Linguagem de Programação I

Aula 11
Introdução ao Desenvolvimento de Aplicações com Interfaces Gráficas e Componentes GUI

l.bertholdo@ifsp.edu.br

Conteúdo

- Aplicações Gráficas
- Frameworks e Bibliotecas
- Exemplo de Aplicação GUI com Swing
  - Componentes de Interface
  - Eventos
  - Caixas de Diálogo
- Menus

Aplicações Gráficas

- Aplicações gráficas, também são chamadas de aplicações GUI (*Graphical User Interface*), disponibilizam ao usuário uma interface padronizada, simples e intuitiva.

- Este tipo de aplicação fornece diversos componentes de interface necessários para interação entre a aplicação e o usuário como: caixas de texto, caixas de seleção, botões, etc.

- Por meio de seus *frameworks*, AWT (*Abstract Window Toolkit*) e Swing, e da biblioteca JavaFX, a plataforma Java ofecere recursos para criação de aplicações gráficas baseada em janelas que podem ser executadas em diferentes sistemas operacionais.

Frameworks e Bibliotecas

- **AWT** foi o primeiro *framework* de componentes de interface da plataforma Java. As classes para criação de aplicações AWT se encontram definidas no pacote **java.awt**.
- Seu objetivo era oferecer um conjunto de componentes e recursos que pudessem ser usados em todas as plataformas suportadas pelo Java, garantindo o máximo de portabilidade.
- Uma aplicação AWT utiliza elementos nativos do sistema operacional em que é executada, ou seja, o botão de uma aplicação AWT usa o componente “botão” equivalente da plataforma.
- Assim, mesmo que a aparência de um componente possa variar ligeiramente, o comportamento durante a operação é o mesmo, independentemente do sistema operacional.

Frameworks e Bibliotecas

- Embora consiga gerar aplicações com aparência e recursos similares para diferentes plataformas, o AWT possui algumas limitações.
- Para usar os componentes nativos do sistema operacional, o AWT implementa um mecanismo complexo, que torna a comunicação entre o sistema operacional e a JVM bastante pesada.
- Outro ponto negativo do AWT é o número relativamente pequeno de componentes disponíveis.

Frameworks e Bibliotecas

- Para superar as limitações do AWT, foi desenvolvido o *framework* **Swing**. As classes para criação de aplicações Swing se encontram definidas no pacote **java.swing**.
- Além de fornecer um conjunto de componentes maior, o Swing possui componentes 100% Java.
- Essa característica faz com que uma aplicação Swing não precise usar componentes nativos do sistema operacional, melhorando a performance da aplicação.
- Tanto no AWT quanto no Swing, todos os componentes são representados por instâncias de classes. Por exemplo: para caixas de texto usa-se a classe **JTextField**, para botões comuns é usada a classe **JButton**, e assim por diante.

Frameworks e Bibliotecas

- Em 2007, a Sun Microsystems lançou uma linguagem semelhante ao JavaScript chamada **JavaFX Script**. Seu objetivo era fornecer recursos mais avançados e fáceis de implementar para criação de aplicações gráficas *desktop* e, posteriormente, *web* e *mobile*.

- Em 2011, a Oracle, que havia adquirido a Sun, lançou a versão 2.0 do **JavaFX.** Desde então, o JavaFX passou a ser uma biblioteca do Java, escrita totalmente nesta linguagem.

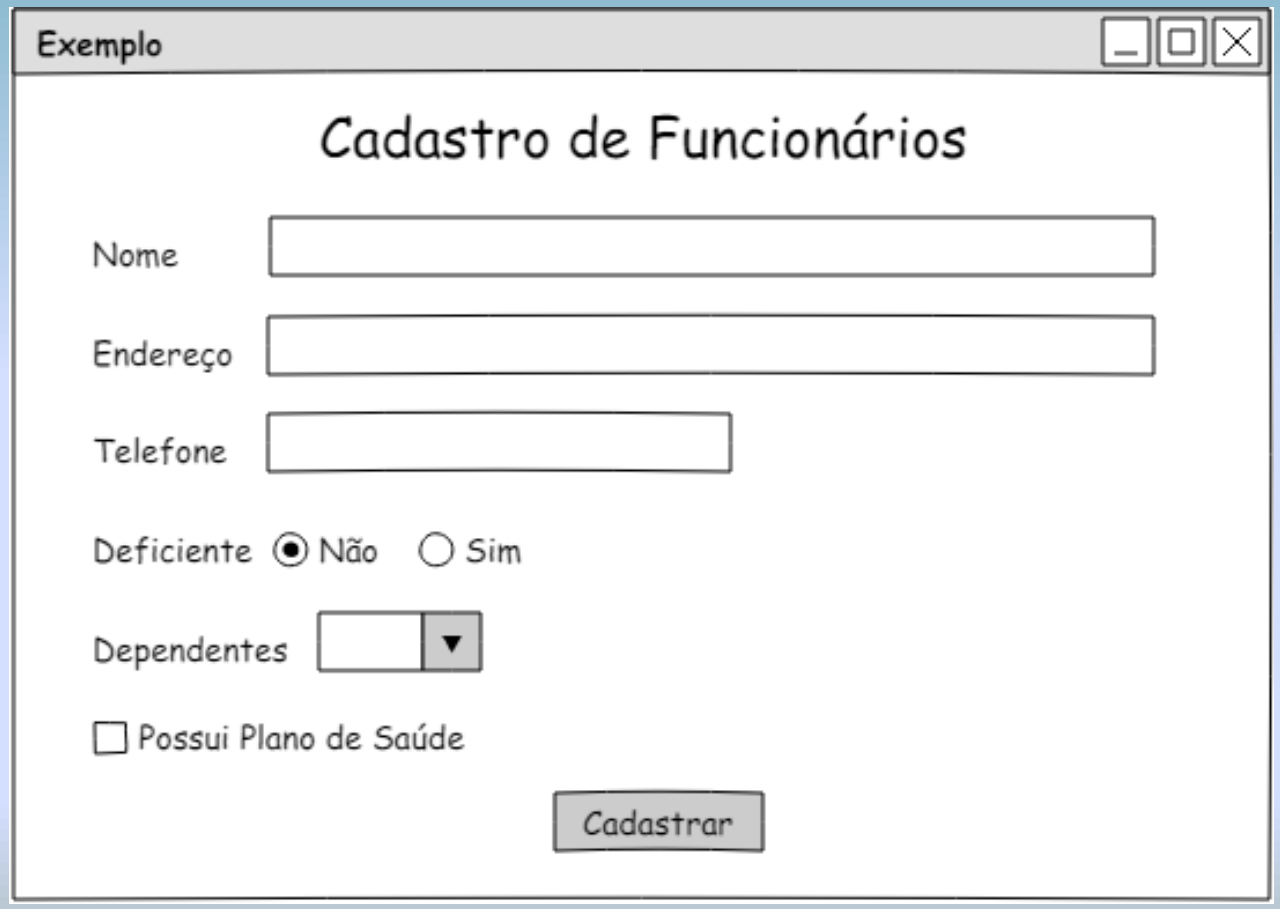
Entre os objetivos da tecnologia JavaFX estão a melhor organização de código, a manutenção mais rápida e a qualidade gráfica proporcionada pelo uso de conceitos de folhas de estilo (CSS – Cascading Style Sheets).

Exemplo de Aplicação GUI com Swing

- A criação de uma aplicação Swing envolve alguns passos:
  1. Desenhar os protótipos das janelas, incluindo todos os componentes de interface.
  2. Para cada janela, criar uma subclasse da classe **JFrame**.
  3. Declarar os respectivos **componentes** de cada janela em sua respectiva classe.
  4. Implementar os **construtores** de cada classe para instanciar, configurar, posicionar e adicionar seus componentes de interface, além de declarar os respectivos eventos de cada componente.
  5. Implementar os **métodos referentes aos eventos** declarados, codificando as ações que devem ser executadas.
  6. Incluir o método **main** na classe correspondente à janela inicial da aplicação.

Exemplo de Aplicação GUI com Swing

1. Desenhando o protótipo da janela.



Exemplo de Aplicação GUI com Swing

2. Criando uma subclasse da classe **JFrame**.

```
import javax.swing.*;

@SuppressWarnings("serial")
public class Exemplo extends JFrame {

}
```

A classe **JFrame** implementa uma janela, na qual serão incluídos todos os outros componentes que permitirão a interação do usuário com a aplicação.

A classe **JFrame** é serializável, ou seja, seus objetos podem ser preservados para futura recuperação, mesmo após o encerramento da aplicação. Ao herdar da classe **JFrame**, a classe **Exemplo** também passa a ser serializável. Esta anotação apenas indica que o compilador deve ignorar a versão de serialização da classe **Exemplo**.

Exemplo de Aplicação GUI com Swing

3. Declarando os componentes da janela na classe.

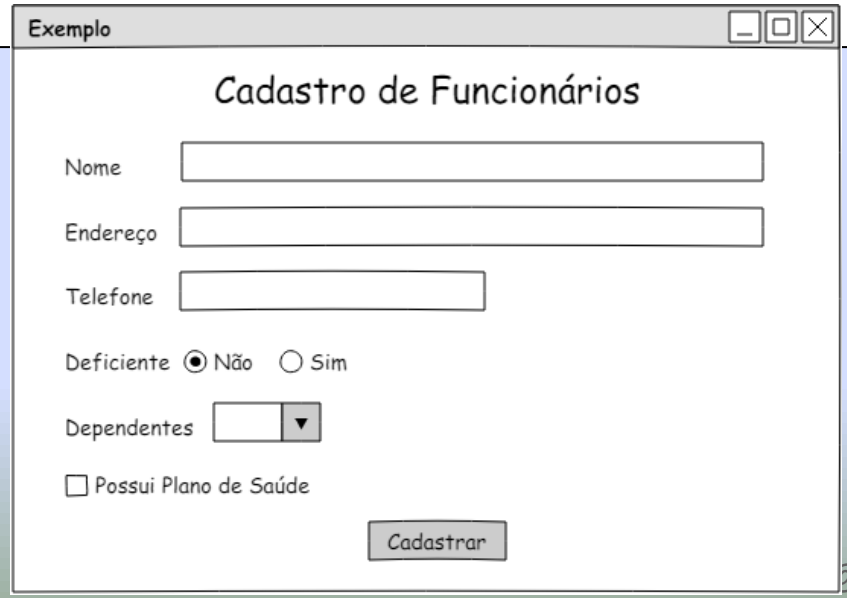
Principais Componentes / Classes	Descrição
JLabel	Texto descritivo para nomes de campos, títulos, legendas, figuras e instruções ao usuário.
JTextField	Campo para entrada de texto. Os dados podem ser fornecidos pelo usuário ou pelo próprio programa.
JRadioButton	Botões de seleção utilizados em grupo, que permitem selecionar uma opção dentre várias previamente configuradas.
ButtonGroup	Componente não gráfico usado para agrupar JRadioButtons.
JComboBox<T>	Campo que permite a seleção de um item dentre vários previamente configurados, sendo **T** o tipo de dado dos itens do campo.
JCheckBox	Caixa de verificação que permite ao usuário indicar se uma determinada opção é verdadeira (caixa marcada) ou falsa (caixa desmarcada).
JButton	Componente que permite disparar uma determinada ação dentro de um programa.

Exemplo de Aplicação GUI com Swing

3. Declarando os componentes da janela na classe.

```
@SuppressWarnings("serial")
public class Exemplo extends JFrame {
    private JLabel lbTitulo, lbNome, lbEndereco, lbTelefone, lbDeficiente, lbDependentes;
    private JTextField tfNome, tfEndereco, tfTelefone;
    private static final String valDef[] = {"Não", "Sim"}; // Valores dos JRadioButtons.
    private JRadioButton rbDeficiente[];
    private ButtonGroup bgDeficiente; // Grupo de botões para agrupar os JRadioButtons.
    private JComboBox<Integer> cbDependentes;
    private static final Integer valDep[] = {1, 2, 3, 4}; // Valores do JComboBox.
    private JCheckBox ckPlanoSaude;
    private JButton btCadastrar;
    private Container cp; // Container para organizar os componentes na janela.
}
```

Em uma aplicação Swing, os nomes dos componentes de interface devem ter uma **convenção**. Não existe uma regra, mas é comum usar duas letras minúsculas para indicar o tipo de componente antes de seu nome.



Exemplo de Aplicação GUI com Swing

4. Implementando o construtor da classe – Instanciando componentes

```
public Exemplo() { // Construtor.
    // Instanciação dos componentes de interface.
    lbTitulo = new JLabel("Cadastro de Funcionários");
    lbNome = new JLabel("Nome");
    lbEndereco = new JLabel("Endereço");
    lbTelefone = new JLabel("Telefone");
    lbDeficiente = new JLabel("Deficiente");
    lbDependentes = new JLabel("Dependentes");
    tfNome = new JTextField();
    tfEndereco = new JTextField();
    tfTelefone = new JTextField();
    rbDeficiente = new JRadioButton[2];
    bgDeficiente = new ButtonGroup();
    cbDependentes = new JComboBox<>(valDep);
    ckPlanoSaude = new JCheckBox("Possui Plano de Saúde");
    btCadastrar = new JButton("Cadastrar");
```

Alguns tipos de componentes de interface permitem que seu texto seja definido por meio do construtor da classe do componente.

Exemplo de Aplicação GUI com Swing

4. Implementando o construtor da classe – Configurando componentes

Cada componente de interface possui um conjunto de métodos, que permitem configurar suas características conforme a necessidade. Por exemplo: cor, tamanho, fonte, posição na tela, etc.

```
// Configuração dos componentes de interface.
setTitle("Exemplo"); // Título da janela.
setSize(500, 300); // Tamanho da janela em pixels.
setLocationRelativeTo(null); // Centraliza a janela na tela.
// Ao fechar a janela, libera todos os recursos usados por ela.
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
lbTitulo.setFont(new Font("Arial", Font.BOLD, 19)); // Ajusta a fonte do JLabel.
for (int i = 0; i < rbDeficiente.length; i++){
    // Adiciona os valores "Não" e "Sim" aos JRadioButtos.
    rbDeficiente[i] = new JRadioButton(valDef[i]);
    rbDeficiente[i].setBackground(new Color(180, 205, 205)); // Cor de fundo dos JRadioButtons.
    bgDeficiente.add(rbDeficiente[i]); // Adiciona os JRadioButtons ao ButtonGroup.
}
rbDeficiente[0].setSelected(true); // Faz com que o JRadioButton "Não" fique marcado.
ckPlanoSaude.setBackground(new Color(180, 205, 205)); // Cor de fundo do JCheckBox.
btCadastrar.setToolTipText("Cadastra o funcionário."); // Inclui uma dica no JButton.
cp = getContentPane(); // Instancia o container da janela.
cp.setLayout(null); // Configura o layout do container como nulo.
cp.setBackground(new Color(180, 205, 205)); // Configura a cor de fundo do container.
```

Métodos do componente **JFrame**.

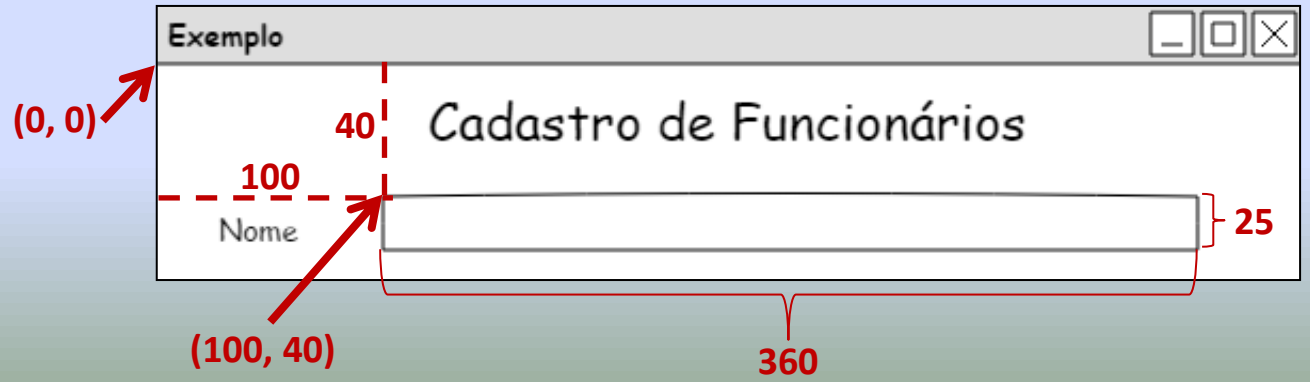
l.bertholdo@ifsp.edu.br

Exemplo de Aplicação GUI com Swing

4. Implementando o construtor da classe – Posicionando componentes

```
// Posicionamento dos componentes de interface na janela.
lbTitulo.setBounds(135, 10, 300, 25); // x, y, largura, altura.
lbNome.setBounds(20, 40, 100, 25);
tfNome.setBounds(100, 40, 360, 25);
lbEndereco.setBounds(20, 70, 100, 25);
tfEndereco.setBounds(100, 70, 360, 25);
lbTelefone.setBounds(20, 100, 100, 25);
tfTelefone.setBounds(100, 100, 120, 25);
lbDeficiente.setBounds(20, 130, 100, 25);
rbDeficiente[0].setBounds(100, 130, 50, 25);
rbDeficiente[1].setBounds(150, 130, 50, 25);
lbDependentes.setBounds(20, 160, 100, 25);
cbDependentes.setBounds(100, 160, 50, 25);
ckPlanoSaude.setBounds(16, 190, 250, 25);
btCadastrar.setBounds(200, 220, 100, 25);
```

Como não foi definido um gerenciador de *layout* para o *container*, é preciso indicar a posição (x, y) e o tamanho (largura e altura) em pixels de cada componente, considerando como base o canto superior esquerdo do componente.



l.bertholdo@ifsp.edu.br

Exemplo de Aplicação GUI com Swing

4. Implementando o construtor da classe – Adicionando componentes

```
// Adição dos componentes de interface ao container.
cp.add(lbTitulo);
cp.add(lbNome);
cp.add(tfNome);
cp.add(lbEndereco);
cp.add(tfEndereco);
cp.add(lbTelefone);
cp.add(tfTelefone);
cp.add(lbDeficiente);
cp.add(rbDeficiente[0]);
cp.add(rbDeficiente[1]);
cp.add(lbDependentes);
cp.add(cbDependentes);
cp.add(ckPlanoSaude);
cp.add(btCadastrar);
```

l.bertholdo@ifsp.edu.br

Exemplo de Aplicação GUI com Swing

4. Implementando o construtor da classe – Declarando eventos
   - Eventos
     - Quando o usuário interage com a aplicação, o sistema operacional envia uma **mensagem** à JVM informando o **tipo de interação**. Exemplos: clique em um botão, alteração no conteúdo de um campo de texto, escolha de um valor em uma caixa de seleção etc.
     - Essas mensagens são denominadas **eventos**, que são convertidos em objetos Java pela JVM e entregues ao **processador de eventos** da aplicação, para que execute alguma tarefa.
     - Os processadores de eventos (*event listeners*) são **interfaces** que possuem métodos específicos para receber e tratar eventos.
     - Qualquer **componente de interface**, capaz de interagir com o usuário, possui um ou mais *event listeners* associados.

Exemplo de Aplicação GUI com Swing

4. Implementando o construtor da classe – Declarando eventos

- Eventos – Exemplo

```
btLimpar.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                btLimparAction();
        }
});

private void btLimparAction() {
        tfUsuario.setText("");
        tfSenha.setText("");
}
```

new ActionListener *cria uma classe anônima (sem nome) que implementa o método* ***actionPerformed*** *da interface* ***ActionListener****. Com isso, o método* ***addActionListener*** *recebe uma instância desta classe anônima. Evita-se assim a declaração de uma classe que implementa a interface* ***ActionListener****.*

- Aqui o método **addActionListener** da classe **JButton** recebe como argumento uma instância (*event listener*) do tipo **ActionListener**.
- Essa instância é criada por meio de uma **classe anônima** que implementa o método **actionPerformed** da interface **ActionListener**.
- O método **actionPerformed**, por sua vez, recebe uma instância de **ActionEvent** (que contém as informações sobre o evento ocorrido) e chama o método **btLimparAction**, que executa a tarefa desejada.

Exemplo de Aplicação GUI com Swing

4. Implementando o construtor da classe – Declarando eventos
   - Alguns tipos de eventos:

Classes de Eventos	Ocorre quando:	Interfaces (*event listeners*)	Métodos das Interfaces	Exemplos de Componentes
ActionEvent	É realizada alguma ação em um componente. Por exemplo, acionar a tecla Enter em um JTextField.	ActionListener	actionPerformed (ActionEvent)	JTextField, JButton, JCheckBox, JComboBox, JRadioButton, JMenuItem
CaretEvent	A posição do cursor de um componente de texto foi alterada.	CaretListener	caretUpdate (CaretEvent)	JTextField, JTextArea
ChangeEvent	O componente passou por alguma alteração. Por exemplo, um JCheckBox é marcado.	ChangeListener	stateChanged (ChangeEvent)	JButton, JCheckBox, JRadioButton, JMenuItem
FocusEvent	O componente ganha ou perde o foco.	FocusListener	focusGained (FocusEvent) focusLost (FocusEvent)	JTextField, JButton, JCheckBox, JComboBox, JRadioButton

Exemplo de Aplicação GUI com Swing

4. Implementando o construtor da classe – Declarando eventos
   - Alguns tipos de eventos:

Classes de Eventos	Ocorre quando:	Interfaces	Métodos das Interfaces	Exemplos de Componentes
ItemEvent	O componente passou por alguma alteração. Por exemplo, um valor de um JComboBox é selecionado.	ItemListener	itemStateChanged(ItemEvent)	JButton, JCheckBox, JComboBox, JRadioButton, JMenuItem
KeyEvent	Uma tecla é acionada enquanto o foco está sobre o componente.	KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)	JTextField, JButton, JCheckBox, JComboBox, JRadioButton
MouseEvent	O mouse é acionado sobre o componente ou o cursor do mouse é passado sobre o componente.	MouseListener	mouseClicked(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent)	JTextField, JButton, JCheckBox, JComboBox, JRadioButton

Exemplo de Aplicação GUI com Swing

4. Implementando o construtor da classe – Declarando eventos

```
    // Declaração do processador de evento referente à seleção do radiobutton Não.
    rbDeficiente[0].addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) { rbNaoAction(); }
    });

    // Declaração do processador de evento referente à seleção do radiobutton Sim.
    rbDeficiente[1].addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) { rbSimAction(); }
    });

    // Declaração do processador de evento referente à alteração do combobox Dependentes.
    cbDependentes.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) { cbDependentesAction(); }
    });

    // Declaração do processador de evento referente à alteração do checkbox Plano de Saúde.
    ckPlanoSaude.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) { ckPlanoSaudeAction(); }
    });

    // Declaração do processador de evento referente ao clique no botão Cadastrar.
    btCadastrar.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) { btCadastrarAction(); }
    });
} // Final do construtor.
```

Exemplo de Aplicação GUI com Swing

5. Implementando os métodos chamados pelos eventos declarados.
   - **Caixas de Diálogo** – Recurso que permite emitir mensagens informativas, ou solicitar dados ou alguma decisão do usuário.
   - Uma das classes que permite criar estas mensagens no Swing é a **JOptionPane**. Para exibir as mensagens são usados os métodos:
     - **showMessageDialog –** Exibe uma janela para informar o usuário contendo apenas um botão OK.
     - **showConfirmDialog –** Exibe uma janela de confirmação com um ou mais botões (Sim, Não, Cancelar).
     - **showInputDialog –** Exibe uma janela para entrada de dados.
     - **showOptionDialog –** Exibe uma janela contendo botões e textos personalizados.

Exemplo de Aplicação GUI com Swing

5. Implementando os métodos chamados pelos eventos declarados.

```
private void rbNaoAction() { // Exibe uma caixa de diálogo.
    JOptionPane.showMessageDialog(this, // 1º) JFrame responsável pela caixa de diálogo.
                                  "Opção 'Não' selecionada.", // 2º) Mensagem a ser exibida.
                                  "Informação", // 3º) Título da caixa de diálogo.
                                  JOptionPane.INFORMATION_MESSAGE); // 4º) Ícone da caixa de diálogo.
}

private void rbSimAction() { // Exibe uma caixa de diálogo.
    JOptionPane.showMessageDialog(this, "Opção 'Sim' selecionada.",
                                  "Informação", JOptionPane.INFORMATION_MESSAGE);
}

private void cbDependentesAction() { // Exibe uma caixa de diálogo.
    JOptionPane.showMessageDialog(this, "Item " + cbDependentes.getSelectedItem() + " selecionado.",
                                  "Informação", JOptionPane.INFORMATION_MESSAGE);
}

private void ckPlanoSaudeAction() { // Exibe uma caixa de diálogo.
    if (ckPlanoSaude.isSelected())
        JOptionPane.showMessageDialog(this, "Checkbox marcado.",
                                      "Informação", JOptionPane.INFORMATION_MESSAGE);
    else
        JOptionPane.showMessageDialog(this, "Checkbox desmarcado.",
                                      "Informação", JOptionPane.INFORMATION_MESSAGE);
}
```

Exemplo de Aplicação GUI com Swing

5. Implementando os métodos chamados pelos eventos declarados.

```
private void btCadastrarAction() { // Exibe uma uma caixa de diálogo contendo os dados informados.
    String ps = "", df = "";
    if (ckPlanoSaude.isSelected()) // Verifica se o JCheckBox está marcado ou não.
        ps = "Sim";
    else
        ps = "Não";

    for (JRadioButton rb : rbDeficiente) // Recupera o texto do JRadionButton selecionado.
        if (rb.isSelected())
            df = rb.getText();

    int resposta = JOptionPane.showConfirmDialog(this, // 1º) JFrame responsável pela caixa de diálogo.
                                                 // 2º) Mensagem a ser exibida na caixa de diálogo.
                                                 "DADOS INFORMADOS: \n" +
                                                 "Nome: " + tfNome.getText() + "\n" +
                                                 "Endereço: " + tfEndereco.getText() + "\n" +
                                                 "Telefone: " + tfTelefone.getText() + "\n" +
                                                 "Deficiente: " + df + "\n" +
                                                 "Dependentes: " + cbDependentes.getSelectedItem() + "\n" +
                                                 "Plano de Saúde: " + ps + "\n\n" +
                                                 "Deseja informar outro funcionário?",
                                                 "Informação", // 3º) Título da caixa de diálogo.
                                                 JOptionPane.YES_NO_OPTION, // 4º) Botões da caixa de diálogo.
                                                 JOptionPane.QUESTION_MESSAGE); // 5º) Ícone da caixa de diálogo.
    if (resposta == 0) // Se resposta for igual a 0 (Sim).
        JOptionPane.showMessageDialog(this, "Você informou Sim.", "Resposta", JOptionPane.INFORMATION_MESSAGE);
    else if (resposta == 1) // Se resposta for igual a 1 (Não).
        JOptionPane.showMessageDialog(this, "Você informou Não.", "Resposta", JOptionPane.INFORMATION_MESSAGE);
}
```

Exemplo de Aplicação GUI com Swing

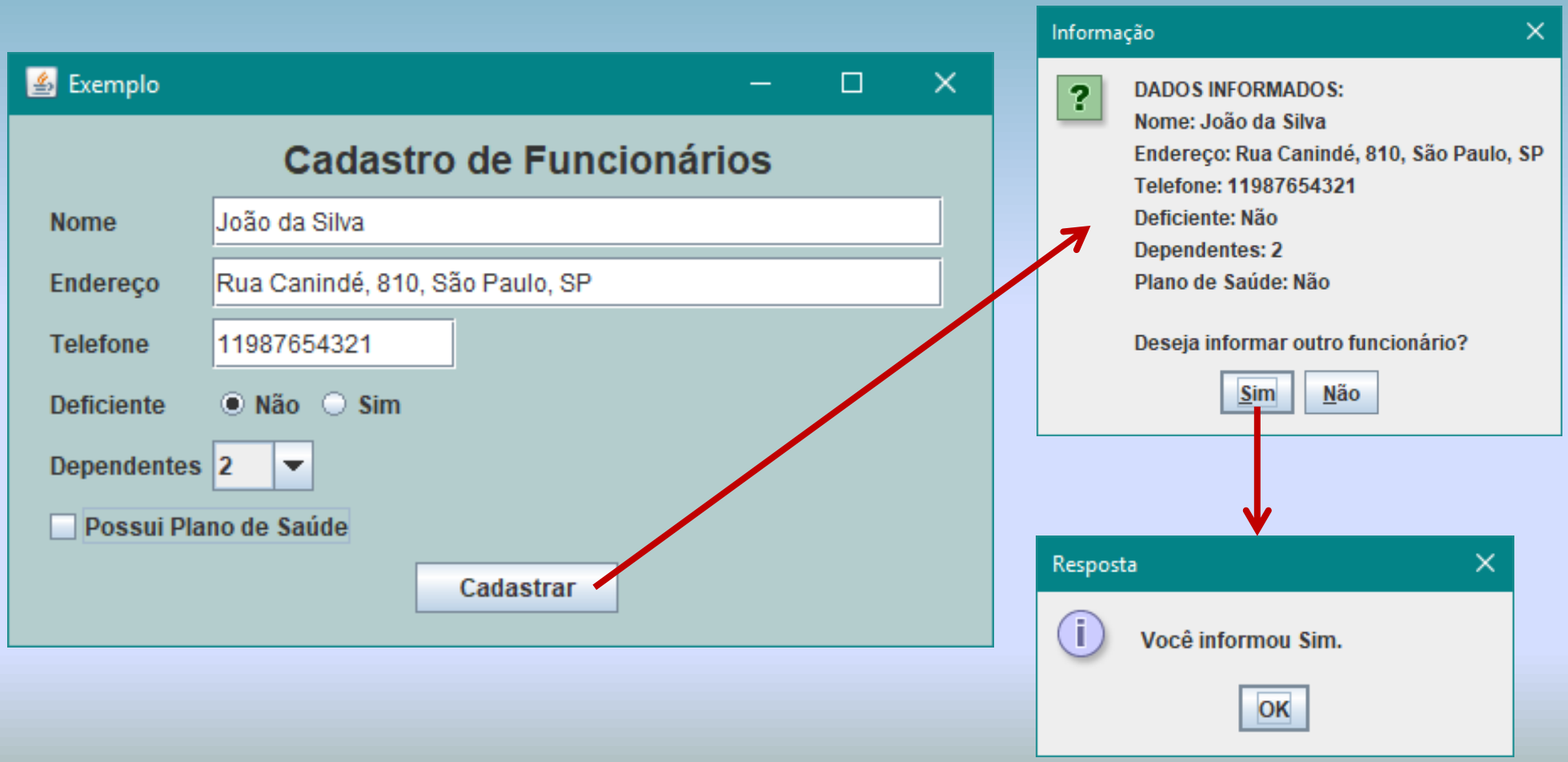
6. Incluir o método **main**.

O método **main** deve ser adicionado apenas à classe responsável pela janela principal da aplicação.

```
public static void main(String[] args) { // Início da aplicação.
    // O método invokeLater da classe SwingUtilities é usado para que a instanciação
    // e exibição da janela sejam feitas de forma assíncrona (em momentos diferentes).
    // Assim, a JVM pode processar os eventos da janela de forma adequada.
    SwingUtilities.invokeLater(new Runnable(){
        @Override
        public void run(){ // Implementa o método run da interface Runnable.
            // "new Exemplo()" instancia a janela e "setVisible(true)" exibe a janela.
            new Exemplo().setVisible(true);
        }});
}
```

Exemplo de Aplicação GUI com Swing

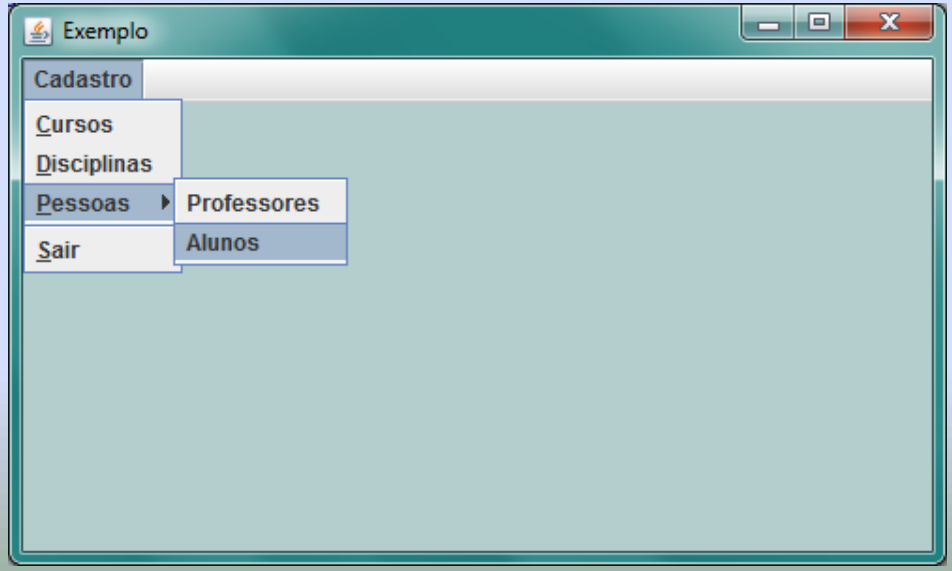
- Aplicação em execução.



l.bertholdo@ifsp.edu.br

Menus

- O Swing oferece três componentes principais para criação de menus em uma janela.
  - **JMenuBar –** Usado para criar uma barra de menus na janela.
  - **JMenu –** Usado para criar menus em uma barra de menus, ou para criar menus dentro de um outro menu.
  - **JMenuItem –** Usado para criar itens dentro de um menu.



l.bertholdo@ifsp.edu.br

Menus

- Exemplo

```
@SuppressWarnings("serial")
public class Exemplo2 extends JFrame {
    private JMenuBar mbBarra; // Barra do menu.
    private JMenu mnCadastro; // Menu Cadastro.

    // Itens e Submenus do menu Cadastro.
    private JMenuItem miCadCurso; // Item Cursos do menu Cadastro.
    private JMenuItem miCadDisciplina; // Item Disciplinas do menu Cadastro.
    private JMenu mnCadPessoa; // Submenu Pessoas do menu Cadastro.
    private JMenuItem miCadPessoa[]; // Itens do submenu Pessoas.
    // Valores dos itens do submenu Pessoas.
    private static final String cadPesItens[] = {"Professores", "Alunos"};
    private JMenuItem miCadSair; // Item Sair do menu Cadastro.

    public Exemplo2() { // Construtor.
        // Instanciação dos componentes de interface.
        mbBarra = new JMenuBar();
        mnCadastro = new JMenu("Cadastro");
        miCadCurso = new JMenuItem("Cursos");
        miCadDisciplina = new JMenuItem("Disciplinas");
        mnCadPessoa = new JMenu("Pessoas");
        miCadPessoa = new JMenuItem[2];
        miCadSair = new JMenuItem("Sair");
```

Menus

- Exemplo

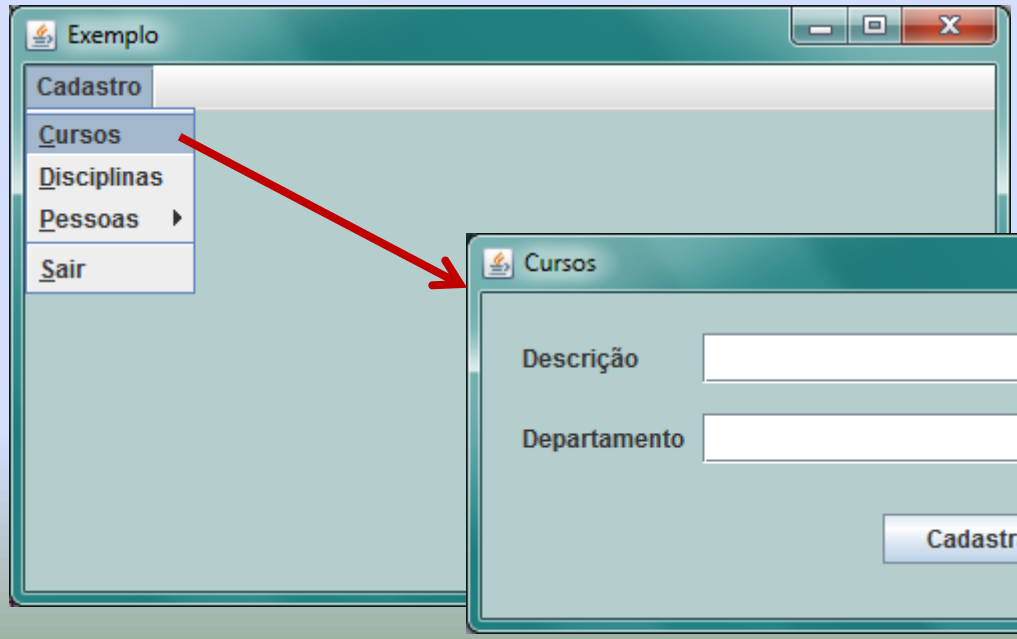
```
    // Configuração dos componentes de interface.
    setTitle("Exemplo"); // Título da janela.
    setSize(500, 300); // Tamanho da janela em pixels.
    setLocationRelativeTo(null); // Centraliza a janela na tela.
    // Ao fechar a janela, libera todos os recursos usados pela aplicação.
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    getContentPane().setBackground(new Color(180, 205, 205)); // Configura a cor de fundo do container.
    miCadCurso.setMnemonic('C'); // Associa uma tecla de atalho ao item Cursos.
    miCadDisciplina.setMnemonic('D'); // Associa uma tecla de atalho ao item Disciplinas.
    mnCadPessoa.setMnemonic('P'); // Associa uma tecla de atalho ao menu Pessoas.
    miCadSair.setMnemonic('S'); // Associa uma tecla de atalho ao item Sair.
    for (int i = 0; i < miCadPessoa.length; i++){
        miCadPessoa[i] = new JMenuItem(cadPesItens[i]); // Configura os valores dos itens do menu Pessoas.
        mnCadPessoa.add(miCadPessoa[i]); // Adiciona os itens ao menu Pessoas.
    }
    mnCadastro.add(miCadCurso); // Adiciona o item Cursos ao menu Cadastro.
    mnCadastro.add(miCadDisciplina); // Adiciona o item Disciplinas ao menu Cadastro.
    mnCadastro.add(mnCadPessoa); // Adiciona o menu Pessoas ao menu Cadastro.
    mnCadastro.addSeparator(); // Adiciona uma linha separadora ao menu.
    mnCadastro.add(miCadSair); // Adiciona o item Sair ao menu Cadastro.
    mbBarra.add(mnCadastro); // Adiciona o menu Cadastro à barra de menus.
    setJMenuBar(mbBarra); // Adiciona a barra de menus à janela.

    // Declaração do processador de evento referente ao clique no item Cursos.
    miCadCurso.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) { miCadCursoAction(); }
    });
} // Final do construtor.
```

Menus

- Exemplo

```
    private void miCadCursoAction() {
        SwingUtilities.invokeLater(new Runnable(){
            @Override
            public void run(){ new Curso().setVisible(true); }});
    }
    public static void main(String[] args) { // Início da aplicação.
        SwingUtilities.invokeLater(new Runnable(){
            @Override
            public void run(){ new Exemplo2().setVisible(true); }});
    }
}
```

É preciso usar **JDialog** para que a janela possa ser **modal** (janela que não permite acesso a outras janelas abertas), além do método **setModal(true)**.




```
@SuppressWarnings("serial")
public class Curso extends JDialog {
    private JLabel lbDescricao, lbDepartamento;
    private JTextField tfDescricao, tfDepartamento;
    private JButton btCadastrar;
```

```
setModal(true);
```

l.bertholdo@ifsp.edu.br

Referências

- Harvey M. Deitel; Paul J. Deitel; Java Como Programar – 4ª Edição. São Paulo: Pearson Education, 2004.

- Peter Jandl Junior; Java – Guia do Programador – 3ª Edição. São Paulo: Novatec Editora, 2015.