



Programação Orientada a Eventos

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Setembro de 2012



Programação Orientada a Eventos

- *A programação orientada a eventos é um paradigma de programação que não segue um fluxo de controle padronizado, sendo que seus fluxos de controle são guiados por sinais externos. Portanto sua aplicação está diretamente ligada com o desenvolvimento de interfaces voltada para o usuário. (Silveira et al. [2007?])*



Evento

- Normalmente um usuário interage com uma GUI do aplicativo para indicar a tarefa que o *software* deve realizar;
- Exemplo:
 - Escrever uma mensagem num aplicativo de correio eletrônico. O ato de clicar no botão “Enviar” instrui o *software* a enviar a mensagem escrita para os endereços especificados no *e-mail*.



GUI's X Eventos

- GUI's são fundamentadas em eventos;
- Quando um usuário interage com um componente GUI, essa interação é conhecida como evento;
- Um evento geralmente guia o programa na realização de uma tarefa.



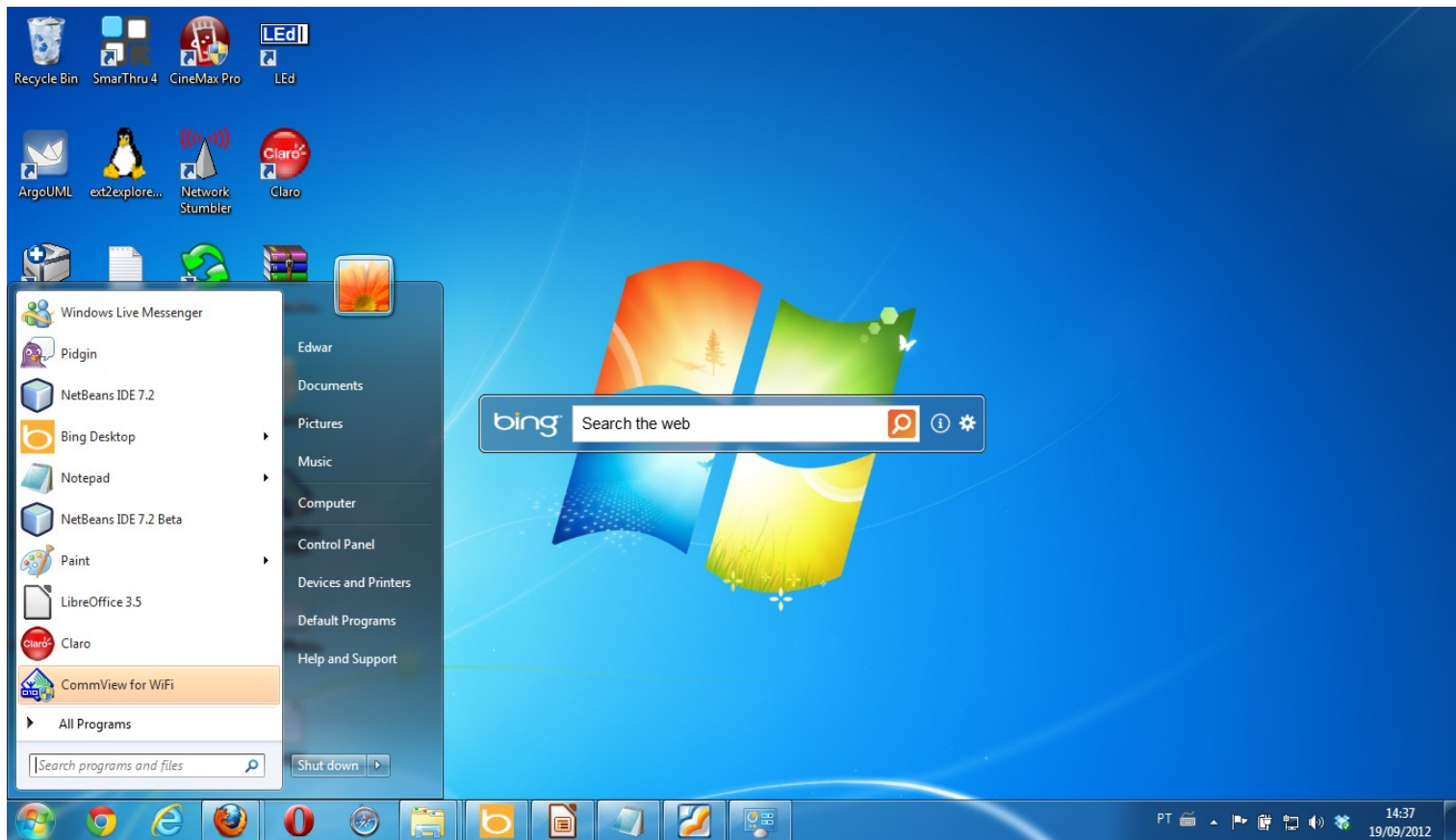
Eventos Comuns

- Eventos comuns nos componentes GUI:
 - clicar em um botão,
 - digitar em um campo de texto,
 - selecionar um item de *menu*,
 - fechar uma janela e
 - mover um *mouse*.
- O código que realiza uma tarefa em resposta a um evento é chamado de: *handler* de evento.



Exemplo de Evento

- Ao clicar no botão iniciar do Windows ele nos mostra uma outra tela composta por programas e serviços do sistema operacional.





Criar e Associar um Evento

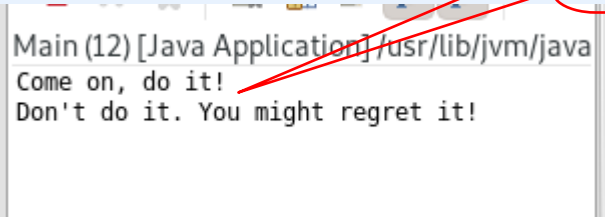
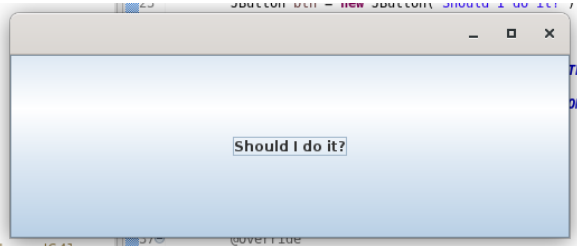
```
1 package principal;
2
3 import java.awt.BorderLayout;
4 import java.awt.event.ActionEvent;
5 import java.awt.event.ActionListener;
6
7 import javax.swing.JButton;
8 import javax.swing.JFrame;
9
10 public class Main {
11     JFrame frame;
12
13
14     public static void main(String[] args) {
15         Main main = new Main();
16
17         main.go();
18     }
19
20     public void go() {
21         frame = new JFrame();
22
23         JButton btn = new JButton("Should I do it?");
24         btn.addActionListener(new AngelListener());
25         btn.addActionListener(new DevilListener());
26     }
27 }
```

Classe Interna

Classe Interna

```
27     frame.getContentPane().add(BorderLayout.CENTER, btn);
28
29     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30
31     frame.setVisible(true);
32     frame.setSize(500, 200);
33 }
34
35 class AngelListener implements ActionListener{
36
37     @Override
38     public void actionPerformed(ActionEvent arg0) {
39         System.out.println("Don't do it. You might regret it!");
40     }
41 }
42
43 class DevilListener implements ActionListener{
44
45     @Override
46     public void actionPerformed(ActionEvent arg0) {
47         System.out.println("Come on, do it!");
48     }
49 }
50
51 }
52 }
```

Após apertar o botão.





Bibliografia

- DEITEL, H. M.; DEITEL, P. J. **Java Como Programar**; 6. ed., São Paulo: Pearson, 2005.
- SILVEIRA, M. P.; RAFAEL; VARGAS, P. T. **Programação Orientada a Eventos**. Disponível em: http://www.dcc.unimontes.br/renato/2009/sdi/materiais/2007_1_paradigmas_orientado_eventos.pdf Acesso em: 19 set. 2012.