar seus itens de menu.

© 2005 by Pearson Education do Brasil

Observação sobre aparência e comportamento 22.7

Separadores podem ser adicionados a um menu para agrupar itens de menu logicamente.

© 2005 by Pearson Education do Brasil

Observação sobre aparência e comportamento 22.8

Qualquer componente GUI 'leve' (isto é, um componente que é uma subclasse de `JComponent`) pode ser adicionado a um `JMenu` ou a um `JMenuBar`.

© 2005 by Pearson Education do Brasil

22.5 JPopupMenu

- **Menu pop-up sensível ao contexto:**
  - **Fornece opções específicas do componente para o qual o evento de gatilho pop-up foi gerado.**
    - **Na maioria dos sistemas, o evento de acionamento de pop-up ocorre quando o usuário pressiona e libera o botão direito do mouse.**
  - **Criado com a classe `JPopupMenu`.**

© 2005 by Pearson Education do Brasil

Observação sobre aparência e comportamento 22.9

O evento de acionamento do pop-up é específico da plataforma. Na maioria das plataformas que utilizam um mouse com múltiplos botões, o evento de acionamento do pop-up ocorre quando o usuário clica com o botão direito do mouse em um componente que suporta um menu pop-up.

© 2005 by Pearson Education do Brasil

Resumo

PopupFrame.java

(1 de 4)

```
1  // Fig. 22.7: PopupFrame.java
2  // Demonstrando JPopupMenus.
3  import java.awt.Color;
4  import java.awt.event.MouseAdapter;
5  import java.awt.event.MouseEvent;
6  import java.awt.event.ActionListener;
7  import java.awt.event.ActionEvent;
8  import javax.swing.JFrame;
9  import javax.swing.JRadioButtonMenuItem;
10 import javax.swing.JPopupMenu;
11 import javax.swing.ButtonGroup;
12
13 public class PopupFrame extends JFrame
14 {
15    private JRadioButtonMenuItem items[]; // armazena itens para cores
16    private final Color colorValues[] =
17       { Color.BLUE, Color.YELLOW, Color.RED }; // cores a serem utilizadas
18    private JPopupMenu popupMenu; // permite ao usuário selecionar cores
19
20    // construtor sem argumentos configura a GUI
21    public PopupFrame()
22    {
23       super( "Using JPopupMenus" );
24
25       ItemHandler handler = new ItemHandler(); // handler para itens de menu
26       String colors[] = { "Blue", "Yellow", "Red" }; // array de cores
27
```

Uma instância da classe **ItemHandler** processará os eventos de item nos itens de menu

© 2005 by Pearson Education do Brasil

Resumo

```
28       ButtonGroup colorGroup = new ButtonGroup(); // gerencia itens de cores
29       popupMenu = new JPopupMenu(); // cria menu pop-up
30       items = new JRadioButtonMenuItem[ 3 ]; // itens para selecionar cor
31
32       // constrói item de menu, adiciona menu popup, ativa tratamento de event
33       for ( int count = 0; count < items.length; count++ )
34       {
35          items[ count ] = new JRadioButtonMenuItem( colors[ count ] );
36          popupMenu.add( items[ count ] ); // adiciona item ao menum pop-up
37          colorGroup.add( items[ count ] ); // adiciona item ao grupo de botão
38          items[ count ].addActionListener( handler ); // adiciona handler
39       } // fim do for
40
41       setBackground( Color.WHITE ); // configura fundo como branco
42
43       // declara um MouseListener para a janela exibir menu pop
44       addMouseListener(
45
46          new MouseAdapter() // classe interna anônima
47          {
48             // trata evento de pressionamento de mouse
49             public void mousePressed( MouseEvent event )
50             {
51                checkForTriggerEvent( event ); // verifica o acionamento
52             } // fim do método mousePressed
53
```

Cria um objeto **`JPopupMenu`**

`PopupFrame.java`

(2 de 4)

Cria e adiciona **`JRadioButtonMenuItem`**s e registra **`ActionListener`**s

Registra um **`MouseListener`** para tratar os eventos de mouse da janela da aplicação

© 2005 by Pearson Education do Brasil

Resumo

PopupFrame.java

(3 de 4)

```
54             // trata eventos de liberação de botão do mouse
55             public void mouseReleased( MouseEvent event )
56             {
57                checkForTriggerEvent( event ); // verifica acionamento
58             } // fim do método mouseReleased
59
60             // determina se o evento deve acionar menu popup
61             private void checkForTriggerEvent( MouseEvent event )
62             {
63                if ( event.isPopupTrigger() )
64                   popupMenu.show(
65                      event.getComponent(), event
66             } // fim do método checkForTriggerEvent
67          } // fim da classe interna anônima
68       ); // fim da chamada para addMouseListe
69    } // fim do construtor PopupFrame
70
```

Se o evento de gatilho pop-up tiver ocorrido, o método **`JPopupMenu show`** exibe o **`JPopupMenu`**

O componente de origem e os argumentos de coordenadas determinam onde o **`JPopupMenu`** aparecerá

© 2005 by Pearson Education do Brasil

Resumo

PopupFrame.java

(4 de 4)

```
71    // classe interna privada para tratar eventos de item de menu
72    private class ItemHandler implements ActionListener
73    {
74       // processa seleções de item de menu
75       public void actionPerformed( ActionEvent event )
76       {
77          // determina qual item de menu foi selecionado
78          for ( int i = 0; i < items.length; i++ )
79          {
80             if ( event.getSource() == items[ i ] )
81             {
82                getContentPane().setBackground( colorValues[ i ] );
83                return;
84             } // fim do if
85          } // fim do for
86       } // fim do método actionPerformed
87    } // fim da classe interna privada ItemHand
88 } // fim da classe PopupFrame
```

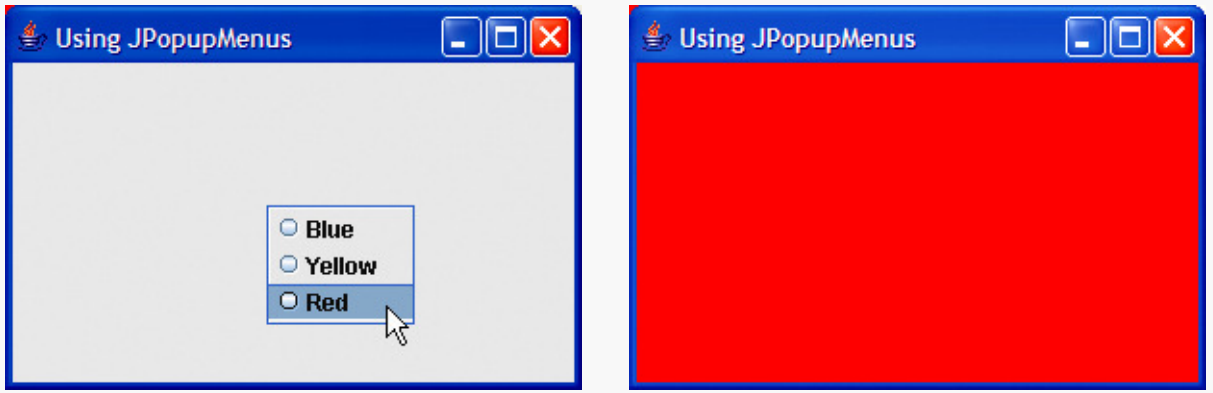
Determina qual **JRadioButtonMenuItem** o usuário selecionou e configura a cor de segundo plano

© 2005 by Pearson Education do Brasil

Resumo

PopupTest.java

```
1  // Fig. 22.8: PopupTest.java
2  // Testando PopupFrame.
3  import javax.swing.JFrame;
4
5  public class PopupTest
6  {
7     public static void main( String args[] )
8     {
9        PopupFrame popupFrame = new PopupFrame(); // cria PopupFrame
10       popupFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
11       popupFrame.setSize( 300, 200 ); // configura o tamanho do frame
12       popupFrame.setVisible( true ); // exibe o frame
13    } // fim do main
14 } // fim da classe PopupTest
```



© 2005 by Pearson Education do Brasil

Observação sobre aparência e comportamento 22.10

Exibir um `JPopupMenu` para o evento de acionamento de pop-up de múltiplos componentes GUI requer o registro de handlers de eventos de mouse para cada um desses componentes GUI.

© 2005 by Pearson Education do Brasil

22.6 Aparência e comportamento plugáveis

- **Aparências das aplicações Java:**
  - **Um programa que utiliza componentes GUI do Abstract Window Toolkit do Java assume a aparência e o comportamento da plataforma:**
    - **Permite aos usuários da aplicação em cada plataforma utilizarem componentes GUI que eles já conhecem.**
    - **Também introduz questões interessantes de portabilidade.**
  - **Os componentes GUI leves do Swing fornecem funcionalidades uniformes:**
    - **Definem uma aparência e um comportamento uniformes entre diferentes plataformas (conhecido como aparência e comportamento metal).**
    - **Também podem personalizar a aparência e o comportamento como um estilo do Windows Microsoft, como um estilo Motif (UNIX) ou uma aparência e comportamento do Macintosh.**

© 2005 by Pearson Education do Brasil

Dica de portabilidade 22.1

Os componentes GUI têm uma aparência diferente em diferentes plataformas e podem exigir quantidades diferentes de espaço para serem exibidos. Isso poderia alterar seus layouts e alinhamentos da GUI.

© 2005 by Pearson Education do Brasil

Dica de portabilidade 22.2

Os componentes GUI em diferentes plataformas têm diferentes funcionalidades-padrão (por exemplo, algumas plataformas permitem que um botão com o foco seja ‘pressionado’ com a barra de espaço e outras, não).

© 2005 by Pearson Education do Brasil

Resumo

LookAndFeelFrame.java

(1 de 4)

```
1  // Fig. 22.9: LookAndFeelFrame.java
2  // Alterando a aparência e comportamento.
3  import java.awt.GridLayout;
4  import java.awt.BorderLayout;
5  import java.awt.event.ItemListener;
6  import java.awt.event.ItemEvent;
7  import javax.swing.JFrame;
8  import javax.swing.UIManager;
9  import javax.swing.JRadioButton;
10 import javax.swing.ButtonGroup;
11 import javax.swing.JButton;
12 import javax.swing.JLabel;
13 import javax.swing.JComboBox;
14 import javax.swing.JPanel;
15 import javax.swing.SwingConstants;
16 import javax.swing.SwingUtilities;
17
18 public class LookAndFeelFrame extends JFrame
19 {
20    // nomes de string das aparências e comportamentos
21    private final String strings[] = { "Metal", "Motif", "Windows" };
22    private UIManager.LookAndFeelInfo looks[]; // aparência e comportamentos
23    private JRadioButton radio[]; // botões de opção para selecionar a aparência e comportamento
24    private ButtonGroup group; // grupo para botões de opção
25    private JButton button; // exibe a aparência do botão
26    private JLabel label; // exibe a aparência do rótulo
27    private JComboBox comboBox; // exibe a aparência da caixa de combinação
28
```

© 2005 by Pearson Education do Brasil

```
29    // configura a GUI
30    public LookAndFeelFrame()
31    {
32       super( "Look and Feel Demo" );
33
34       JPanel northPanel = new JPanel(); // cria o painel North
35       northPanel.setLayout( new GridLayout( 3, 1, 0, 5 ) );
36
37       label = new JLabel( "This is a Metal look-and-feel",
38          SwingConstants.CENTER ); // cria o rótulo
39       northPanel.add( label ); // adiciona o rótulo ao painel
40
41       button = new JButton( "JButton" ); // cria o botão
42       northPanel.add( button ); // adiciona o botão ao painel
43
44       comboBox = new JComboBox( strings ); // cria a caixa de combinação
45       northPanel.add( comboBox ); // adiciona a caixa de combinação ao painel
46
47       // cria um array para botões de opção
48       radio = new JRadioButton[ strings.length ];
49
50       JPanel southPanel = new JPanel(); // cria o painel South
51       southPanel.setLayout( new GridLayout( 1, radio.length ) );
52
53       group = new ButtonGroup(); // grupo de botões para a aparência e comportamento
54       ItemHandler handler = new ItemHandler(); // handler da aparência e comportamento
55
```

Resumo

LookAndFeelFrame.java

(2 de 4)

© 2005 by Pearson Education do Brasil

Resumo

LookAndFeelFrame.java

```
56       for ( int count = 0; count < radio.length; count++ )
57       {
58          radio[ count ] = new JRadioButton( strings[ count ] );
59          radio[ count ].addItemListener( handler ); // adiciona handler
60          group.add( radio[ count ] ); // adiciona botões de opção ao grupo
61          southPanel.add( radio[ count ] ); // adiciona botões de opção ao painel
62       } // fim do for
63
64       add( northPanel, BorderLayo
65       add( southPanel, BorderLayo
66
67       // obtém as informações sobre a aparência e comportamento instaladas
68       looks = UIManager.getInstalledLookAndFeels();
69       radio[ 0 ].setSelected( true ); // configura a seleção padrão
70    } // fim do construtor LookAndFeelFrame
71
72    // utiliza UIManager para alterar a aparência e comportamento da GUI
73    private void changeTheLookAndFeel( int value )
74    {
75       try // muda a aparência e comportamento
76       {
77          // configura a aparência e comportamento para esse aplicativo
78          UIManager.setLookAndFeel( looks[ value ].getClassName() );
79
80          // atualiza os componentes nesse aplicativo
81          SwingUtilities.updateComponentTreeUI( this );
82       } // fim do try
```

Obtém o array dos objetos **`UIManager.LookAndFeelInfo`** que descrevem cada aparência e comportamento disponível no seu sistema

Invoca o método **`static setLookAndFeel`** para alterar a aparência e comportamento

Invoca o método **`static updateComponentTreeUI`** para alterar a aparência e comportamento de cada componente GUI anexado à aplicação

© 2005 by Pearson Education do Brasil

```
83          catch ( Exception exception )
84          {
85             exception.printStackTrace();
86          } // fim do catch
87       } // fim do método changeTheLookAndFeel
88
89       // classe interna private para tratar eventos de botão de opção
90       private class ItemHandler implements ItemListener
91       {
92          // processa a seleção de aparência e comportamento feita pelo usuário
93          public void itemStateChanged( ItemEvent event )
94          {
95             for ( int count = 0; count < radio.length; count++ )
96             {
97                if ( radio[ count ].isSelected() )
98                {
99                   label.setText( String.format( "This is a %s look-and-feel",
100                     strings[ count ] ) );
101               comboBox.setSelectedIndex( count ); // configura o índice da caixa de combinação
102                  changeTheLookAndFeel( count ); // muda a aparência e comportamento
103               } // fim do if
104            } // fim do for
105         } // fim do método itemStateChanged
106      } // fim da classe interna private ItemHand
107   } // fim da classe LookAndFeelFrame
```

Chama o método utilitário `changeTheLookAndFeel`

Resumo

`LookAndFeelFrame.java`

(4 de 4)

© 2005 by Pearson Education do Brasil

Dica de desempenho 22.1

Cada aparência e comportamento são representados por uma classe Java. O método `UIManager getInstalledLookAndFeels` não carrega cada classe. Em vez disso, fornece os nomes das classes de aparência e comportamento disponíveis de modo que uma escolha possa ser feita (presumivelmente, uma vez na inicialização do programa). Isso reduz o overhead de ter de carregar todas as classes de aparência e comportamento mesmo se o programa não utilizar algumas delas.

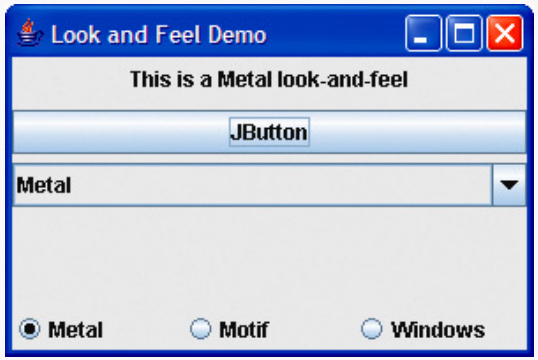
© 2005 by Pearson Education do Brasil

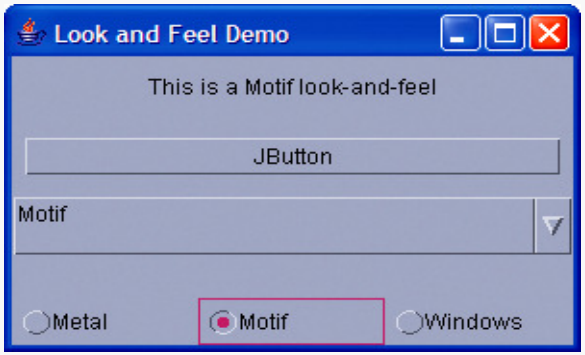
Resumo

LookAndFeelDemo.java

(1 de 2)

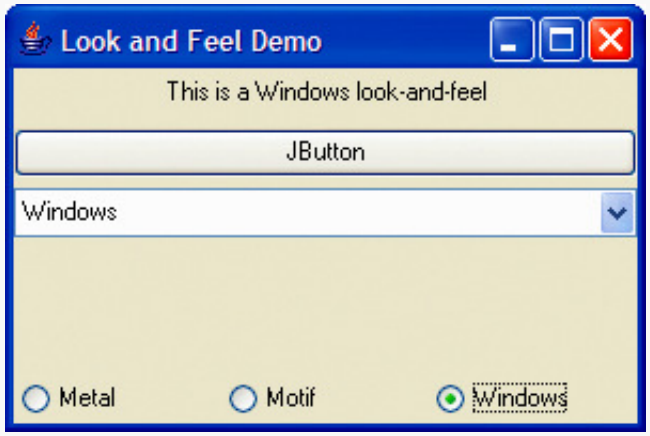
```
1  // Fig. 22.10: LookAndFeelDemo.java
2  // Mudando a aparência e o comportamento.
3  import javax.swing.JFrame;
4
5  public class LookAndFeelDemo
6  {
7     public static void main( String args[] )
8     {
9        LookAndFeelFrame lookAndFeelFrame = new LookAndFeelFrame();
10       lookAndFeelFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
11       lookAndFeelFrame.setSize( 300, 200 ); // configura o tamanho do frame
12       lookAndFeelFrame.setVisible( true ); // exibe o frame
13    } // fim do main
14 } // fim da classe LookAndFeelDemo
```





© 2005 by Pearson Education do Brasil

Resumo



`LookAndFeelDemo.java`

(2 de 2)

© 2005 by Pearson Education do Brasil

22.7 JDesktopPane e JInternalFrame

- **Interface de múltiplos documentos:**
  - **Uma janela principal (chamada janela-pai) contém outras janelas (chamadas janelas-filhas).**
  - **Gerencia vários documentos abertos que estão sendo processados em paralelo.**
  - **Implementada pelo `JDesktopPane` e `JInternalFrame` do Swing.**

© 2005 by Pearson Education do Brasil

Resumo

DeskTopFrame.java

(1 de 4)

```
1  // Fig. 22.11: DesktopFrame.java
2  // Demonstrando JDesktopPane.
3  import java.awt.BorderLayout;
4  import java.awt.Dimension;
5  import java.awt.Graphics;
6  import java.awt.event.ActionListener;
7  import java.awt.event.ActionEvent;
8  import java.util.Random;
9  import javax.swing.JFrame;
10 import javax.swing.JDesktopPane;
11 import javax.swing.JMenuBar;
12 import javax.swing.JMenu;
13 import javax.swing.JMenuItem;
14 import javax.swing.JInternalFrame;
15 import javax.swing.JPanel;
16 import javax.swing.ImageIcon;
17
18 public class DesktopFrame extends JFrame
19 {
20    private JDesktopPane theDesktop;
21
22    // configura a GUI
23    public DesktopFrame()
24    {
25       super( "Using a JDesktopPane" );
26
27       JMenuBar bar = new JMenuBar(); // cria a barra de menu
28       JMenu addMenu = new JMenu( "Add" ); // cria o menu Add
29       JMenuItem newFrame = new JMenuItem( "Internal Frame" );
30
```

Cria um **JMenuBar**, um **JMenu** e um **JMenuItem**

© 2005 by Pearson Education do Brasil

```
31       addMenu.add( newFrame ); // adiciona um novo item de frame ao menu Add
32       bar.add( addMenu ); // adiciona o menu Add à barra de menus
33       setJMenuBar( bar ); // configura a barra de menus
34
35       theDesktop = new JDesktopPane(); // cria o painel
36       add( theDesktop ); // adiciona painel de área de trabalho ao frame
37
38       // configura o ouvinte para o item de menu newF
39       newFrame.addActionListener(
40
41          new ActionListener() // classe interna anôni
42          {
43             // exibe a nova janela interna
44             public void actionPerformed( ActionEvent event )
45             {
46                // cria o frame interno
47                JInternalFrame frame = new JInternalFrame(
48                   "Internal Frame", true, true, true, true );
49
50                MyJPanel panel = new MyJPanel(); // cria um novo painel
51                frame.add( panel, BorderLayout.CENTER ); // adiciona o painel
52                frame.pack(); // configura frame interno de acordo c/ o tamanho do conteúdo
53
```

Adiciona o **JMenuItem** ao **JMenu**, e o **JMenu** ao **JMenuBar**, e configura o **JMenuBar** para a janela da aplicação

DeskTopFrame

O **JDesktopPane** será utilizado para gerenciar as janelas-filhas do **JInternalFrame**

Cria um objeto **JInternalFrame**

Os argumentos de construtor especificam a string da barra de título e se o usuário pode ou não redimensionar, fechar, maximizar e minimizar o frame interno

Configura o tamanho da janela-filha

© 2005 by Pearson Education do Brasil

Resumo

```
54                theDesktop.add( frame ); // anexa frame interno
55                frame.setVisible( true ); // mostra o frame interno
56             } // fim do método actionPerformed
57          } // fim da classe interna anÔnima
58       ); // fim da chamada para addActionListener
59    } // fim do construtor DesktopFrame
60 } // fim da classe DesktopFrame
61
62 // classe para exibir um ImageIcon em um painel
63 class MyJPanel extends JPanel
64 {
65    private static Random generator = new Random();
66    private ImageIcon picture; // imagem a ser exibida
67    private String[] images = { "yellowflowers.png", "purpleflowers.png",
68       "redflowers.png", "redflowers2.png", "lavenderflowers.png" };
69
70    // carrega a imagem
71    public MyJPanel()
72    {
73       int randomNumber = generator.nextInt( 5 );
74       picture = new ImageIcon( images[ randomNumber ] ); // configura o ícone
75    } // fim do construtor MyJPanel
76
```

Adiciona o `JInternalFrame` a `theDesktop` e exibe o `JInternalFrame`

`.java`

(3 de 4)

© 2005 by Pearson Education do Brasil

Resumo

```
77    // exibe o imageIcon no painel
78    public void paintComponent( Graphics g )
79    {
80       super.paintComponent( g );
81       picture.paintIcon( this, g, 0, 0 ); // exibe o ícone
82    } // fim do método paintComponent
83
84    // retorna dimensões da imagem
85    public Dimension getPreferredSize()
86    {
87       return new Dimension( picture.getIconWidth(),
88          picture.getIconHeight() );
89    } // fim do método getPreferredSize
90 } // fim da classe MyJPanel
```

Especifica o tamanho preferido do painel para uso pelo método **pack**

DeskTopFrame
.va

(4 de 4)

© 2005 by Pearson Education do Brasil

Resumo

DeskTopTest.java

(1 de 3)

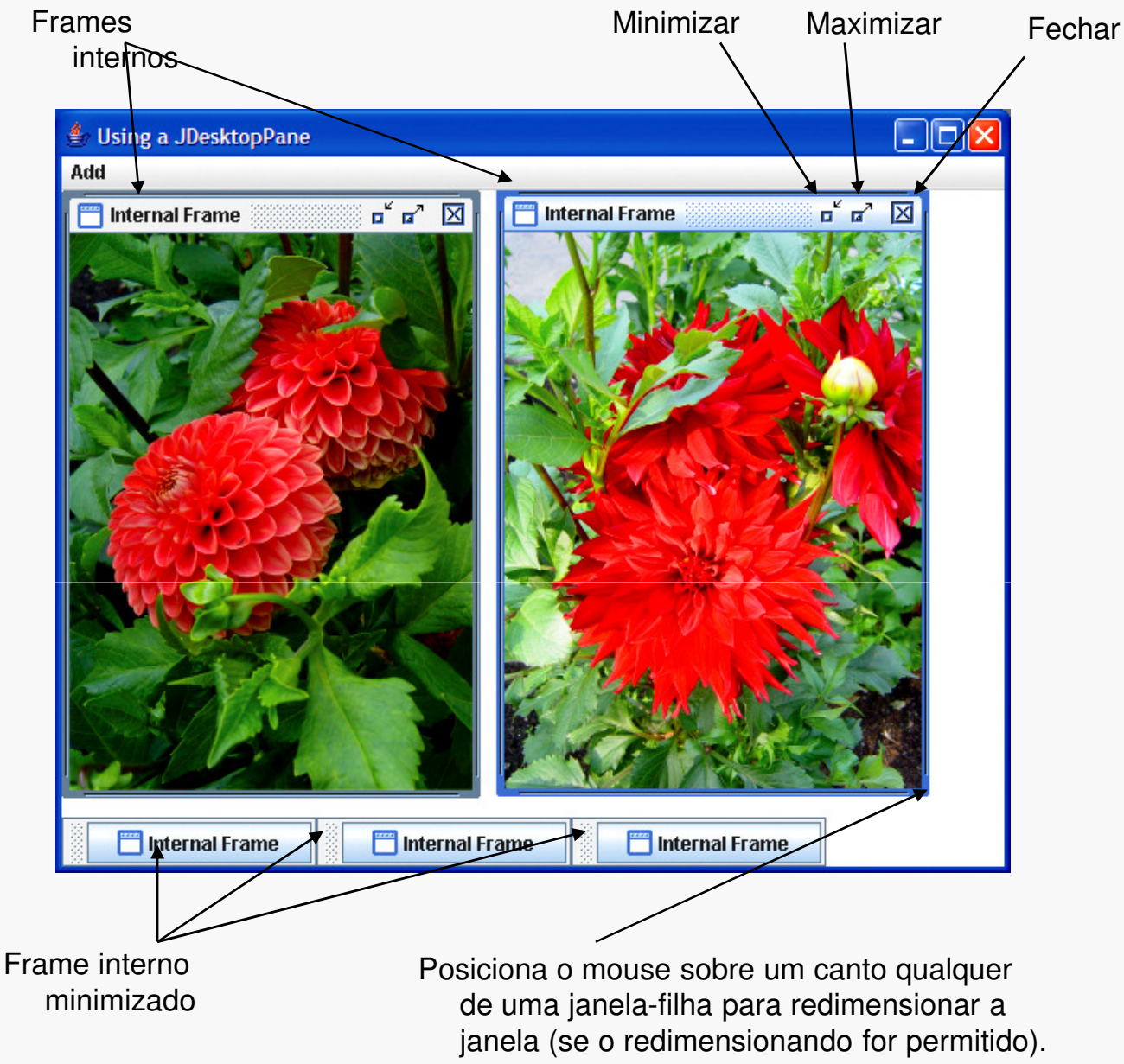
```
1  // Fig. 22.12: DesktopTest.java
2  // Demonstrando JDesktopPane.
3  import javax.swing.JFrame;
4
5  public class DesktopTest
6  {
7     public static void main( String args[] )
8     {
9        DesktopFrame desktopFrame = new DesktopFrame();
10       desktopFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
11       desktopFrame.setSize( 600, 480 ); // configura o tamanho do frame
12       desktopFrame.setVisible( true ); // exibe o frame
13    } // fim do main
14 } // fim da classe DesktopTest
```

© 2005 by Pearson Education do Brasil

Resumo

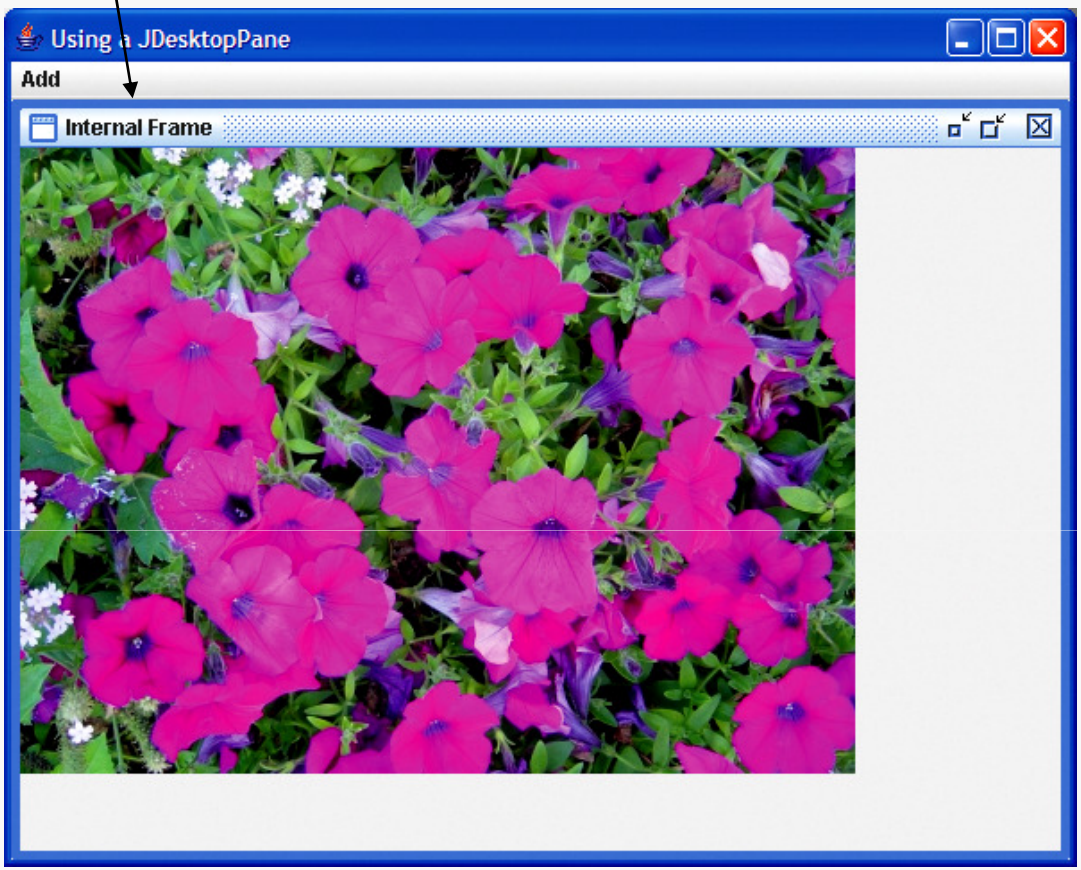
`DeskTopTest.java`

(2 de 3)



© 2005 by Pearson Education do Brasil

Frame interno maximizado



Resumo

`DeskTopTest.java`

(3 de 3)

© 2005 by Pearson Education do Brasil

Exercícios

1) Implementar os exemplos dessa apresentação:

- **MenuFrame.java e MenuTest.java**
- **PopupFrame.java e PopupTest.java**
- **LookAndFeelFrame.java e LookAndFeelTest.java**
- **DesktopFrame.java e DesktopTest.java**

2) Implementar uma classe para compor o seguinte menu:

Cadastros	Movimentos	Financeiro	Sair
Funcionarios	Matrícula	Contas a Receber	
Alunos	Mensalidades	Contas a Pagar	
Professores		Fluxo de Caixa	

© 2005 by Pearson Education do Brasil