21

Multimídia: applets e aplicativos

OBJETIVOS

- Neste capítulo, você aprenderá:
- Como obter e exibir imagens.
- Como criar animações de seqüências de imagens.
- Como criar mapas de imagem.
- Como obter, reproduzir, repetir (loop) e interromper sons, utilizando um `AudioClip`.
- Como reproduzir vídeo utilizando a interface `Player`.

Sumário

21.1 Introdução

- **Multimídia — a ‘nova onda’ do Java:**
  - **Som, imagens, imagens gráficas e vídeo.**
  - **Uma enorme área de programação.**
  - **Demanda extraordinário poder de computação.**
- **Muitos usuários de computador querem imagens gráficas coloridas, tridimensionais e de alta resolução.**
- **O Java fornece instalações multimídia extensas, incluindo:**
  - **API do Java 3D – para criar aplicações de imagens em 3D.**
  - **API do Java Media Framework (JMF) – para adicionar áudio e vídeo a uma aplicação.**
  - **API do Java Sound – para reproduzir, gravar e modificar áudio.**
  - **API do Java Speech – para entrada e saída de comandos de voz.**

21.2 Carregando, exibindo e dimensionando imagens

- Classes `Image` e `ImageIcon` — utilizadas para carregar e exibir imagens.
- Exibindo imagens:
  - Método `drawImage` de `Graphics` – utilizado para desenhar uma imagem referenciada pelo objeto `Image` (pode ser dimensionado).
  - O método `ImageIcon` `paintIcon` pode ser utilizado para desenhar uma imagem referenciada pelo objeto `ImageIcon`.
- Carregando imagens:
  - O método `Applet` `getImage` carrega uma `Image` em um applet.
  - O método `Applet` `getDocumentBase` retorna a localização do arquivo HTML do applet na Internet.
  - `ImageObservers` recebe notificações à medida que `Image` é carregada e atualiza a imagem na tela se ela não estiver completa quando exibida.
- O Java suporta vários formatos de imagem, incluindo GIF, JPEG e PNG.

Resumo

```
1  // Fig. 21.1: LoadImageAndScale.java
2  // Carrega uma imagem e exibe-a em seu tamanho original e no dobro de seu
3  // tamanho original. Carrega e exibe a mesma imagem que um ImageIcon.
4  import java.awt.Graphics;
5  import java.awt.Image;
6  import javax.swing.ImageIcon;
7  import javax.swing.JApplet;
8
9  public class LoadImageAndScale extends JApplet
10 {
11    private Image image1; // cria um objeto Image
12    private ImageIcon image2; // cria um objeto ImageIcon
13
14    // carrega a imagem quando o applet é carregado
15    public void init()
16    {
17       image1 = getImage( getDocumentBase(), "redflowers.png" );
18       image2 = new ImageIcon( "yellowflowers.png" );
19    } // fim do método init
20
21    // exibe a imagem
22    public void paint( Graphics g )
23    {
24       super.paint( g );
25
26       g.drawImage( image1, 0, 0, this ); // desenha imagem original
27
28       // desenha a imagem para ajustar a largura e a altura menos 120 pix
29       g.drawImage( image1, 0, 120, getWidth(), getHeight() - 120, this );
30
```

LoadImageAndScale.java

Retorna a localização do arquivo HTML como objeto URL

O método `getImage` retorna o objeto `Image` para o arquivo `redflowers.jpg`

Retorna a localização do arquivo HTML como objeto URL

Cria um objeto `ImageIcon` para o arquivo `yellowflowers.jpg`

Desenha a imagem armazenada em `redflowers.jpg`

Desenha a mesma imagem dimensionada com um tamanho diferente

```
31       // desenha o ícone utilizando seu método paintIcon
32       image2.paintIcon( this, g, 180, 0 );
33    } // fim do método paint
34 } // fim da classe LoadImageAndScale
```

Resumo

LoadImageAndScale.java

(2 de 2)

Dica de portabilidade 21.1

A classe `Image` é uma classe `abstract` — como resultado, os programas não podem instanciar a classe `Image` para criar objetos. Para alcançar independência de plataforma, a implementação do Java em cada plataforma fornece sua própria subclasse de `Image` para armazenar as informações da imagem.

21.3 Animação de uma série de imagens

- A animação pode ser criada exibindo uma seqüência de imagens semelhantes.
- O objeto `Timer` pode ser utilizado para especificar quando cada imagem é exibida.
- Os objetos `Timer` geram `ActionEvents` em intervalos fixos:
  - Método `start` – `Timer` deve começar a gerar eventos.
  - Método `stop` – `Timer` deve parar de gerar eventos.
  - Método `restart` – `Timer` deve começar a gerar eventos novamente.
- Método `getPreferredSize` de `Component` determina a largura e altura preferidas de um componente.
- O método `Component` `getMinimumSize` determina a largura e altura mínimas de um componente.

Resumo

LogoAnimatorJPanel.java

(1 de 4)

```
1  // Fig. 21.2: LogoAnimatorJPanel.java
2  // Animação de uma série de imagens.
3  import java.awt.Dimension;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import java.awt.Graphics;
7  import javax.swing.ImageIcon;
8  import javax.swing.JPanel;
9  import javax.swing.Timer;
10
11 public class LogoAnimatorJPanel extends JPanel
12 {
13    private final static String IMAGE_NAME = "deitel"; // nome básico de imagem
14    protected ImageIcon images[]; // array de imagens
15    private final int TOTAL_IMAGES = 30; // número de imagens
16    private int currentImage = 0; // índice de imagem atual
17    private final int ANIMATION_DELAY = 50; // retardo em milissegundos
18    private int width; // largura da imagem
19    private int height; // altura da imagem
20
21    private Timer animationTimer; // O Timer guia a animação
22
23    // construtor inicializa LogoAnimatorJPanel carregando imagens
24    public LogoAnimatorJPanel()
25    {
26       images = new ImageIcon[ TOTAL_IMAGES ];
27
```

Será utilizado para armazenar imagens a serem animadas

Resumo

```
28       // carrega 30 imagens
29       for ( int count = 0; count < images.length; count++ )
30          images[ count ] = new ImageIcon( getClass().getResource(
31             "images/" + IMAGE_NAME + count + ".gif" ) );
32
33       // esse exemplo assume que todas as imagens têm a mesma largura e altura
34       width = images[ 0 ].getIconWidth();   // obtém
35       height = images[ 0 ].getIconHeight(); // obtém
36    } // fim do construtor LogoAnimatorJPanel
37
38    // exibe imagem atual
39    public void paintComponent( Graphics g )
40    {
41       super.paintComponent( g ); // chama superclasse paintComponent
42
43       images[ currentImage ].paintIcon( this, g, 0, 0 );
44
45       // configura próxima imagem a ser desenhada apenas se o timer
46       if ( animationTimer.isRunning() )
47          currentImage = ( currentImage + 1 ) % TOTAL_IMAGES;
48    } // fim do método paintComponent
49
```

Cria e armazena `ImageIcon` para cada imagem

(2 de 4)

Configura a próxima imagem somente se `Timer` ainda estiver em execução

Resumo

```
50    // inicia a animação ou reinicia se a janela for reexibida
51    public void startAnimation()
52    {
53       if ( animationTimer == null )
54       {
55          currentImage = 0; // exibe a primeira imagem
56
57          // cria o timer
58          animationTimer =
59             new Timer( ANIMATION_DELAY, new TimerHandler() );
60
61          animationTimer.start(); // inicia o timer
62       } // fim do if
63       else // animationTimer já existe, reinicia animação
64       {
65          if ( ! animationTimer.isRunning() )
66             animationTimer.restart();
67       } // fim do else
68    } // fim do método startAnimation
69
70    // pára o timer da animação
71    public void stopAnimation()
72    {
73       animationTimer.stop();
74    } // fim do método stopAnimation
75
```

Cria `Timer` de modo que as imagens sejam exibidas em intervalos da duração `ANIMATION_DELAY`

(3 de 4)

Ativa `Timer` para iniciar a gerar eventos

Ativa `Timer` para iniciar a gerar eventos novamente

Pára a geração de eventos pelo `Timer`

Resumo

```
76    // retorna o tamanho mínimo da animação
77    public Dimension getMinimumSize()
78    {
79       return getPreferredSize();
80    } // fim do método getMinimumSize
81
82    // retorna tamanho preferido da animação
83    public Dimension getPreferredSize()
84    {
85       return new Dimension( width, height );
86    } // fim do método getPreferredSize
87
88    // classe interna para tratar eventos de ação do Timer
89    private class TimerHandler implements ActionListener
90    {
91       // responde ao evento do Timer
92       public void actionPerformed( ActionEvent actionEvent )
93       {
94          repaint(); // pinta animador novamente
95       } // fim do método actionPerformed
96    } // fim da classe TimerHandler
97 } // fim da classe LogoAnimatorJPanel
```

Define o tamanho mínimo para `JPanel`

Define tamanho preferido para `JPanel`

`LogoAnimator JPanel.java`

e 4)

Resumo

LogoAnimator.java

(1 de 2)

```
1  // Fig. 21.3: LogoAnimator.java
2  // Animação de uma série de imagens.
3  import javax.swing.JFrame;
4
5  public class LogoAnimator
6  {
7     // executa a animação em um JFrame
8     public static void main( String args[] )
9     {
10       LogoAnimatorJPanel animation = new LogoAnimatorJPanel();
11
12       JFrame window = new JFrame( "Animator test" ); // configura a janela
13       window.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
14       window.add( animation ); // adiciona painel ao frame
15
16       window.pack();  // aumenta a janela apenas o suficiente para sua GUI
17       window.setVisible( true );   // exibe a janela
18
19       animation.startAnimation();  // inicia a animação
20    } // fim do main
21 } // fim da classe LogoAnimator
```

Resumo

`LogoAnimator.java`

(2 de 2)

Observação de engenharia de software 21.1

Ao criar uma animação para utilizar em um applet, forneça um mecanismo para desativar a animação quando o usuário navegar para uma nova página Web diferente daquela em que o applet de animação reside.

Observação sobre aparência e comportamento 21.1

O tamanho-padrão de um objeto `JPanel` são 10 pixels de largura e 10 pixels de altura.

Observação sobre aparência e comportamento 21.2

Ao subclassificar `JPanel` (ou qualquer outro `JComponent`), anule o método `getPreferredSize` se o novo componente precisar ter uma largura e uma altura específicas preferidas.

Observação sobre aparência e comportamento 21.3

Se um novo componente GUI tiver uma largura e altura mínimas (isto é, se as pequenas dimensões resultarem em um componente ineficiente na tela), sobrescreva o método `getMinimumSize` para retornar a largura e a altura mínimas como uma instância da classe `Dimension`.

Observação sobre aparência e comportamento 21.4

Para muitos componentes GUI, o método `getMinimumSize` é implementado para retornar o resultado de uma chamada ao método `getPreferredSize` desse componente.

21.4 Mapas de imagem

- **Mapas de imagens são utilizados para criar páginas Web interativas.**
- **Contêm áreas ativas (*hot areas*) em que o usuário pode clicar para realizar uma tarefa.**
- **Quando o usuário posiciona o ponteiro do mouse sobre uma área ativa, normalmente uma mensagem descritiva é exibida.**
- **O método `Applet` `showStatus` exibe texto na barra de status do contêiner de um applet.**

Resumo

ImageMap.java

(1 de 5)

```
1  // Fig. 21.4: ImageMap.java
2  // Demonstrando uma mapa da imagem.
3  import java.awt.event.MouseAdapter;
4  import java.awt.event.MouseEvent;
5  import java.awt.event.MouseMotionAdapter;
6  import java.awt.Graphics;
7  import javax.swing.ImageIcon;
8  import javax.swing.JApplet;
9
10 public class ImageMap extends JApplet
11 {
12    private ImageIcon mapImage;
13
14    private static final String captions[] = { "Common Programming Error",
15       "Good Programming Practice", "Graphical User Interface Tip",
16       "Performance Tip", "Portability Tip",
17       "Software Engineering Observation", "Error-Prevention Tip" };
18
19    // configura ouvientes de mouse
20    public void init()
21    {
22       addMouseListener(
23
```

```
24          new MouseAdapter() // classe interna anônima
25          {
26             // indica quando o ponteiro do mouse sai da área do applet
27             public void mouseExited( MouseEvent event )
28             {
29                showStatus( "Pointer outside applet" );
30             } // fim do método mouseExited
31          } // fim da classe interna anônima
32       ); // fim da chamada para addMouseListener
33
34       addMouseMotionListener(
35
36          new MouseMotionAdapter() // classe interna anônima
37          {
38             // determina ícone sobre o qual o mouse aparece
39             public void mouseMoved( MouseEvent event )
40             {
41                showStatus( translateLocation(
42                   event.getX(), event.getY() ) );
43             } // fim do método mouseMoved
44          } // fim da classe interna anônima
45       ); // fim da chamada para addMouseMotionListener
46
47       mapImage = new ImageIcon( "icons.png" ); // obtém imagem
48    } // fim do método init
49
```

Resumo

ImageMap.java

(2 de 5)

Resumo

```
50    // exibe mapImage
51    public void paint( Graphics g )
52    {
53       super.paint( g );
54       mapImage.paintIcon( this, g, 0, 0 );
55    } // fim do método paint
56
57    // retorna legenda de dica com base nas coordenadas do m
58    public String translateLocation( int x, int y )
59    {
60       // se coordenar fora da imagem, retorna imediatament
61       if ( x >= mapImage.getIconWidth() || y >= mapImage.getIconHeight() )
62          return "";
63
64       // determina o número de ícone (0 - 6)
65       double iconWidth = ( double ) mapImage.getIconWidth() / 7.0;
66       int iconNumber = ( int )( ( double ) x / iconWidth );
67
68       return captions[ iconNumber ]; // retorna a legenda de ícone apropriada
69    } // fim do método translateLocation
70 } // fim da classe ImageMap
```

ImageMap.java

Método chamado quando o mouse é movido

Coordenadas do mouse atuais

Não faz nada se o mouse não estiver sobre um ícone

Determina sobre qual ícone o mouse está

Exibe o texto para a imagem sobre a qual o mouse está

Resumo

`ImageMap.java`

(4 de 5)

Resumo

`ImageMap.java`

(5 de 5)

21.5 Carregando e reproduzindo clipes de áudio

- **Programas Java podem reproduzir e manipular clipes de áudio.**
- **Reproduzindo sons em um applet:**
  - **O método `play` de `Applet` – carrega som e o reproduz uma vez.**
  - **Métodos `play`, `loop` e `stop` de `AudioClip`.**
  - **Capacidades adicionais fornecidas pelas APIs Java Media Framework e Java Sound.**
- **Carregando sons em um applet:**
  - **O método `Applet` `getAudioClip` – recupera som, retorna a referência a um `AudioClip`.**
  - **O método `play` de `Applet` carrega o som.**
- **Formatos de arquivos suportados incluem os formatos de arquivos Sun Audio, Windows Wave e MIDI.**

Resumo

LoadAudioAndPlay.java

(1 de 4)

```
1  // Fig. 21.5: LoadAudioAndPlay.java
2  // Carrega um clipe de áudio e o reproduz.
3  import java.applet.AudioClip;
4  import java.awt.event.ItemListener;
5  import java.awt.event.ItemEvent;
6  import java.awt.event.ActionListener;
7  import java.awt.event.ActionEvent;
8  import java.awt.FlowLayout;
9  import javax.swing.JApplet;
10 import javax.swing.JButton;
11 import javax.swing.JComboBox;
12
13 public class LoadAudioAndPlay extends JApplet
14 {
15    private AudioClip sound1, sound2, currentSound;
16    private JButton playJButton, loopJButton, stopJButton;
17    private JComboBox soundJComboBox;
18
19    // carrega a imagem quando o applet começa executar
20    public void init()
21    {
22       setLayout( new FlowLayout() );
23
24       String choices[] = { "Welcome", "Hi" };
25       soundJComboBox = new JComboBox( choices ); // cria a JComboBox
26
27       soundJComboBox.addItemListener(
28
```

`AudioClip` utilizado para representar arquivos de áudio

```
29          new ItemListener() // classe interna anônima
30          {
31             // interrompe o som e muda para o som selecionado pelo usuário
32             public void itemStateChanged( ItemEvent e )
33             {
34                currentSound.stop();
35                currentSound = soundJComboBox.getSelectedIndex() == 0 ?
36                   sound1 : sound2;
37             } // fim do método itemStateChanged
38          } // fim da classe interna anônima
39       ); // fim da chamada de método addItemListener
40
41       add( soundJComboBox ); // adiciona JComboBox ao applet
42
43       // configura o handler de evento de botão e os botões
44       ButtonHandler handler = new ButtonHandler();
45
46       // cria Play JButton
47       playJButton = new JButton( "Play" );
48       playJButton.addActionListener( handler );
49       add( playJButton );
50
```

Resumo

LoadAudioAndPlay.java

(2 de 4)

Resumo

LoadAudioAndPlay.java

(3 de 4)

```
51       // cria Loop JButton
52       loopJButton = new JButton( "Loop" );
53       loopJButton.addActionListener( handler );
54       add( loopJButton );
55
56       // cria Stop JButton
57       stopJButton = new JButton( "Stop" );
58       stopJButton.addActionListener( handler );
59       add( stopJButton );
60
61       // carrega sons e configura currentSound
62       sound1 = getAudioClip( getDocumentBase(), "welcome.wav" );
63       sound2 = getAudioClip( getDocumentBase(), "hi.au" );
64       currentSound = sound1;
65    } // fim do método init
66
67    // pára o som quando o usuário alterna entre páginas da Web
68    public void stop()
69    {
70       currentSound.stop(); // pára AudioClip
71    } // fim do método stop
72
```

Carrega clipes de áudio

Resumo

```
73    // private inner class to handle button events
74    private class ButtonHandler implements ActionListener
75    {
76       // process play, loop and stop button events
77       public void actionPerformed( ActionEvent actionEvent )
78       {
79          if ( actionEvent.getSource() == playJButton )
80             currentSound.play(); // play AudioClip once
81          else if ( actionEvent.getSource() == loopJButton )
82             currentSound.loop(); // play AudioClip continuously
83          else if ( actionEvent.getSource() == stopJButton )
84             currentSound.stop(); // stop AudioClip
85       } // end method actionPerformed
86    } // end class ButtonHandler
87 } // end class LoadAudioAndPlay
```

LoadAudioAndPlay

Reproduz um clipe

Reproduz um clipe múltiplas vezes

Pára a reprodução do clipe de áudio

Observação sobre aparência e comportamento 21.5

Ao reproduzir clipes de áudio em um applet ou aplicativo, forneça um mecanismo para o usuário desativar o áudio.

21.6 Reproduzindo vídeo e outros tipos de mídia com o Java Media Framework

- **Um vídeo simples pode transmitir muitas informações de maneira concisa e eficiente.**
- **A API do JMF permite que os programadores Java reproduzam, editem, transmitam por stream e capturem tipos de mídia populares.**
- **Os tipos de arquivos suportados incluem o Microsoft Audio/Video Interleave, filmes Macromedia Flash 2, Future Splash, MPEG Layer 3 Audio, MIDI, MPEG-1 videos e filmes QuickTime.**

Criando um Media Player simples

- A interface `Player` é utilizada para reproduzir vídeos.
- A classe `Manager` declara métodos utilitários para acessar os recursos do sistema a fim de reproduzir e manipular mídia.
- O método gerenciador `createRealizedPlayer` obtém um `Player` para um clipe de mídia especificado.
- Carregando e reproduzindo vídeo:
  - O método Player `getVisualComponent` obtém um componente que exibe o aspecto visual de um arquivo de mídia.
  - O método `Player` `getControlPanelComponent` obtém um componente que fornece controles de playback e de mídia.
  - O método `Player` `start` inicia a reprodução de um arquivo de mídia.

Resumo

MediaPanel.java

(1 de 2)

```
1  // Fig. 21.6: MediaPanel.java
2  // Um JPanel reproduz mídia de um URL
3  import java.awt.BorderLayout;
4  import java.awt.Component;
5  import java.io.IOException;
6  import java.net.URL;
7  import javax.media.CannotRealizeException;
8  import javax.media.Manager;
9  import javax.media.NoPlayerException;
10 import javax.media.Player;
11 import javax.swing.JPanel;
12
13 public class MediaPanel extends JPanel
14 {
15    public MediaPanel( URL mediaURL )
16    {
17       setLayout( new BorderLayout() ); // utiliza um BorderLayout
18
19       // Utiliza componentes leves para compatibilidade Swing
20       Manager.setHint( Manager.LIGHTWEIGHT_RENDERER, true );
21
22       try
23       {
24          // cria um player para reproduzir a mídia es
25          Player mediaPlayer = Manager.createRealizedP
26
27          // obtém os componentes para o vídeo a os controles de reprodução
28          Component video = mediaPlayer.getVisualComponent();
29          Component controls = mediaPlayer.getControlPanelComponent();
30
```

Utiliza um renderizador leve

Cria `Player` para um arquivo especificado por `mediaURL`

Recupera os componentes a fim de exibir o vídeo e os controles para pausar e executar o vídeo

Resumo

```
31          if ( video != null )
32             add( video, BorderLayout.CENTER ); // adiciona componente de vídeo
33
34          if ( controls != null )
35             add( controls, BorderLayout.SOUTH ); // adiciona con
36
37          mediaPlayer.start(); // inicia a reprodução do clipe de mídia
38       } // fim do try
39       catch ( NoPlayerException noPlayerException )
40       {
41          System.err.println( "No media player found" );
42       } // fim do catch
43       catch ( CannotRealizeException cannotRealizeException )
44       {
45          System.err.println( "Could not realize media player" );
46       } // fim do catch
47       catch ( IOException iOException )
48       {
49          System.err.println( "Error reading from the source" );
50       } // fim do catch
51    } // fim do construtor MediaPanel
52 } // fim da classe MediaPanel
```

Reproduz um clipe

MediaPanel.java

(2 de 2)

Resumo

MediaTest.java

(1 de 3)

```
1  // Fig. 21.7: MediaTest.java
2  // Um Media Player simples
3  import java.io.File;
4  import java.net.MalformedURLException;
5  import java.net.URL;
6  import javax.swing.JFileChooser;
7  import javax.swing.JFrame;
8
9  public class MediaTest
10 {
11    // carrega o aplicativo
12    public static void main( String args[] )
13    {
14       // cria um chooser de arquivo
15       JFileChooser fileChooser = new JFileChooser();
16
17       // mostra diálogo de arquivo aberto
18       int result = fileChooser.showOpenDialog( null );
19
20       if ( result == JFileChooser.APPROVE_OPTION ) // usuário escolheu um arquivo
21       {
22          URL mediaURL = null;
23
24          try
25          {
26             // obtém o arquivo como URL
27             mediaURL = fileChooser.getSelectedFile().toURL();
28          } // fim do try
```

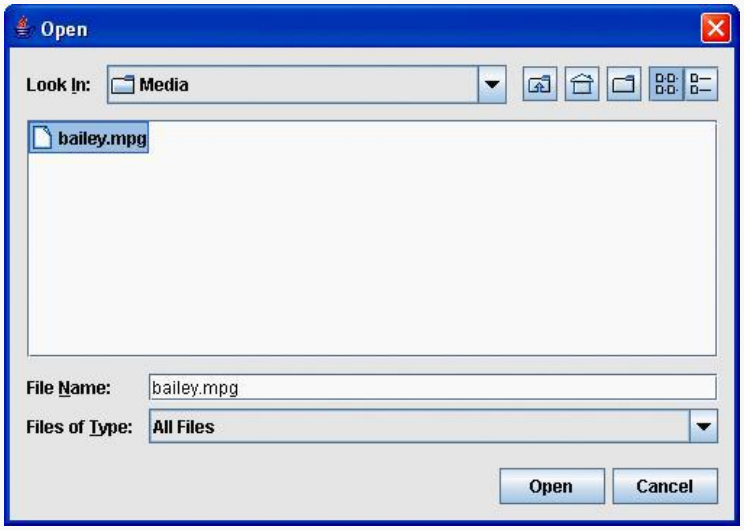
Recupera o arquivo especificado pelo usuário

Resumo

MediaTest.java

(2 de 3)

```
29          catch ( MalformedURLException malformedURLException )
30          {
31             System.err.println( "Could not create URL for the file" );
32          } // fim do catch
33
34          if ( mediaURL != null ) // exibe somente se houver um URL válido
35          {
36             JFrame mediaTest = new JFrame( "Media Tester" );
37             mediaTest.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
38
39             MediaPanel mediaPanel = new MediaPanel( mediaURL );
40             mediaTest.add( mediaPanel );
41
42             mediaTest.setSize( 300, 300 );
43             mediaTest.setVisible( true );
44          } // fim do if interno
45       } // fim do if externo
46    } // fim do main
47 } // fim da classe MediaTest
```



Resumo

`MediaTest.java`

(3 de 3)