Programação Orientada a Eventos

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

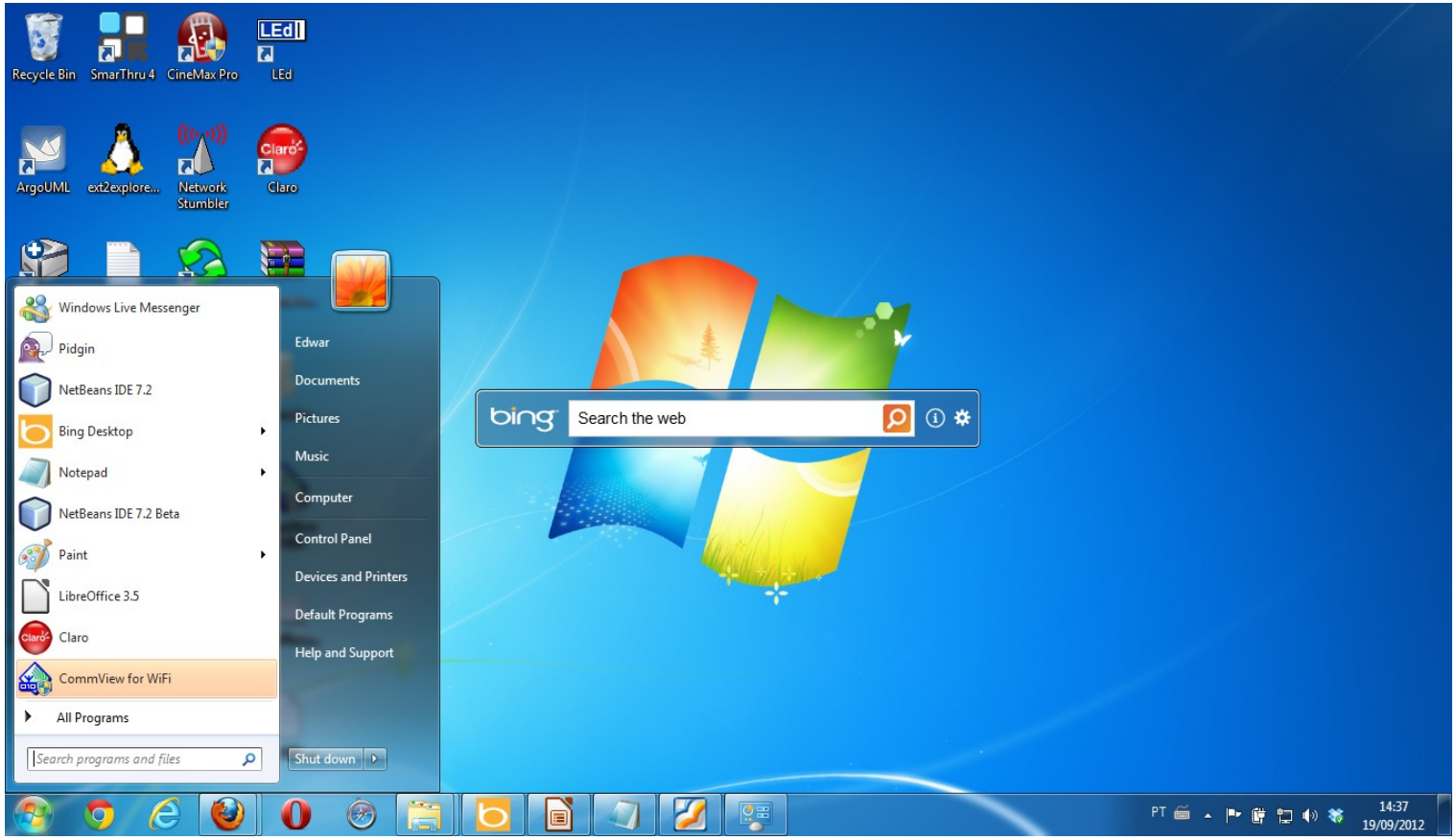
- GUI's são fundamentadas em eventos;
- Quando um usuário interage com um componente GUI, essa interação é conhecida como evento;
- Um evento geralmente guia o programa na realização de uma tarefa.

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.

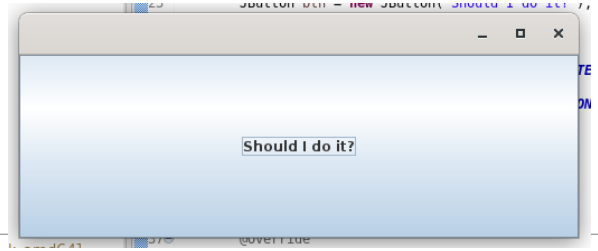


Criar e Associar um Evento

```java
package principal;

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

import javax.swing.JButton;
import javax.swing.JFrame;

public class Main {

    JFrame frame;

    public static void main(String[] args) {
        Main main = new Main();

        main.go();
    }

    public void go() {
        frame = new JFrame();

        JButton btn = new JButton("Should I do it?");
        btn.addActionListener(new AngelListener());
        btn.addActionListener(new DevilListener());
        frame.getContentPane().add(BorderLayout.CENTER, btn);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setVisible(true);
        frame.setSize(500, 200);
    }

    class AngelListener implements ActionListener{

        @Override
        public void actionPerformed(ActionEvent arg0) {
            System.out.println("Don't do it. You might regret it!");
        }

    }

    class DevilListener implements ActionListener{

        @Override
        public void actionPerformed(ActionEvent arg0) {
            System.out.println("Come on, do it!");
        }

    }
}
```

Classe Interna

Classe Interna




```
Main (12) [Java Application] /usr/lib/jvm/java
Come on, do it!
Don't do it. You might regret it!
```

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.