11

Aula 2
Eventos do Mouse

11.13 Tratamento de evento de mouse

- **Eventos de mouse:**
  - **Cria um objeto** `MouseEvent`.
  - **Tratado por** `MouseListeners` **e** `MouseMotionListeners`**.**
  - `MouseInputListener` **combina as duas interfaces.**
  - **A interface** `MouseWheelListener` **declara o método** `mouseWheelMoved` **para tratar** `MouseWheelEvents`.

Métodos de interface `MouseListener` e `MouseMotionListener`

Métodos de interface ***`MouseListener`***

`public void mousePressed( MouseEvent event )`

Chamado quando um botão do mouse é pressionado enquanto o cursor de mouse estiver sobre um componente.

`public void mouseClicked( MouseEvent event )`

Chamado quando um botão do mouse é pressionado e liberado enquanto o cursor do mouse pairar sobre um componente. Esse evento é sempre precedido por uma chamada para `mousePressed`.

`public void mouseReleased( MouseEvent event )`

Chamado quando um botão do mouse é liberado depois de ser pressionado. Esse evento sempre é precedido por uma chamada para mousePressed e um ou mais chamadas para `mouseDragged`.

`public void mouseEntered( MouseEvent event )`

Chamado quando o cursor do mouse entra nos limites de um componente.

Figura 11.27 | Métodos de interface `MouseListener` e `MouseMotionListener`. (Parte 1 de 2.)

Métodos de interface `MouseListener` e `MouseMotionListener`	
`public void mouseExited( MouseEvent event )`	Chamado quando o cursor do mouse deixa os limites de um componente.
Métodos de interface `MouseMotionListener`	
`public void mouseDragged( MouseEvent event )`	Chamado quando o botão do mouse é pressionado enquanto o cursor de mouse estiver sobre um componente e o mouse é movido enquanto o botão do mouse permanecer pressionado. Esse evento é sempre precedido por uma chamada para `mousePressed`. Todos os eventos de arrastar são enviados para o componente em que o usuário começou a arrastar o mouse.
`public void mouseMoved( MouseEvent event )`	Chamado quando o mouse é movido quando o cursor de mouse estiver sobre um componente. Todos os eventos de movimento são enviados para o componente sobre o qual o mouse atualmente está posicionado.

Figura 11.27 | Métodos de interface `MouseListener` e `MouseMotionListener`. (Parte 2 de 2.)

Observação sobre a aparência e comportamento 11.12

As chamadas de método para `mouseDragged` e `mouseReleased` são enviadas ao `MouseMotionListener` do `Component` em que uma operação de arrastar do mouse se iniciou. De maneira semelhante, a chamada de método `mouseReleased` no fim de uma operação de arrastar é enviada para o `MouseListener` de `Component` em que a operação de arrastar se iniciou.

Resumo

MouseTracker Frame.java

(1 de 4)

```
1  // Fig. 11.28: MouseTrackerFrame.java
2  // Demonstrando eventos de mouse.
3  import java.awt.Color;
4  import java.awt.BorderLayout;
5  import java.awt.event.MouseListener;
6  import java.awt.event.MouseMotionListener;
7  import java.awt.event.MouseEvent;
8  import javax.swing.JFrame;
9  import javax.swing.JLabel;
10 import javax.swing.JPanel;
11
12 public class MouseTrackerFrame extends JFrame
13 {
14    private JPanel mousePanel; // painel em que eventos de mouse ocorrerão
15    private JLabel statusBar; // rótulo que exibe informações sobre evento
16
17    // construtor MouseTrackerFrame configura GUI e
18    // registra handlers de evento de mouse
19    public MouseTrackerFrame()
20    {
21       super( "Demonstrating Mouse Events" );
22
23       mousePanel = new JPanel(); // cria painel
24       mousePanel.setBackground( Color.WHITE ); // configura cor de fundo
25       add( mousePanel, BorderLayout.CENTER ); // adiciona
26
27       statusBar = new JLabel( "Mouse outside JPanel" );
28       add( statusBar, BorderLayout.SOUTH ); // adiciona rótulo ao JFrame
29
```

Cria `JPanel` para capturar eventos de mouse

Configura o fundo como branco

Cria `JLabel` e o adiciona à aplicação

```
30       // cria e registra listener para mouse e eventos de movimento de mouse
31       MouseHandler handler = new MouseHandler();
32       mousePanel.addMouseListener( handler );
33       mousePanel.addMouseMotionListener( handler );
34    } // fim do construtor MouseTrackerFrame
35
36    private class MouseHandler implements MouseListener,
37       MouseMotionListener
38    {
39       // handlers de evento MouseListener
40       // trata evento quando o mouse é liberado logo depois de pressionado
41       public void mouseClicked( MouseEvent event )
42       {
43          statusBar.setText( String.format( "Clicked at [%d, %d]",
44             event.getX(), event.getY() ) );
45       } // fim do método mouseClicked
46
47       // trata evento quando mouse é pressionado
48       public void mousePressed( MouseEvent event )
49       {
50          statusBar.setText( String.format( "Pressed at [%d, %d]",
51             event.getX(), event.getY() ) );
52       } // fim do método mousePressed
53
54       // trata evento quando mouse é liberado depois da operação de arrastar
55       public void mouseReleased( MouseEvent event )
56       {
57          statusBar.setText( String.format( "Released at [%d, %d]",
58             event.getX(), event.getY() ) );
59       } // fim do método mouseReleased
```

Cria handler de evento para eventos de mouse

Registra um handler de evento

Implementa interfaces ouvintes de mouse

(2 de 4)

Declara o método `mouseClicked`

Determina a localização do clique de mouse

Declara o método `mousePressed`

Declara o método `mouseReleased`

Resumo

```
60
61       // trata evento quando mouse entra na área
62       public void mouseEntered( MouseEvent event )
63       {
64          statusBar.setText( String.format( "Mouse entered at [%d, %d]",
65             event.getX(), event.getY() ) );
66          mousePanel.setBackground( Color.GREEN );
67       } // fim do método mouseEntered
68
69       // trata evento quando mouse sai da área
70       public void mouseExited( MouseEvent event )
71       {
72          statusBar.setText( "Mouse outside JPanel" );
73          mousePanel.setBackground( Color.WHITE );
74       } // fim do método mouseExited
75
```

Declara o método `mouseEntered`

Configura o segundo plano de `JPanel`

`MouseTracker`

(3 de 4)

Declara o método `mouseExited`

Configura o segundo plano de `JPanel`

Resumo

```
76       // MouseMotionListener event handlers
77       // trata evento MouseMotionListener
78       public void mouseDragged( MouseEvent event )
79       {
80          statusBar.setText( String.format( "Dragged at [%d, %d]",
81             event.getX(), event.getY() ) );
82       } // fim do método mouseDragged
83
84       // trata evento quanto usuário move o mouse
85       public void mouseMoved( MouseEvent event )
86       {
87          statusBar.setText( String.format( "Moved at [%d, %d]",
88             event.getX(), event.getY() ) );
89       } // fim do método mouseMoved
90    } // fim da classe MouseHandler interna
91 } // fim da classe MouseTrackerFrame
```

Declara o método `mouseDragged`

Declara o método `mouseMoved`

`MouseTracker Frame.java`

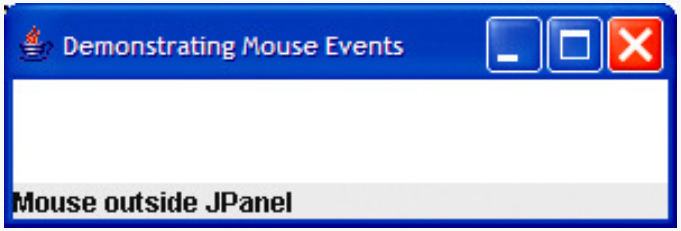
(4 de 4)

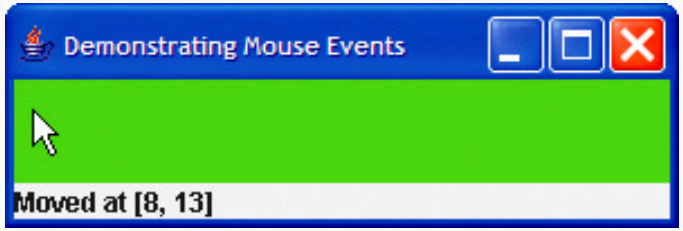
Resumo

MouseTracker Frame.java

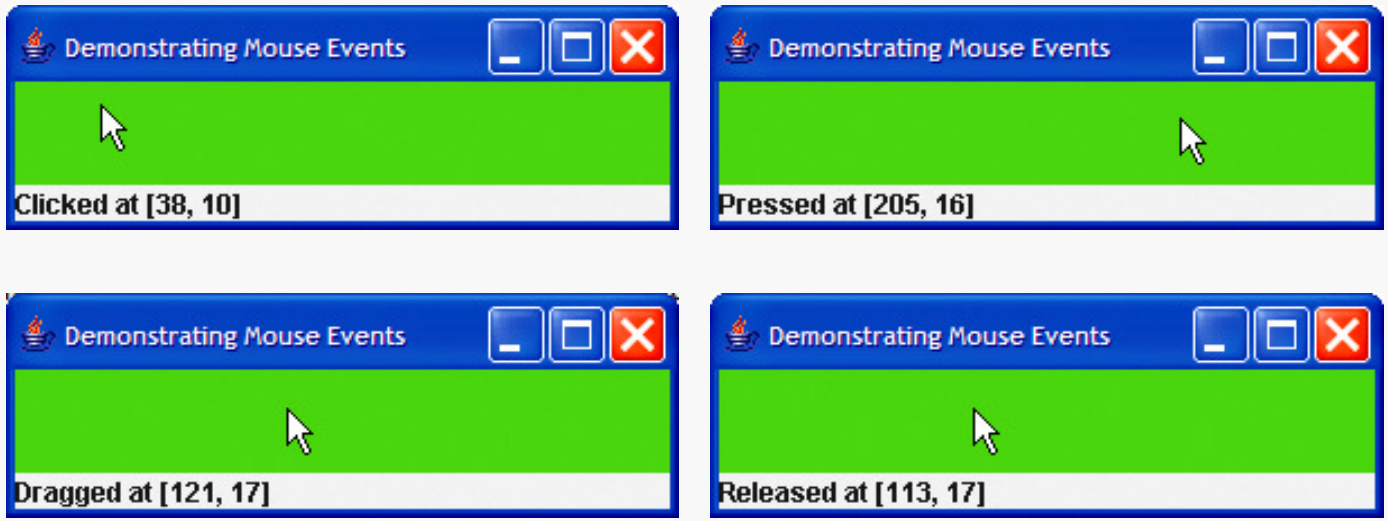
(1 de 2)

```
1  // Fig. 11.29: MouseTrackerFrame.java
2  // Testando MouseTrackerFrame.
3  import javax.swing.JFrame;
4
5  public class MouseTracker
6  {
7     public static void main( String args[] )
8     {
9        MouseTrackerFrame mouseTrackerFrame = new MouseTrackerFrame();
10       mouseTrackerFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
11       mouseTrackerFrame.setSize( 300, 100 ); // configura o tamanho do frame
12       mouseTrackerFrame.setVisible( true ); // exibe o frame
13    } // fim de main
14 } // fim da classe MouseTracker
```





Resumo



`MouseTracker Frame.java`

(2 de 2)

11.14 Classes adaptadoras

- **Classe adaptadora:**
  - **Implementa interface ouvinte de evento.**
  - **Fornece implementação-padrão para todos os métodos de tratamento de eventos.**

Observação de engenharia de software 11.7

Quando uma classe implementa uma interface, a classe tem um relacionamento *'é um'* com essa interface. Todas as subclasses diretas e indiretas dessa classe herdam essa interface. Portanto, um objeto de uma classe que estende uma classe adaptadora de evento *é um* objeto do tipo ouvinte de eventos correspondente (por exemplo, um objeto de uma subclasse de `MouseAdapter` é um `MouseListener`).

Herdando `MouseAdapter`

- `MouseAdapter`:
  - **Classe adaptadora para as interfaces `MouseListener` e `MouseMotionListener`.**
  - **Estender a classe permite sobrescrever somente os métodos que você deseja utilizar.**

Erro comum de programação 11.4

Se você estender uma classe adaptadora e digitar incorretamente o nome do método que você está sobrescrevendo, o método simplesmente torna-se outro método na classe. Esse é um erro de lógica difícil de ser detectado, visto que o programa chamará a versão vazia do método herdado da classe adaptadora.

Classe adaptadora de evento em `java.awt.event`	Implementa interface
`ComponentAdapter`	`ComponentListener`
`ContainerAdapter`	`ContainerListener`
`FocusAdapter`	`FocusListener`
`KeyAdapter`	`KeyListener`
`MouseAdapter`	`MouseListener`
`MouseMotionAdapter`	`MouseMotionListener`
`WindowAdapter`	`WindowListener`

Figura 11.30 | Classes adaptadoras de evento e as interfaces que elas implementam no pacote `java.awt.event`.

Resumo

MouseDetails Frame.java

(1 de 2)

```
1  // Fig. 11.31: MouseDetailsFrame.java
2  // Demonstrando cliques de mouse e distinguindo entre botões do mouse.
3  import java.awt.BorderLayout;
4  import java.awt.Graphics;
5  import java.awt.event.MouseAdapter;
6  import java.awt.event.MouseEvent;
7  import javax.swing.JFrame;
8  import javax.swing.JLabel;
9
10 public class MouseDetailsFrame extends JFrame
11 {
12    private String details; // representação String
13    private JLabel statusBar; // JLabel que aparece no botão de janela
14
15    // construtor configura barra de título String e registra o listener de mouse
16    public MouseDetailsFrame()
17    {
18       super( "Mouse clicks and buttons" );
19
20       statusBar = new JLabel( "Click the mouse" );
21       add( statusBar, BorderLayout.SOUTH );
22       addMouseListener( new MouseClickHandler() ); // adiciona handler
23    } // fim do construtor MouseDetailsFrame
24
```

Registra um handler de evento

Resumo

MouseDetails Frame.java

(2 de 2)

```
25    // classe interna para tratar eventos de mouse
26    private class MouseClickHandler extends MouseAdapter
27    {
28       // trata evento de clique de mouse e determina qual botão foi pressionado
29       public void mouseClicked( MouseEvent event )
30       {
31          int xPos = event.getX(); // obtém posição x do mouse
32          int yPos = event.getY(); // obtém posição y do mouse
33
34          details = String.format( "Clicked %d time(s)",
35             event.getClickCount() );
36
37          if ( event.isMetaDown() ) // botão direito do mouse
38             details += " with right mouse button";
39          else if ( event.isAltDown() ) // botão do meio do mouse
40             details += " with center mouse button";
41          else // botão esquerdo do mouse
42             details += " with left mouse button";
43
44          statusBar.setText( details ); // exibe mensagem na statusBar
45       } // fim do método mouseClicked
46    } // fim da classe interna private MouseClickHandler
47 } // fim da classe MouseDetailsFrame
```

Obtém o número de vezes que o botão do mouse foi clicado

Testa se o botão direito do mouse foi clicado

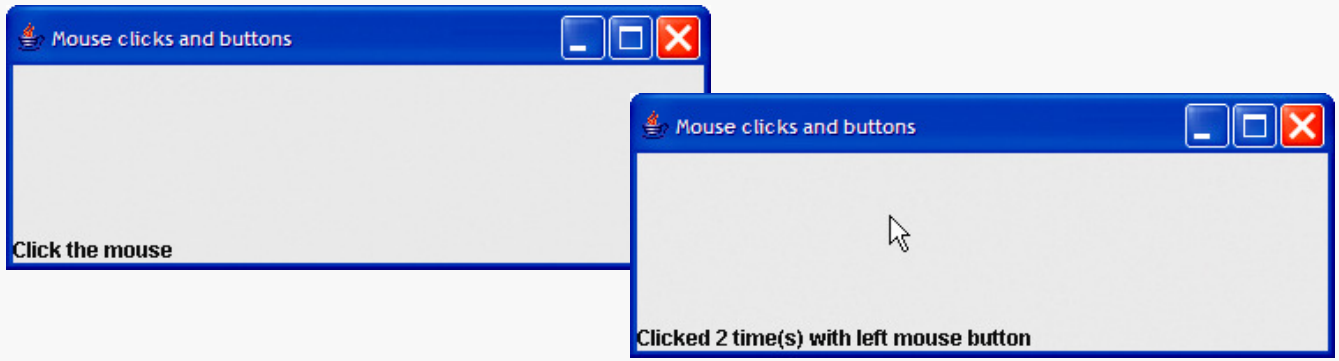
Testa se o botão do meio do mouse foi clicado

Resumo

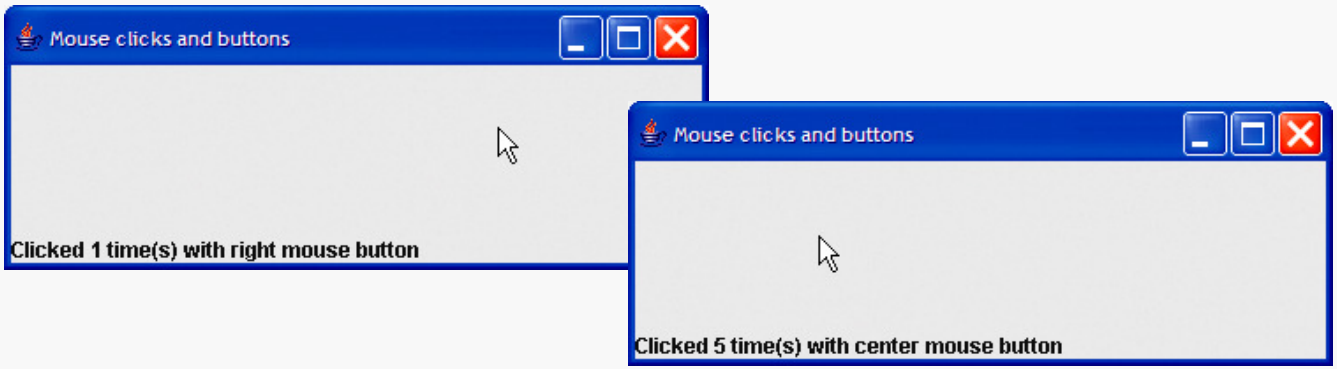
MouseDetails.java

(1 de 2)

```
1  // Fig. 11.32: MouseDetails.java
2  // Testando MouseDetailsFrame.
3  import javax.swing.JFrame;
4
5  public class MouseDetails
6  {
7     public static void main( String args[] )
8     {
9        MouseDetailsFrame mouseDetailsFrame = new MouseDetailsFrame();
10       mouseDetailsFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
11       mouseDetailsFrame.setSize( 400, 150 ); // configura o tamanho do frame
12       mouseDetailsFrame.setVisible( true ); // obtém o frame
13    } // fim de main
14 } // fim da classe MouseDetails
```



Resumo



`MouseDetails.java`

(2 de 2)

Método `InputEvent`	Descrição
`isMetaDown()`	Retorna `true` quando o usuário clica no botão direito do mouse em um mouse com dois ou três botões. Para simular um clique de botão direito com um mouse de um botão, o usuário pode manter pressionada a tecla *Meta* no teclado e clicar no botão do mouse.
`isAltDown()`	Retorna `true` quando o usuário clica no botão do mouse do meio em um mouse com três botões. Para simular um clique com o botão do meio do mouse em um mouse com um ou dois botões, o usuário pode pressionar a tecla *Alt* no teclado e clicar no único botão ou no botão esquerdo do mouse.

Figura 11.33 | Os métodos `InputEvent` que ajudam a distinguir entre os cliques do botão esquerdo, do centro e direito do mouse.