Menus

- Adicionados a componentes que possuem o método **`setJMenuBar`**
  - **`JFrame`** and **`JApplet`**
- Classes usadas:
  - **`JMenuBar`** - barra de menu
  - **`JMenuItem`** - item de menu
  - **`JMenu`** - um menu
    - tem itens de menu e são inseridos na MenuBar
    - podem funcionar como submenu
  - **`JCheckBoxMenuItem`**
    - Item de menu do tipo (Yes/No)
  - **`JRadioButtonMenuItem`**
    - Item de menu funcionando como radio
- Usando menus
  - Cria a barra de menu
  - Cria os menus
  - Cria os itens de menu
  - Adiciona os itens de menu aos menus
  - Se precisa de submenu insere-os nos menus
  - Adiciona os menus a barra de menu

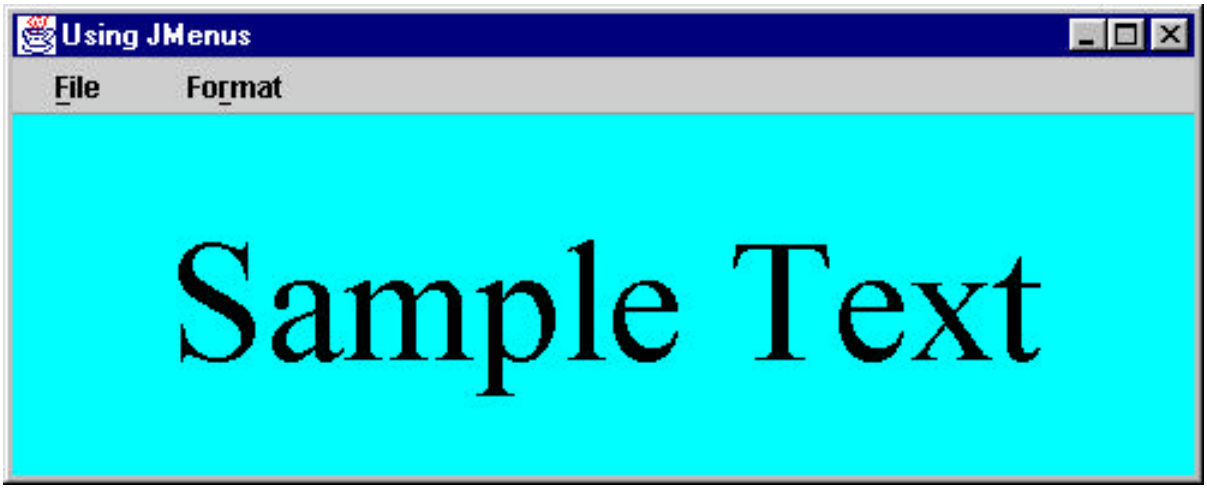
```
public class MenuTest extends JFrame {
  private Color colorValues[] =
   {Color.black,Color.blue,Color.red,Color.green };
  private JRadioButtonMenuItem colorItems[],fonts[];

   private JCheckBoxMenuItem styleItems[];
   private JLabel display;
   private ButtonGroup fontGroup, colorGroup;
   private int style;
   public MenuTest() {
      super( "Using JMenus" );
      JMenuBar bar = new JMenuBar();
      setJMenuBar( bar );  // set the menubar
      // create File menu and Exit menu item
       JMenu fileMenu = new JMenu( "File" );
      fileMenu.setMnemonic( 'F' );
      JMenuItem aboutItem=new JMenuItem("About...");
                     aboutItem.setMnemonic( 'A' );
      aboutItem.addActionListener(
         new ActionListener() {
            public void actionPerformed(
                                 ActionEvent e ) {
               JOptionPane.showMessageDialog(
                    MenuTest.this,
                   ”Exemplo de uso de menus",
                   "About",JOptionPane.PLAIN_MESSAGE);
            }
         }
      ); // end of addActionListener
      fileMenu.add( aboutItem );
      JMenuItem exitItem = new JMenuItem( "Exit" );
```

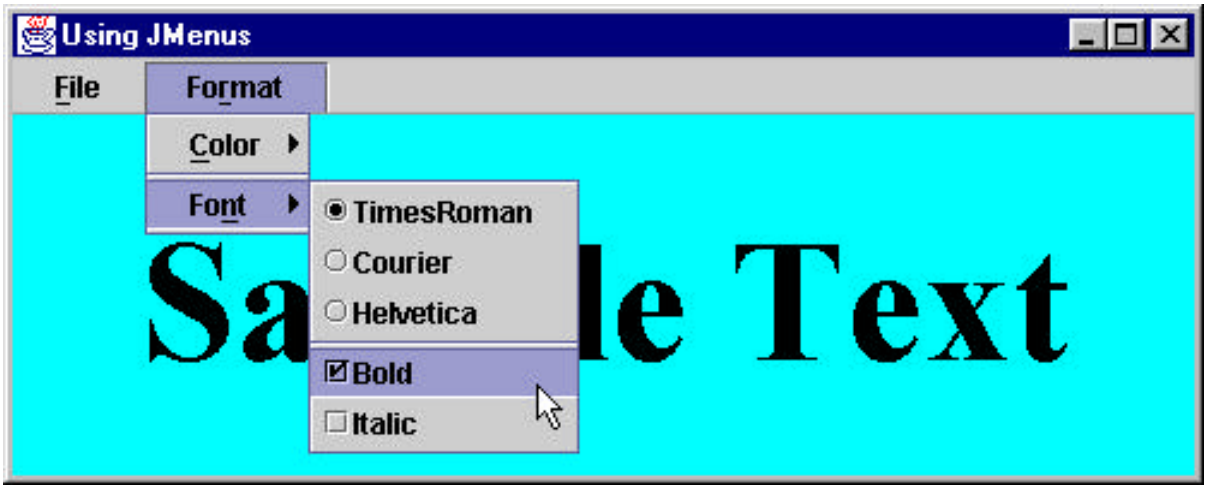
```
exitItem.setMnemonic( 'x' );
exitItem.addActionListener(
   new ActionListener() {
    public void actionPerformed(ActionEvent e)
      {
         Svstem.exit( 0 );
      }
   }
);
fileMenu.add( exitItem );
bar.add( fileMenu );    // add File menu
JMenu formatMenu = new JMenu( "Format" );
formatMenu.setMnemonic( 'r' );
String colors[] =
   { "Black", "Blue", "Red", "Green" };
JMenu colorMenu = new JMenu( "Color" );
colorMenu.setMnemonic( 'C' );
colorGroup = new ButtonGroup();
ItemHandler itemHandler = new ItemHandler();
for ( int i = 0; i < colors.length; i++ ) {
 colorItems[ i ] =
 new JRadioButtonMenuItem( colors[ i ] );
 colorItems = new JRadioButtonMenuItem[
                                 colors.length ];
 colorMenu.add( colorItems[ i ] );
 colorGroup.add( colorItems[ i ] );
 colorItems[ i ].addActionListener(
                                 itemHandler );
}
colorItems[ 0 ].setSelected( true );
formatMenu.add( colorMenu );
formatMenu.addSeparator();
String fontNames[] =
   { "TimesRoman", "Courier", "Helvetica" };
JMenu fontMenu = new JMenu( "Font" );
```

```
   fontMenu.setMnemonic( 'n' );
   fonts=new JRadioButtonMenuItem[fontNames.length];
   fontGroup = new ButtonGroup();

   for ( int i = 0; i < fonts.length; i++ ) {

      fonts[ i ] =
        new JRadioButtonMenuItem(fontNames[ i ]);
      fontMenu.add( fonts[ i ] );
      fontGroup.add( fonts[ i ] );

      fonts[ i ].addActionListener(itemHandler);

   }
   fonts[ 0 ].setSelected( true );
   fontMenu.addSeparator();
   String styleNames[] = { "Bold", "Italic" };
   styleItems = new
              JCheckBoxMenuItem[styleNames.length];
   StyleHandler styleHandler = new StyleHandler()
   for ( int i = 0; i<styleNames.length; i++) {
     styleItems[i] = new
                 JCheckBoxMenuItem(styleNames[i]);

     fontMenu.add( styleItems[ i ] );
      styleItems[i].addItemListener(styleHandler);
   show();
}
public static void main( String args[] ) {
   MenuTest app = new MenuTest();

   app.addWindowListener(
      new WindowAdapter() {
         public void windowClosing( WindowEvent e )

         { System.exit( 0 ); }

      }
   );
}
```

```
   class ItemHandler implements ActionListener {
      public void actionPerformed( ActionEvent e ) {
         for ( int i = 0; i < colorItems.length; i++ )
            if ( colorItems[ i ].isSelected() ) {
              display.setForeground(colorValues[i]);
               break;
            }
         for ( int i = 0; i < fonts.length; i++ )
            if ( e.getSource() == fonts[ i ] ) {
               display.setFont( new Font(
                  fonts[ i ].getText(), style, 72 ) );
               break;
            }
         repaint();
      }
   }
   class StyleHandler implements ItemListener {
      public void itemStateChanged( ItemEvent e ) {
         style = 0;
         if ( styleItems[ 0 ].isSelected() )
            style += Font.BOLD;
         if ( styleItems[ 1 ].isSelected() )
            stvle += Font.ITALIC;
         display.setFont( new Font(
            display.getFont().getName(),style,72) );
         repaint();
      }
   }
}
```

Interface com o Menu





Eventos de Mouse

- Gerados por qualquer component
- Métodos de tratamento de eventos de mouse:
  - recebem objeto **`MouseEvent`** com informação sobre o evento (e.g. **`getX`** e **`getY`**)
- Interfaces **`MouseListener`** e **`MouseMotionListener`**
  - **`addMouseListener`**
  - **`addMouseMotionListener`**
- Interface **`MouseListener`**

`public void mousePressed( MouseEvent e )`

- botao do mouse pressionado

`public void mouseClicked( MouseEvent e )`

- botao do mouse pressionado e solto

`public void mouseReleased(MouseEvent e)`

- botao do mouse solto

`public void mouseEntered( MouseEvent e )`

- mouse entrou na area do componente

`public void mouseExited( MouseEvent e )`

- mouse deixou a area do componente

Eventos de Mouse

- Interface **MouseMotionListener**

public void mouseDragged(MouseEvent e)

- mouse pressionado e movendo-se

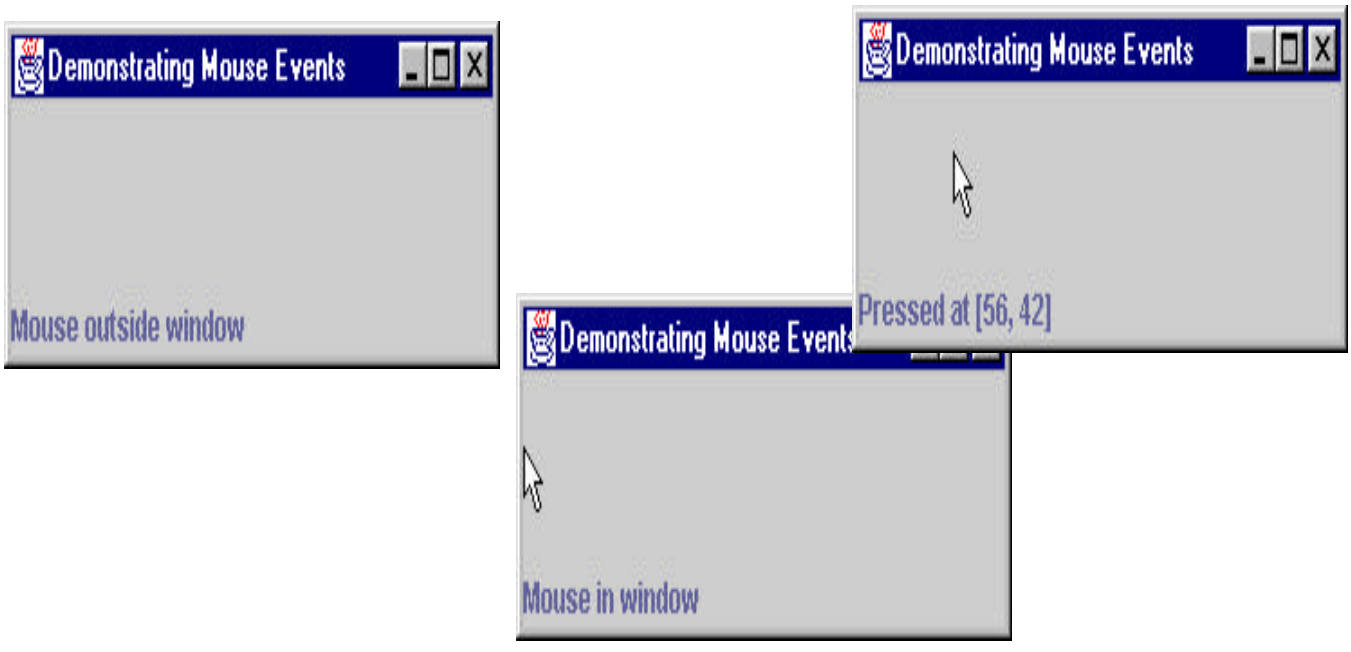
public void mouseMoved(MouseEvent e)

- mouse se movendo quando sobre o componente

```
8  public class MouseTracker extends JFrame
9                implements MouseListener,
10    private JLabel statusBar;

12    public MouseTracker() {

14       super( "Demonstrating Mouse Events" );
15
16       statusBar = new JLabel();
17       getContentPane().add( statusBar,
                               BorderLayout.SOUTH );
19       // application listens to its own mouse
20       addMouseListener( this );
21       addMouseMotionListener( this );
23       setSize( 275, 100 );
24       show();
25    }
```

```
27    // MouseListener event handlers
28    public void mouseClicked( MouseEvent e ){
30       statusBar.setText( "Clicked at [" +
31                          ", " + e.getY()+"]");
32    }
34    public void mousePressed( MouseEvent e ){
36       statusBar.setText( "Pressed at [" +
37                          ", "+e.getY()+"]" );
38    }
40    public void mouseReleased( MouseEvent e ){
42       statusBar.setText( "Released at [" +
43                          ", " + e.getY()+"]");
44    }
46    public void mouseEntered( MouseEvent e ){
48       statusBar.setText( "Mouse in window" );
49    }
51    public void mouseExited( MouseEvent e )
53       statusBar.setText( "Mouse outside
54    }
57    public void mouseDragged( MouseEvent e )
58    {
59       statusBar.setText( "Dragged at
               ["+e.getX()+ ", " + e.getY()+"]");
61    }
63    public void mouseMoved(MouseEvent e){
65       statusBar.setText( "Moved at ["+e.getX()
66                          ", " + e.getY() + "]" );
67    }
```

```
69    public static void main( String args[] )
70    {
71       MouseTracker app = new MouseTracker();
72
73       app.addWindowListener(
74          new WindowAdapter() {
75             public void windowClosing(
                                      WindowEvent e )
76             {
77                System.exit( 0 );
78             }
79          }
80       );
81    }
82 }
```



Java2D

- Conjunto de classes para criar gráficos
- Exemplo

```
C:> cd \jdk1.2\demo\jfc\Java2D
C:> java Java2Demo
```

- Desenhando em componentes
  - redefine o método paint - recebe um Graphics
  - Graphics-> objeto que representa o contexto gráfico
  - cast para Graphics2D

```
public void paint(Graphics g) {
  Graphics2D g2 = (Graphics2D)g;
  // Now we can do cool 2D stuff.
}
```

  - componente pode representar a tela ou um dispositivo qualquer
  - Em componentes Swing deve redefinir paintComponent() ao invés de paint().
    - Swing usa paint() para desenhar os componentes filhos

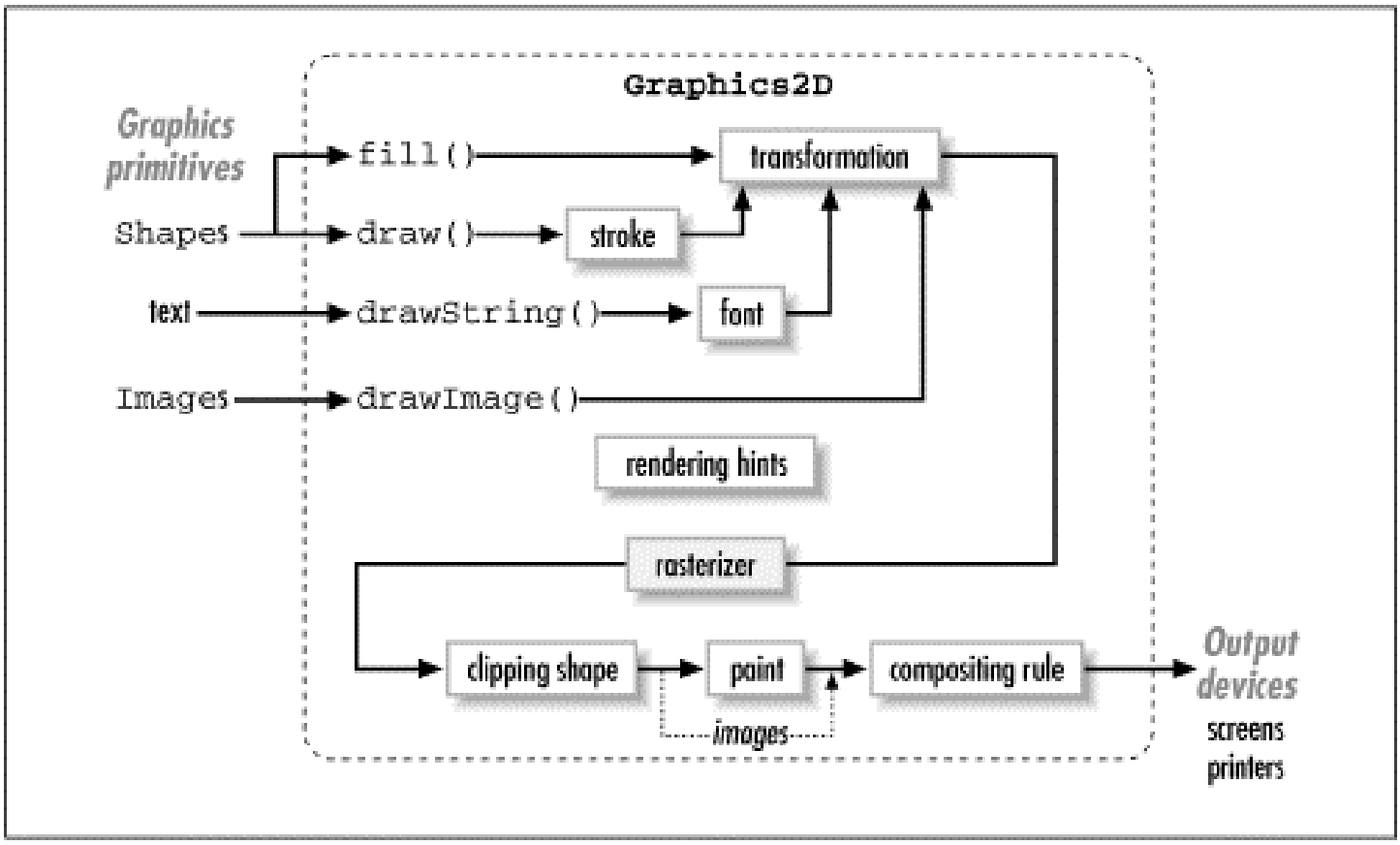
ApplicationFrame

- Aplicação Exemplo - Deriva e implementa paintComponent:

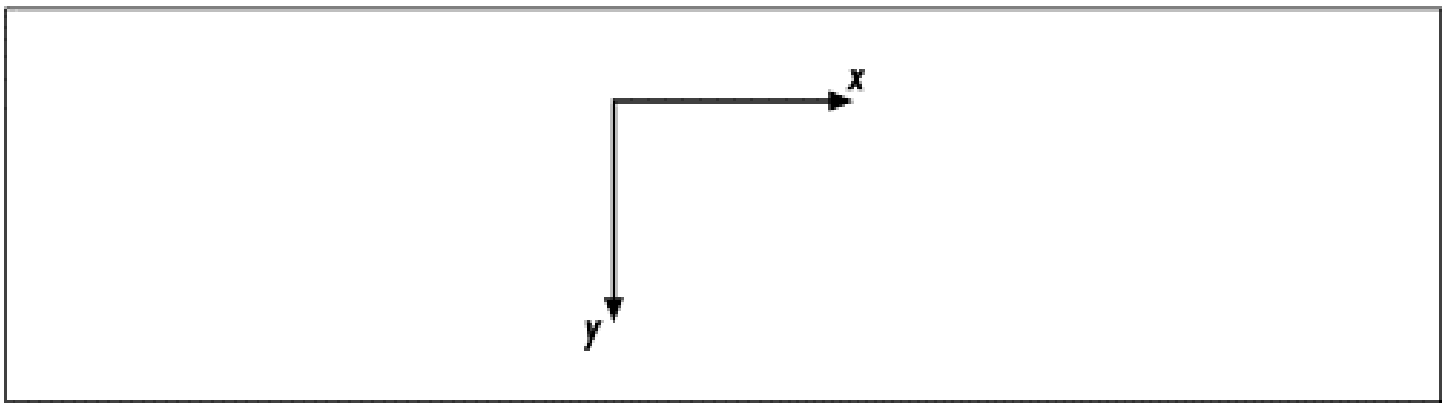
```
public class ApplicationFrame extends JFrame {
  public ApplicationFrame() {
   this("ApplicationFrame v1.0"); }
  public ApplicationFrame(String title) {
    super(title);
    createUI();
  }
  protected void createUI() {
    setSize(500, 400);
    center();
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e){
        dispose();
        System.exit(0);
      }
    });
  }
  public void center() {
    Dimension screenSize =
   Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = getSize();
    int x = (screenSize.width -
   frameSize.width) / 2;
    int y = (screenSize.height -
   frameSize.height) / 2;
    setLocation(x, y);
  }
}
```

Graphics2D e Sistema de Coordenadas

- Fluxo de Rendering



- Objetos (*User Space) s*ão desenhados no *Device Space*



- Transformação: 72 coordenadas US x 1” DS

Desenhando Linhas, Retângulos e Elipses

- Métodos para desenhar formas
  - **`drawLine( x1, y1, x2, y2 )`**
    - Linha de **`x1`**, **`y1`** para **`x2`**, **`y2`**
  - **`drawRect( x1, y1, width, height)`**
    - retângulo com canto superior esquerdo em **`x1, y1`**
  - **`fillRect( x1, y1, width, height)`**
    - Preenche o retângulo
  - **`clearRect (x1, y1, width, height)`**
    - preenche o retângulo com a cor de fundo
  - **`drawOval( x, y, width, height )`**
    - desenha uma elipse contida no retângulo
  - **`fillOval ( x, y, width, height )`**
    - **`preenche a elipse`**

Desenhando Polígonos

- **`drawPolygon( xPoints[], yPoints[], points )`**
  - Desenha um poligono com os vertices (xi,yi) especificados no vetor.
  - Desenha poligono fechado
- **`drawPolyline ( xPoints[], yPoints, points )`**
  - desenha uma poligonal aberta.
- **`drawPolygon( myPolygon )`**
  - Desenha o poligono especificado
- **`fillPolygon( xPoints[], yPoints[], points )`**
  - desenha um poligono preenchido
- **`fillPolygon( myPolygon )`**
  - desenha um poligono preenchido
- **`Polygon( xValues[], yValues[], numberOfPoints )`**
  - constroi um objeto Polygon
- **`myPolygon.addPoint( x, y )`**
  - adiciona um vertice ao objeto Polygon

JPanel

- Pode ser usado como area dedicada de desenho
  - Recebe eventos do mouse
  - Pode ser extendida para criara novos componentes
- Método **`paintComponent`**
  - todo componente derivado de **`Jcomponent`** `possui este método`
  - `ajuda a desenhar corretamente`
  - Redefine:

```
public void paintComponent(Graphics
  g ){
     super.paintComponent( g );
     // your additional drawing code
}
```

  - primeiro chama o construtor da superclasse
- **`JFrame`** and **`JApplet`**
  - não são subclasses de **`JComponent`**
    - deve redefinir o método **`paint`**
- Cria subclasses customizadas
  - Herda de **`JPanel`**
  - Redefine o método **`paintComponent`**

Exemplo

```
6 public class CustomPanel extends JPanel {
7    public final static int CIRCLE = 1, SQUARE
= 2; private int shape;
8
10   public void paintComponent( Graphics g ){
12      super.paintComponent( g );
13
14      if ( shape == CIRCLE )
15         g.fillOval( 50, 10, 60, 60 );
16      else if ( shape == SQUARE )
17         g.fillRect( 50, 10, 60, 60 );
18   }
19
20   public void draw( int s ) {
22      shape = s;
23      repaint();
24   }
25}
```

```
32 public class CustomPanelTest extends JFrame {
33    private JPanel buttonPanel;
34    private CustomPanel myPanel;
35    private JButton circle, square;
37    public CustomPanelTest(){
39       super( "CustomPanel Test" );
41       myPanel = new CustomPanel();
42       myPanel.setBackground( Color.green );
44       square = new JButton( "Square" );
45       square.addActionListener(
46          new ActionListener() {
47             public void actionPerformed(
                                ActionEvent e ) {
49               myPanel.draw(CustomPanel.SQUARE);
50             }
51          }
52       );
54       circle = new JButton( "Circle" );
55       circle.addActionListener(
56          new ActionListener() {
57             public void actionPerformed(
                                ActionEvent e ) {
59               myPanel.draw(CustomPanel.CIRCLE);
60             }
61          }
62       );
```

```
64          buttonPanel = new JPanel();
65          buttonPanel.setLayout( new GridLayout(
                                          1, 2 ) );
66          buttonPanel.add( circle );
67          buttonPanel.add( square );
69          Container c = getContentPane();
70          c.add( myPanel, BorderLayout.CENTER );
71          c.add( buttonPanel, BorderLayout.SOUTH );
72
73          setSize( 300, 150 );
74          show();
75       }
76
77       public static void main( String args[] )
78       {
79          CustomPanelTest app=new
                                    CustomPanelTest();
81          app.addWindowListener(
82             new WindowAdapter() {
83                public void windowClosing(
                                    WindowEvent e ) {
85                   System.exit( 0 );
86                }
87             }
88          );
89       }
90    }
```

Criado uma subclasse autocontida

- Eventos
  - **`JPanels`** não reconhecem eventos próprios
  - Reconhece eventos de nível mais baixo
    - Eventos de mouse e de teclado
- Exemplo
  - Crie uma subclasse de **`JPanel`** nomeado **`SelfContainedPanel`** que escuta seus evento do mouse
    - desenhe uma elipse redefinindo **`paintComponent`**
  - Importe **`SelfContainedPanel`** em outra classe
    - A outra classe possui seus próprios genrenciadores de eventos de mouse
  - Adicione uma instância de **`SelfContainedPanel`** ao content pane

```
9  public class SelfContainedPanelTest extends
10    private SelfContainedPanel myPanel;
12    public SelfContainedPanelTest() {
14       myPanel = new SelfContainedPanel();
15       myPanel.setBackground( Color.yellow );
17       Container c = getContentPane();
18       c.setLayout( new FlowLayout() );
19       c.add( myPanel );
21       addMouseMotionListener(
22          new MouseMotionListener() {
23             public void mouseDragged(
                                  FMouseEvent e ) }
25                setTitle( "Dragging: x=" +
26                          "; y=" + e.getY() );
27             }
29             public void mouseMoved(
                                  MouseEvent e ) {
31                setTitle( "Moving: x=" + e.getX(
32                          "; y=" + e.getY() );
33             }
34          }
35       );
37       setSize( 300, 200 );
38       show();
39    }
```

```
41    public static void main( String args[] )
42    {
43       SelfContainedPanelTest app =
44          new SelfContainedPanelTest();
45
46       app.addWindowListener(
47          new WindowAdapter() {
48             public void windowClosing(
49             {
50                System.exit( 0 );
51             }
52          }
53       );
54    }
55}
```

```
65 public class SelfContainedPanel extends
66    private int x1, y1, x2, y2;
67
68    public SelfContainedPanel()
69    {
70       addMouseListener(
71          new MouseAdapter() {
72             public void mousePressed(
                                  MouseEvent e ) {
74                x1 = e.getX();
75                y1 = e.getY();
76             }
78             public void mouseReleased(
                                    MouseEvent e ) {
80                x2 = e.getX();
81                y2 = e.getY();
82                repaint();
83             }
84          }
85       );
87       addMouseMotionListener(
88          new MouseMotionAdapter() {
91                x2 = e.getX();
92                y2 = e.getY();
93                repaint();
94             }
95          }
96       );
97    }
```

```
98
99    public Dimension getPreferredSize()
100       {
101          return new Dimension( 150, 100 );
102       }
103
104       public void paintComponent( Graphics
105       {
106          super.paintComponent( g );
107
108          g.drawOval( Math.min( x1, x2 ),
109                      Math.abs( x1 - x2 ),
110       }
111    }
```