Lição 8

Tratamento de Eventos em Interfaces Gráficas

Objetivos

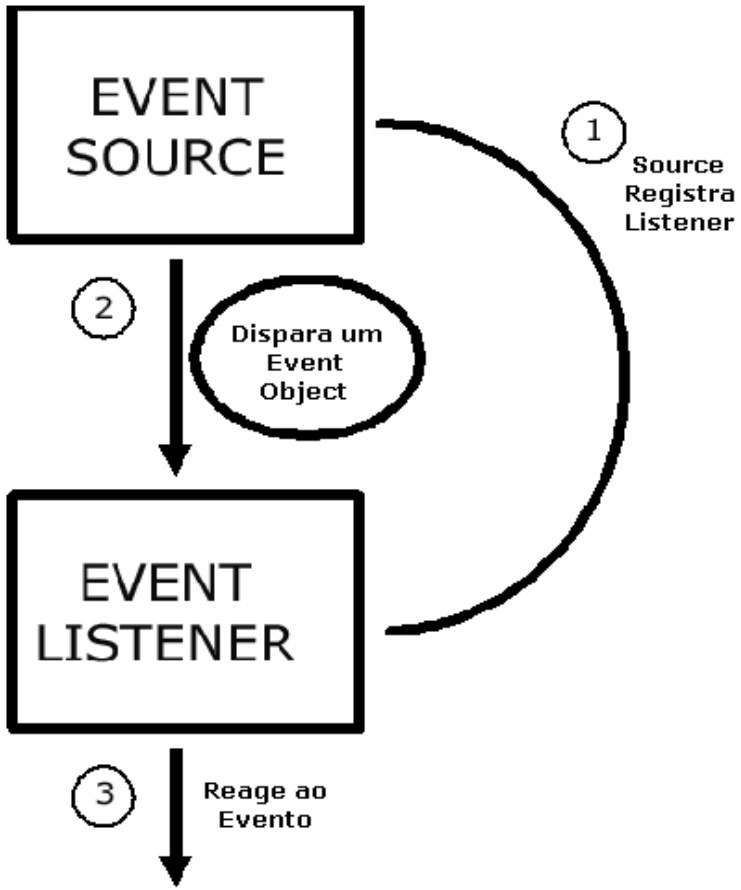
Ao final desta lição, o estudante será capaz de:

- Enumerar os componentes do modelo de delegação de eventos
- Explicar como o modelo de delegação de eventos funciona
- Criar aplicações gráficas que interajam com o usuário
- Discutir os benefícios das classes *Adapter*
- Discutir as vantagens de utilizar *Inner Class* e *Anonymous Inner Class*

Modelo de delegação de eventos

- Modelo de delegação de eventos
- Componentes:
  - *Event Source*
  - *Event Listener/Handler*
  - *Event Object*

Modelo de delegação de eventos



Registro de *Listener*

- *Event source* registrando um *listener*:

```
void add<Tipo>Listener(<Tipo>Listener
  listenerObj)
```

- *Event source* removendo um *listener*:

```
void remove<Tipo>Listener(<Tipo>Listener
  listenerObj)
```

Classes de evento

- Um *event object* tem uma classe evento como seu tipo de dado de referência
- Classe *EventObject*
- Classe *AWTEvent*

Classes de evento

- *ComponentEvent*
- *InputEvent*
- *ActionEvent*
- *ItemEvent*
- *KeyEvent*
- *MouseEvent*
- *TextEvent*
- *WindowEvent*

Listener de evento

- Classes que implementam as interfaces *<Tipo>Listener*

- Interfaces *<Tipo>Listener*:
  - *ActionListener*
  - *MouseListener*
  - *MouseMotionListener*
  - *WindowListener*

Método de *ActionListener*

```
public void actionPerformed(ActionEvent e)
```

Métodos de *MouseListener*

```
public void mouseClicked(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
```

Métodos de *MouseMotionListener*

```
public void mouseDragged(MouseEvent e)
public void mouseMoved(MouseEvent e)
```

Métodos de *WindowListener*

```
public void windowOpened(WindowEvent e)
public void windowClosing(WindowEvent e)
public void windowClosed(WindowEvent e)
public void windowActivated(WindowEvent e)
public void windowDeactivated(WindowEvent e)
public void windowIconified(WindowEvent e)
public void windowDeiconified(WindowEvent e)
```

Guia para criação de aplicações gráficas

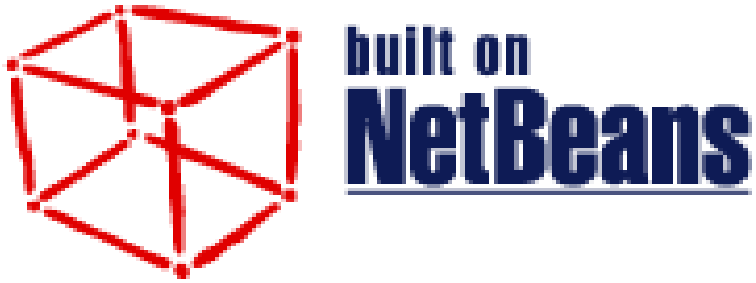
- Etapas:
  1. Criar uma classe que descreva e mostre a aparência da sua aplicação gráfica
  2. Criar uma classe que implemente a interface *listener* apropriada. Esta classe poderá estar inserida na classe do primeiro passo
  3. Na classe implementada, sobrepor TODOS os métodos da interface *listener*. Descreva em cada método como o evento deve ser tratado. Pode-se deixar vazio o método que não se desejar tratar
  4. Registre o objeto *listener* (a instância da classe *listener* do passo 2) no *event source* utilizando o método add<Tipo>Listener

Anonymous Inner Class

- *Anonymous Inner Class*, não são declaradas
- Porque usar *anonymous inner class*?

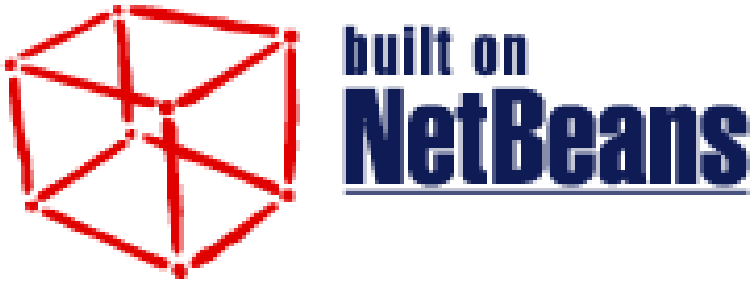
Action Events: Exemplo

- Passaremos agora para o NetBeans



Close Window: Exemplo

- Passaremos agora para o NetBeans

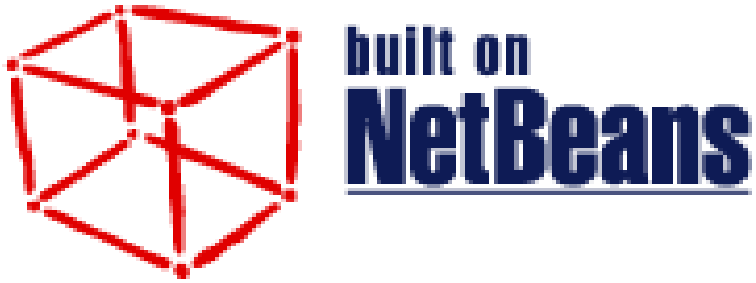


Classes *Adapter*

- Porque usar classes *Adapter*?
- Classes *Adapter*

Classes *Adapter*: Exemplo *Close Window*

- Passaremos agora para o NetBeans

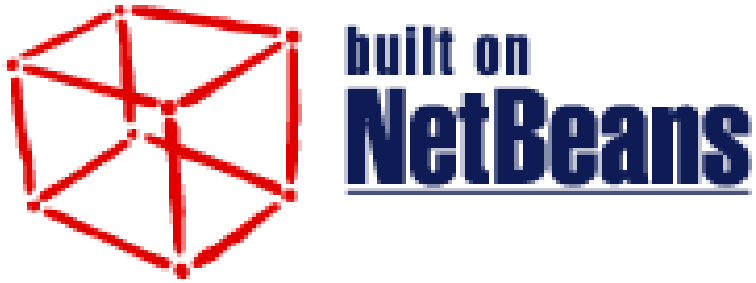


Inner Class

- Classe declarada dentro de outra classe
- Porque usar inner class?

Inner Class: Exemplo *Close Window*

- Passaremos agora para o NetBeans



Sumário

- Modelo de delegação de eventos
- Componentes do modelo de delegação de eventos
- Classes *Event*
- *Anonymous Inner Class*
- *Listeners* de evento
- Criando aplicações gráficas com tratamento de eventos
- Simplificando seu código:
  - Classes *Adapter*
  - *Inner Class*

Parceiros

- **Os seguintes parceiros tornaram JEDI™ possível em Língua Portuguesa:**