

# Padrões de projeto, testes automatizados e XML

---

## 04. Interfaces gráficas com swing



Esp. Márcio Palheta

Gtalk: [marcio.palheta@gmail.com](mailto:marcio.palheta@gmail.com)



# Interfaces gráficas com Swing

---

- Java 1.1 – **AWT** a primeira API
- A partir da Java 1.2 – **SWING** – padrão de mercado atual;
- Bibliotecas com grande número de componentes;
- Nativas da JRE;

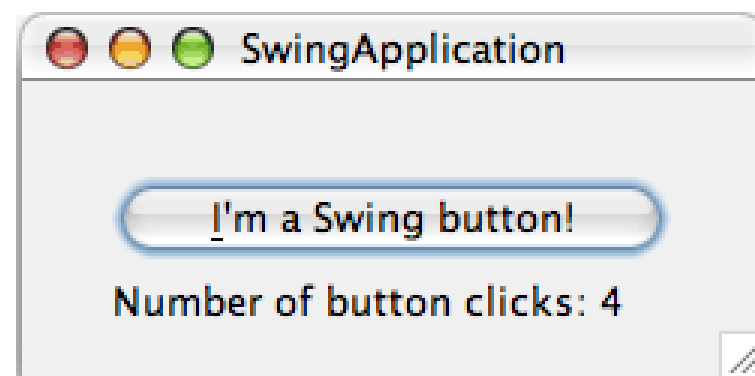
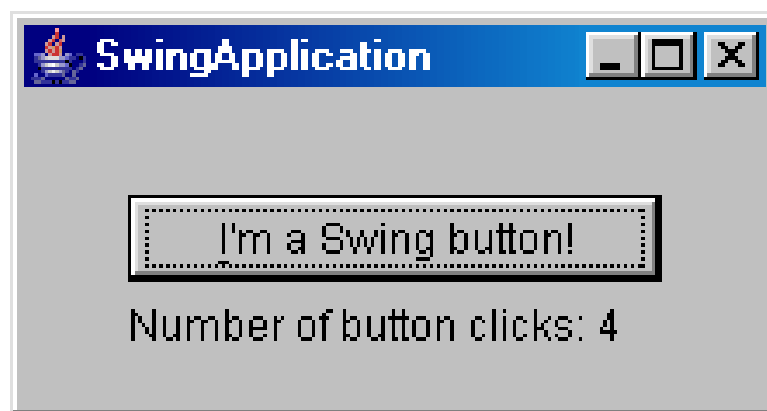
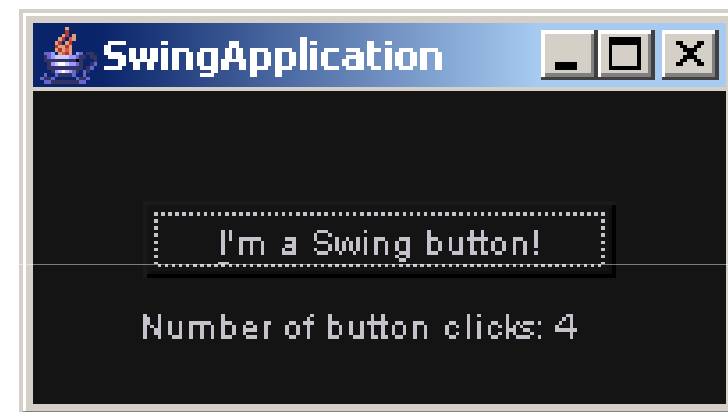


# Pensando em portabilidade

---

- Look-and-feel: componente de definição de propriedades como cores e tamanhos
- Mesmo conjunto de componentes para qualquer Sistema operacional;
- O próprio java já define um **padrão de cores**, mas que pode ser alterado;

# Screenshots do SWING

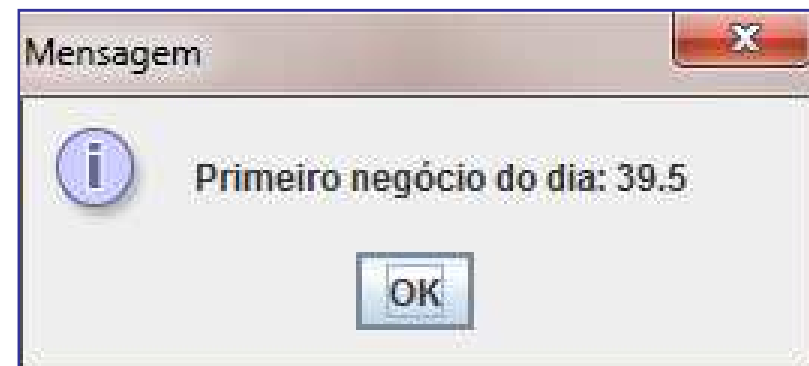
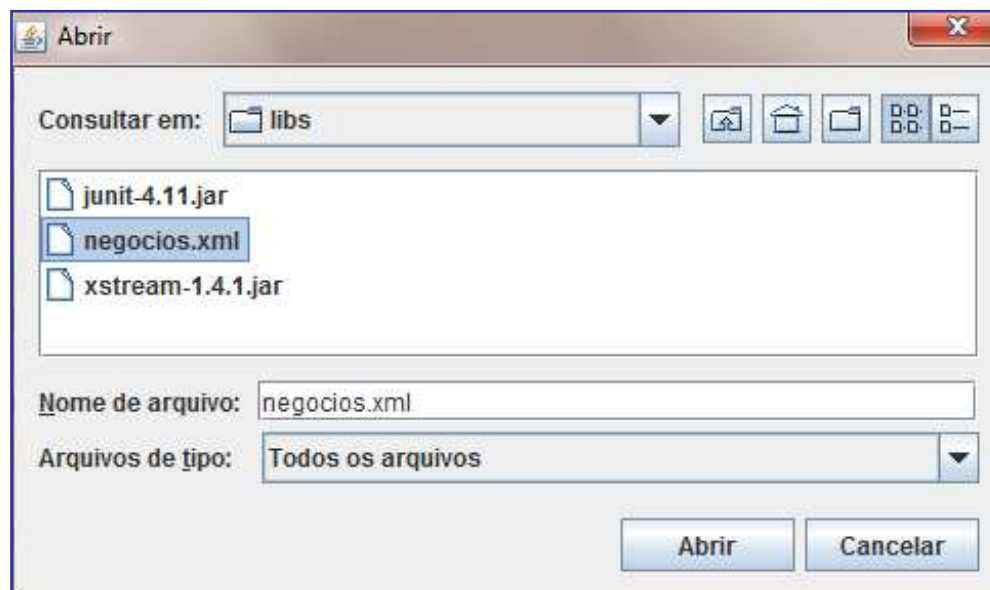


# Classe para escolha de XML

```
package br.com.bolsadevalores.view;
import java.io.FileNotFoundException;
public class XMLChooser {
    public void escolher() {
        try {
            JFileChooser fileChooser = new JFileChooser();
            int retorno = fileChooser.showOpenDialog(null);
            if (retorno == JFileChooser.APPROVE_OPTION) {
                FileReader reader = new FileReader(fileChooser.getSelectedFile());
                List<Negocio> negocios = new XMLReader().carregar(reader);
                Negocio primeiroNegocio = negocios.get(0);
                String msg = "Primeiro negócio do dia: "
                    + primeiroNegocio.getPreco();
                JOptionPane.showMessageDialog(null, msg);
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

# Execução da classe

```
public static void main(String[] args) {  
    new XMLChooser().escolher();  
}
```



# Melhorando o código

```
public class XMLChooser {  
    public void escolher() {  
        try {  
            //selecionando a pasta do projeto com "."  
            JFileChooser fileChooser = new JFileChooser(".");  
  
            //Selecionar apenas arquivo ".xml"  
            fileChooser.setFileFilter(new FileNameExtensionFilter("Apenas XML", "xml"));  
            int retorno = fileChooser.showOpenDialog(null);  
            if (retorno == JFileChooser.APPROVE_OPTION) {  
                FileReader reader = new FileReader(fileChooser.getSelectedFile());  
                List<Negocio> negocios = new XMLReader().carregar(reader);  
                Negocio primeiroNegocio = negocios.get(0);  
                String msg = "Primeiro negócio do dia: "  
                    + primeiroNegocio.getPreco();  
                JOptionPane.showMessageDialog(null, msg);  
            }  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

# Montando nossa tela: pensando em componentes

- Queremos montar uma tela com:
  - Um formulário;
  - Dois botões:
    - Um para carregar o XML; e
    - Outro para sair do programa;







# O design pattern Composite: Component e Container

---

- Swing e arquitetura flexível;
- O padrão **Composite**:
  - Todo **componente** é também um **container** de outros componentes;
  - Hierarquia composta por componentes que ficam dentro de outros componentes;
  - JButton → JPanel → JFrame;



# Tratamento de eventos

---

- **Botões** e o evento de **clique**;
- **Listeners**: interfaces que definem métodos que são disparados por eventos;
- O swing **captura** os eventos do usuário e invoca os métodos implementados;
- O botão e a interface **ActionListener**;
- O método **actionPerformed**;
- Como **implementar** ActionListener?



# Classes internas

---

- Numa tela, muitos componentes disparam eventos;
- Criar uma **classe** para cada **evento**?
- Listeners são objetos com pouco código
- É comum usarmos **classes internas**

```
public class Externa {  
    public class Interna{  
        // Classe interna  
    }  
}
```



# Classes internas

---

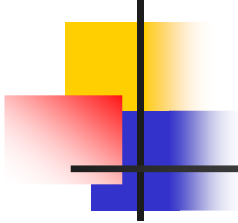
- A classe interna é um objeto da classe externa:
  - Externa.Interna
- Vantagem: Não precisa de novo arquivo `.java`;
- Podem ser marcadas como `private`;
- São classes normais – podem ter instâncias, implementar interfaces



# Classes anônimas

- É uma forma específica de classe **interna**
- Muito comum em código do SWING;
- Classe anônima para tratar o evento:

```
ActionListener eventoSair = new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        System.exit(0);  
    }  
};  
JButton botaoSair = new JButton("Sair");  
botaoSair.addActionListener(eventoSair);
```



## Simplificando – sem variável

- Não precisamos da variável de referência;
- Classe anônima declarada como parâmetro de método;

```
 JButton botaoSair = new JButton("Sair");  
 botaoSair.addActionListener(new ActionListener() {  
     public void actionPerformed(ActionEvent e) {  
         System.exit(0);  
     }  
 });
```



# Por dentro do código

---

- A sintaxe indica que estamos dando **new** em uma nova classe que implementa a interface **ActionListener**; e
- Possui o método **actionPerformed** que chama o **exit**.
- Mas qual é o nome dessa classe?
- Não sei, por isso é classe **anônima**. 😊

# Exercício:

## Tela principal

```
1 package br.com.bolsadevalores.view;
2 import java.awt.event.ActionEvent;
8 public class TelaPrincipal {
9
10     //Definicao de atributos
11     private JFrame janela;
12     private JPanel painel;
13     //Definicao do formulario principal
14     private void montaJanela() {
15         janela = new JFrame("Bolsa de valores");
16         janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17     }
18     //Definicao do painel principal
19     private void montaPainel() {
20         painel = new JPanel();
21         janela.add(painel);
22     }
23     // Continua...
```



# Exercício:

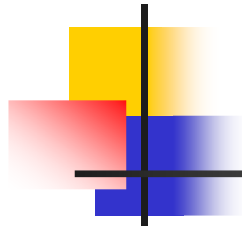
## Tela principal (cont...)

```
23 // Continua...
24 private void montaBotaoCarregar() {
25     JButton botaoCarregar = new JButton("Carregar XML");
26     // Definindo o evento
27     botaoCarregar.addActionListener(new ActionListener() {
28         public void actionPerformed(ActionEvent e) {
29             new XMLChooser().escolher();
30         }
31     });
32     painel.add(botaoCarregar);
33 }
34
35 private void montaBotaoSair() {
36     JButton botaoSair = new JButton("Sair");
37     botaoSair.addActionListener(new ActionListener() {
38         public void actionPerformed(ActionEvent e) {
39             System.exit(0);
40         }
41     });
42     painel.add(botaoSair);
43 }
```

# Exercício:

## Tela principal (final)

```
44 // Continua...
45 private void mostraJanela() {
46     // Ajuste o tamanho do form aos componente
47     janela.pack();
48     // Ou... Defina o tamanho do formulario
49     janela.setSize(540, 540);
50     // Exibe o formulario
51     janela.setVisible(true);
52 }
53 public void montaTela() {
54     montaJanela();
55     montaPainel();
56     montaBotaoCarregar();
57     montaBotaoSair();
58     mostraJanela();
59 }
60 public static void main(String[] args) {
61     new TelaPrincipal().montaTela();
62 }
63 }
```




# Trabalhando com tabelas

---

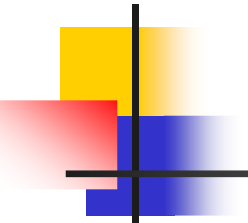
- Exibir o resultado em uma tabela
  - Linhas: negócios do XML;
  - 3 colunas: Preço, quantidade e data;
- Componente para exibição:
  - `javax.swing.table.JTable`
- Funções como: reorganização das colunas, drag and drop, ordenação e outras;

# Altere a classe XMLChooser



```
public class XMLChooser {  
    //Retorna uma colecao de negocios  
    public List<Negocio> escolher() {  
        try {  
            //selecionando a pasta do projeto com "."  
            JFileChooser fileChooser = new JFileChooser(".");  
            //Selecionar apenas arquivo ".xml"  
            fileChooser.setFileFilter(  
                new FileNameExtensionFilter("Apenas XML", "xml"));  
            int retorno = fileChooser.showOpenDialog(null);  
            if (retorno == JFileChooser.APPROVE_OPTION) {  
                FileReader reader = new FileReader(  
                    fileChooser.getSelectedFile());  
                //Retorna a lista processada  
                return new XMLReader().carregar(reader);  
            }  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        }  
        //Retorna uma lista generica, melhor que NULL  
        return Collections.emptyList();  
    }  
}
```

# Atualização da tela principal



```
public class TelaPrincipal {
    // Definicao de atributos
    private JFrame janela;
    private JPanel painel;
    private JTable tabela;

    private void montaTabelaComScroll() {
        //Configuracao da tabela
        tabela = new JTable();
        tabela.setBorder(new LineBorder(Color.black));
        tabela.setGridColor(Color.black);
        tabela.setShowGrid(true);
        //Configuracao da barra de rolagem
        JScrollPane scroll = new JScrollPane();
        scroll.getViewport().setBorder(null);
        scroll.getViewport().add(tabela);
        scroll.setSize(450, 450);
        //Adicione a barra ao painel principal
        painel.add(scroll);
    }
}
```

# Atualizar método

## TelaPrincipal.montarTela()

- Atualize o método montarTela();
- Inclua a chamada ao novo método:
  - montarTabelaComScroll()

```
public void montarTela() {  
    montaJanela();  
    montaPainel();  
    montarTabelaComScroll();  
    montaBotaoCarregar();  
    montaBotaoSair();  
    mostraJanela();  
}
```

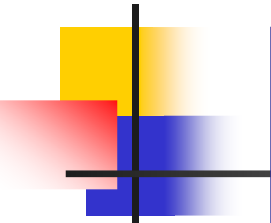


# Implementando TableModel

---

- Um **table model** é responsável por devolver para a **tabela** os dados a serem exibidos;
- A classe **AbstractTableModel**:
  - **getColumnCount** - devolve a quantidade de colunas
  - **getRowCount** - devolve a quantidade de linhas
  - **getValueAt(row, column)** - dada uma linha e uma coluna devolve o valor correspondente

# Implementando TableModel



```
package br.com.bolsadevalores.view;
import java.util.List;
@SuppressWarnings("serial")
public class NegociosTableModel extends AbstractTableModel {
    private final List<Negocio> negocios;
    public NegociosTableModel(List<Negocio> negocios) {
        this.negocios = negocios;
    }
    public int getColumnCount() { return 3; }

    public int getRowCount() { return negocios.size(); }

    public Object getValueAt(int rowIndex, int columnIndex) {
        Negocio n = negocios.get(rowIndex);
        switch (columnIndex) {
            case 0: return n.getPreco();
            case 1: return n.getQuantidade();
            case 2: return n.getData();
        }
        return null;
    }
}
```

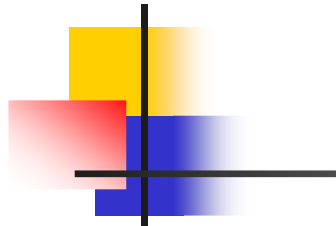


# Atualização da classe TelaPrincipal

- Atualize a classe interna para atualizar o TableModel da tabela:

```
private void montaBotaoCarregar() {  
    JButton botaoCarregar = new JButton("Carregar XML");  
    // Definindo o evento  
    botaoCarregar.addActionListener(new ActionListener() {  
        public void actionPerformed(ActionEvent e) {  
            List<Negocio> negocios = new XMLChooser().escolher();  
            NegociosTableModel model = new NegociosTableModel(negocios);  
            tabela.setModel(model);  
        }  
    });  
    painel.add(botaoCarregar);  
}
```

# Executando a tela principal



Bolsa de valores		
A	B	C
39.5	1076	java.util.GregorianCalendar...
40.45	1033	java.util.GregorianCalendar...
39.82	1118	java.util.GregorianCalendar...
39.21	1144	java.util.GregorianCalendar...
39.86	1081	java.util.GregorianCalendar...
39.06	1166	java.util.GregorianCalendar...
38.22	1188	java.util.GregorianCalendar...
37.58	1138	java.util.GregorianCalendar...
37.11	1046	java.util.GregorianCalendar...
37.32	990	java.util.GregorianCalendar...
37.62	1023	java.util.GregorianCalendar...
38.37	1076	java.util.GregorianCalendar...
38.72	1162	java.util.GregorianCalendar...
38.14	1105	java.util.GregorianCalendar...
37.28	1041	java.util.GregorianCalendar...
37.96	1031	java.util.GregorianCalendar...
38.1	1081	java.util.GregorianCalendar...
38.43	1002	java.util.GregorianCalendar...
39.41	953	java.util.GregorianCalendar...
39.8	896	java.util.GregorianCalendar...
38.8	993	java.util.GregorianCalendar...
38.98	920	java.util.GregorianCalendar...
38.86	969	java.util.GregorianCalendar...
39.61	895	java.util.GregorianCalendar...
39.26	871	java.util.GregorianCalendar...

Carregar XML Sair

# Atualização da classe NegociosTableModel

```
public class NegociosTableModel extends AbstractTableModel {  
    private final List<Negocio> negocios;  
    @Override  
    //Marcando a tabela com "somente leitura"  
    public boolean isCellEditable(int rowIndex, int columnIndex) {  
        return false;  
    }  
    @Override  
    //Metodo para informar o nome da coluna  
    public String getColumnName(int column) {  
        switch (column) {  
            case 0: return "Preço";  
            case 1: return "Quantidade";  
            case 2: return "Data";  
        }  
        return null;  
    }  
}
```


# Execute a tela principal



The screenshot shows a Java Swing window titled "Bolsa de valores". It contains a table with three columns: "Preço", "Quantidade", and "Data". The table lists 25 rows of stock data. The "Data" column contains the string "java.util.GregorianCalendar..." for all entries. At the bottom of the window, there are two buttons: "Carregar XML" and "Sair".

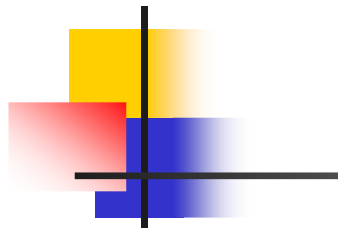
Preço	Quantidade	Data
39.5	1076	java.util.GregorianCalendar...
40.45	1033	java.util.GregorianCalendar...
39.82	1118	java.util.GregorianCalendar...
39.21	1144	java.util.GregorianCalendar...
39.86	1081	java.util.GregorianCalendar...
39.06	1166	java.util.GregorianCalendar...
38.22	1188	java.util.GregorianCalendar...
37.58	1138	java.util.GregorianCalendar...
37.11	1046	java.util.GregorianCalendar...
37.32	990	java.util.GregorianCalendar...
37.62	1023	java.util.GregorianCalendar...
38.37	1076	java.util.GregorianCalendar...
38.72	1162	java.util.GregorianCalendar...
38.14	1105	java.util.GregorianCalendar...
37.28	1041	java.util.GregorianCalendar...
37.96	1031	java.util.GregorianCalendar...
38.1	1081	java.util.GregorianCalendar...
38.43	1002	java.util.GregorianCalendar...
39.41	953	java.util.GregorianCalendar...
39.8	896	java.util.GregorianCalendar...
38.8	993	java.util.GregorianCalendar...
38.98	920	java.util.GregorianCalendar...
38.86	969	java.util.GregorianCalendar...
39.61	895	java.util.GregorianCalendar...
39.26	871	java.util.GregorianCalendar...

# Tabela com título



```
private void montaTitulo() {  
    JLabel titulo = new JLabel("Lista de Negócios");  
    titulo.setFont(new Font("Verdana", Font.BOLD, 25));  
    titulo.setForeground(new Color(50, 50, 100));  
    titulo.setHorizontalAlignment(SwingConstants.CENTER);  
    painel.add(titulo);  
}  
  
public void montaTela() {  
    montaJanela();  
    montaPainel();  
    montaTitulo();  
    montaTabelaComScroll();  
    montaBotaoCarregar();  
    montaBotaoSair();  
    mostraJanela();  
}
```

# Tabela com título



Bolsa de valores

## Lista de Negócios

Preço	Quantidade	Data
39.5	1076	java.util.GregorianCalendar...
40.45	1033	java.util.GregorianCalendar...
39.82	1118	java.util.GregorianCalendar...
39.21	1144	java.util.GregorianCalendar...
39.86	1081	java.util.GregorianCalendar...
39.06	1166	java.util.GregorianCalendar...
38.22	1188	java.util.GregorianCalendar...
37.58	1138	java.util.GregorianCalendar...
37.11	1046	java.util.GregorianCalendar...
37.32	990	java.util.GregorianCalendar...
37.62	1023	java.util.GregorianCalendar...
38.37	1076	java.util.GregorianCalendar...
38.72	1162	java.util.GregorianCalendar...
38.14	1105	java.util.GregorianCalendar...
37.28	1041	java.util.GregorianCalendar...
37.96	1031	java.util.GregorianCalendar...
38.1	1081	java.util.GregorianCalendar...
38.43	1002	java.util.GregorianCalendar...
39.41	953	java.util.GregorianCalendar...
39.8	896	java.util.GregorianCalendar...
38.8	993	java.util.GregorianCalendar...
38.98	920	java.util.GregorianCalendar...
38.86	969	java.util.GregorianCalendar...
39.61	895	java.util.GregorianCalendar...

Carregar XML Sair



# Exibindo dados formatados

- Altere o método `getValueAt` da classe **NegociosTableModel**:

```
public Object getValueAt(int rowIndex, int columnIndex) {  
    Negocio n = negocios.get(rowIndex);  
    switch (columnIndex) {  
        case 0:  
            NumberFormat numberFormat = NumberFormat.getCurrencyInstance();  
            return numberFormat.format(n.getPreco());  
        case 1:  
            return n.getQuantidade();  
        case 2:  
            DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");  
            return dateFormat.format(n.getData().getTime());  
    }  
    return null;  
}
```

# Exibindo dados formatados



Bolsa de valores

## Lista de Negócios

Preço	Quantidade	Data
R\$ 39,50	1076	12/09/2008
R\$ 40,45	1033	12/09/2008
R\$ 39,82	1118	12/09/2008
R\$ 39,21	1144	12/09/2008
R\$ 39,86	1081	12/09/2008
R\$ 39,06	1166	12/09/2008
R\$ 38,22	1188	12/09/2008
R\$ 37,58	1138	12/09/2008
R\$ 37,11	1046	12/09/2008
R\$ 37,32	990	12/09/2008
R\$ 37,62	1023	13/09/2008
R\$ 38,37	1076	13/09/2008
R\$ 38,72	1162	13/09/2008
R\$ 38,14	1105	13/09/2008
R\$ 37,28	1041	13/09/2008
R\$ 37,96	1031	14/09/2008
R\$ 38,10	1081	14/09/2008
R\$ 38,43	1002	14/09/2008
R\$ 39,41	953	14/09/2008
R\$ 39,80	896	14/09/2008
R\$ 38,80	993	14/09/2008
R\$ 38,98	920	14/09/2008
R\$ 38,86	969	15/09/2008
R\$ 39,61	895	15/09/2008
R\$ 39,26	871	15/09/2008

Carregar XML Sair





# Bibliografia

---

- Java - Como programar, de Harvey M. Deitel
- Use a cabeça! - Java, de Bert Bates e Kathy Sierra
- (Avançado) Effective Java Programming Language Guide, de Josh Bloch



# Referências WEB

---

- **Site oficial:**

- SUN: [www.java.sun.com](http://www.java.sun.com)

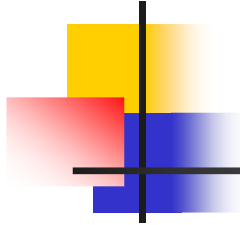
- **Fóruns e listas:**

- Javaranch: [www.javaranch.com](http://www.javaranch.com)
- GUJ: [www.guj.com.br](http://www.guj.com.br)

- **Apostilas:**

- Argonavis: [www.argonavis.com.br](http://www.argonavis.com.br)
- Caelum: [www.caelum.com.br](http://www.caelum.com.br)

# Padrões de projeto, testes automatizados e XML



## 04. Interfaces gráficas com swing



Esp. Márcio Palheta

Gtalk: [marcio.palheta@gmail.com](mailto:marcio.palheta@gmail.com)