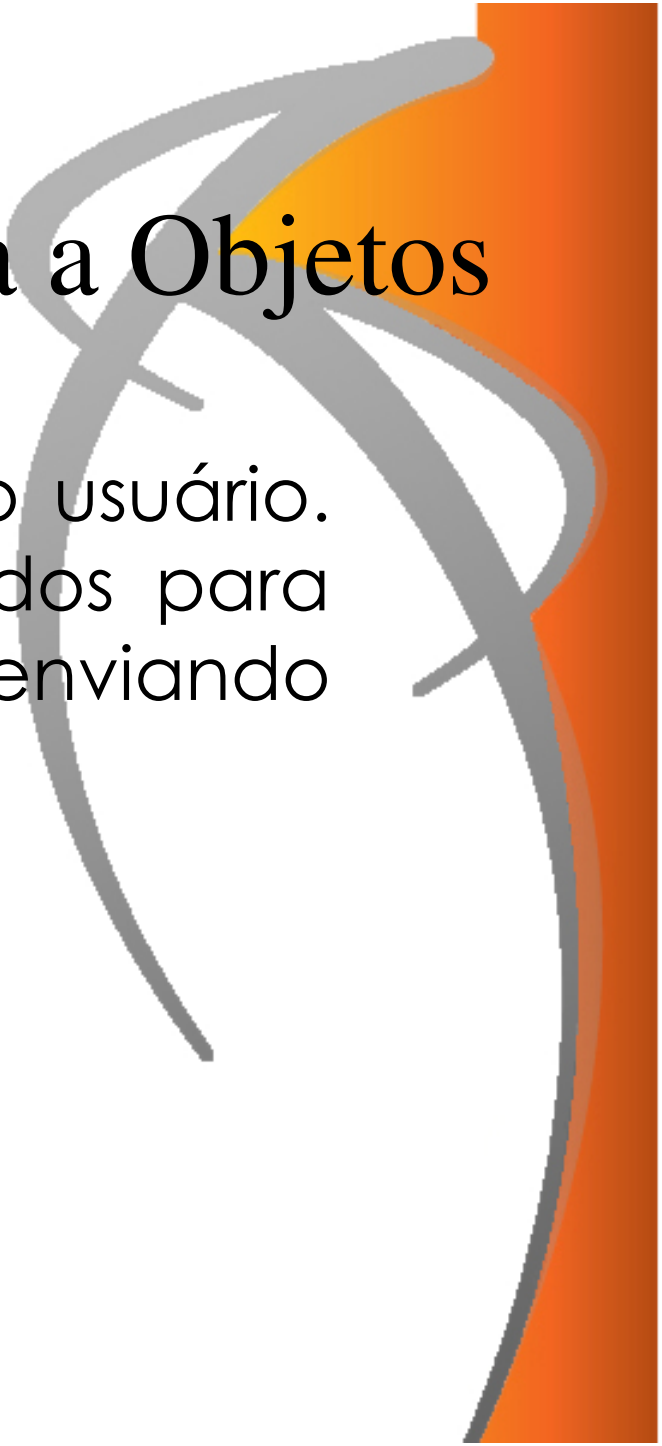
INTERFACE COM O USUÁRIO (em Java)

Programação Orientada a Objetos

educação a distância
UFSCar
virtual

Programação Orientada a Objetos

Chegamos à interface com o usuário. Você já utilizou alguns métodos para dialogar com o usuário, quer enviando mensagens quer lendo dados.



Programação Orientada a Objetos

Para ler dados utilizando uma janela de diálogo, fizemos uso do método:

JOptionPane.**showInputDialog**

que recebe uma cadeia de caracteres (mensagem para o usuário) e devolve uma string com o valor digitado pelo usuário.

Programação Orientada a Objetos

Modo de uso:

A = JOptionPane.**showInputDialog("msg");**

Onde msg seria a cadeia de caracteres de mensagem para o usuário.

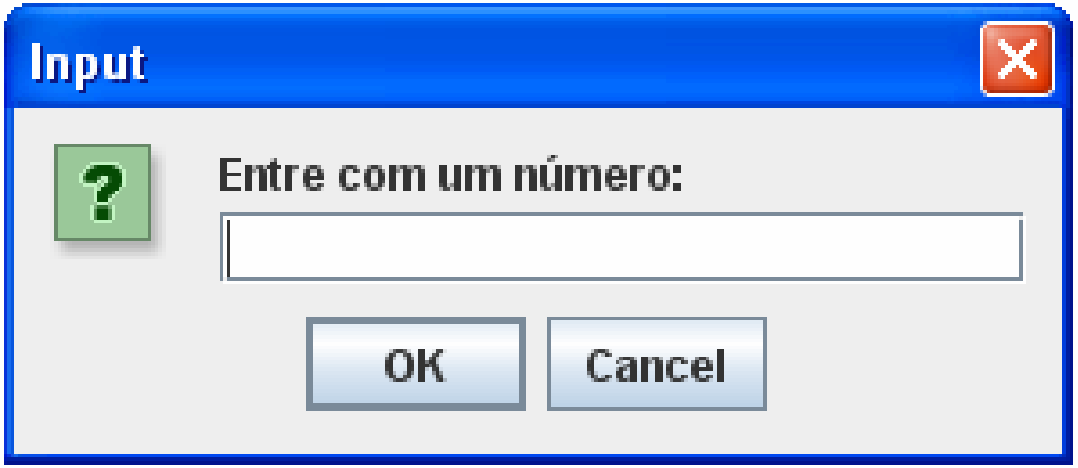
A seria uma variável do tipo String.

Programação Orientada a Objetos

Para transformar o valor armazenado em A em um número inteiro, por exemplo, devemos fazer uma conversão de tipo:

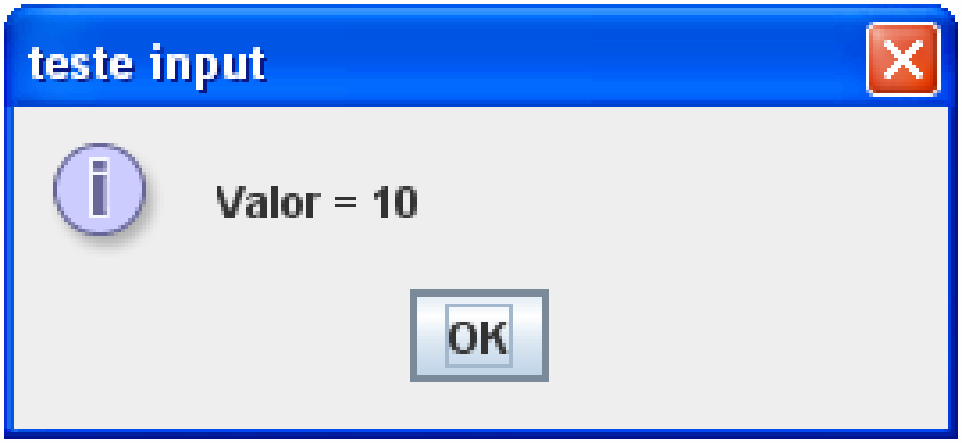
```
x = Interger.parseInt(A);
```

Programação Orientada a Objetos



Visão de uma janela de diálogo com o usuário para obtenção de um valor.

Programação Orientada a Objetos



Visão de uma janela de mensagens para o usuário.

Programação Orientada a Objetos

```
import javax.swing.JOptionPane;

class teste_input{

    public static void main(String args[])
    {
      String A;
      int x;

      A = JOptionPane.showInputDialog("Entre com um número:");

      x = Integer.parseInt(A);

      JOptionPane.showMessageDialog(null,"Valor = "+x,"teste
input",JOptionPane.INFORMATION_MESSAGE);

    }
}
```

Programação Orientada a Objetos

Saídas com mais informações podem ocorrer com a utilização de uma área de textos:

```
JTextArea saida = new JTextArea();
Container c = getContentPane();
c.add(saida);
...
msg += "outra mensagem";  // String
...
saida.setText(msg)
```

Programação Orientada a Objetos

Mas podemos incrementar a nossa interface um pouco mais, colocando botões, checkboxes, radioboxes etc...

Mas para isso precisaremos entender um pouco sobre interface gráfica com o usuário (GUI ou graphical user interface) e tratramento de eventos.

Programação Orientada a Objetos

Alguns componentes GUI básicos são:

- JLabel — Texto não editável
- JTextField — Leitura de dados
- JButton — Botão
- JCheckBox
- JComboBox
- JList — Lista de itens
- JPanel — Conteiner para os componentes

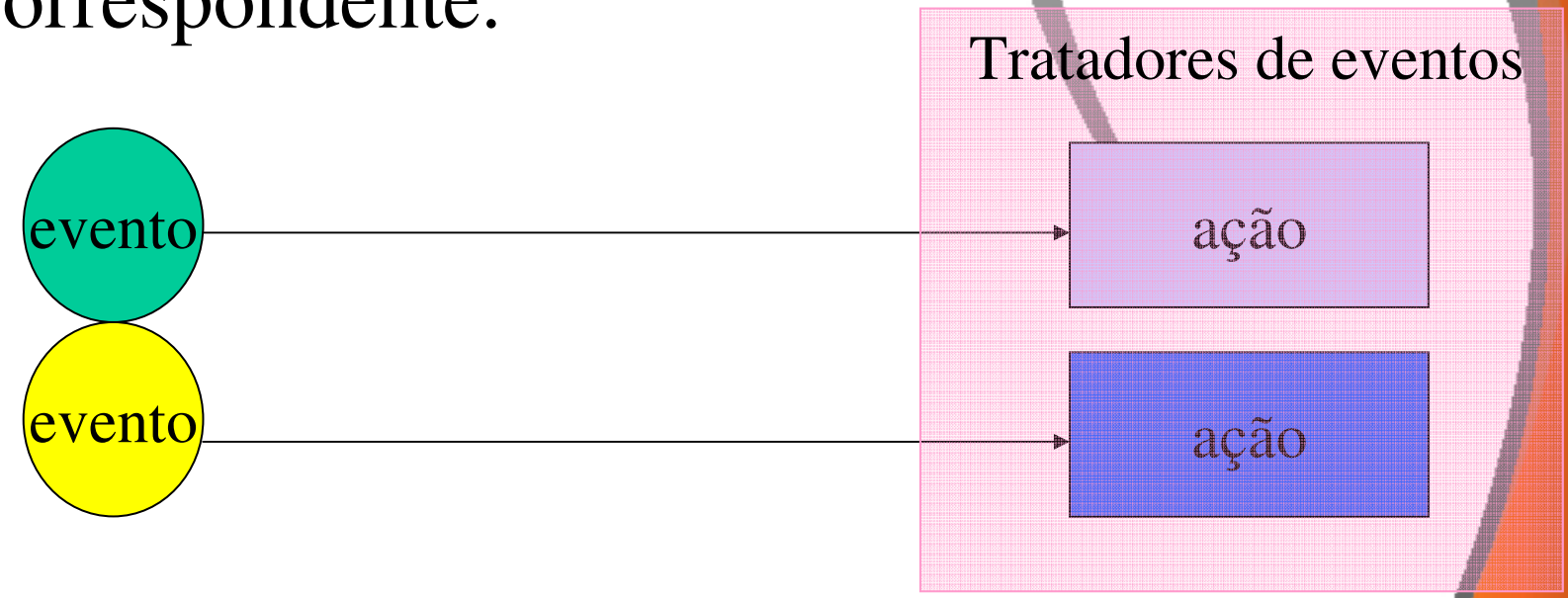
Programação Orientada a Objetos

Uma interface gráfica normalmente está associada com eventos. Um evento pode ser descrito como sendo uma ação que o usuário realiza (movimentar o mouse, clicar, pressionar uma tecla, etc).

É, aparentemente, óbvio que precisaremos tratar eventos quando estivermos trabalhando com interfaces gráficas.

Programação Orientada a Objetos

Normalmente teremos que fazer um mapeamento entre o evento gerado (de um determinado componente) e a ação correspondente.



Programação Orientada a Objetos

java.util.EventListener

- ActionListener
- ComponentListener
- ItemListener
- MouseListener
- MouseMotionListener
- WindowListener
- ...

Programação Orientada a Objetos

Vejamos um exemplo...

Programação Orientada a Objetos

Vamos supor que desejássemos incluir mais algumas informações:

- Sexo → radiobox
- Pai e/ou mãe vivos → checkbox

Programação Orientada a Objetos

Como temos sexo masculino e feminino, podemos utilizar um radiobox da seguinte forma:

```
private JRadioButton masc, fem;
private ButtonGroup sexo;
```

Programação Orientada a Objetos

Criamos os botões individuais e os adicionamos ao container:

```
masc = new JRadioButton("masculino",true);
container.add(masc);
fem =  new JRadioButton("feminino",false);
container.add(fem);
```

Programação Orientada a Objetos

E fazemos o relacionamento lógico:

```
sexo = new ButtonGroup();
sexo.add(masc);
sexo.add(fem);
```

Programação Orientada a Objetos

Registramos o tratador de eventos:

```
masc.addActionListener( tratador);
fem.addActionListener( tratador);
```

Programação Orientada a Objetos

E adicionamos o código no tratador de eventos para informar a escolha ao usuário:

```
if (evento.getSource() == masc)
        string = "masculino";
if (evento.getSource() == fem)
        string = "feminino";
```

Programação Orientada a Objetos

Algo parecido é feito com os botões checkbox. A diferença, do ponto de vista lógico, é que podemos ter mais que um checkbox ativado.
Também precisamos ter atenção para o fato que um checkbox selecionado uma vez indica ativação. Selecionado uma segunda vez, indica desativação.

Programação Orientada a Objetos

Declaração:

```
private JCheckBox pai, mae;
```

Programação Orientada a Objetos

```
pai = new JCheckBox("pai ");
mae = new JCheckBox("mãe ");
container.add(pai);
container.add(mae);
```

Programação Orientada a Objetos

```
CHECKBOX t_chckbox = new CHECKBOX();
pai.addItemListener( t_chckbox);
mae.addItemListener( t_checkbox);
```

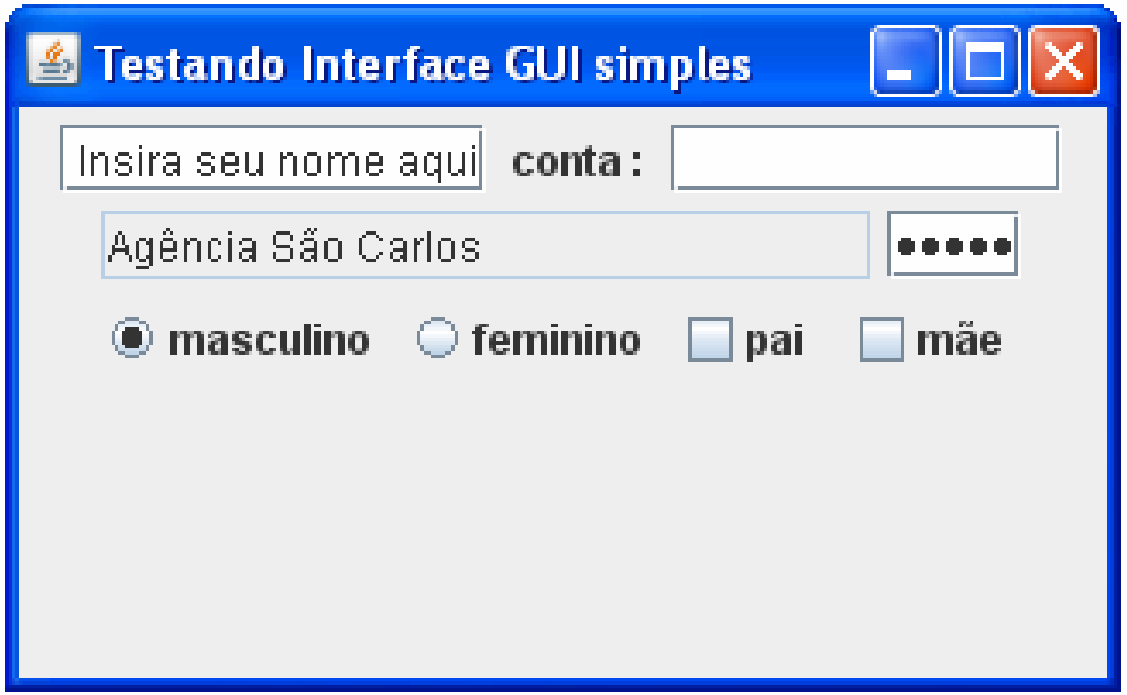
Programação Orientada a Objetos

Método de tratamento de evento checkbox:

```
private class CHECKBOX implements ItemListener
{
    public void itemStateChanged( ItemEvent evento)
    {
        String string = " ";

        if (evento.getSource() == pai)
            if (evento.getStateChange() == ItemEvent.SELECTED)
                    string = "pai vivo";
            else
                    string = "pai não está mais vivo";
     ...
    }
}
```

Programação Orientada a Objetos



Programação Orientada a Objetos

O processo é muito semelhante para utilização de:

- botões;
- listas de seleção simples;
- listas de seleção múltipla

Gostou?

Programação Orientada a Objetos

FIM