Manipulação de Eventos
Mouse e Teclado

Aula 02 · Listeners, classes adaptadoras e programação orientada a eventos

Práticas de POO

Ciência da Computação · Graduação

`ActionListener` · `MouseListener` · `MouseMotionListener` · `KeyListener` · `Adapter`

AULA 02

O que você vai aprender hoje

01 **Programação orientada a eventos**
Entender por que apps gráficos são assíncronos e como os eventos fluem.

02 **Modelo de delegação**
Conhecer o caminho do evento: usuário, SO, JVM e tratador.

03 **Eventos de componentes**
Usar ActionListener e o método actionPerformed.

04 **Eventos do mouse**
Tratar cliques, entrada, saída e arrasto com MouseListener.

05 **Eventos de teclado**
Capturar teclas pressionadas e liberadas com KeyListener.

06 **Classes adaptadoras**
Simplificar com MouseAdapter, KeyAdapter e cia.

CONCEITO

O que são eventos?

- Programas gráficos são **assíncronos**: qualquer coisa pode acontecer a qualquer momento.
- O programa não sabe predizer quando o usuário vai clicar em um botão.

EXEMPLOS DE EVENTOS

- Clique do mouse
- Movimentação do mouse
- Tecla pressionada
- Janela aberta ou fechada
- Componente ganhando foco

A SOLUÇÃO

Orientação a eventos

As interações do usuário são capturadas e passadas para o programa processar, por meio do modelo de delegação de eventos.

CONCEITO

Como um evento funciona

1

Usuário age

Clica, move o mouse ou pressiona uma tecla.

2

SO detecta

O sistema operacional monitora a ação do hardware.

3

SO avisa a JVM

O evento é entregue ao programa em execução.

4

Tratador executa

O método registrado para aquele evento é chamado.

O papel do programador

Construir tratadores de eventos capazes de processar cada tipo de ocorrência (clique, movimento, tecla) e registrá-los nos componentes para que a JVM possa usá-los.

ARQUITETURA

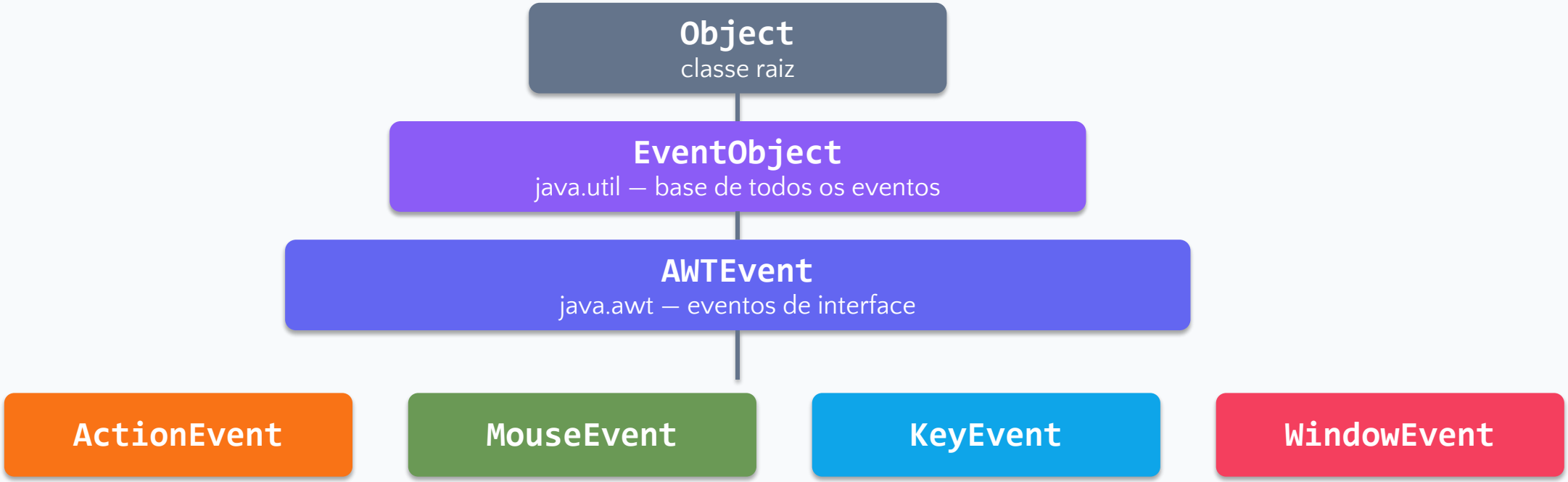
O caminho de um evento

EXEMPLO REAL

- Quando o mouse entra em um componente, o evento `MouseEntered` é gerado.
- A JVM verifica se o componente tem um listener registrado para esse tipo. Se tiver, o controle é passado e o método associado é executado.

ARQUITETURA

Hierarquia das classes de eventos



Cada tipo de evento tem sua própria classe.

MouseEvent carrega as coordenadas; KeyEvent carrega a tecla; ActionEvent indica uma ação de componente.

CONCEITO

O que é um Listener?

- `Listener` significa **"ouvinte"**.
- É uma interface que fica à espera de um evento e, quando ele ocorre, aciona um método específico.

1 Implementar

Sua classe implementa a interface listener.

```
implements MouseListener
```

2 Sobrescrever

Você escreve o corpo dos métodos de tratamento.

```
public void mouseClicked(...)
```

3 Registrar

O listener é anexado ao componente.

```
botao.addMouseListener(handler)
```

COMPONENTES

ActionListener: eventos de ação

- Trata ações de componentes, como o clique em um **`JButton`**.
- Possui um único método: **`actionPerformed`**.

IMPORTS NECESSÁRIOS

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

`ActionEvent`

é o objeto que carrega as informações da ação, como qual componente a disparou.

O MÉTODO

```
public void actionPerformed(ActionEvent e) {
    // ações executadas
    // quando o evento ocorre
}
```

É aqui dentro que você programa o que o componente faz — por exemplo, o que acontece ao clicar em "Gravar".

EXEMPLO

ActionListener — classe separada (1/5)

TrataEvento.java — tratador dedicado

```java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JOptionPane;

// Classe dedicada a tratar os eventos dos botões
public class TrataEvento implements ActionListener {

    private JButton botaoOK, botaoCancela;

    public TrataEvento(JButton botaoOK, JButton botaoCancela) {
        this.botaoOK = botaoOK;
        this.botaoCancela = botaoCancela;
    }

```

EXEMPLO

ActionListener — classe separada (2/5)

TrataEvento.java — tratador dedicado

```
15
16      @Override
17      public void actionPerformed(ActionEvent evento) {
18          if (evento.getSource() == botaoOK) {
19              JOptionPane.showMessageDialog(null, "Botão OK foi clicado");
20          }
21          if (evento.getSource() == botaoCancela) {
22              JOptionPane.showMessageDialog(null, "Botão CANCELA foi clicado");
23          }
24      }
25  }
```

EXEMPLO

ActionListener — classe separada (3/5)

EventoComponente.java — registrando o tratador

```java
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JButton;

public class EventoComponente extends JFrame {

    private JButton btnOK = new JButton("OK");
    private JButton btnCancela = new JButton("Cancela");
    private TrataEvento evento;   // referência ao tratador

```

EXEMPLO

ActionListener — classe separada (4/5)

EventoComponente.java — registrando o tratador

```
10
11      public EventoComponente() {
12          super("Tratamento de Eventos de Componentes");
13          setLayout(new FlowLayout());
14          evento = new TrataEvento(btnOK, btnCancela);
15
16          btnOK.addActionListener(evento);       // registra o listener
17          add(btnOK);
18          btnCancela.addActionListener(evento);  // registra o listener
19          add(btnCancela);
20      }
21
```

EXEMPLO

ActionListener — classe separada (5/5)

EventoComponente.java — registrando o tratador

```
21
22      public static void main(String[] args) {
23          EventoComponente obj = new EventoComponente();
24          obj.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25          obj.setSize(350, 200);
26          obj.setLocation(300, 300);
27          obj.setVisible(true);
28      }
29  }
```

DESAFIO

ActionListener — direto no componente

- Usando como modelo a classe anterior `EventoComponente()`, implemente a classe `EventoComponenteDireto`, e adicione os eventos diretamente aos botões `btnOK` e `btnCancela` abaixo.

Tratamento direto com classe anônima

```
01  btnOK.addActionListener(new ActionListener() {
02      @Override
03      public void actionPerformed(ActionEvent e) {
04          JOptionPane.showMessageDialog(null, "Clicou no Botão OK");
05      }
06  });
07
08  btnCancela.addActionListener(new ActionListener() {
09      @Override
10      public void actionPerformed(ActionEvent e) {
11          JOptionPane.showMessageDialog(null, "Clicou no Botão CANCELA");
12          System.exit(0);   // encerra a aplicação
13      }
14  });
```

DECISÃO

Classe separada x tratamento direto

Classe que implementa a interface

- Centraliza o tratamento em um só lugar
- Um tratador para vários componentes
- Código mais organizado em telas grandes
- Usa getSource() para identificar a origem

Tratamento direto (classe anônima)

- Cada componente tem o próprio tratamento
- Código fica junto do componente
- Mais rápido para poucos componentes
- Dispensa getSource()

O resultado final das duas abordagens é o mesmo — a escolha depende do tamanho e da organização do projeto.

MOUSE

MouseListener: 5 eventos do mouse

- Os eventos gerados por ações do mouse são tratados pela interface `MouseListener`, que disponibiliza cinco métodos.

`mouseClicked(MouseEvent e)`	quando um botão do mouse é clicado
`mousePressed(MouseEvent e)`	quando um botão do mouse é pressionado
`mouseReleased(MouseEvent e)`	quando um botão do mouse é solto
`mouseEntered(MouseEvent e)`	quando o mouse entra em um componente
`mouseExited(MouseEvent e)`	quando o mouse sai de um componente

MOUSE

Entendendo cada método do mouse

O OBJETO MouseEvent

- getX() — posição horizontal do clique
- getY() — posição vertical do clique
- getButton() — qual botão foi usado
- getClickCount() — nº de cliques

Com getX() e getY() você sabe exatamente onde o usuário clicou dentro do componente.

EXEMPLO: mouseClicked

```
@Override
public void mouseClicked(MouseEvent e) {
    int x = e.getX();
    int y = e.getY();
    statusBar.setText(
        "Clicado em [" + x + ", " + y + "]"
    );
}
```

MOUSE

MouseMotionListener: movimento do mouse

- Os eventos de **movimento** do mouse são tratados pela interface **`MouseMotionListener`**, com dois métodos.

`mouseMoved(MouseEvent e)`

Disparado quando o mouse se move dentro do componente ou janela, sem nenhum botão pressionado.

`mouseDragged(MouseEvent e)`

Disparado quando o mouse se move com um botão pressionado — base do recurso arrastar-e-soltar.

Registro separado

MouseMotionListener é registrado com addMouseMotionListener(), diferente do addMouseListener() usado pelo MouseListener.

EXEMPLO

Eventos do Mouse — 1/6: estrutura

EventosMouse.java — parte 1

```java
import java.awt.Color;
import java.awt.BorderLayout;
import java.awt.event.*;
import javax.swing.*;

public class EventosMouse extends JFrame {

    private JPanel mousePanel;   // painel onde os eventos ocorrerão
    private JLabel statusBar;    // rótulo que exibe informações do evento

```

EXEMPLO

Eventos do Mouse — 2/6: estrutura

EventosMouse.java — parte 1

```
11      public EventosMouse() {
12          super("Demonstração de Eventos do Mouse");
13
14          mousePanel = new JPanel();
15          mousePanel.setBackground(Color.WHITE);
16          add(mousePanel, BorderLayout.CENTER);
17
18          statusBar = new JLabel("Mouse fora do JPanel");
19          add(statusBar, BorderLayout.SOUTH);
20
21          MouseHandler handler = new MouseHandler();
22          mousePanel.addMouseListener(handler);
23          mousePanel.addMouseMotionListener(handler);
24      }
```

EXEMPLO

Eventos do Mouse — 3/6: MouseListener

EventosMouse.java — parte 2

```
24
25  private class MouseHandler implements MouseListener, MouseMotionListener {
26
27      public void mouseClicked(MouseEvent e) {
28          statusBar.setText(String.format("Clicado em [%d, %d]", e.getX(), e.getY()));
29      }
30      public void mousePressed(MouseEvent e) {
31          statusBar.setText(String.format("Pressionado em [%d, %d]", e.getX(), e.getY()));
32      }
33      public void mouseReleased(MouseEvent e) {
34          statusBar.setText(String.format("Liberado em [%d, %d]", e.getX(), e.getY()));
35      }
36
```

EXEMPLO

Eventos do Mouse — 4/6: movimento

EventosMouse.java — parte 2

```
36
37      public void mouseEntered(MouseEvent e) {
38          statusBar.setText("Mouse na área");
39          mousePanel.setBackground(Color.BLUE);
40      }
41      public void mouseExited(MouseEvent e) {
42          statusBar.setText("Mouse fora do JPanel");
43          mousePanel.setBackground(Color.WHITE);
44      }
45
```

EXEMPLO

Eventos do Mouse — 5/6: movimento

EventosMouse.java — parte 3

```
46
47      // métodos do MouseMotionListener
48      public void mouseDragged(MouseEvent e) {
49          statusBar.setText(String.format("Arrastando em [%d, %d]", e.getX(), e.getY()));
50      }
51      public void mouseMoved(MouseEvent e) {
52          statusBar.setText(String.format("Movido em [%d, %d]", e.getX(), e.getY()));
53      }
54  }
55
```

EXEMPLO

Eventos do Mouse — 6/6: main

EventosMouse.java — parte 3

```
55
56      public static void main(String[] args) {
57          // Boa prática: cria a GUI na Event Dispatch Thread
58          SwingUtilities.invokeLater(() -> {
59              EventosMouse janela = new EventosMouse();
60              janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
61              janela.setSize(500, 300);
62              janela.setLocationRelativeTo(null);
63              janela.setVisible(true);
64          });
65      }
66  }
```

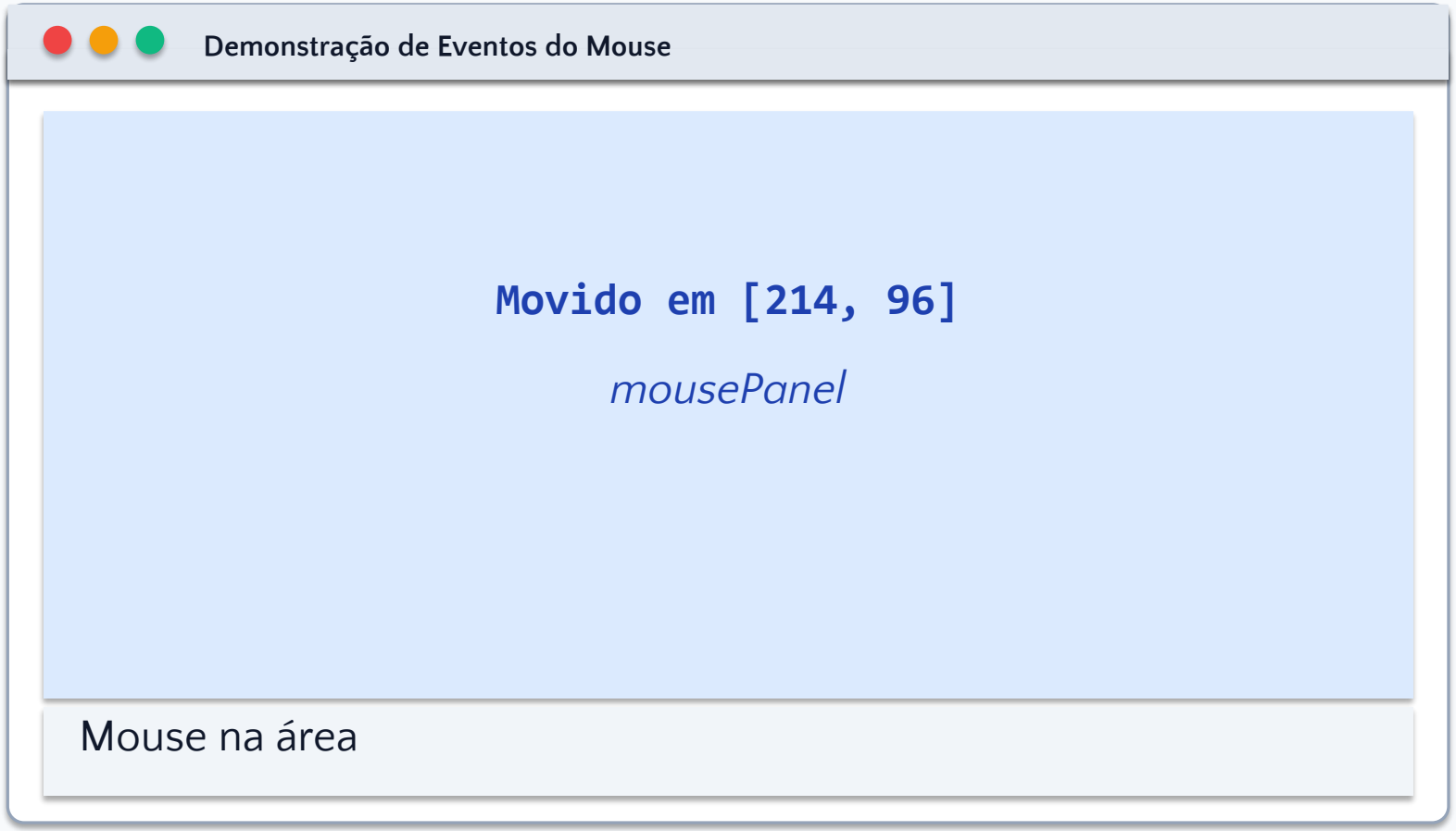
RESULTADO

O exemplo do mouse em execução

mouseEntered

mouseMoved

statusBar



TECLADO

KeyListener: eventos de teclado

- Eventos de teclado são gerados quando teclas são pressionadas ou liberadas, tratados pela interface `KeyListener`.

`keyPressed(KeyEvent e)`	Invocado quando uma tecla é pressionada.
`keyReleased(KeyEvent e)`	Invocado quando uma tecla é solta após ser pressionada.
`keyTyped(KeyEvent e)`	Invocado quando um caractere é digitado (captura a tecla).

Métodos úteis de KeyEvent:
`getKeyChar()` · `getKeyCode()` · `getKeyText()` · `isActionKey()`

EXEMPLO

Eventos de Teclado — 1/4: estrutura

EventosTeclado.java — parte 1

```java
import java.awt.Color;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.JFrame;
import javax.swing.JTextArea;

public class EventosTeclado extends JFrame implements KeyListener {

    private String line1 = "", line2 = "", line3 = "";
    private JTextArea textArea;   // área que exibe a saída

```

EXEMPLO

Eventos de Teclado — 2/4: estrutura

EventosTeclado.java — parte 1

```
11
12      public EventosTeclado() {
13          super("Demonstrando Eventos de Teclado");
14
15          textArea = new JTextArea(10, 15);
16          textArea.setText("Pressione uma tecla...");
17          textArea.setEnabled(false);
18          textArea.setDisabledTextColor(Color.BLACK);
19          add(textArea);
20
21          addKeyListener(this);   // o frame processa os eventos de teclado
22      }
23
```

EXEMPLO

Eventos de Teclado — 3/4: os três métodos

EventosTeclado.java — parte 2

```
23
24      public void keyPressed(KeyEvent e) {
25          line1 = String.format("Tecla pressionada: %s",
26                                KeyEvent.getKeyText(e.getKeyCode()));
27      }
28      public void keyReleased(KeyEvent e) {
29          line1 = String.format("Tecla liberada: %s",
30                                KeyEvent.getKeyText(e.getKeyCode()));
31      }
32      public void keyTyped(KeyEvent e) {
33          line1 = String.format("Tecla digitada: %s", e.getKeyChar());
34      }
35
```

EXEMPLO

Eventos de Teclado — 4/4: método main

EventosTeclado.java — parte 2

```
35
36      public static void main(String[] args) {
37          SwingUtilities.invokeLater(() -> {
38              EventosTeclado janela = new EventosTeclado();
39              janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
40              janela.setSize(400, 250);
41              janela.setLocationRelativeTo(null);
42              janela.setVisible(true);
43          });
44      }
45  }
```

SIMPLIFICAÇÃO

Interfaces x Classes Adaptadoras

- Muitas interfaces têm **vários métodos**, mas nem sempre você precisa de todos.
- Implementar a interface obriga a escrever todos, mesmo os vazios.

O PROBLEMA

```
// é preciso implementar TODOS
public void mouseClicked(MouseEvent e) { }
public void mousePressed(MouseEvent e) { ... }
public void mouseReleased(MouseEvent e) { }
public void mouseEntered(MouseEvent e) { }
public void mouseExited(MouseEvent e) { }
```

A SOLUÇÃO: ADAPTER

```
// estende a adaptadora e sobrescreve
// só o que precisa
public class Handler extends MouseAdapter {
    @Override
    public void mousePressed(MouseEvent e) {
        // apenas este método
    }
}
```

REFERÊNCIA

Classes adaptadoras disponíveis

Classe Adaptadora	Interface Listener
`ComponentAdapter`	`ComponentListener`
`ContainerAdapter`	`ContainerListener`
`FocusAdapter`	`FocusListener`
`KeyAdapter`	`KeyListener`
`MouseAdapter`	`MouseListener`
`MouseMotionAdapter`	`MouseMotionListener`
`WindowAdapter`	`WindowListener`

ActionListener não possui adaptadora
— ele tem apenas um método (actionPerformed), então não há necessidade.

EXEMPLO

Usando MouseAdapter na prática — 1/2: estrutura

EventoAdapter.java — sobrescrevendo apenas um método

```java
import java.awt.FlowLayout;
import java.awt.event.*;
import javax.swing.*;

public class EventoAdapter extends JFrame {

    private JButton btnOK = new JButton("Botão OK");

    public EventoAdapter() {
        setTitle("Classe Adaptadora de Evento");
        setSize(400, 300);
        setLocationRelativeTo(null);
        setLayout(new FlowLayout());
        add(btnOK);

```

EXEMPLO

Usando MouseAdapter na prática — 2/2: estrutura e main

EventoAdapter.java — sobrescrevendo apenas um método

```
15
16          // estende MouseAdapter e sobrescreve só o mousePressed
17          addMouseListener(new MouseAdapter() {
18              @Override
19              public void mousePressed(MouseEvent e) {
20                  JOptionPane.showMessageDialog(null, "Mouse Pressionado");
21              }
22          });
23      }
24
25      public static void main(String[] args) {
26          SwingUtilities.invokeLater(() -> new EventoAdapter().setVisible(true));
27      }
28  }
```

DECISÃO

Quando usar interface x adaptadora

Implementar a interface

- Use quando precisar de TODOS os métodos
- Sua classe pode estender outra classe
- Obrigatório para ActionListener
- Mais verboso, porém flexível

Estender a adaptadora

- Use quando precisar de POUCOS métodos
- Código mais limpo e direto
- Sua classe fica "presa" à herança
- Ideal para MouseAdapter e KeyAdapter

Regra prática: precisa de quase todos os métodos → interface. Precisa de só um ou dois → adaptadora.

MAPA

Todos os listeners desta aula

Listener	Método principal	O que trata
`ActionListener`	`actionPerformed`	Ações de componentes (botões)
`MouseListener`	`5 métodos`	Clique, entrada e saída do mouse
`MouseMotionListener`	`mouseMoved / Dragged`	Movimento e arrasto do mouse
`KeyListener`	`keyPressed/Typed/Released`	Teclas pressionadas e liberadas

Registro
`addActionListener()` · `addMouseListener()` · `addMouseMotionListener()` · `addKeyListener()`

ATENÇÃO

Erros comuns ao tratar eventos

Esquecer de registrar o listener

Sem addMouseListener() ou addActionListener(), o evento nunca é tratado.

Criar GUI fora da EDT

Use SwingUtilities.invokeLater() para evitar travamentos.

Implementar interface e esquecer métodos

Todos os métodos da interface precisam existir, mesmo vazios.

Usar getSource() sem comparar direito

Compare com == apenas para objetos; para texto, use equals().

Tratar evento no componente errado

O listener deve ser registrado no componente que gera o evento.

Confundir keyPressed com keyTyped

keyTyped captura caracteres; keyPressed captura qualquer tecla.

PRÁTICA

Exercícios: programe os eventos

EXERCÍCIO 1 — Fácil

Contador de cliques

Crie um frame com um JButton e um JLabel. A cada clique no botão (ActionListener), incremente um contador e atualize o JLabel mostrando "Cliques: N".

EXERCÍCIO 2 — Médio

Radar do mouse

Crie um frame com um JPanel. Usando MouseListener e MouseMotionListener, mostre em um JLabel as coordenadas do mouse e mude a cor de fundo quando o mouse entrar/sair do painel.

EXERCÍCIO 3 — Desafio

Detector de teclas

Crie um frame com KeyListener que mostre em um JTextArea o nome da tecla pressionada, se é tecla de função e se há modificador (Ctrl, Shift). Use uma classe adaptadora para simplificar.

Dica: comece registrando o listener e deixando os métodos vazios, depois preencha um método por vez.

Resumo da aula

Sua interface agora responde ao usuário

- Programação orientada a eventos
- Modelo de delegação: usuário → SO → JVM → tratador
- ActionListener e actionPerformed
- MouseListener e MouseMotionListener
- KeyListener e eventos de teclado
- Classes adaptadoras (MouseAdapter, KeyAdapter)

PRÓXIMA AULA

Menus e organização de módulos de sistema

Vamos criar menus personalizados para executar os módulos e formulários de sistema.

```
JMenu menu = new JMenu()
```