 UNIVERSIDADE PAULISTA

ALPOO

Teoria - Java

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

1

 UNIVERSIDADE PAULISTA

Disciplina: ALPOO

Tema: Criando Interfaces – Listeners

Professor: Ricardo Drudi

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

2

POO - Java

 UNIVERSIDADE PAULISTA




Orientação por Eventos

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

7

Orientação por Eventos

 UNIVERSIDADE PAULISTA

- **Motivação**
 - Um modelo de programação que tornou-se bastante difundido com o uso de interfaces gráficas foi a *programação orientada por eventos*.
 - Segundo esse modelo, o programa deixa de ter o controle do fluxo de execução, que passa a um módulo encarregado de gerenciar a *interface*.
 - Assim, o programa passa a ser chamado pelo módulo quando algum *evento* é gerado na *interface*.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

8

Orientação por Eventos



- Callback
 - para que o programa possa ser chamado pelo módulo, ele deve registrar uma **função** para cada evento de *interface* que ele desejar tratar.
 - essas funções são chamadas de **callbacks** por permitirem que o programa principal 'chame de volta' o controle do fluxo para si.
 - o modelo orientado a eventos é ortogonal ao modelo de orientação por objetos.
 - problema: não possui o conceito de função. Como resolver então?

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

9

Orientação por Eventos



- Solução – Objeto Callback
 - A solução é utilizar um **objeto** que faça o papel de *callback*. Ou seja, onde registraríamos uma função, passamos a registrar um objeto. Assim, quando o sistema precisar executar a *callback*, ele deverá executar um determinado método do objeto. Esse método, então, fará o tratamento do evento.
- Análise de Caso – `java.awt`
 - O pacote `java.awt` é um pacote padrão de Java que implementa um sistema de *interface*. Esse sistema possui janelas, textos, botões e diversos outros elementos.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

10

Pacote `java.awt`



- Como a maioria dos sistemas de interface, o AWT é um sistema orientado por eventos, ou seja, o tratamento dos eventos deve ser feito por meio de *callbacks*.
- Callbacks em Java
 - Como Java é uma linguagem puramente OO, *callbacks* devem ser implementadas através de objetos.
 - Um objeto que implementa uma *callback* é chamado de *listener*.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

11

Observer

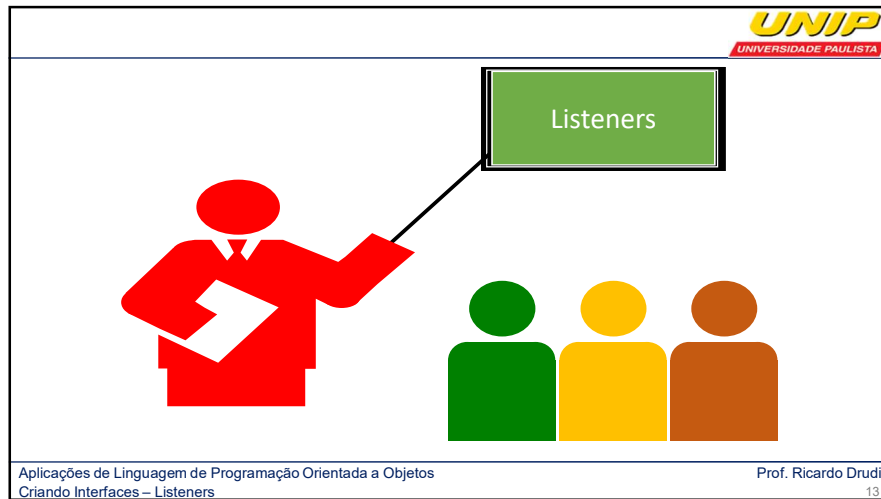


- Observer é um dos **design patterns** mais importantes do Java.
- A implementação de um *Listener* segue esse padrão.
- Um Observer é formado por dois objetos interagindo:
 - Um (ou mais) Observados, que são objetos que têm seus estados monitorados.
 - Um (ou mais) Observadores, que são objetos que ficam atentos para qualquer mudança de estado dos Observados, e podem realizar ações quanto algum estado interessante é alterado.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

12



Listeners

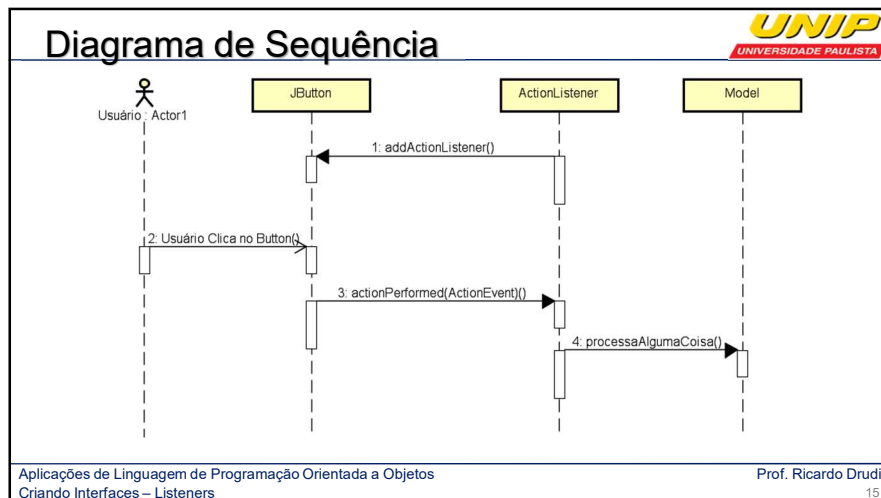
- Em um frame Swing, os **observados** são os componentes com os quais o usuário pode interagir e que provocam ações a cada interação.
- Os **observadores** são os listeners associados ao componentes Swing do frame.
- Exemplos: um JButton pode ter um ActionListener associado a ele, que será executado sempre que o botão for clicado.
- Eventos distintos (clique no botão, movimento do *mouse*) são capturados por Listeners apropriados para cada situação.

UNIP
UNIVERSIDADE PAULISTA

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

14



Criando Listeners

- A criação de Listener segue 3 etapas:
 - Declaração de uma classe que implementa a *interface* do listener apropriado.
 - Associação (registro) de uma instância da classe listener com o objeto que terá seu comportamento observado
 - A implementação do método que responde à mudança de estado do objeto observado.

UNIP
UNIVERSIDADE PAULISTA

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

16

Exemplo 1 - ActionListener



```
public class BotaoListener implements ActionListener {
    public void actionPerformed (ActionEvent ev) {
        System.out.println("O botão foi clicado!");
    }
}

BotaoListener bl = new BotaoListener();
botao.addActionListener(bl);
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi
17

Exemplo 2 – Anonymous Inner Class



```
btAnonymous.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ev) {
        System.out.println("O botão foi clicado!");
    }
});
```

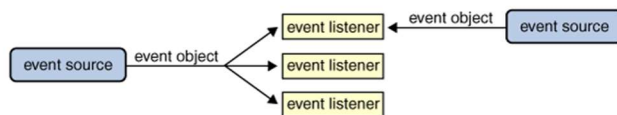
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi
18

Múltiplos Eventos



- Múltiplos *listeners* podem ser adicionado a um único componente
- Vários componentes podem ser associados ao mesmo *listener*



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi
19

Listeners Gerais



- Há duas classes de listeners disponíveis no Swing:
 - Gerais:** herdados do JComponent, disponíveis para todos os componentes
 - Específicos:** disponíveis somente para alguns componentes em particular

Listener	Eventos Monitorados
Component	Alteração no tamanho, posição ou visibilidade do componente.
Focus	Recebimento ou perda do foco da aplicação.
Key	Pressionamento da tecla monitorada – somente se o objeto tiver o foco.
Mouse	Clique, duplo-clique, pressionamento/liberação de botão.
Mouse-Motion	Alterações na posição (x,y) do ponteiro do mouse sobre o componente.
Mouse-Wheel	Movimentação da “rodinha” do mouse.
Hierarchy	Alterações no contêiner onde o componente está alocado.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi
20

Listeners Específicos

UNIP UNIVERSIDADE PAULISTA

- Variam conforme o tipo de interação que o componente tem com o usuário:
 - Button: ActionListener, ChangeListener, ItemListener
 - TextField: ActionListener, CaretListener, DocumentListener

Component	Action Listener	Caret Listener	Change Listener	Document Listener, Undoable Edit Listener	Item Listener
button	✓		✓		✓
check box	✓		✓		✓
color chooser			✓		
combo box	✓				✓

A lista completa pode ser obtida em:
<https://docs.oracle.com/javase/tutorial/uiswing/events/eventsandcomponents.html>

Aplicações de Linguagem de Programação Orientada a Objetos
 Criando Interfaces – Listeners

Prof. Ricardo Drudi
21

Mouse Listeners

UNIP UNIVERSIDADE PAULISTA

Aplicações de Linguagem de Programação Orientada a Objetos
 Criando Interfaces – Listeners

Prof. Ricardo Drudi
22

Eventos de Mouse

UNIP UNIVERSIDADE PAULISTA

- **Mouse Listener**: Ocorrem quando o usuário usa o *mouse* (ou outro dispositivo similar) para interagir com o componente.
- **Mouse Motion Listener**: Ocorrem quando há movimentação do ponteiro do *mouse*. Muito mais custoso do que o **Mouse Listener**.
- **Mouse Wheel Listener**: Ocorrem quando há movimentação na rodinha do *mouse*.
- **MouseInputAdapter**: classe que implementa as interfaces **Mouse Listener** e **Mouse Motion Listener**.
- **MouseAdapter**: classe que implementa as 3 interfaces de controle do *mouse*.

Aplicações de Linguagem de Programação Orientada a Objetos
 Criando Interfaces – Listeners

Prof. Ricardo Drudi
23

Eventos de Mouse

UNIP UNIVERSIDADE PAULISTA

Aplicações de Linguagem de Programação Orientada a Objetos
 Criando Interfaces – Listeners

Prof. Ricardo Drudi
24

Exemplo 3 - MouseListener



```
public class CapturaMouse implements MouseListener {
    public void mousePressed(MouseEvent e) {
        System.out.println("O botão do mouse foi pressionado!");
    }
    public void mouseReleased(MouseEvent e) {
        System.out.println("O botão do mouse foi solto!");
    }
    public void mouseEntered(MouseEvent e) {
        System.out.println("O ponteiro do mouse entrou na área do componente!");
    }
    public void mouseExited(MouseEvent e) {
        System.out.println("O ponteiro do mouse saiu da área do componente!");
    }
    public void mouseClicked(MouseEvent e) {
        System.out.println("O botão do mouse foi clicado!");
    }
}
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi
25

Exemplo 4 - MouseAdapter



```
public class CapturaMouseAdapter extends MouseAdapter {

    @Override
    public void mouseMoved(MouseEvent e) {
        System.out.println("O evento mouseMoved nem sempre funciona!");
    }
    @Override
    public void mouseClicked(MouseEvent e) {
        System.out.println("O mouse está na posição ("
            + e.getX() + ";" + e.getY() + ")");
    }
}
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi
26

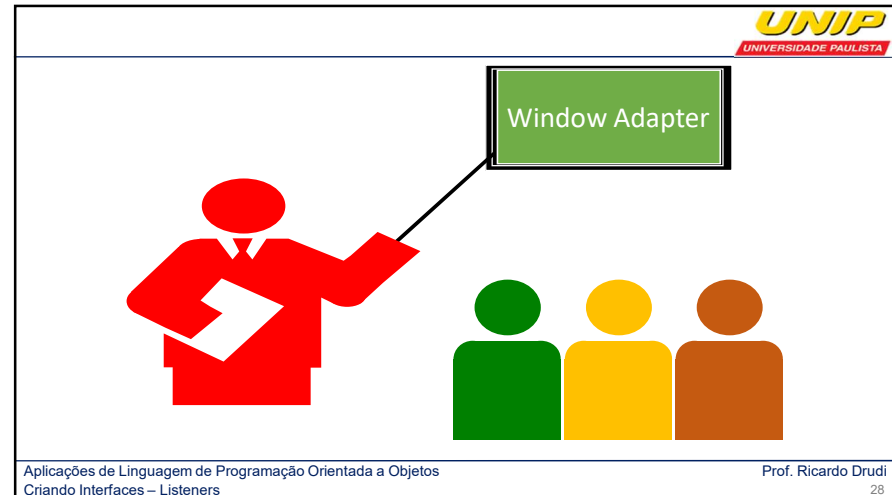
Exemplo 5 – MouseAdapter Anonymous



```
this.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        System.out.println("O mouse está na posição ("
            + e.getX() + ";" + e.getY() + ")");
    }
});
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi
27



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi
28

WindowAdapter

- WindowAdapter** é uma classe abstrata que implementa as *interfaces* `WindowFocusListener`, `WindowListener`, `WindowStateListener`, que monitoram eventos da *view*.

Método	Eventos Monitorados
<code>windowActivated</code>	Invocado quando uma janela é ativada.
<code>windowClosed</code>	Invocado quando uma janela foi fechada.
<code>windowClosing</code>	Invocado durante o processo de fechamento de uma janela.
<code>windowGainedFocus</code>	Invocado quando uma janela (ou algum componente) recebe o foco do teclado.
<code>windowLostFocus</code>	Invocado quando a janela perde o foco do teclado.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

Exemplo 6 – windowClosing

```

setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
...
this.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        int resp = JOptionPane.showConfirmDialog(null,
            "Tem certeza que deseja sair?",
            "Confirmação",
            JOptionPane.OK_CANCEL_OPTION);
        if (resp == JOptionPane.OK_OPTION) System.exit(0);
    }
});

```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

FocusListener

- Monitora os eventos de receber/perder o foco do teclado para cada componente do Swing.
- A classe `FocusAdapter` implementa a *interface* `FocusListener` com métodos vazios, que podem ser sobrescritos (`Override`).
- Se o comportamento desejado for o mesmo para todos os componentes, é possível criar apenas um `listener` e associá-lo a vários componentes.

Método	Eventos Monitorados
<code>focusGained</code>	Invocado quando o componente recebe o foco.
<code>focusLost</code>	Invocado quando o componente perde o foco.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners


Prof. Ricardo Drudi

Exercício 1 - Cores

Crie uma aplicação conforme a tela abaixo, onde a cor da fonte da mensagem **Cor Seleccionada** é modificada conforme o mouse passe por cima dos painéis coloridos abaixo.

Se o usuário clicar no **X** para sair, confirme antes de efetivamente fechar a aplicação.

Você pode incrementar sua aplicação utilizando tags HTML, se quiser.



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners


Prof. Ricardo Drudi

54

Exercício 2 – Focus + Formatter

Crie uma aplicação onde o usuário digite seu nome, telefone fixo, telefone celular e o CPF. Inclua as seguintes funcionalidades:

- 1) O campo que estiver sendo digitado no momento deve ficar amarelo;
- 2) Os demais campos devem ser brancos;
- 3) Os campos telefone, celular e CPF devem ser formatados;
- 4) Quando o usuário clicar no botão Imprime, os dados digitados devem ser exibidos na console;
- 5) Utilize FocusListener e MaskFormatter.




Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

55

Obrigado.

When your teacher is talking about Java and you remember Minecraft was made with Java



Copyright © 2018, Prof. Ricardo Drudi
Todos direitos reservados. A reprodução ou divulgação total ou parcial deste documento é expressamente proibida sem o consentimento formal, por escrito, do professor Ricardo Drudi.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – Listeners

Prof. Ricardo Drudi

56