

## Programação Orientada a Objetos

Fausto Maranhão Ayres

### 5

## Aplicação gráfica (parte3) - Componentes

### JButton

```
button.setEnabled(true);           //habilita o botão
button.setEnabled(false);          //desabilita o botão
button.setBorderPainted(false);    //desabilita a borda
button.setToolTipText("dica");     //dica pro usuario
button.setMnemonic(KeyEvent.VK_F5); //tecla de atalho
button.setMnemonic('a');           //tecla de atalho alt + a

//chamar o actionPerformed() de um button
button.doClick();
```

# JLabel

```
label.setOpaque(true);           //habilita cor de fundo
label.setForeground(Color.RED);   //cor da fonte
label.setBackground(Color.RED);   //cor do fundo
label.setFont(new Font("Serif", Font.PLAIN, 14));
label.setBorder(new LineBorder(Color.BLACK, 4, true));
```

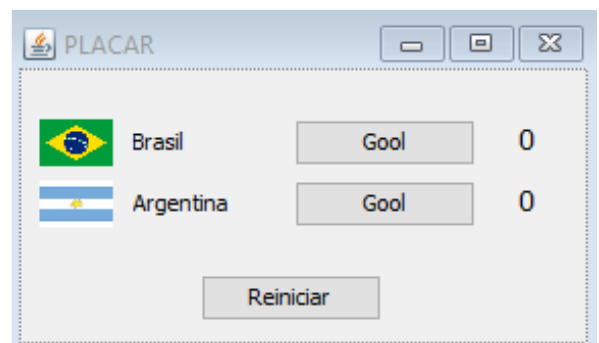
3

Fausto M. Ayres

## Placar.java

### ■ Botões Gool

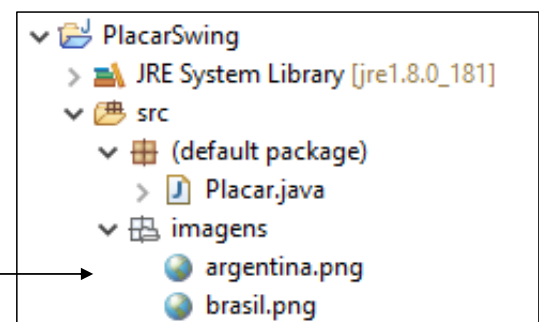
- Incrementa cada contador
- Se o contador atingir 5, o nome do vencedor será exibido e os 2 botões Gool serão desabilitados



### ■ Botão Reiniciar

- Habilita os 2 botões Gool
- Zera os contadores

Baixe as imagens das bandeiras do Brasil e Argentina e copie para o **package** imagens



4

Fausto M. Ayres

# Adicionando imagem

```
//Após a criação do label  
//carregar imagem de arquivo  
...
```

```
ImageIcon icon =
```

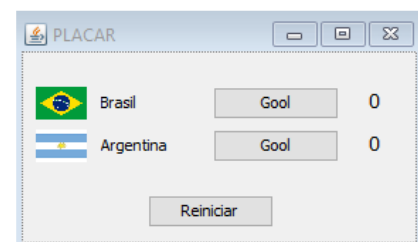
```
new ImageIcon(Placar.class.getResource("/imagens/brasil.png"));
```



```
//escalando imagem para o mesmo tamanho do label
```

```
icon.setImage(icon.getImage().getScaledInstance(  
    label.getWidth(),  
    label.getHeight(),  
    Image.SCALE_DEFAULT  
));
```

```
label.setIcon(icon);
```

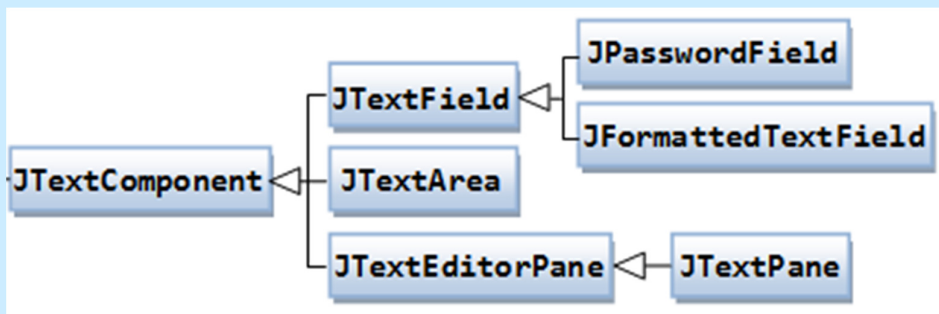


fausto.ayres@ifpb.edu.br

5

# Gerar arquivo executável





# COMPONENTES TEXTUAIS

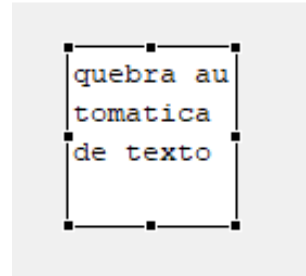
## JTextField

```
textField.setEditable(false); //trava a digitação  
textField.requestFocus(); //recebe o cursor  
textField.selectAll();  
textField.setHorizontalAlingn(JTextField.LEFT);  
textField.setVerticalAlingn(JTextField.TOP);
```

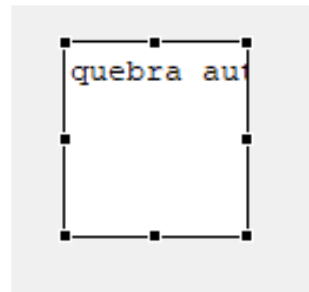
# JTextArea

```
textArea.append("nova linha \n");  
textArea.append("nova linha \n");  
textArea.setEditable(true); //permite digitação
```

```
//Quebra automática de texto  
textArea.setLineWrap(true);
```



```
textArea.setLineWrap(false);
```

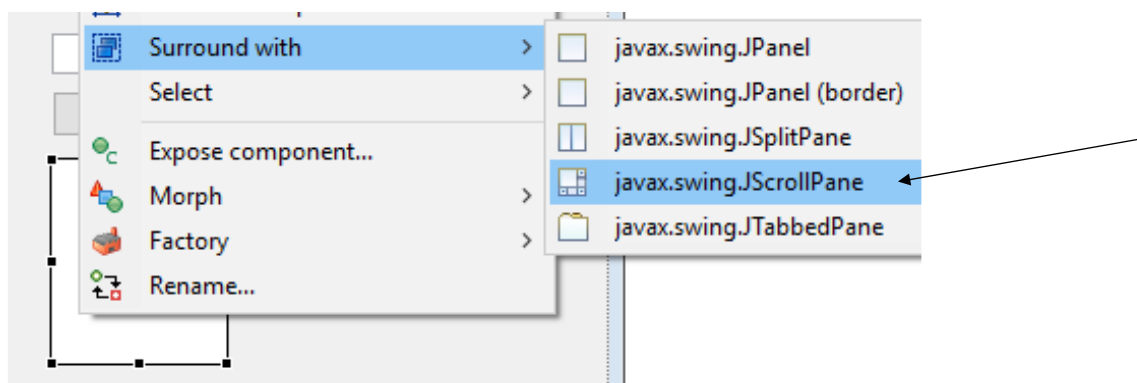
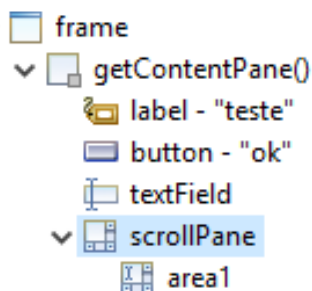


Fausto M. Ayres

9

## JTextArea + JScrollPane

```
JTextArea area1 = new JTextArea(10,20);  
JScrollPane scrollPane = new JScrollPane(area1);
```



fausto.ayres@ifpb.edu.br

10

# JFormattedTextField

```
MaskFormatter mask= new MaskFormatter("(##)####-####");  
mask.setPlaceholderCharacter('_');
```



```
JFormattedTextField tx = new JFormattedTextField(mask);  
String s = tx.getText();
```

fausto.ayres@ifpb.edu.br

11

## Máscaras

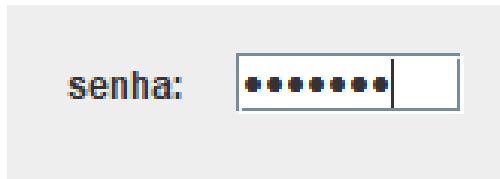
Máscaras prontas:

**telefone:** "(##)####-####"  
**data:** "##/##/####"  
**cep:** "#####-###"  
**cpf:** "###.###.###-##"  
**cnpj:** "##.###.###/####-##"  
**placa:** "UUU-####" (antigo)

- # indica que qualquer número poderá ser inserido (0-9);
- U indica que qualquer letra (a-z) poderá ser inserida.  
A máscara converterá letras minúsculas em maiúsculas;
- L indica qualquer letra (a-z) poderá ser inserida.  
A máscara converterá letras maiúsculas em minúsculas;
- ? indica qualquer letra (a-z) poderá ser inserida.  
A máscara manterá a letra inserida;
- A indica qualquer letra ou numero (0-9 e a-z) poderá ser inserido;
- \* indica qualquer coisa, incluindo caracteres especiais poderão ser inseridos

# JPasswordField

```
JPasswordField pf = new JPasswordField("ifpb");
```



*Esconde os caracteres*

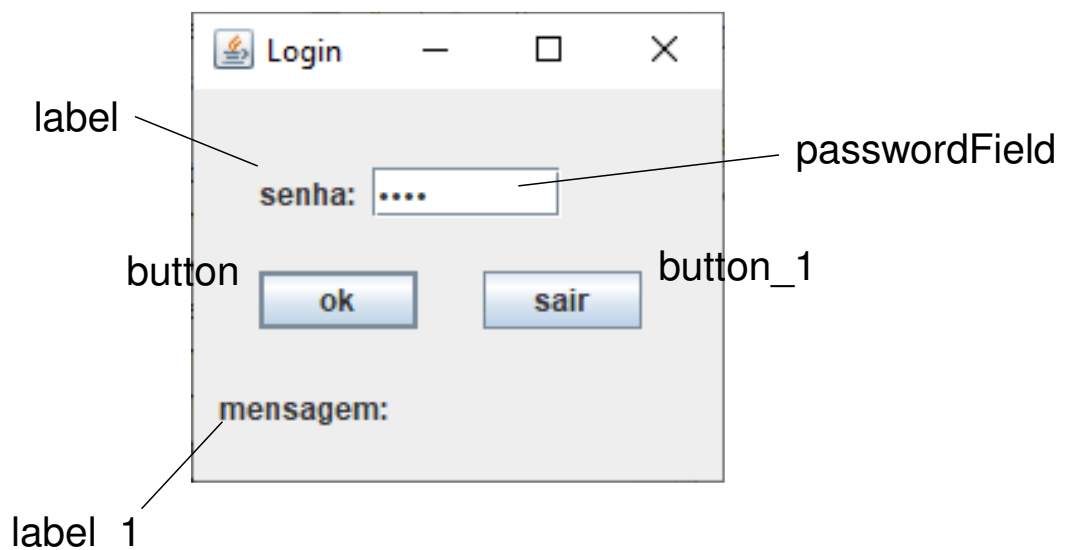
## Obs:

Por questões de segurança, ele armazena o conteúdo no formato **char[ ]**

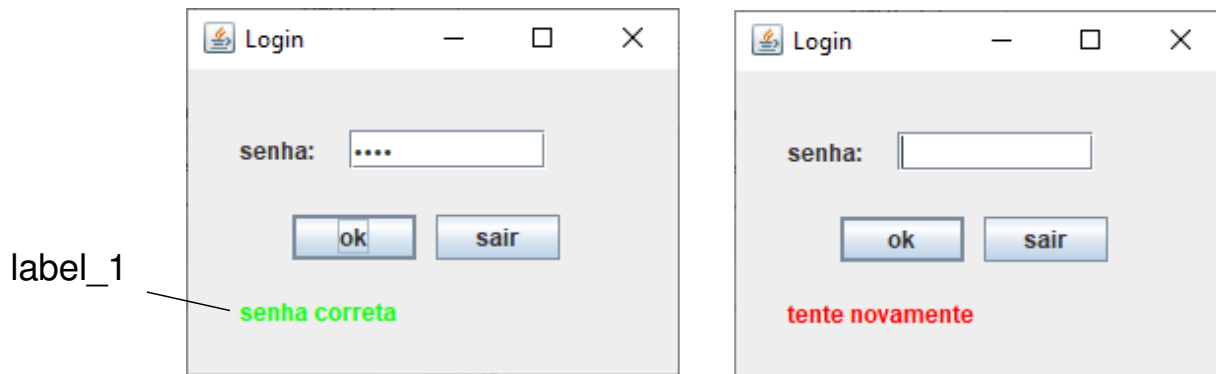
No entanto, pode-se criar a **String** com o **char[ ]**

```
String texto = new String(pf.getPassword());
```

## Login.java



# Login.java



## MÚTIPLAS JANELAS



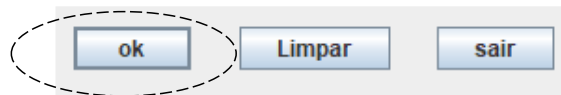
## JFrame

```
frame.setResizable(false); //fixa tamanho
frame.setCursor(Cursor.CURSOR_POINTING_HAND);

//ícone na barra de titulo da janela
ImageIcon icon = new ImageIcon("c:/fig.jpg");
frame.setIconImage(icon.getImage());

//maximizar e minimizar
frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
frame.setExtendedState(JFrame.ICONIFIED);

//Botão default do frame (ativado com ENTER)
frame.getRootPane().setDefaultButton(button);
```



```
frame.setVisible(false); //oculta janela mas não destrói
frame.dispose(); //destrói a janela
```

17

## Fechamento (destruição) da Janela

```
frame.setVisible(false); //oculta janela mas não fecha
frame.dispose(); //fecha (destroi) a janela
```

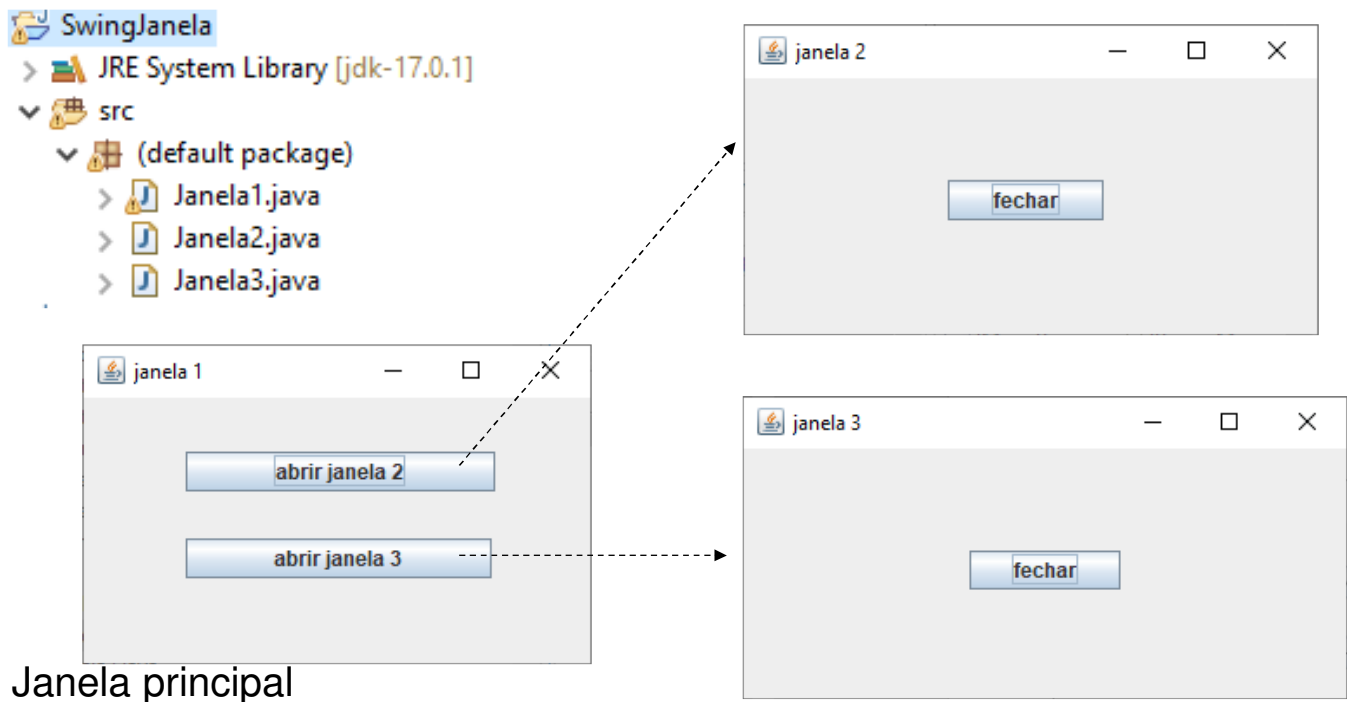
```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
```

### Formas de fechamento da janela

<code>JFrame.EXIT_ON_CLOSE</code>	fecha todas as janelas
<code>JFrame.DISPOSE_ON_CLOSE</code>	fecha somente a janela atual
<code>JFrame.HIDE_ON_CLOSE</code>	não fecha, apenas oculta
<code>JFrame.DO_NOTHING_ON_CLOSE</code>	não fecha nem oculta

# Usando múltiplas janelas

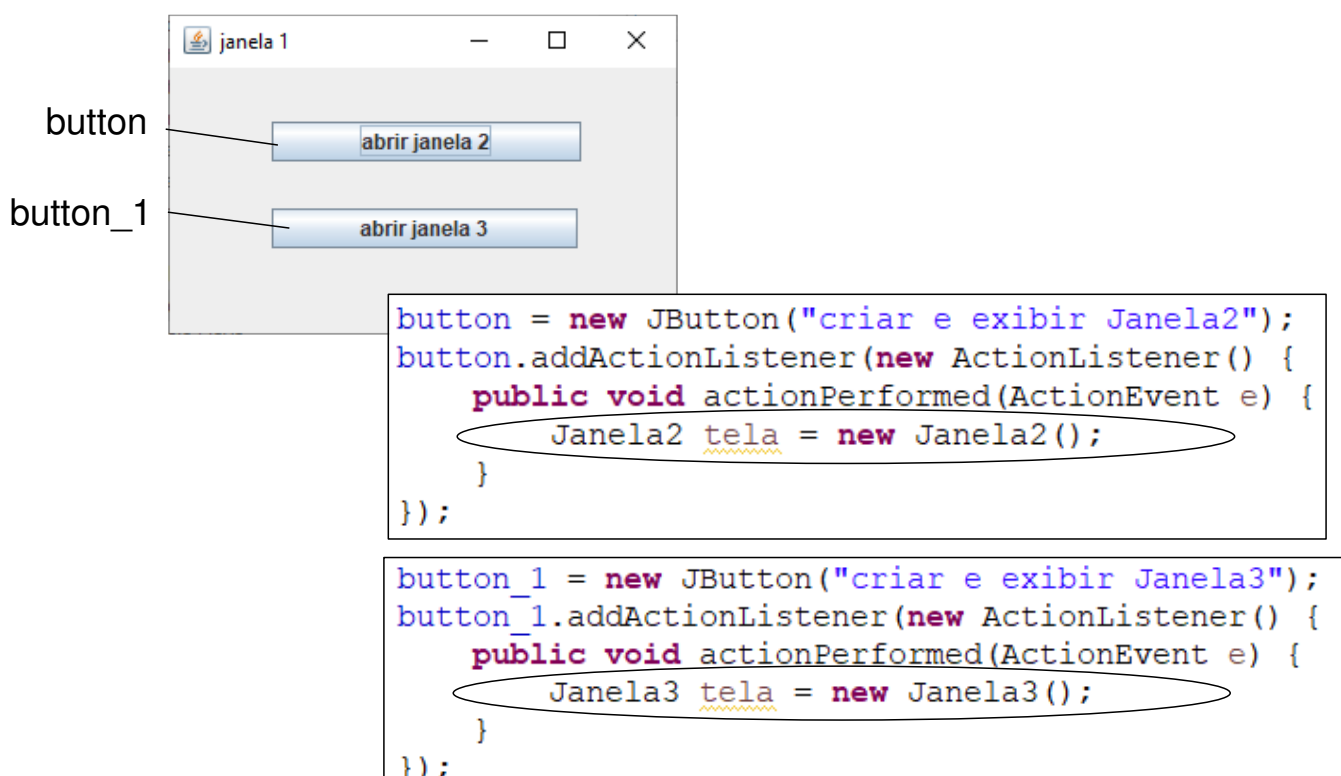
- Crie um projeto com as 3 janelas abaixo



Fausto M. Ayres

19

## Criando Janela2 e Janela3



Fausto M. Ayres

20

# Exibindo Janela2 e Janela3

- Serão exibidas somente se existir a instrução `frame.setVisible(true)` no construtor delas

```
public Janela2() {  
    initialize();  
    frame.setVisible(true);  
}
```



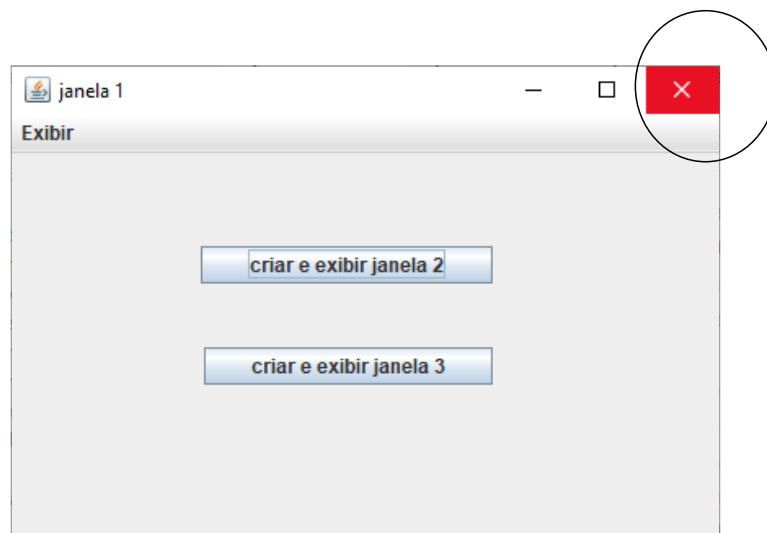
```
public Janela3() {  
    initialize();  
    frame.setVisible(true);  
}
```



Fausto M. Ayres

21

# Fechando a Janela1



Cuidado!

O fechamento da Janela1 provoca o fechamento das demais janelas, ou seja, o **término da aplicação**

Fausto M. Ayres

22

# Configurar a forma de fechamento



`frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)` *default*



`frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE)`

Fausto M. Ayres

23

## Fechando Janela2 e Janela3



Ação

Ação

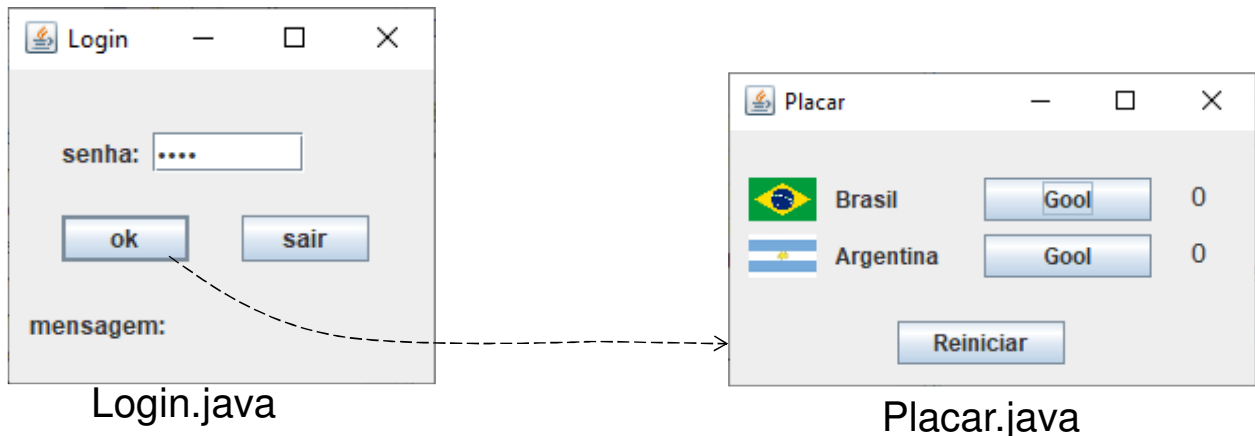
```
button = new JButton("fechar janela");
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        frame.dispose();
    }
});
```

Fausto M. Ayres

24

# Exemplo: Telas Login + Aplicação

- A aplicação Login **abre** a aplicação Placar:
- A aplicação Login será **fechada sem fechar** a aplicação Placar

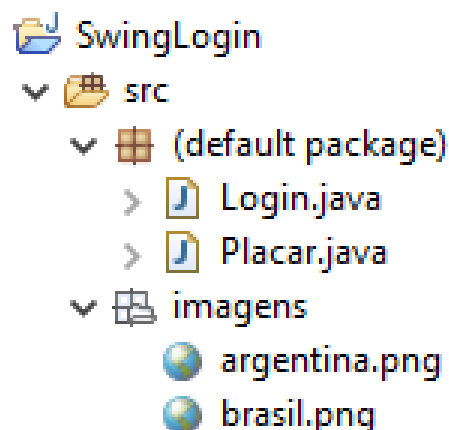


Fausto M. Ayres

25

## Preparação

Junte os arquivos **Login.java** e **Placar.java** no mesmo projeto



Fausto M. Ayres

26

# Aplicação Login

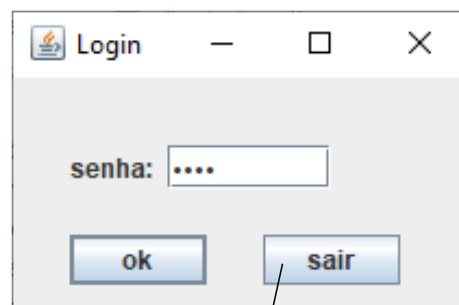


```
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        String texto = new String(passwordField.getPassword());
        if (texto.equals("ifpb")) {
            frame.dispose();
            Placar tela = new Placar();
        } else {
            label_1.setText("tente novamente");
            passwordField.setText("");
            passwordField.requestFocus();
        }
    }
});
```



27

# Aplicação Login



```
button_2 = new JButton("sair");
button_2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        frame.dispose();
    }
});
```

Fechar Login

28

# Aplicação Placar

- A aplicação Placar só aparecerá se existir **frame.setVisible(true)** no construtor

```
public Placar() {  
    initialize();  
    frame.setVisible(true);  
}
```



Fausto M. Ayres

29

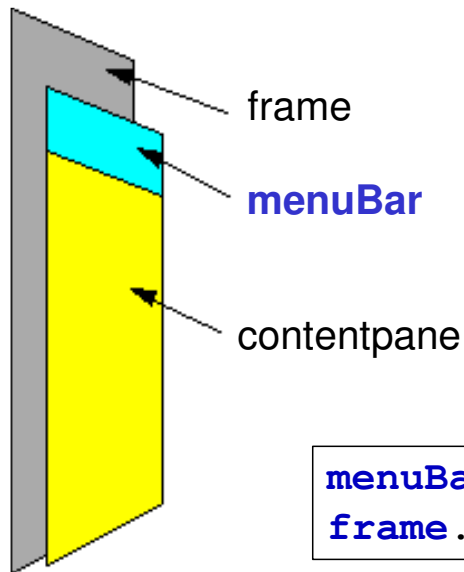
## COMPONENTES DE MENU

Fausto M. Ayres

30

# Barra de Menus

- A barra de menus é independente do painel de conteúdo.



```
menuBar = new JMenuBar();  
frame.setJMenuBar(menuBar);
```

Fausto M. Ayres

31

## JMenuBar, JMenu e JMenuItem

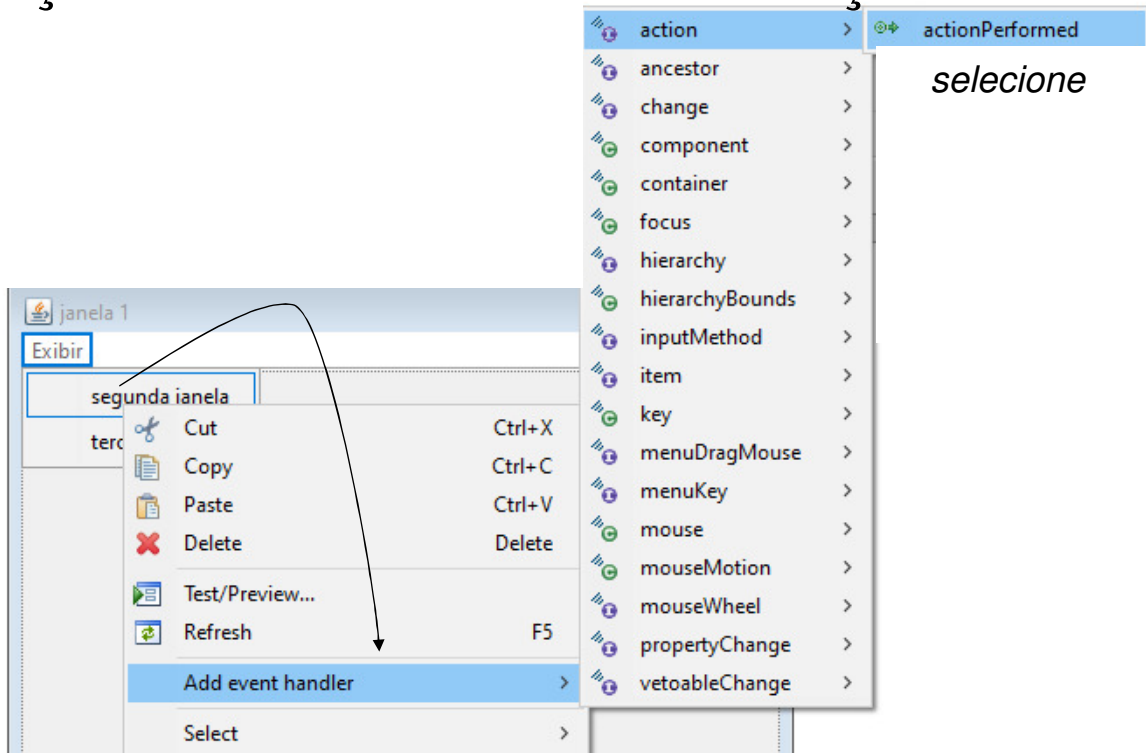
Fausto M. Ayres

32



# ActionListener/actionPerformed()

- Ação do item de menu é similar ação do botão



Fausto M. Ayres

33

# ActionListener/actionPerformed()

- Ação:
  - chamar **button.doClick()** para abrir a janela

```
menuItem = new JMenuItem("segunda janelas");
menuItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        button.doClick();
    }
});
```

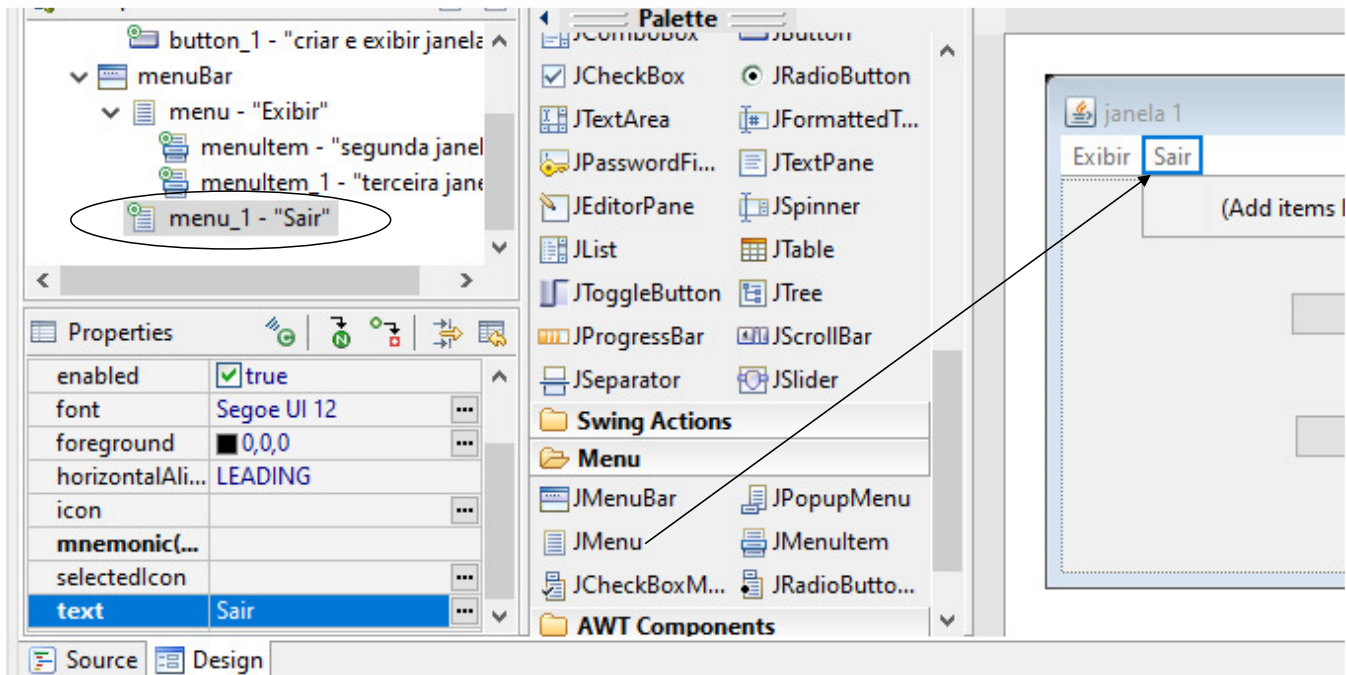
Redireciona para actionPerformed() do button

Fausto M. Ayres

34

# Menu “Sair”

- Utiliza-se um JMenu sem nenhum item

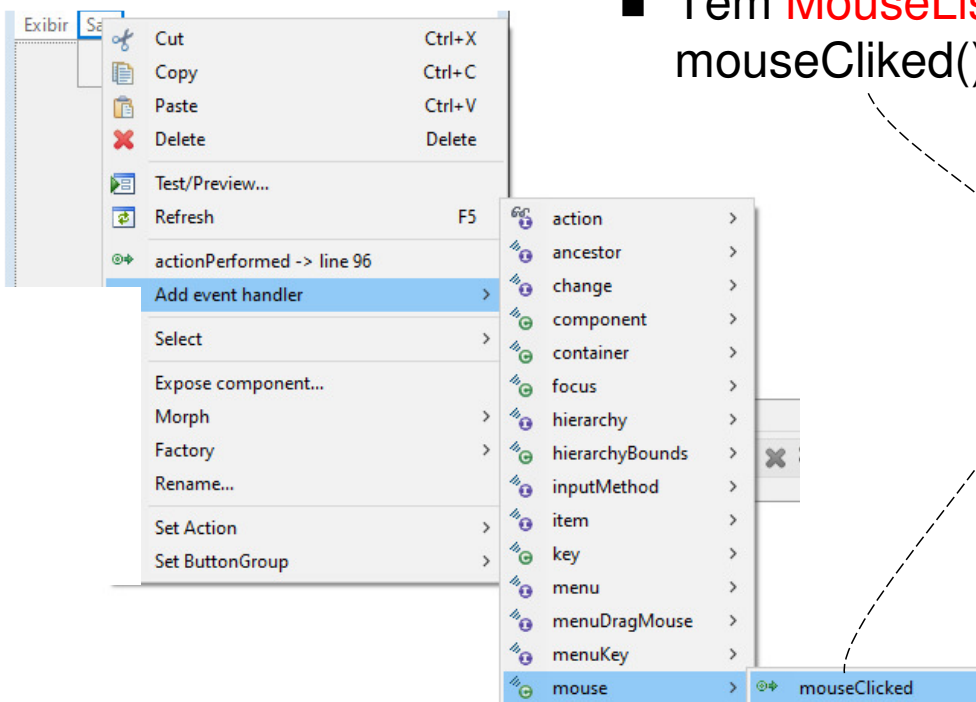


Fausto M. Ayres

35

# Menu “Sair”

- **JMenu** não tem **ActionListener**
- Tem **MouseListener** com método `mouseClicked()`



36

# Menu "Sair"

## ■ MouseListener + mouseClicked()

```
menu_1 = new JMenu("Sair");  
menu_1.addMouseListener(new MouseAdapter() {  
    @Override  
    public void mouseClicked(MouseEvent e) {  
        frame.dispose();  
    }  
});
```



Fausto M. Ayres

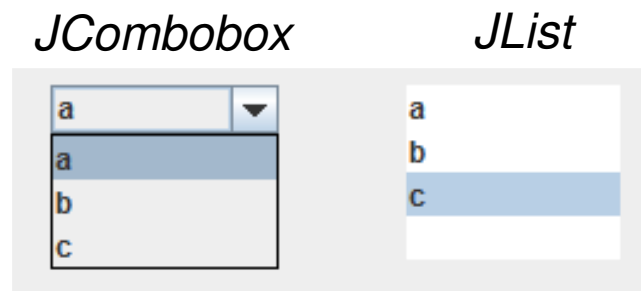
37

## COMPONENTES DE LISTA

38

# Componentes JList e JComboBox

- Exibem dados na forma de lista
- Esses dados são manipulados separadamente

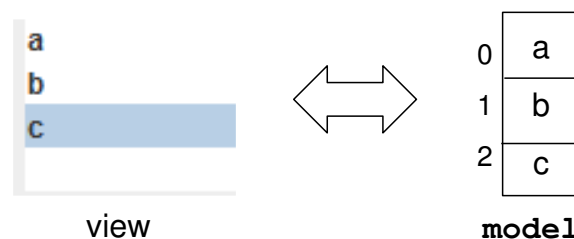


Fausto M. Ayres

39

## Separação View-Model

- Os componentes separam a exibição dos dados (view) da estrutura dos dados (model)



- Classe `DefaultListModel` (`Jlist`)
- Classe `DefaultComboBoxModel` (`JComboBox`)

Fausto M. Ayres

40

# Principais métodos

Método JList, JComboBox	Descrição
<code>componente.getSelectedIndex()</code>	Obtém a posição selecionada
<code>componente.getSelectedItem()</code>	Obtém objeto selecionado no JComboBox
<code>componente.getSelectedValue()</code>	Obtém objeto selecionado no JList
<code>componente.clearSelection()</code>	Limpa a seleção
<code>componente.getModel()</code>	Acessa o model
<code>componente.setModel(model)</code>	Substitui o model

Método do model	Descrição
<code>model.addElement(e)</code>	Adiciona elemento
<code>model.remove(index)</code>	Remove da posição
<code>model.get(index)</code>	Acessa elemento
<code>model.clear()</code>	Remove todos elementos
<code>model.contains(x)</code>	Contém elemento

Fausto M. Ayres

41

## JList

```
//String[] itens = {"a", "b", "c"};
//list = new JList(itens);           // itens fixos

list = new JList();
list.setSelectionMode(ListSelectionMode.SINGLE_SELECTION);

DefaultListModel<String> model = new DefaultListModel<>();
list.setModel(model);

model.addElement("a");
model.addElement("b");
model.addElement("c");

int index = list.getSelectedIndex();
String item = (String) list.getSelectedValue();

model.remove(index);
if (model.contains("c"))
    model.clear();
```

A manipulação dos  
elementos é feita via model

# JComboBox

```
//String[] itens = {"a", "b", "c"};
//combobox = new JComboBox(itens);           //itens fixos

combobox = new JComboBox();
DefaultComboBoxModel<String> model=new DefaultComboBoxModel<>()
combobox.setModel(model);

model.addElement("a");
model.addElement("b");
model.addElement("c");
...
int index = list.getSelectedIndex();
String item = (String) list.getSelectedItem();

model.remove(index);
if (model.contains("c"))
    model.clear();
```

A manipulação dos elementos  
é feita via model

fausto.ayres@ifpb.edu.br

43

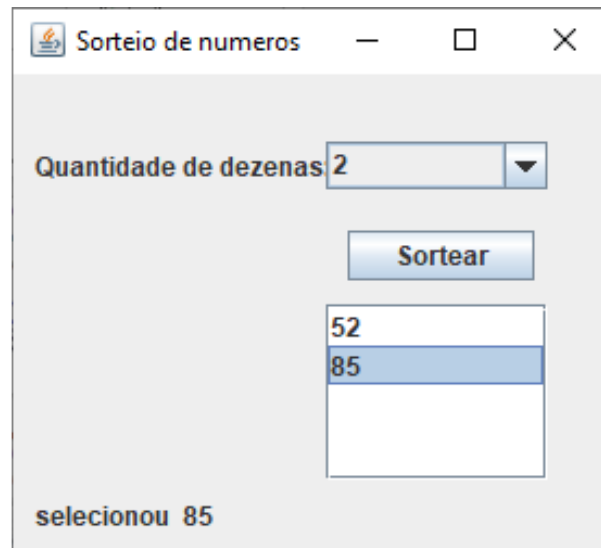
## Seleção de uma linha do JList

- **JList** utiliza um ouvinte **MouseListener** para processar click do mouse (encapsulado num objeto **MouseAdapter**)

```
...
list.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        //int index = list.getSelectedIndex();
        String item = (String) list.getSelectedValue();
        label_1.setText("selecionou "+ item);
    }
});
```

## Exercício

- O usuário escolhe a quantidade de dezenas ( $N=1,2,\dots,10$ ) e o programa sorteia  $N$  dezenas e as exibe numa lista

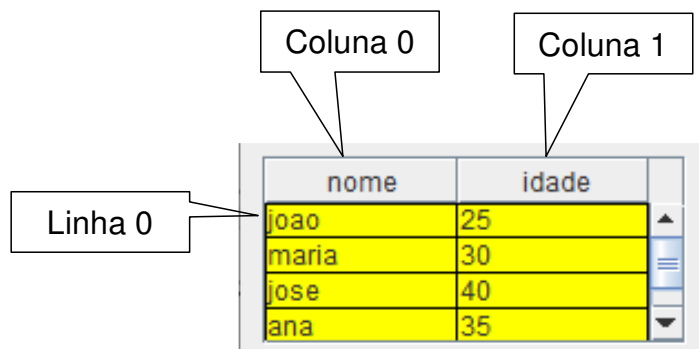


Fausto M. Ayres

45

## Tabela (JTable)

- É um grid (linhas e colunas) cujos dados estão armazenados num **DefaultTableModel**



nome	idade
joao	25
maria	30
jose	40
ana	35

Fausto M. Ayres

46

# Principais métodos JTable

Método de JTable	Descrição
<code>table.getSelectedRow()</code>	Obtém a linha selecionada
<code>table.clearSelection()</code>	Limpa a seleção
<code>table.getModel()</code>	Acessa o model
<code>table.setModel(model)</code>	Substitui o model
<code>table.getValueAt(linha,coluna)</code>	Acessa elemento da linha e coluna

Método do model	Descrição
<code>model.addColumn(titulo)</code>	Cria coluna com titulo
<code>model.addRow(Object[ ])</code>	Cria linha com os objetos
<code>model.removeRow(index)</code>	Remove a linha
<code>model.getRowCount()</code>	Quantidade de linhas
<code>model.getColumnCount()</code>	Quantidade de colunas

Fausto M. Ayres

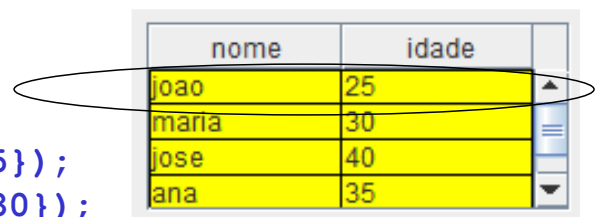
47

## JTable

```
JTable table = new JTable();
table.setRowSelectionAllowed(true);
table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
table.setBorder(new LineBorder(Color.BLACK));
table.setShowGrid(true);
...
```

```
DefaultTableModel model = new DefaultTableModel();
table.setModel(model);
```

```
model.addColumn("nome");
model.addColumn("idade");
model.addRow(new Object[]{"joao", 25});
model.addRow(new Object[]{"maria", 30});
...
```



```
String nome = (String) table.getValueAt(0, 0); //lin 0,col 0
int idade = (Integer) table.getValueAt(0, 1); //lin 0,col 1
```

Fausto M. Ayres

48



## Seleção de uma linha da tabela

- **JTable** utiliza um ouvinte **MouseListener** para processar click do mouse (encapsulado num objeto **MouseAdapter**)

```
...
table.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        int index = table.getSelectedRow();
        label.setText("selecionou a linha" +index);
    }
});
```

## Bloquear edição de células da tabela

- Sobrescreve-se o método `isCellEditable()` na criação da tabela

```
table = new JTable() {
    public boolean isCellEditable(int rowIndex, int vColIndex) {
        return false;
    }
};
```