Programação por eventos

Um *evento* ocorre quando o utilizador interage com um objecto gráfico:

. manipular um botão com o rato;

. introduzir texto num campo de texto

. seleccionar um item de menu

. ...

Num modelo de programação por eventos, programam-se objectos ("*event listeners*") que reagem a alterações no estado de outros objectos ("*event sources*"). Um "event listener" inclui um método que será executado em resposta ao (aos) evento(s) gerado(s).

Para *tratar um evento* é necessário associar (registar) um ou vários "listeners" ao objecto que gera o evento.

Quando um evento é gerado o sistema de execução notifica o objecto de escuta ("listener") correspondente, invocando o método que trata o evento.

Caso não exista nenhum "listener" registado no objecto que gera o evento, este não terá qualquer efeito.

Para cada tipo de evento temos um "listener" correspondente.

Um objecto pode ser registado como um "listener" se for uma instância de uma classe que implementa uma determinada interface.

```
interface ActionListener
interface MouseListener
interface KeyListener
 ...
```

Exemplo:

```
public interface ActionListener {
        public void actionPerformed(ActionEvent e);
}

import java.awt.event.*;
public classe TrataBotao implements ActionListener {
    public void actionPerformed ( ActionEvent e ){
                // ... Tratar evento
     }
}
```

Para registar um objecto "ActionListener" no objecto que gera o evento, usamos o método

void addActionListener (ActionListener)

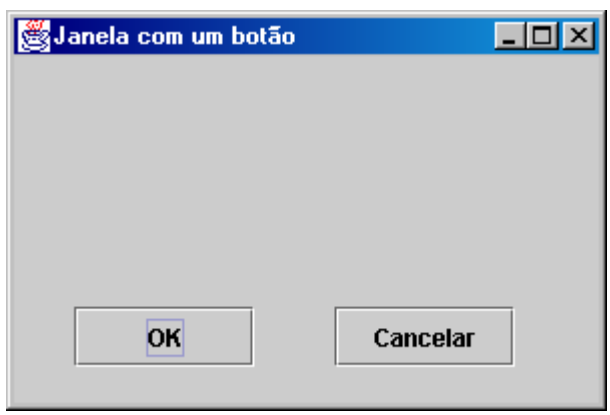
```
b1 = new JButton ( "OK“ );

TrataBotao  tb= new TrataBotao( );

b1.addActionListener ( tb );
```

Obs.

Um único "listener" pode ser associado a múltiplos objectos geradores de eventos.

Vários "lisneters" podem ser associadas a um único evento

Suponhamos a seguinte janela com dois botões:



Tratar eventos

Queremos que o título da janela mude consoante o botão que for pressionado.

O método que vai tratar o evento terá a seguinte estrutura:

```
public void actionPerformed ( ActionEvent e ){

    (1) String textoDoBotao =
            aceder ao texto do objecto que gerou o evento;

    (2) JFrame janela =
            aceder à janela que contém o objecto gerador do evento

   janela.setTitle ("Pressionou o botão " +  textoDoBotao );
}
```

(1) Pode obter-se de duas formas:

a) pelo método "String getActionCommand()", da classe ActionEvent

```
String textoDoBotao = e.getActionCommand ();
```

b) através do método "Object getSource()", da classe ActionEvent

```
JButton botaoClicado = (JButton) e.getSource();

String textoDoBotao = botaoClicado.getText();
```

(2) Para obtermos a janela que contém o gerador do evento, é necessário:

. Obter o painel que contém o objecto gerador do evento

. Obter a janela que contém o painel.

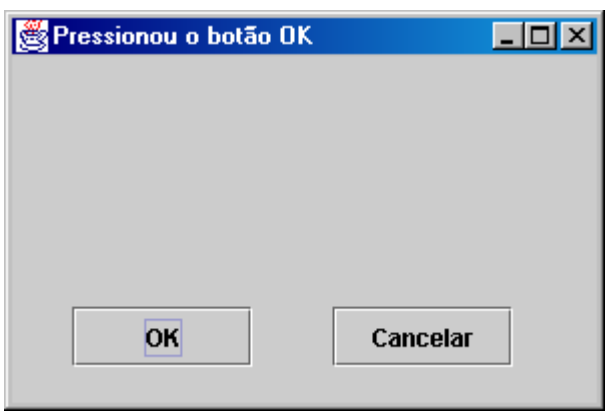
```
JRootPane rootPane =  botaoClicado.getRootPane();

JFrame frame = (JFrame)  rootPane.getParent();
```

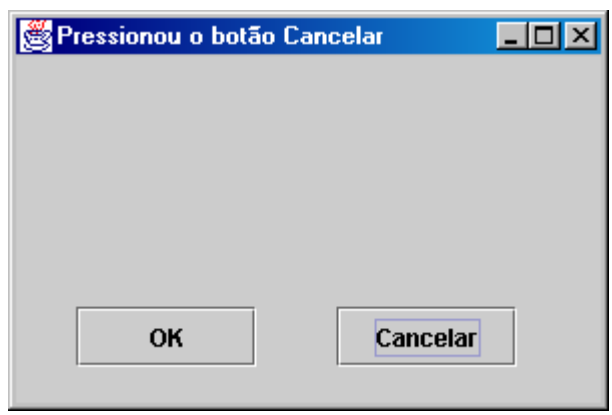
Uma janela pode conter vários painéis encadeados. O painel de topo é o "root pane"

A própria janela pode ser o "listener" dos objectos gráficos que contém:

```
public class Janela2 extends JFrame implements
    ActionListener {
  private JButton b1, b2;
  ...
  public Janela2(){
     ...
    Container contentor = getContentPane();
    b1 = new JButton("OK");
    b2 = new JButton("Cancelar");
    b1.setBounds(30,125,90,30);
    b2.setBounds(160,125,90,30);

    //registar o listener em cada botão
    b1.addActionListener(this);
    b2.addActionListener(this);

    contentor.add(b1);
    contentor.add(b2);
  }
```

```
  public void  actionPerformed(ActionEvent e){
        String textoDoBotao;
                        textoDoBotao = e.getActionCommand();

        setTitle("Pressionou o botão " + textoDoBotao);
  }
public static void main(String[] args)  {
  Janela2 j = new Janela2();
  j.setVisible(true);
 }
} // fim da classe Janela2
```





A classe JLabel

. Permite mostrar texto não editável

```
public class Janela3 extends JFrame {
   private JLabel l;
   ...
 public Janela3(){
    ...
 l = new JLabel("Vingança !!!!");
 l.setBounds(205,100,70,20);
contentor.add(l);

 li = new JLabel (new ImageIcon ("d://_POO/rato.jpg"));
 li.setBounds(0,0,200,200);
 contentor.add(li);
```