

**E buscar-me-eis, e me achareis, quando me  
buscardes com todo o vosso coração.**

**Jeremias 29:13**

**Engenharia de Computação**

# LINGUAGEM DE PROGRAMAÇÃO ORIENTADA A OBJETOS

- Prof: José Antonio Gonçalves
- [zag655@gmail.com](mailto:zag655@gmail.com)

Nestes slides:

## **Orientação a Objetos em Java:**

Como construir interfaces gráficas com Java **sem a utilização das ferramentas de suporte gráfico da IDE:**

- Construção de Janelas;
- Adição de rótulos, caixas de entrada de dados, botões, menus, submenus e itens de menus à janela;
- Como adicionar funcionalidades por meio da Interface ActionListener.

Pacotes: Swing e AWT (Abstract Window Toolkit)

**Interface Gráfica:**

**Criar Janela e adicionar  
componentes gráficos sem ajuda da  
IDE**

# Criar uma janela: JFrame

```
import javax.swing.JFrame;

public class criaJan {

    public static void main(String args[]){

        JFrame jan1 = new JFrame();
        jan1.setVisible(true);

    }

}
```

# Dimensões de uma janela: JFrame

```
import javax.swing.JFrame;
```

```
public class criaJan {
```

```
    public static void main(String args[]){
```

```
        JFrame jan1 = new JFrame();
```

```
        int larg = 300, alt =250;
```

```
        jan1.setSize(larg,alt);
```

```
        jan1.setVisible(true);
```

```
    }
```

```
}
```

# Definindo o **Título** para uma janela: JFrame

```
import javax.swing.JFrame;
```

```
public class criaJan {
```

```
    public static void main(String args[]){
```

```
        JFrame jan1 = new JFrame();
```

```
        int larg = 300, alt =250;
```

```
        jan1.setSize(larg,alt);
```

```
        jan1.setTitle("Primeira Janela – Form de Entrada de Dados");
```

```
        jan1.setVisible(true);
```

```
    }
```

```
}
```

# Adicionando um **Rótulo** (label) na janela

```
import javax.swing.JFrame;  
import javax.swing.JLabel;
```

```
public class criaJan {
```

```
    public static void main(String args[]){
```

```
        JFrame jan1 = new JFrame();
```

```
        int larg = 300, alt = 250;
```

```
        jan1.setSize(larg,alt);
```

```
        jan1.setTitle("Primeira Janela - Form de Entrada de Dados ");
```

```
        JLabel rot1 = new JLabel();
```

```
        rot1.setText("Rotulo 1 - Nome");
```

```
        jan1.add(rot1);
```

```
        jan1.setVisible(true);
```

```
    }
```

```
}
```



# Adicionando um **Caixa de Texto** na janela: **JTextField**

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class criaJan {
    public static void main(String args[]){
        JFrame jan1 = new JFrame();
        int larg = 300, alt =250;
        jan1.setSize(larg,alt);
        jan1.setTitle("Primeira Janela - Form de Entrada de Dados ");
        JLabel rot1 = new JLabel();
        rot1.setText("Rotulo 1 - Nome");
        jan1.add(rot1);
        JTextField entrada1 = new JTextField(20);
        jan1.add(entrada1);
        jan1.setVisible(true);
    }
}
```

# Organizando o Layout da janela: AWT: FlowLayout()

```
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class criaJan {
    public static void main(String args[]){
        JFrame jan1 = new JFrame();
        int larg = 300, alt =250;
        jan1.setSize(larg,alt);
        jan1.setTitle("Primeira Janela - Form de Entrada de Dados ");
        JLabel rot1 = new JLabel();
        rot1.setText("Rotulo 1 - Nome");
        jan1.add(rot1);
        JTextField entrada1 = new JTextField(20);
        jan1.add(entrada1);
        jan1.setLayout(new FlowLayout());
        jan1.setVisible(true);
    }
}
```

# Adicionando Botões a janela: JButton

```
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;

public class criaJan {
    public static void main(String args[]){
        JFrame jan1 = new JFrame();
        int larg = 300, alt =250;
        jan1.setSize(larg,alt);
        jan1.setTitle("Primeira Janela - Form de Entrada de Dados ");
        JLabel rot1 = new JLabel();
        rot1.setText("Rotulo 1 - Nome");
        jan1.add(rot1);
        JTextField entrada1 = new JTextField(20);
        jan1.add(entrada1);
        JButton bt1 = new JButton("Clique aqui");
        //bt1.setMnemonic('C'); //descomente e veja que traceja o caracter C (teclas de atalho)
        jan1.setLayout(new FlowLayout());
        jan1.setVisible(true);
    }
}
```

Pacotes: Swing e AWT (Abstract Window Toolkit)

## Interface Gráfica: Criar Menus, SubMenus e Itens de menu

# Janelas: Menus

```
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
public class MenuSimples{
    private JMenuBar bm;
    private JMenu menuzaum;
    private JMenu arq;
    private JMenuItem abr;
    public MenuSimples(){
        JFrame jan2 = new JFrame();
        jan2.setTitle("Colocando Menus e SubMenus");
        jan2.setSize(400,400);
        abr = new JMenuItem("abrir"); //criando os Itens-de-Menu para o Menu Arquivo
        abr.setMnemonic('a');
        arq = new JMenu("Arquivo"); //criando o Menu
        arq.setMnemonic('A');
        arq.add(abr); //adicionando Itens-de-Menu ao Menu Arquivo

        //TRABALHANDO COM SUBMENUS
        menuzaum = new JMenu("Menuzaum");//criando menu que conterá o SubMenu Arquivo
        menuzaum.setMnemonic('M');
        menuzaum.add(arq); //adicionando o Menu Arquivo como submenu de Menuzaum
        bm = new JMenuBar(); //criando a Barra de Menus
        bm.add(menuzaum); //adicionando o menu Menuzaum a Barra-de-Menus
        jan2.setJMenuBar(bm); //adicionando à janela (jan2)Barra de Menus com seus submenus
        jan2.setLayout(new FlowLayout()); //Organizando a janela
        jan2.setVisible(true); //tornando a janela visível
    }
    public static void main(String args[]){
        new MenuSimples();
    }
}
```

Adicionando **Funcionalidades** a Botões e Menus:

# **Interface ActionListener**

# Estudo de caso (*contêineres com conteúdo sem evento*)

```
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*.*;
```

```
public class CriaJan {
    private static JTextField entrada1 = new JTextField(20);
    private static JButton bt1 = new JButton("Clique aqui");
    private static JFrame jan1 = new JFrame("Título da Janela");
    private static JLabel rot1 = new JLabel("Rotulo 1 - Nome");
    private static CriaJan cj = new CriaJan();

    public static void main(String args[]){
        int larg = 300, alt =250;
        jan1.setSize(larg,alt);
        jan1.add(rot1);
        jan1.add(entrada1);
        bt1.setMnemonic('C');
        jan1.add(bt1);
        jan1.setLayout(new FlowLayout());
        jan1.setVisible(true);
    }
}
```

*Código da criação de  
Janela (pouco alterado)*

# Interface ActionListener

Uma das formas através das quais conseguimos atribuir funcionalidades a objetos é utilizando a Interface ActionListener .

Logo se é uma Interface **devemos implementar o método contido nela**. Vejamos a assinatura do método:

```
void actionPerformed(ActionEvent e)
```

O parâmetro **ActionEvent** deste método trata-se de um objeto da classe ActionEvent (cujo pacote é: **java.awt.event.ActionEvent** ).

No exemplo apresentado, o parâmetro é “passado” através do método **addActionListener(ActionEvent act\_event)**, caso o objeto, que sofrer alguma ação (um clique do mouse, por exemplo), suporte – mesmo que através de herança – este método.



# Estudo de caso (*adicionando evento aos Botões*)

```
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
```

```
public class CriaJan implements ActionListener{
```

```
    private static JTextField entrada1 = new JTextField(20);
    private static JButton bt1 = new JButton("Clique aqui");
    private static JFrame jan1 = new JFrame("Título da Janela");
    private static CriaJan cj = new CriaJan();
    private static JLabel rot1 = new JLabel("Rotulo 1 - Nome");
```

```
    public static void main(String args[]){
```

```
        int larg = 300, alt = 250;
        jan1.setSize(larg, alt);
        jan1.add(rot1);
        jan1.add(entrada1);
        bt1.setMnemonic('C');
        jan1.add(bt1);
        jan1.setLayout(new FlowLayout());
        jan1.setVisible(true);
```

```
        bt1.addActionListener(cj);
```

```
    } //fim do método main
```

```
@Override
```

```
    public void actionPerformed(ActionEvent evt){
```

```
        Object obj = evt.getSource();
```

```
        if(obj.equals(bt1)){
```

```
            String frase = "É fácil...";
            entrada1.setText(frase);
            jan1.setVisible(true);
```

```
            JOptionPane.showMessageDialog(null, "Você escolheu a opção: "+frase, "Exibição de Dados", JOptionPane.INFORMATION_MESSAGE);
```

```
        } //fim do IF
```

```
    } //fim do método actionPerformed
```

```
} //fim da classe
```

# Estudo de caso (*Menus – sem evento*)

```
import java.awt.FlowLayout;
import javax.swing.*;
```

```
public class MenuSimples{
    private JMenuBar barMenu = new JMenuBar();
    private JMenu menuGeral = new JMenu("Menu Geral");
    private JMenu subMenuArq = new JMenu("Arquivo");
    private JMenuItem itemMenuAbr = new JMenuItem("abrir");
    private JFrame janPrincipal = new JFrame("Colocando Menus e SubMenus");
    private JTextField texto = new JTextField(30);

    public MenuSimples(){
        janPrincipal.setSize(400,400);
        itemMenuAbr.setMnemonic('a');
        subMenuArq.setMnemonic('A');
        subMenuArq.add(itemMenuAbr); //add o ITEM de menu ABR ao SUBMENU ARQ
        menuGeral.setMnemonic('M');
        menuGeral.add(subMenuArq); //add SUBMENU ARQ (item ABR) ao MENU menuGeral
        barMenu.add(menuGeral); //add MENU menuGeral a BARRA DE MENU barMenu
        janPrincipal.setJMenuBar(barMenu); //add BARRA DE MENU barMenu ao JFrame janPrincipal
        janPrincipal.setLayout(new FlowLayout());
        janPrincipal.setVisible(true);
    }
    public static void main(String args[]){
        new MenuSimples();
    }
}
```

*Código de "menu"  
(pouco alterado)*

# Estudo de caso (*Menus – adicionado evento*)

```
import java.awt.FlowLayout;
import javax.swing.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class MenuSimples implements ActionListener{
    private JMenuBar barMenu = new JMenuBar();
    private JMenu menuGeral = new JMenu("Menu Geral");
    private JMenu subMenuArq = new JMenu("Arquivo");
    private JMenuItem ItemMenuAbr = new JMenuItem("abrir");
    private JFrame janPrincipal = new JFrame("Colocando Menus e SubMenus");
    private JTextField texto = new JTextField(30);

    public MenuSimples(){
        janPrincipal.setSize(400,400);
        ItemMenuAbr.setMnemonic('a');
        subMenuArq.setMnemonic('A');
        subMenuArq.add(ItemMenuAbr);
        menuGeral.setMnemonic('M');
        menuGeral.add(subMenuArq);
        barMenu.add(menuGeral);
        janPrincipal.setJMenuBar(barMenu);
        janPrincipal.setLayout(new FlowLayout());
        janPrincipal.setVisible(true);
        ItemMenuAbr.addActionListener(this); //invocando o método ActionPerformed
    }
}
```

# Estudo de caso (*Menus – adicionado evento*)

## ...Continuação do slide anterior

```
public MenuSimples(){
    janPrincipal.setSize(400,400);
    ItemMenuAbr.setMnemonic('a');
    subMenuArq.setMnemonic('A');
    subMenuArq.add(ItemMenuAbr);
    menuGeral.setMnemonic('M');
    menuGeral.add(subMenuArq);
    barMenu.add(menuGeral);
    janPrincipal.setJMenuBar(barMenu);
    janPrincipal.setLayout(new FlowLayout());
    janPrincipal.setVisible(true);
    ItemMenuAbr.addActionListener(this); //invocando o método ActionPerformed
}

public void actionPerformed(ActionEvent ativa){
    Object obj = ativa.getSource();
    if(obj.equals(ItemMenuAbr)){
        JOptionPane.showMessageDialog(janPrincipal,"Clicou no menu abrir");
        texto.setText("Não é difícil");
        janPrincipal.add(texto);
        janPrincipal.setVisible(true);
    }
}

public static void main(String args[]){
    new MenuSimples();
}
```

## **Importante**

**Para os exercícios não se preocupe com a posição dos objetos na janela para resolução dos exercícios.**

**Simplesmente use o gerenciador de layout  
FlowLayout (sem parâmetros):**

**“suaJanela.setLayout(new **FlowLayout()** );”**

# Exercícios para fixação...

Crie uma nova aplicação, nesta deverá:

- 1)\_ juntar as características e funcionalidades das duas classes anteriores, a saber: CriaJan e MenuSimples;
- 2)\_ Adicionar mais um botão rótulo “Sair” que, ao ser acionado, permita sair da aplicação;
- 3)\_ Adicionar uma opção na “barra de menus” com o rótulo “Sair” e, idem ao item 2, ao ser acionado também permita abandonar a aplicação;
- 4)\_ “Dentro” do menu geral, adicione uma opção chamada “Calcular” que ao ser acionada:
  - 4.a)\_ Abra, em seqüência, duas caixas de entrada de dados solicitando dois valores numéricos e inteiros
  - 4.b)\_ Os valores obtidos no item 4.a deverão ser somados e exibidos numa caixa de mensagem

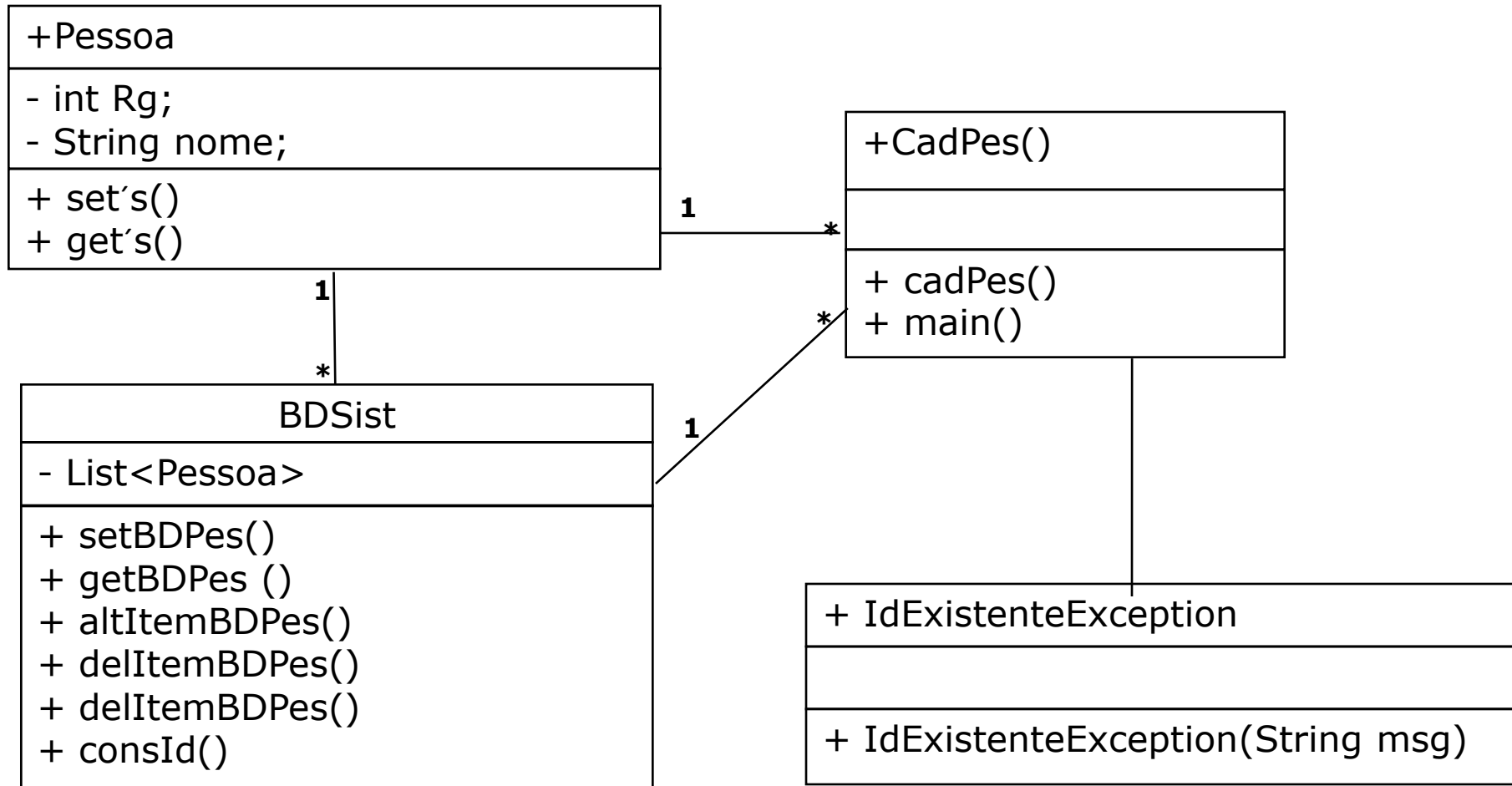
**Nota:**

**Método para sair:**

`suaJanela.setDefaultCloseOperation(jan1.EXIT_ON_CLOSE);` //executa o System exit

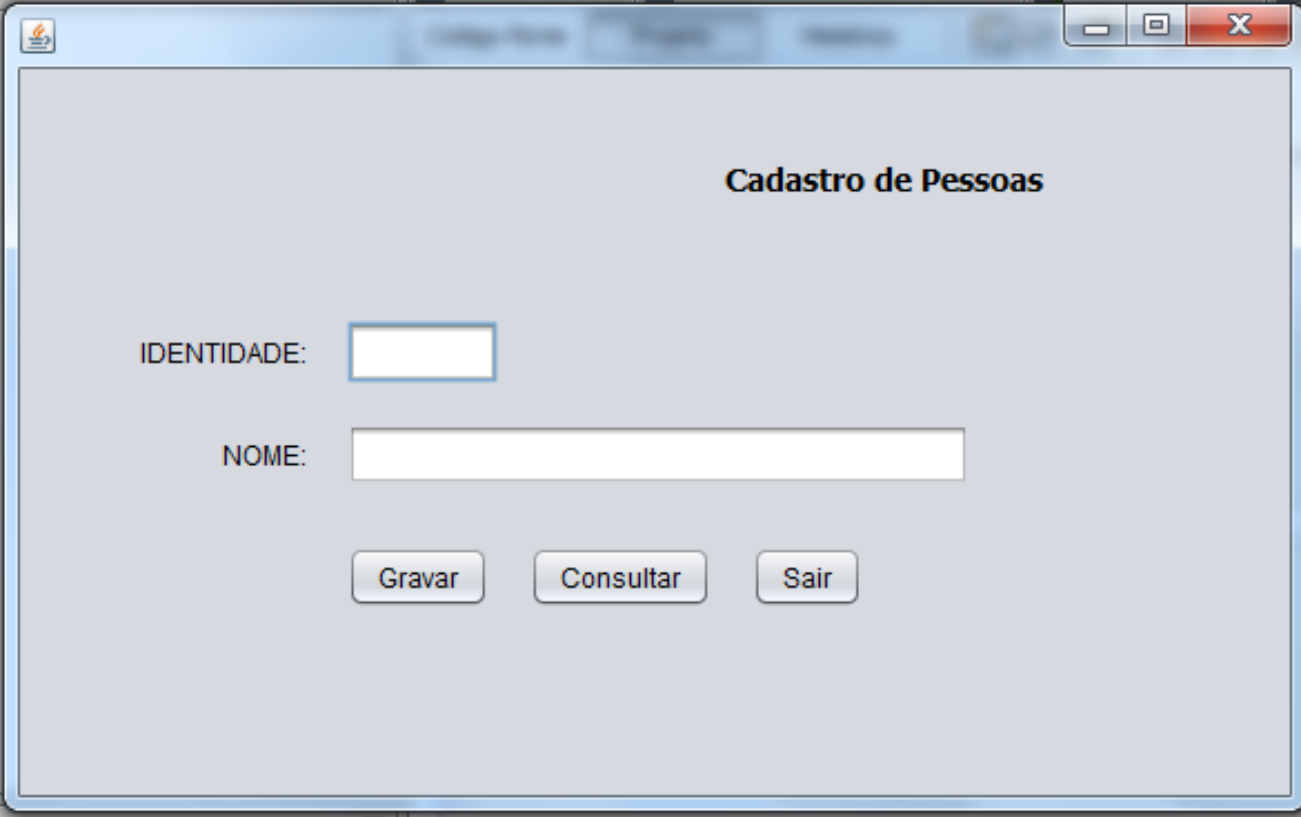
# Exercícios para fixação... continuação

Observe o diagrama a seguir:



# Exercícios para fixação... continuação

Observe a Janela seguir



**Cadastro de Pessoas**

IDENTIDADE:

NOME:



# Exercícios para fixação... continuação

**GRAVAR:** Ao preencher os dois campos e clicar no gravar, os dados serão armazenados no BDSist. Deverá aparecer uma caixa de mensagem avisando.



The image shows a software window titled "Cadastro de Pessoas". It contains two input fields: "IDENTIDADE:" with the value "1" and "NOME:" with the value "Jesus". Below these fields are three buttons: "Gravar", "Consultar", and "Sair". A smaller dialog box is open in the foreground, titled "Cadastro de Pessoa OK", with the message "Pessoa Cadastrada com sucesso...." and an "OK" button.

# Exercícios para fixação... continuação

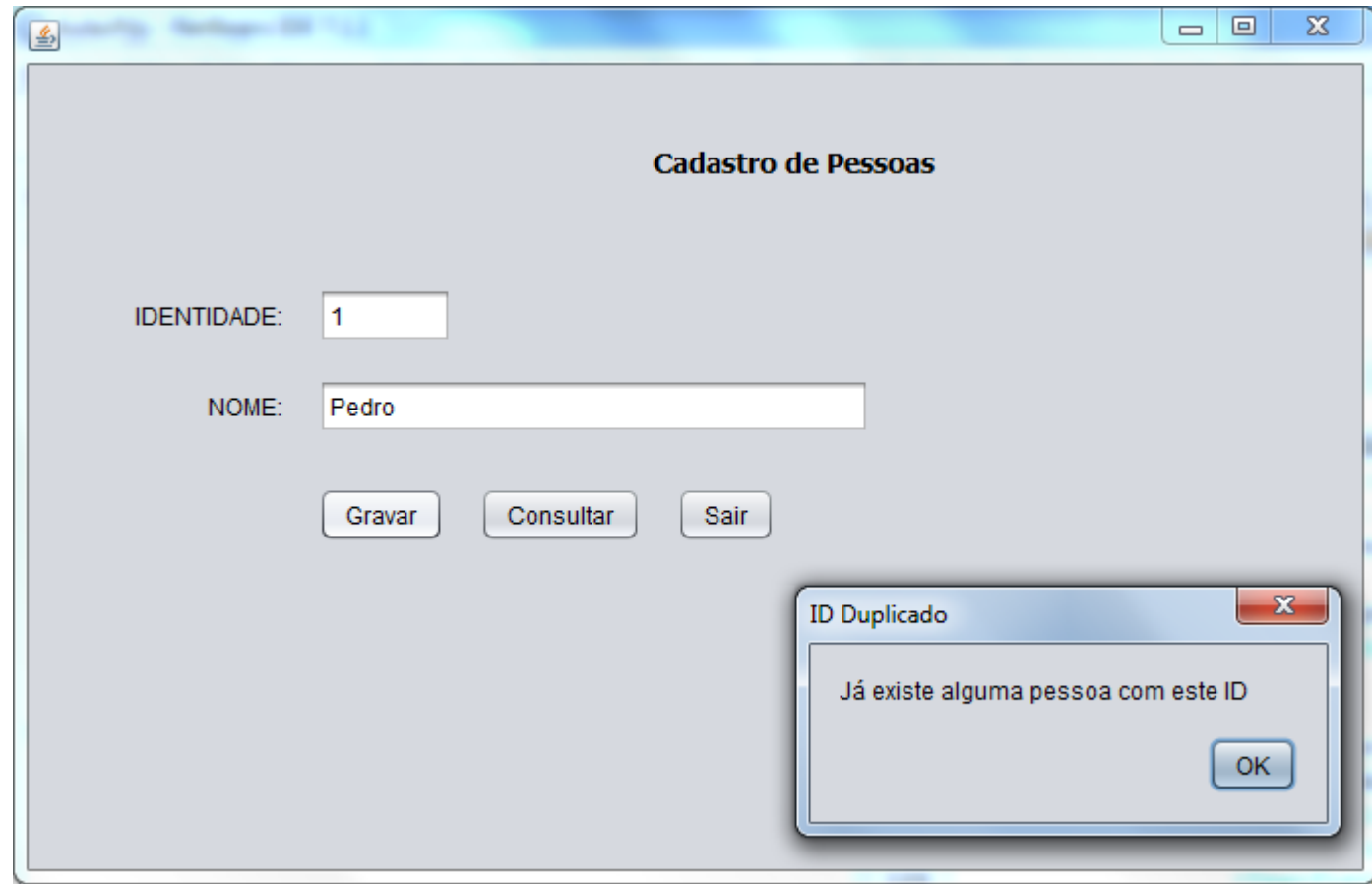
**GRAVAR:** Se ao preencher o campo “identidade” o usuário informar algo diferente de número inteiro :

- 1)\_ Disparará uma exceção que deverá ser tratada;
- 2)\_ Também deverá aparecer uma mensagem informando que “entrou com tipo de dados errado”



# Exercícios para fixação... continuação

**GRAVAR:** Ao preencher os dois campos e clicar no gravar, os dados serão armazenados no BDSist. Porém:  
1)\_ Deve testar se já não existe alguém com este ID. Se existir, deverá apresentar uma mensagem avisando:



# Exercícios para fixação... continuação

## GRAVAR

Ao preencher os dois campos e clicar no **gravar**, os dados serão armazenados no BDSist. Porém:

1)\_ Deve testar se já não existe alguém com este ID. Se existir, a classe *CadPes*, deverá tratar a exceção

**Para isso:**

1.a)\_ na classe BDSist, crie o método ***public Pessoa consId(Pessoa p)*** que recebe uma Pessoa e compara (em todo List) se existe outra pessoa com o mesmo Id. **Se achar retorna a pessoa** do List (a que encontrou), **se não, retorna null**;

1.b)\_ ainda na classe *BDSist*, crie o método ***setBDPes()***: Caso encontre alguém com o mesmo ID, deverá **disparar a exceção**.

Para tanto crie a classe *IdExistenteException* com seu método construtor. No construtor deverá receber e imprimir uma mensagem (que já existe uma pessoa com este id) passada por parâmetro.

1.c)\_ Na classe *CadPes*, onde foi chamado o método *setBDpes(Pessoa p)*, deverá capturar e tratar a exceção.

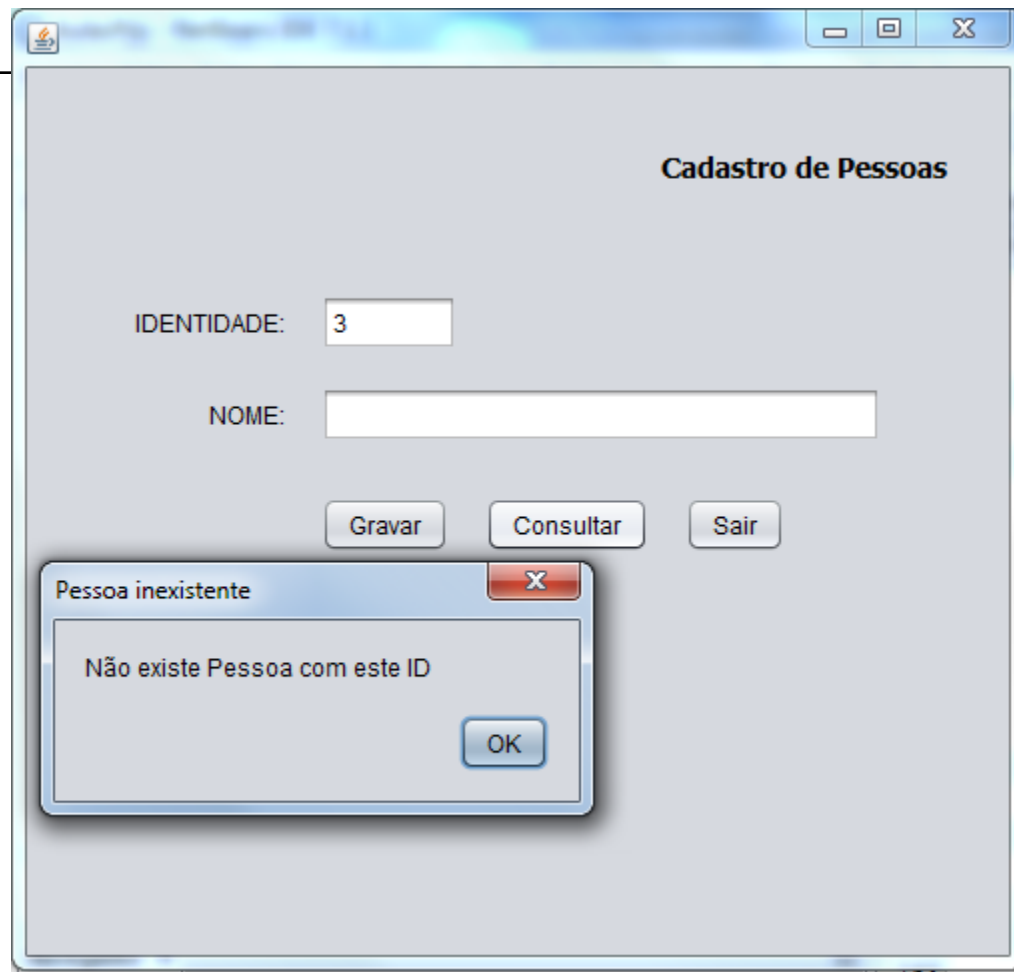
**CONSULTAR:** Ao preencher o campo identidade e clicar no consultar, caso exista alguma pessoa com o numero de identidade (ID) igual no BDPes, seu nome deverá ser exibido no campo nome

**SAIR:** Ao clicar no sair, deverá abandonar a aplicação

# Exercícios para fixação... continuação

## CONSULTAR

Ao preencher o campo identidade e clicar no **consultar**, caso exista alguma pessoa com o numero de identidade (ID) igual no BDPes, seu nome deverá ser exibido no campo nome. Para tanto use o método: `public Pessoa consId(Pessoa p)` da classe `BDSist` e imprima os dados do objeto de retorno de método. Caso não haja ninguém, deverá aparecer uma mensagem avisando:



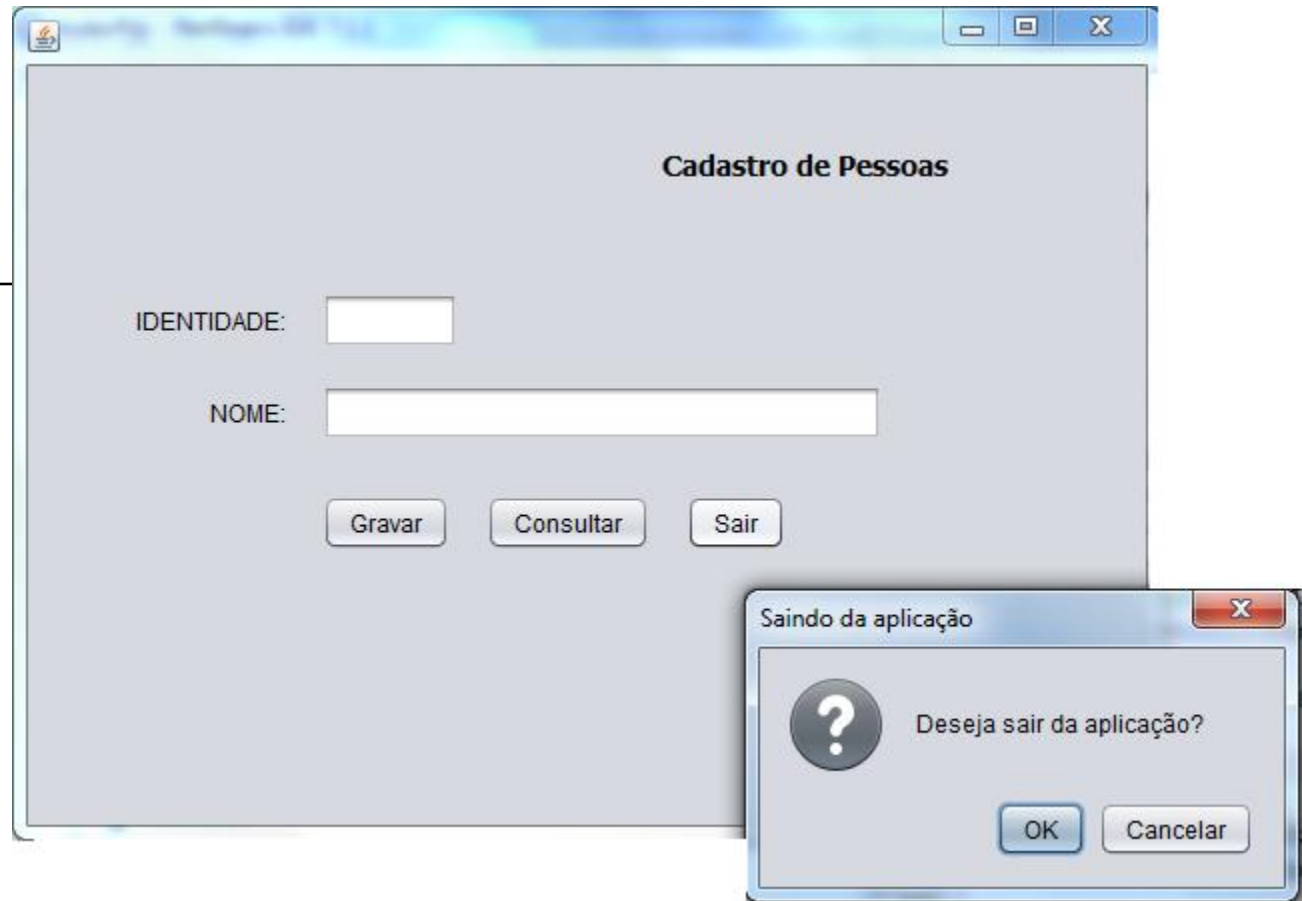
# Exercícios para fixação... continuação

## SAIR

Ao clicar no **sair**, deverá apresentar uma caixa de confirmação. Caso o usuário opte por cancelar deverá continuar na aplicação. Se não, deverá abandoná-la:

Pode usar:

*System.exit(0);*



# Outras Interfaces

# Outras Interfaces

## focusListener

**void focusGained (FocusEvent e):** Invocado quando um componente ganha o foco do teclado

**void focusLost (FocusEvent e):** Invocado quando um componente perde o foco do teclado.



# Outras Interfaces

## KeyListener

**void keyPressed (KeyEvent e):** Chamada quando uma tecla foi pressionada.

**void keyReleased (KeyEvent e):** Chamada quando uma tecla foi liberada.

**void keyTyped (KeyEvent e):** Chamada quando uma tecla foi digitado

# Outras Interfaces

## MouseListener

**Void mouseClicked (MouseEvent e):** Chamado quando o botão do mouse foi clicado (pressionado e liberado) em um componente.

**void mouseEntered (MouseEvent e):** Chamado quando o mouse entra um componente.

**void mouseExited (MouseEvent e):** Chamada quando o mouse sai de um componente.

**void mouseReleased (MouseEvent e):** Chamada quando um botão do mouse foi lançado em um componente.

# Outras Interfaces

## WindowListener

**void windowActivated (WindowEvent e):** Chamado quando a janela estiver definida para ser a janela ativa.

**void windowClosed (WindowEvent e):** Invocado quando uma janela tiver sido fechada como o resultado da chamada `dispose` sobre a janela.

**void windowClosing void (WindowEvent e):** Chamado quando o usuário tenta fechar a janela do menu da janela do sistema.

**void windowDeactivated (WindowEvent e):** Chamado quando uma janela já não é a janela ativa.

**void windowDeiconified (WindowEvent e):** Chamado quando uma janela é alterado a partir de uma reduzida a um estado normal.

**void windowIconified (WindowEvent e):** Invocado quando uma janela é alterada de um normal para um estado minimizado.

**void windowOpened (WindowEvent e) :** Invocada pela primeira vez, uma janela é tornada visível.

## Referências Bibliográficas:

- DEITEL, H.; DEITEL, P. JAVA – Como Programar. 3.ed. Porto Alegre: Bookman, 2001.
- ECKEL, B. Thinking in Java , 2nd edition, EUA: Prentice Hall, 2000.
- HORSTMANN, C. Core Java – Advanced Features. EUA: Prentice Hall, 2000. Volume II.
- HORSTMANN, C. Core Java – Fundamentals. EUA: Prentice Hall, 2000. Volume I.
- <http://docs.oracle.com/javase/6/docs/api> (em 14/06/2012)
- <http://www.guj.com.br> (em 14/06/2012)
- <http://www.caelum.com.br> (em 14/06/2012)
- <http://www.argonavis.com.br>(em 14/06/2012)