

**FCT/Unesp – Presidente Prudente**  
**Departamento de Matemática e Computação**

**Programação Orientada  
a Objetos**  
**Interface Gráfica**  
**Parte III**

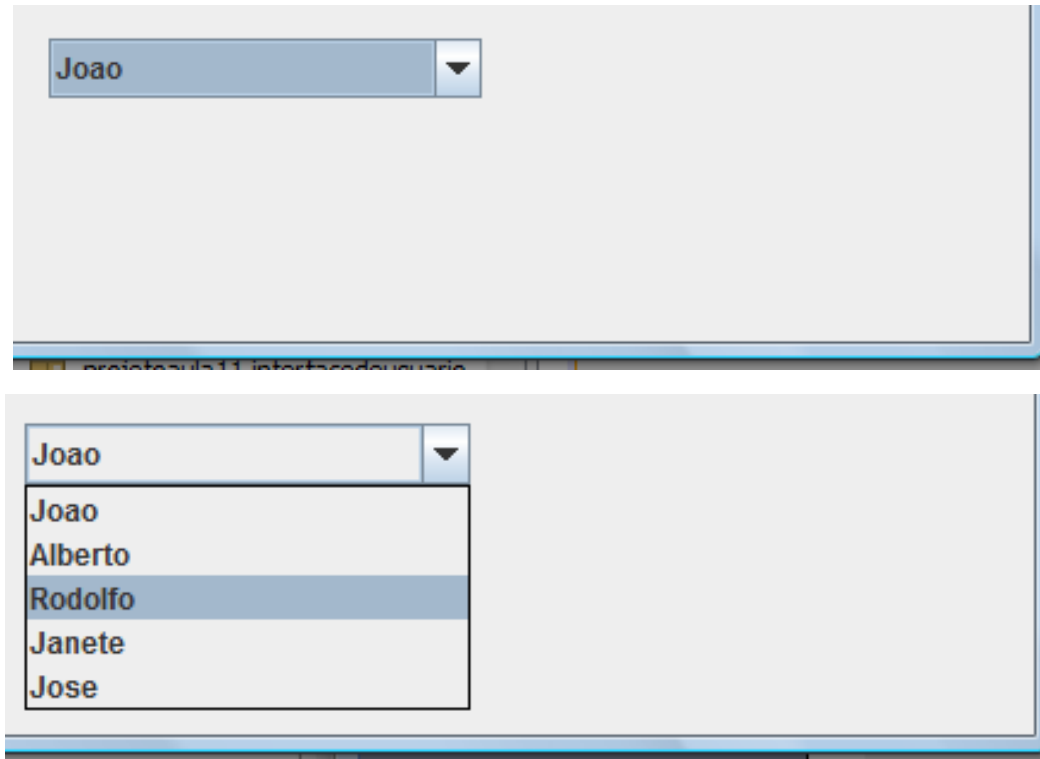
**Prof. Dr. Danilo Medeiros Eler**  
**danilo.eler@unesp.br**

# Aula de Hoje

- ComboBox
  - JComboBox
- ListBox
  - Jlist
- CheckBox
  - JCheckBox
- RadioButton
  - JRadioButton

# JComboBox

- Apresenta uma listagem que se expande quando o usuário clica sobre ela



# JComboBox

- O JComboBox é muito simples de utilizar e permite que objetos sejam adicionados a ele por meio do método **addItem**
  - Por exemplo, adicionar uma String vinda de um JText

```
palavras_Combo.addItem(palavra_Text.getText());
```

# JComboBox

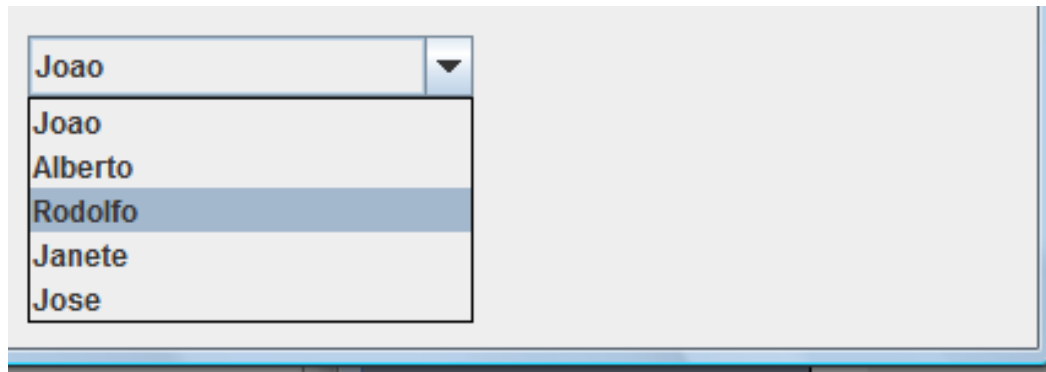
- Para recuperar o objeto selecionado basta utilizar o método **getSelectedItem()** e fazer
  - No exemplo também é mostrado também como obter o índice do objeto selecionado

```
String palavra = (String) palavras_Combo.getSelectedItem();  
int indice = palavras_Combo.getSelectedIndex();
```

# JComboBox

- Qualquer objeto pode ser adicionado ao JComboBox, por exemplo, objetos do tipo Cliente

```
Cliente clientes[] = control.getClientes();  
for (int i=0; i<clientes.length; i++){  
    clientes_Combo.addItem( clientes[i] );  
}
```



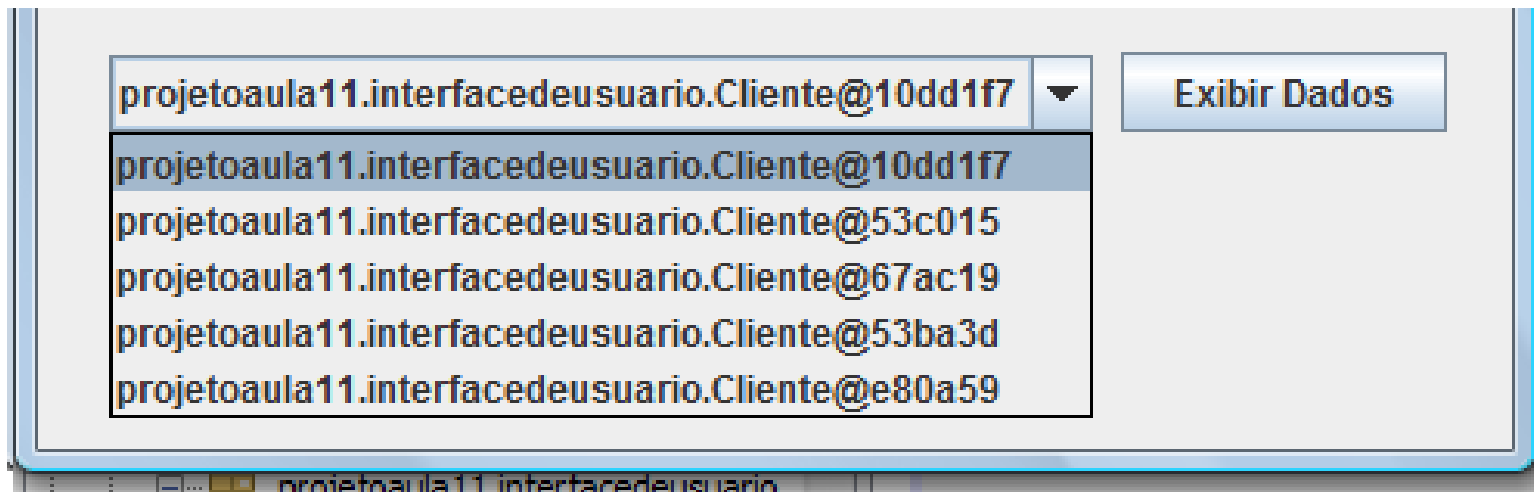
# JComboBox

- Será exibido no JComboBox o retorno do método **toString** do objeto
- Portanto, é necessário implementar esse método para retornar o que desejamos exibir
- No caso do exemplo anterior

```
@Override  
public String toString(){  
    return this.nome;  
}
```

# JComboBox

- Se o método **toString** não for sobrescrito será utilizada a implementação original, por exemplo



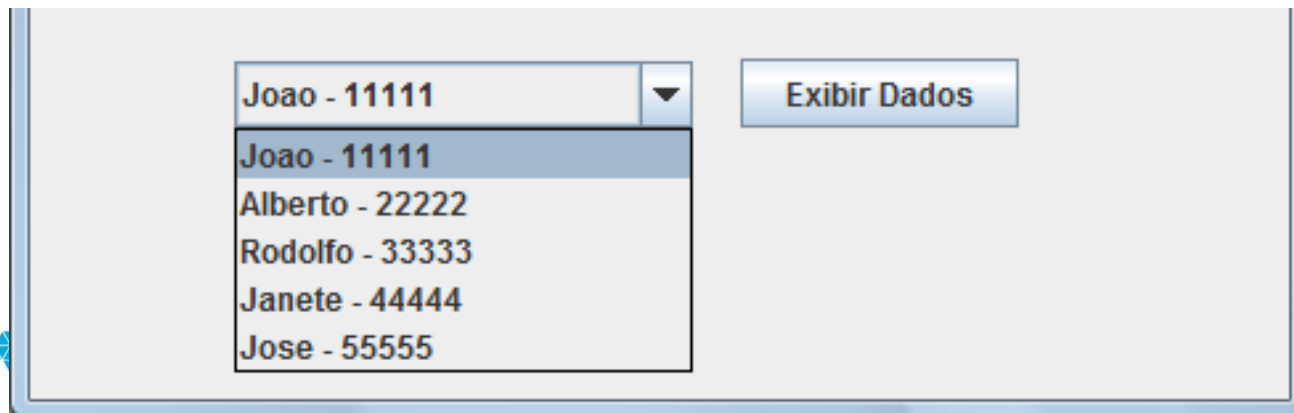


# JComboBox

- Podemos retornar os dados que desejarmos no método **toString**. Assim, quando for um objeto Cliente for exibido, será utilizada a String especificada pelo programador. Novo exemplo, com nome e CPF

@Override

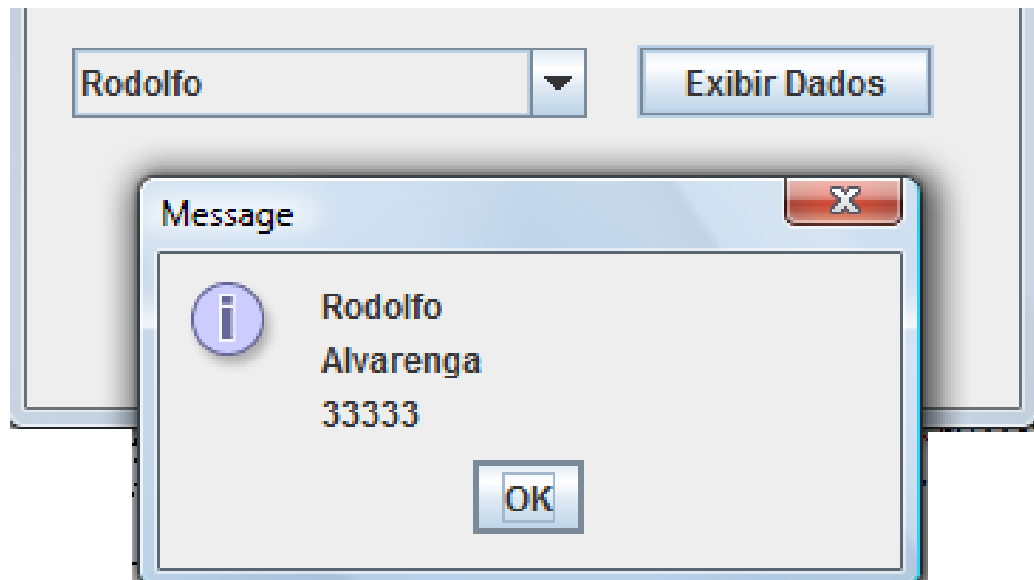
```
public String toString(){  
    return this.nome + " - " + this.CPF;  
}
```



# JComboBox

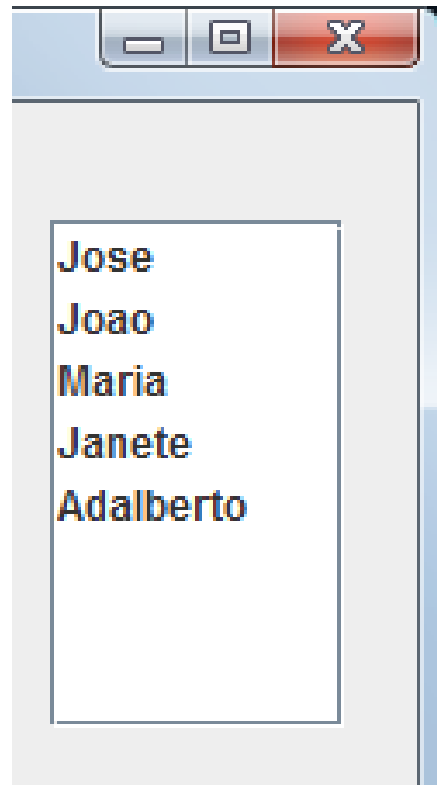
- Recuperando o objeto Cliente

```
Cliente cliente = (Cliente) clientes_Combo.getSelectedItem();  
JOptionPane.showMessageDialog(this, cliente.getNome()+"\n"  
    +cliente.getSobrenome()+"\n"+cliente.getCPF());
```



# JList

- O JList tem uma função de exibir itens em uma lista, como ilustrado abaixo



# JList

- Para adicionar elementos ao JList é necessário especificar um modelo para ele, semelhante ao que fizemos com o JTable
- Como, nesse caso, o JList é inicializado sem modelo, vamos criar um atributo e adicionar ao JList no construtor da classe

```
public class IUPrincipal extends javax.swing.JFrame {  
    private DefaultListModel modeloList = new DefaultListModel();  
    /** Creates new form IUPrincipal */  
    public IUPrincipal() {  
        initComponents();  
        palavras_List.setModel(modeloList);  
    }  
}
```

# JList

- Para adicionar elementos ao JList devemos acessar o modelo (atributo da classe) que criamos e invocar o método **addElement**, passando um objeto como parâmetro
  - Assim como o JComboBox, qualquer objeto pode ser adicionado ao JList
- Exemplo com String

```
modeloList.addElement( palavra_Text.getText() );
```

# JList

- Para recuperar um elemento do JList devemos utilizar o modelo criado e acessar o elemento desejado por meio do método **getElementAt**
  - Note que esse método exige um inteiro, isto é, o índice do elemento desejado
- No exemplo abaixo é mostrado como obter o índice de um elemento selecionado pelo usuário

```
int indice = palavras_List.getSelectedIndex();
```

```
String palavra = (String) modeloList.getElementAt( indice );
```

ou

```
String palavra = (String) palavras_List.getSelectedValue();
```

# JList

- Remoção

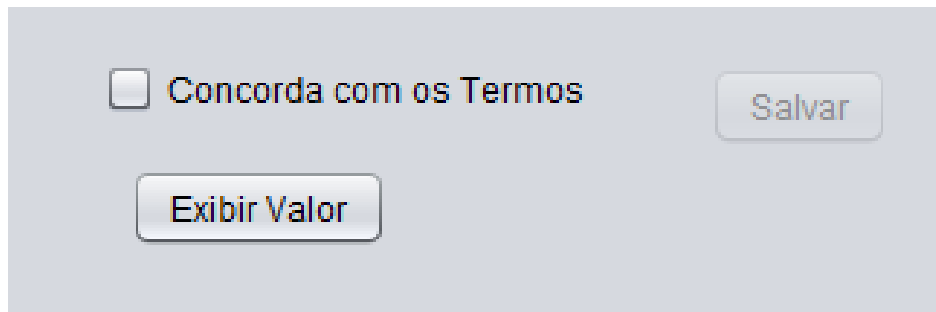
```
int index = palavras_List.getSelectedIndex();  
if (index >= 0) {  
    modeloList.removeElementAt(index);  
}
```

- Limpar todos os itens

```
modeloList.removeAllElements();
```

# CheckBox

- Exemplo 1

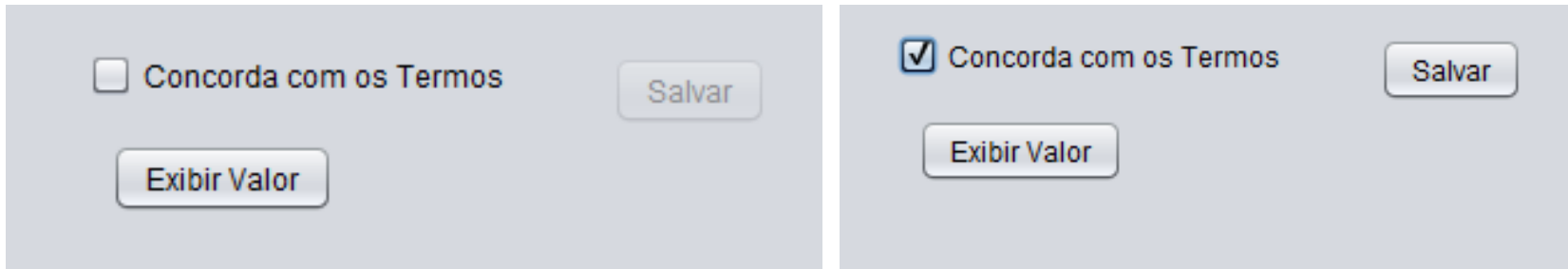


```
if (checkConcorda.isSelected()) {  
    System.out.println("concorda");  
}else{  
    System.out.println("não concorda");  
}
```



# CheckBox

- Exemplo 2
  - Evento StateChanged

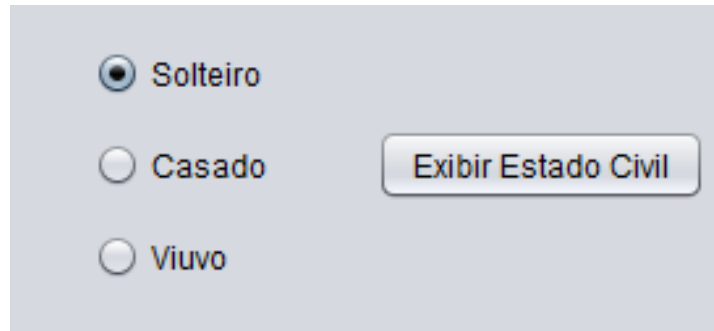


- Botão Salvar é habilitado quando o CheckBox está selecionado

```
butSalvar.setEnabled(checkConcorda.isSelected());
```

# RadioButton

- Usa-se um ButtonGroup para associar mais de um RadioButton



```
if (radioSolteiro.isSelected()) {  
    System.out.println("Selecionou Solteiro");  
} else if (radioCasado.isSelected()) {  
    System.out.println("Selecionou Casada");  
} else if (radioViuvo.isSelected()) {  
    System.out.println("Selecionou Viuvo");  
}
```

# Links

- <http://docs.oracle.com/javase/6/docs/api/javax/swing/JComboBox.html>
- <http://docs.oracle.com/javase/6/docs/api/javax/swing/JList.html>
- <http://docs.oracle.com/javase/1.5.0/docs/api/javax/swing/DefaultListModel.html>
- <https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html>
- <https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html>
- <https://docs.oracle.com/javase/7/docs/api/javax/swing/ButtonGroup.html>

# Referências

## **BIBLIOGRAFIA BÁSICA**

1. SINTES, A., Aprenda programação orientada a objetos em 21 dias, Pearson Education do Brasil, 2002.
2. VAREJÃO, F., Linguagens de programação : Java, C e C++ e outras : conceitos e técnicas, Campus, 2004.
3. DEITEL, H. M., DEITEL, P. J., **Java:** como programar, São Paulo: Pearson Education do Brasil, 2010. 1144p.
4. DEITEL, H. M., DEITEL, P. J., **Java:** como programar, Porto Alegre: Bookman, 2003. 1386p.
5. SAVITCH, W. J., C++ absoluto, Pearson Education : Addison Wesley, 2004.

## **BIBLIOGRAFIA COMPLEMENTAR**

1. BERMAN, A. M. *Data Structures via C++: Objects by Evolution*, Oxford University Press Inc., 1997.
2. BARNES, D.J. & KÖLLING, M., Programação orientada a objetos com Java, Pearson Education : Prentice Hall, 2004.
3. DEITEL, H. M. e DEITEL, P. J. *C++: Como Programar*, Bookman, 2001.
4. GILBERT, R. F. e FOROUZAN, B. A. *Data Structures: A Pseudo Approach with C++*, Brooks/Cole Thomson Learning, 2001.
5. MUSSER, D. R. e SAINI, A. *STL Tutorial and Reference Guide: Programming with the Standard Template Library*, Addison-Wesley, 1996.
6. SEBESTA, R. W. *Conceitos de Linguagem de Programação*, 4ª Ed., Bookman, 2003.
7. SEDGEWICK, R. *Algorithms in C++*, Addison-Wesley, 2002.
8. STROUSTRUP, B. *A Linguagem de Programação C++*, 3ª Ed., Bookman, 2000.

**FCT/Unesp – Presidente Prudente**  
**Departamento de Matemática e Computação**

# Programação Orientada a Objetos

## Interface de Usuário

### ComboBox e ListBox

Prof. Dr. Danilo Medeiros Eler  
daniloeler@fct.unesp.br