Graphical user Interfaces

Objetivos:

. Construir programas com interfaces gráficas

"Graphical User Interface (GUI) application programs"

- Utilizar classes do package **javax.swing**

- Usar um modelo de programação por eventos

("Java's delegation-based event model")

Os componentes gráficos da linguagem Java estão definidos nos packages:

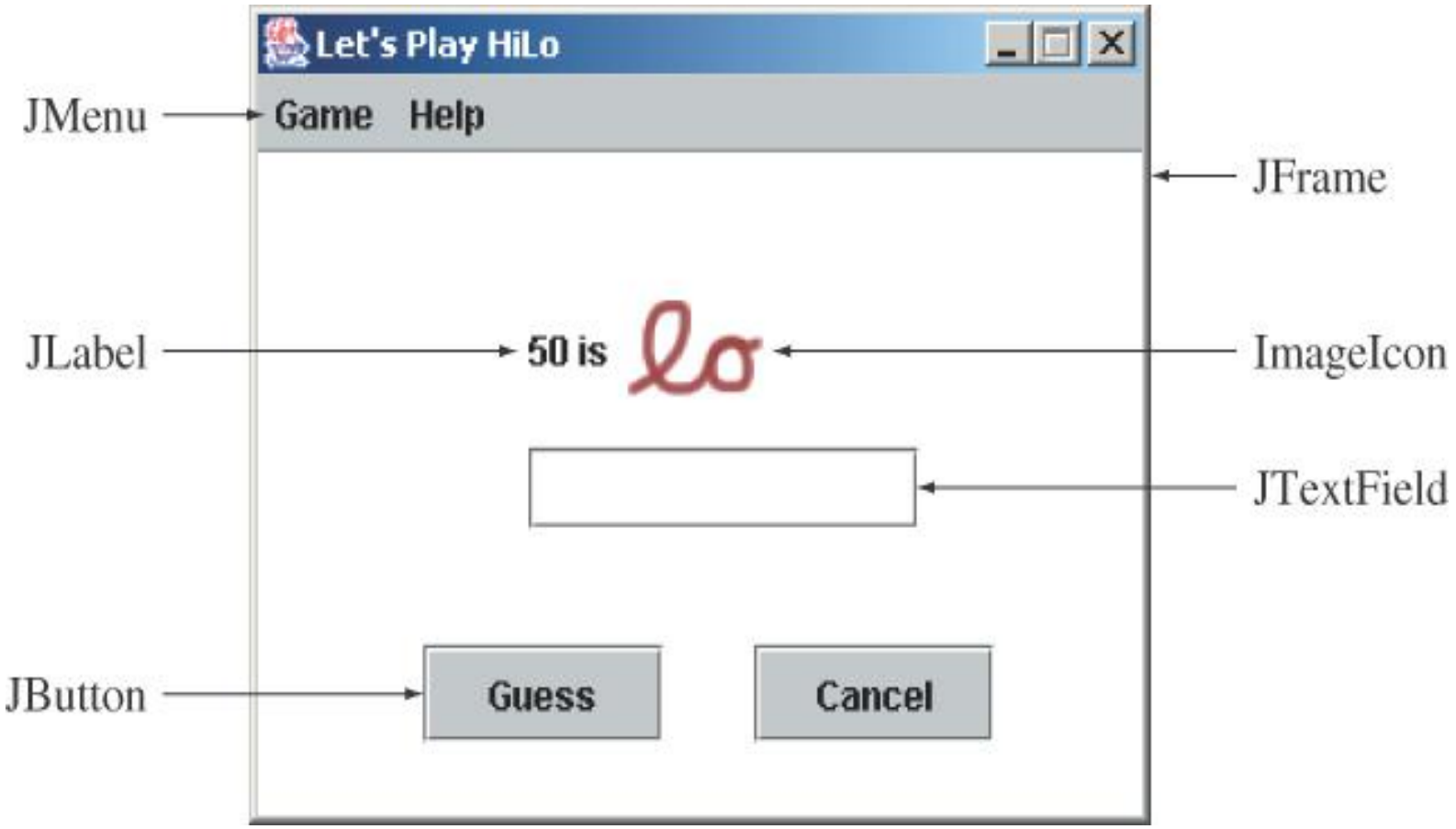
java.awt

javax.swing (surge com a versão Java2 SDK1.2)

Nova plataforma:

javafx

Alguns componentes gráficos do package javax.swing



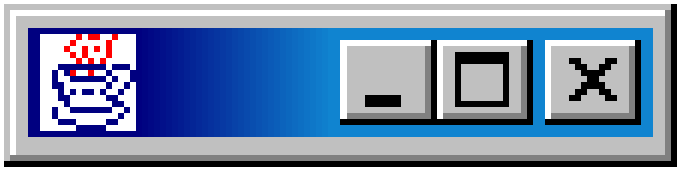
Definição de uma janela (JFrame)

- Um objeto do tipo JFrame contém os elementos básicos para manipularmos uma janela:

abrir, fechar, mover e redimensionar

- A uma janela poderemos adicionar outros componentes gráficos

```
import javax.swing.*;
public class Janela {
  public static void main (String [] args){
     JFrame janela = new JFrame();
     janela.setVisible(true);
  } }
```



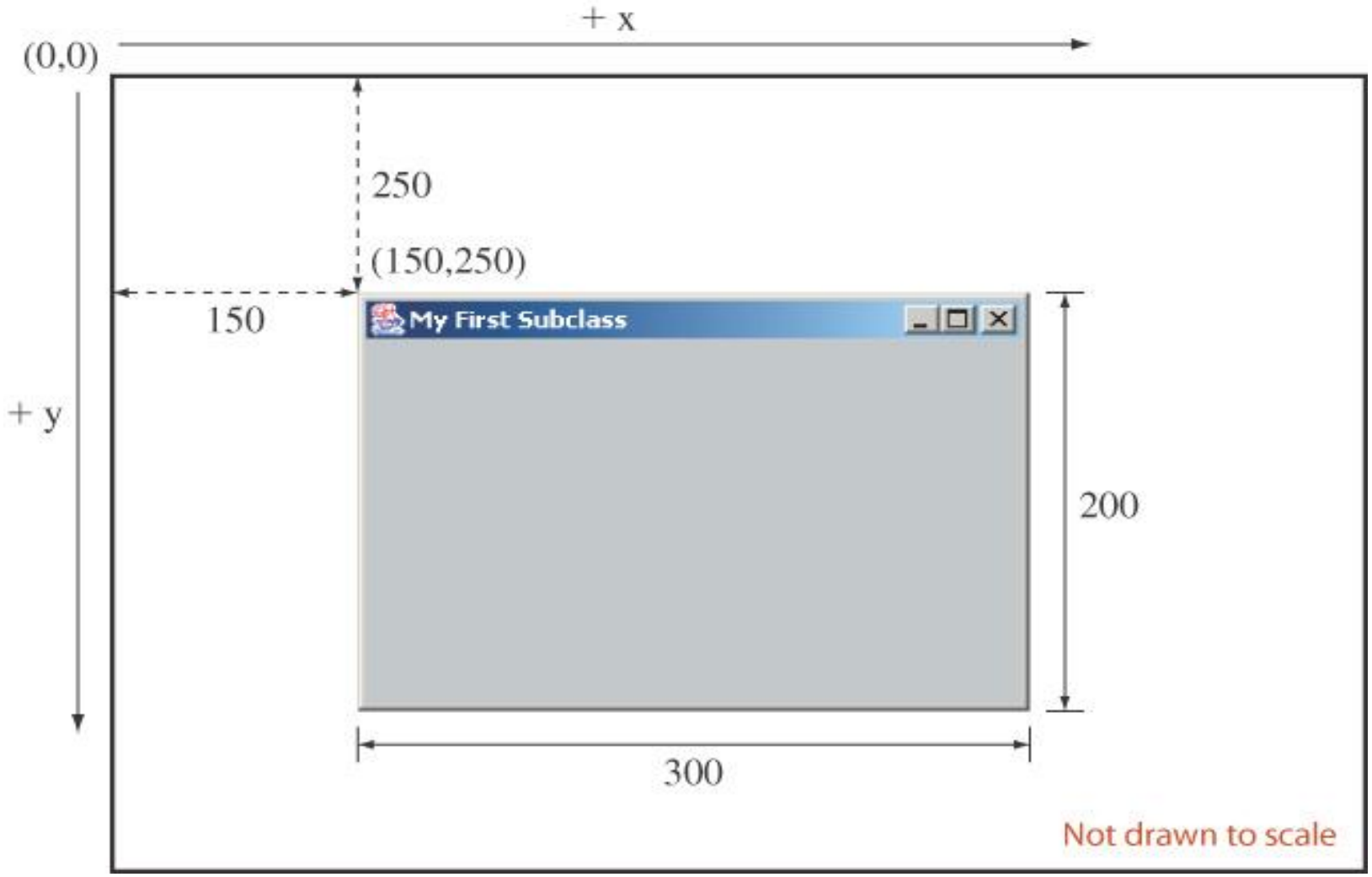
Alguns métodos da classe JFrame:

- atribuir um título
setTitle (“My First Subclass");

-dimensionar a janela
setSize (300, 200);
(300 pixels de largura e 200 pixels de altura)

- posicionar a janela no ponto de coordenadas (150, 250)
setLocation (150, 250);

- requerer que o programa termine quando a janela é fechada
setDefaultCloseOperation (EXIT_ON_CLOSE);


+ x
(0,0)
250
(150,250)
150
My First Subclass
+ y
200
300
Not drawn to scale

Para construir uma interface gráfica podemos definir uma subclasse de JFrame.

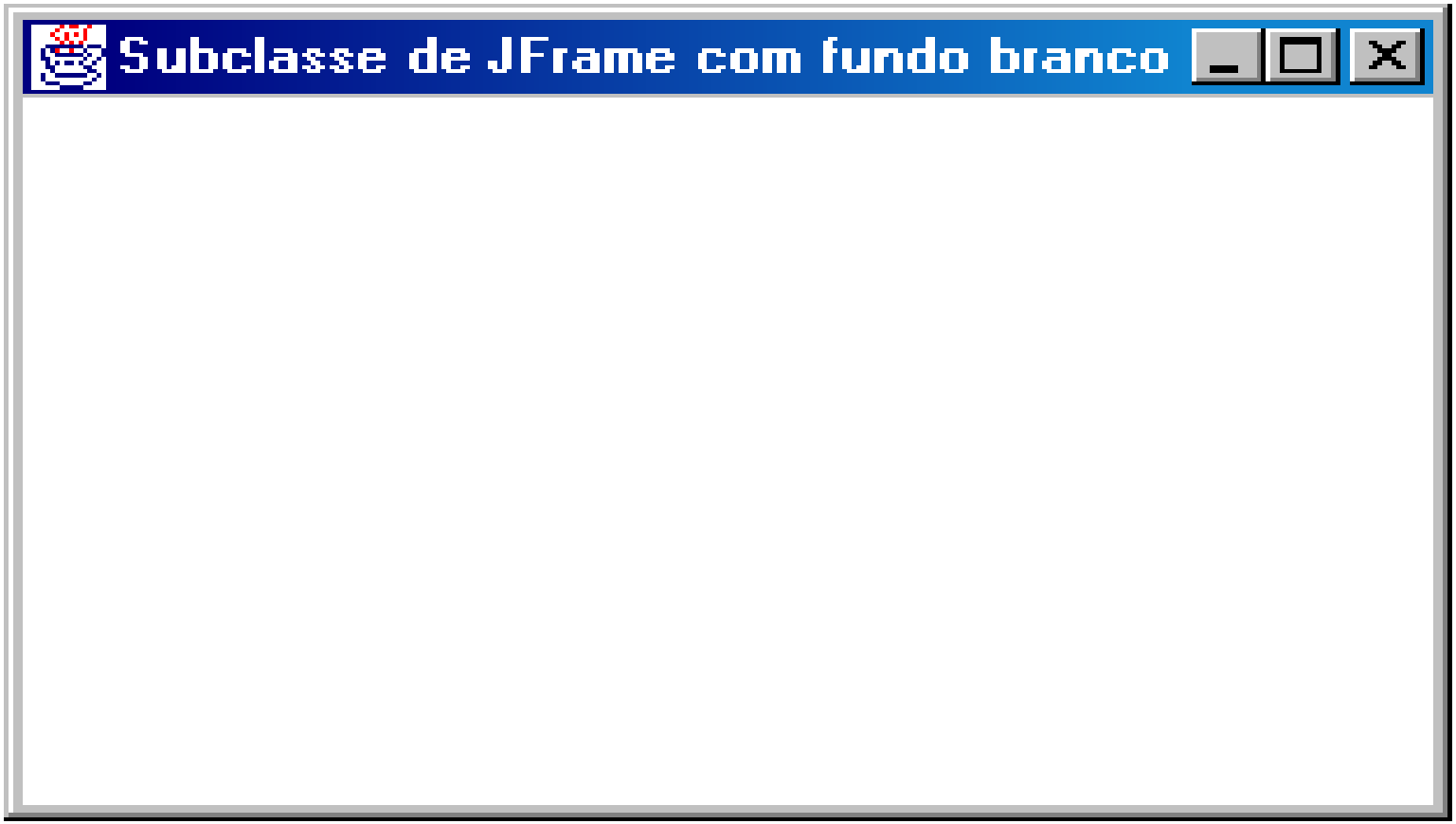
Nessa subclasse iremos adicionar o comportamento necessário à interface gráfica que pretendemos.

A classe Janela1 define uma janela com as características anteriores e com o fundo branco:

```
import javax.swing.*;
import java.awt.*;
public class Janela1  extends JFrame {
private static final int FRAME_WIDTH    = 300;
private static final int FRAME_HEIGHT   = 200;
private static final int FRAME_X_ORIGIN = 150;
private static final int FRAME_Y_ORIGIN = 250;
```

```
…
public Janela1(){
setTitle ( “Subclasse de JFrame com fundo branco" );
setSize  (FRAME_WIDTH, FRAME_HEIGHT );
setLocation (FRAME_X_ORIGIN, FRAME_Y_ORIGIN );
setDefaultCloseOperation( EXIT_ON_CLOSE );
alterarCorFundo( );
}
//continua …
```

```
private void alterarCorFundo() {
    Container contentor = getContentPane();
    contentor.setBackground(Color.white);
}
public static void main(String[] args) {
  Janela1 j = new Janela1();
  j.setVisible(true);
 }
}// fim da classe Janela1
```


Subclasse de JFrame com fundo branco

Painel de conteúdos "content pane"

Designa a área da janela em que se pode mostrar conteúdo como texto, imagem, etc.

(i. é, toda a área, com exceção do título, da barra de menus e do bordo da janela)

Para aceder ao painel de conteúdos de uma janela usamos o método:

Container getContentPane(), definido na classe JFrame

O método devolve um objeto do tipo Container, classe do package java.awt

Adicionar botões, objetos do tipo JButton, ao
painel de conteúdos de uma janela

Há duas formas de adicionar objetos gráficos no painel
de conteúdos de uma janela:

1 – usar um objeto do tipo **LayoutManager** para
controlar a colocação dos objetos

2 – explicitar o tamanho e a posição específica onde queremos
o objeto (**posicionamento absoluto**)

Para usarmos a segunda possibilidade devemos desativar o layout manager, da classe Container, invocando o o método:

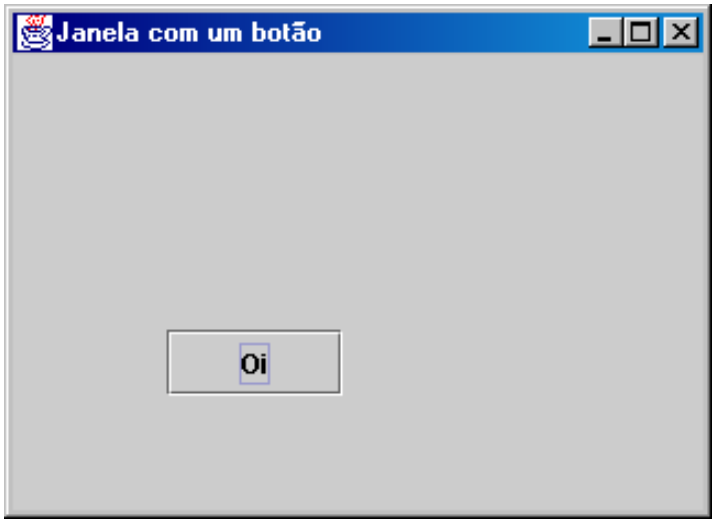
void setLayout(LayoutManager mgr) com o valor null

contentor.setLayout(null);

Para colocar o botão numa dada posição usamos o método da classe JButton:

void setBounds (x, y, width , height);

```
Container contentor = getContentPane();

JButton botao = new JButton(“Oi”);

contentor.setLayout(null);

botao.setBounds ( 70, 125, 80, 30 );

//Finalmente, vamos colocar o botão na janela:

contentor.add( botao );
```



Vamos agora ver como fazer o programa reagir
a um “click” sobre o botão ...

Programação por eventos

Um evento ocorre quando o utilizador interage com um objeto gráfico:

. manipular um botão com o rato;

. introduzir texto num campo de texto

. selecionar um item de menu

. …

Programação por eventos

...

Num modelo de programação por eventos, programam-se objetos ("event listeners") que reagem a alterações no estado de outros objetos ("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 objeto que gera o evento.

Quando um evento é gerado o sistema de execução notifica o objeto 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 objeto 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 objeto "ActionListener" no objeto que gera o evento, usamos o método:

void addActionListener (ActionListener)

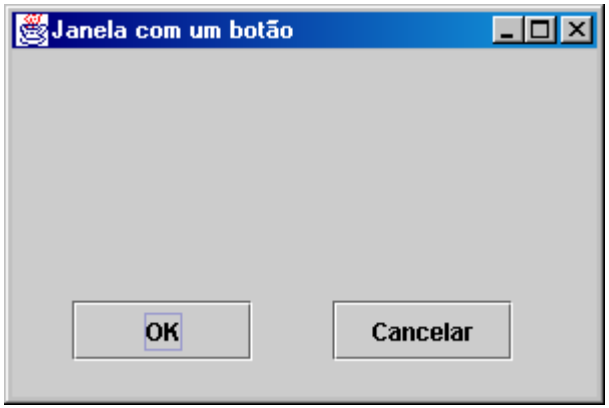
```
b1 = new JButton ( "OK“ );

TrataBotao  tb= new TrataBotao( );

b1.addActionListener ( tb );
```

Obs.

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

Vários "listeners" podem ser associados 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 objeto que gerou o evento;

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

. Obter a janela que contém o painel.

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

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

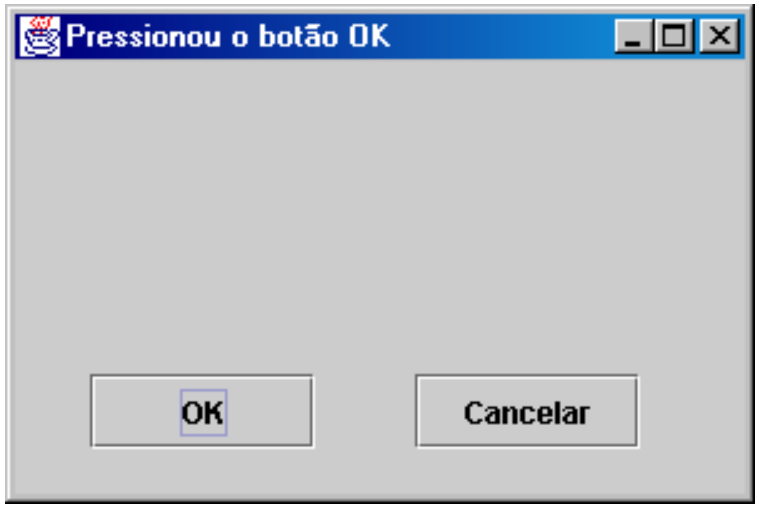
Uma janela pode conter vários painéis encadeados. O painel de topo é o “root pane”

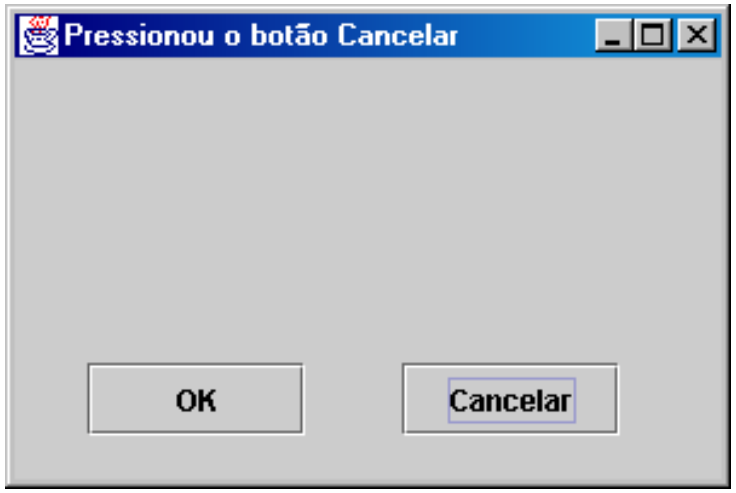
A própria janela pode ser o “listener” dos objetos gráficos que contém.

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

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

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

//Continua …
```

```
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
```


Pressionou o botão OK
OK
Cancelar


Pressionou o botão Cancelar
OK
Cancelar

Uma forma mais rápida de construir aplicações com interfaces gráficas é usar o “NetBeans IDE's GUI builder”.

Use o tutorial do link abaixo para construir um pequeno exemplo:
http://docs.oracle.com/javase/tutorial/uiswing/learn/index.html

JavaFx
https://docs.oracle.com/javase/8/javafx/get-started-tutorial/javafx_get_started.htm

http://docs.oracle.com/javase/8/javafx/get-started-tutorial/hello_world.htm