Interface Gráfica

Prof. Marcelo Cohen

(material da Profa. Luciana Nedel)

1

Graphical User Interface (GUI)

- **Existe uma infinidade de funcionalidades disponíveis nas bibliotecas de classes Java, destinadas a prover a comunicação homem-máquina gráfica.**
- **Os elementos básicos necessários para criar um GUI residem em dois pacotes:**
  - **java.awt: Abstract Windowing Toolkit (classes básicas);**
  - **javax.swing: Swing Components - fornece melhores alternativas aos componentes definidos na classe java.awt. Exemplo: estudaremos a classe JButton do Swing no lugar da classe Button, do java.awt.**

2

Graphical User Interface (GUI)

- As classes Swing são parte de um conjunto mais genérico de capacidades gráficas, chamado de Java Foundation Classes, ou JFC.

- JFC suporta:
  - definição de botões, menus, etc.
  - desenho 2D (java.awt.geom)
  - funcionalidades drag-and-drop (java.awt.dnd)
  - API com acessibilidade a usuários (javax.accessibility)

- Swing é mais flexível que java.awt porque é implementada toda em Java, enquanto que java.awt é implementada em código nativo.

3

Graphical User Interface (GUI)

Exemplo 1:

```
import javax.swing.*;
public class TestaJanela
{   // Objeto Janela
 public static void main (String args[])
 {
    static JFrame janela = new JFrame("Título da janela");
    janela.setBounds(50, 100, 400, 150); // Seta posicao e tamanho
    janela.setDefaultCloseOperation( WindowConstants.
                                        DISPOSE_ON_CLOSE);
    janela.setVisible(true);  // Exibe a janela
 }
}
```

Argumentos do método setDefaultCloseOperation:
DISPOSE_ON_CLOSE - Destróia a janela
DO_NOTHING_ON_CLOSE - Desabilita opção
HIDE_ON_CLOSE - Apenas fecha a janela

4

Exercícios

1) Teste a classe exemplo com os diferentes argumentos para o método setDefaultCloseOperation().

2) Faça um trecho de programa que anime uma janela, variando sua posição e tamanho.

5

Containers e componentes

- Uma interface gráfica em Java é baseada em dois elementos:
  - containers: servem para agrupar e exibir outros componentes
  - componentes: botões, labels, scrollbars, etc.
- Dessa forma, todo programa que ofereça uma interface vai possuir pelo menos um container, que pode ser:
  - JFrame: janela principal do programa
  - JDialog: janela para diálogos
  - JApplet: janela para Applets

6

Containers e componentes

- **Para construirmos uma interface gráfica em JAVA, adicionamos componentes (Botões, Menus, Textos, Tabelas, Listas, etc.) sobre a área da janela.**
- **Por essa razão a área da janela é um container, ou seja, um elemento capaz de armazenar uma lista de componentes.**

7

Containers e Componentes

Exemplo 2:

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

public class TestaContainer
{
 public static void main (String args[ ])
 {
   int i;
   JFrame janela = new JFrame("Título da janela");
   janela.setBounds(50, 100, 400, 150); // Seta posição e tamanho
   janela.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

    FlowLayout flow = new FlowLayout();     // Define o layout do
  container
   Container caixa = janela.getContentPane(); // Define o tamanho
   caixa.setLayout(flow);   // Seta layout do container
   for (i=1; i<=6; i++)
      caixa.add(new JButton("Aperte " + i));  // Adiciona um botão
   janela.setVisible(true);  // Exibe a janela
 }
}
```

8

Exercícios

3) Redimensione interativamente a janela da aplicação e observe o comportamento dos botões da interface.

4) Troque o argumento de FlowLayout(), para FlowLayout (FlowLayout.LEFT) e observe.

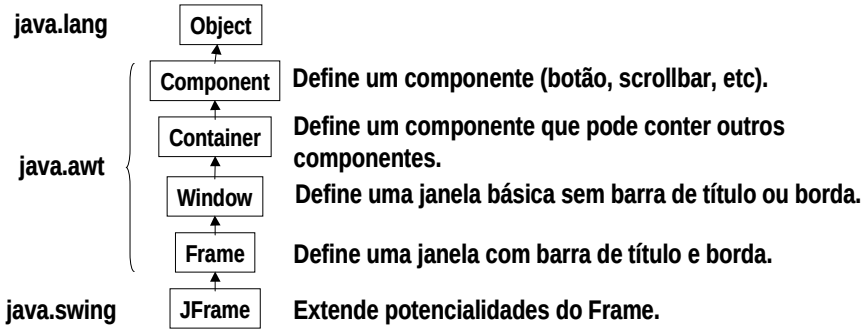
5) Adicione no programa acima, os seguintes componentes:

```
JLabel label = new JLabel("Exemplo de texto:");
caixa.add(label);
JTextField campo = new JTextField(15);
caixa.add(campo);
janela.pack(); // Redimensiona a janela
```

9

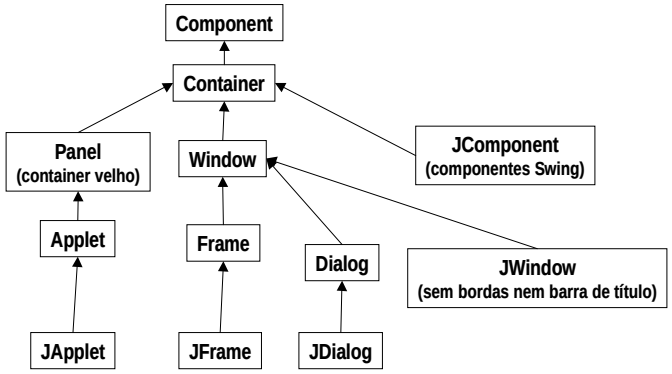
Criando uma janela

- Uma janela em Java é representada por um objeto da classe Window.



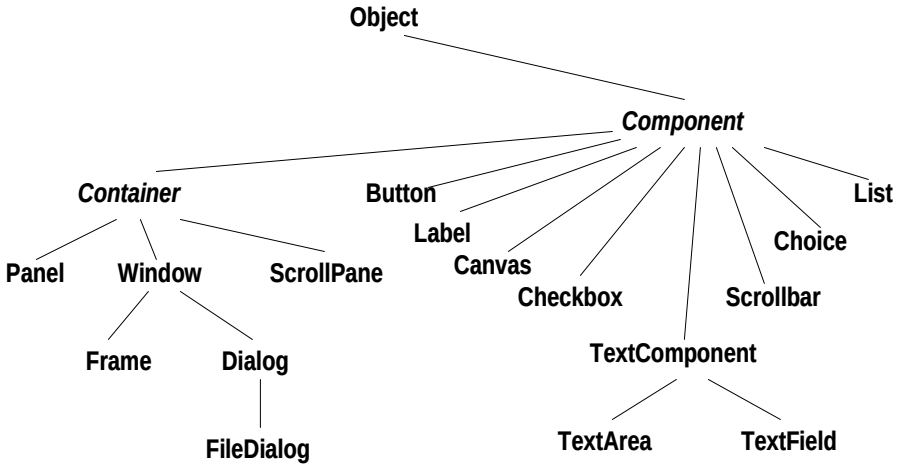
10

Criando uma janela



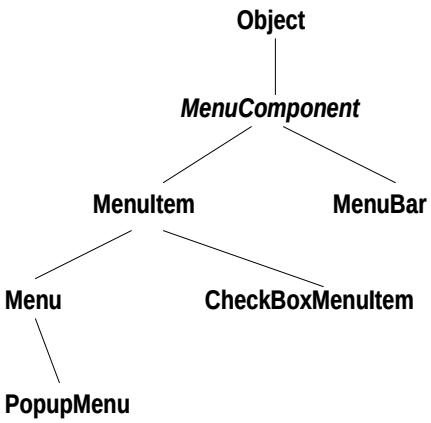
11

Classes de AWT (1)



12

Classes de AWT (2)



13

Package Swing

- Criado em 1997
- Extensão da AWT - Abstract Window Toolkit
- Classes implementadas inteiramente em Java
- Mesma estrutura que os componentes AWT
- Componentes que fornecem melhores alternativas para a implementação de interfaces gráficas
  - JButton no lugar de Button,
  - JFrame no lugar de Frame, etc.
- As classes de Swing fazem parte de um conjunto mais genérico de classes com capacidades gráficas: JFC (Java Foundation Classes)

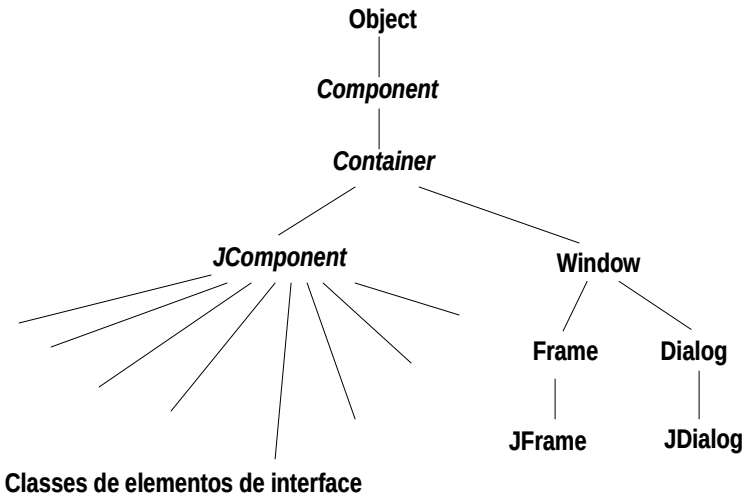
14

Componentes Swing

- **Componentes swing são divididos em:**
  - **Visuais:**
    - **botões, menus, barras de ferramentas, etc.**
  - **Não-visuais, de auxílio aos outros:**
    - **root pane, panel, layered pane, etc.**
- **Pacote javax.swing**

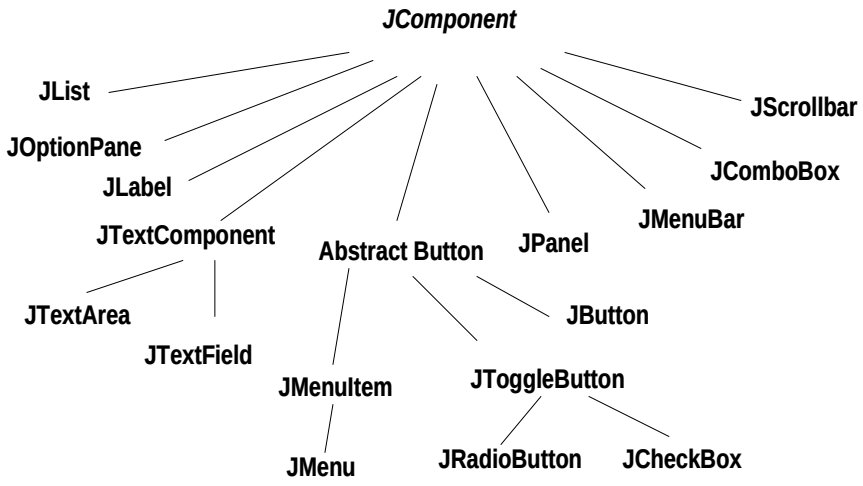
15

Classes de Swing



16

Hierarquia de componentes



17

Complexidade da hierarquia

- Algumas classes da hierarquia são bastante simples, outras são complexas
- JFrame, por exemplo:
  - 2 construtores, 236 métodos
    - 23 métodos declarados
    - 17 métodos herdados de Frame
    - 19 métodos herdados de Window
    - 44 métodos herdados de Container
    - 124 métodos herdados de Component
    - 9 métodos herdados de Object

18

Complexidade da hierarquia

- **Não é possível dominar as facilidades da linguagem com a mesma velocidade com que se domina os comandos de uma linguagem procedural**
- **Uso da documentação da API durante a programação é rotineiro e essencial**

19

Containers e Componentes

- **Desde que a classe Container é uma sub-classe da classe Component, todo objeto container é também um Componente.**
- **A diferença básica entre um objeto JFrame e um objeto Window é que o JFrame representa a janela principal de uma aplicação.**
- **Antes de criar um objeto Window, é preciso criar um objeto JFrame.**
- **Como a classe JDialog deriva diretamente da classe Window, pode-se criar objetos JDialog somente no contexto de um objeto JFrame.**

20

Containers e Componentes

- **JFrame: Um objeto desta classe possui uma barra de título e características para receber menus e outros componentes.**
- **JDialog: Usada para definir janelas de diálogo para entrada de dados. Normalmente usada em resposta a uma opção de menu selecionada. Definida em função de um objeto JFrame.**
- **JApplet: Classe base para applets Java. É possível desenhar e adicionar menus e outros componentes em um JApplet.**
- **JComponent: As subclasses de JComponent definem um conjunto de componentes standard (menus, botões, checkboxes, etc.)**

21

Containers e Componentes

- **Alguns atributos de componentes:**
  - **posição (x,y): posição do objeto em relação ao seu container;**
  - **nome do componente (myWindow.setName("Teste"););**
  - **tamanho: altura e largura;**
  - **cor do objeto e cor de fundo;**
  - **fonte**
  - **aparência do cursor;**
  - **objeto habilitado ou não (isEnabled(), myWindow.setEnabled);**
  - **objeto visível ou não (isVisible(), myWindow.setVisible);**
  - **objeto válido ou não.**
- **Todos atributos são private.**

22

Containers e Componentes

- **Tamanho e posição de um componente:**
  - **posição é definida por x, y (tipo int) ou por um objeto do tipo Point. Um objeto Point possui dois argumentos public int (x e y);**
  - **dimensões definidas por width e height (tipo int) ou por um objeto do tipo Dimension (dois argumentos public int - width e height).**
  - **Tamanho e posição também podem ser definidos juntos pelo objeto Rectangle (4 argumentos public int - x, y representando o topo à esquerda e width e height definindo o tamanho).**

23

Containers e Componentes

- **Exemplos de métodos:**
  - **void setBounds(int x, int y, int width, int height);**
  - **void setBounds(Rectangle rect);**
  - **Rectangle getBounds();**
  - **void setSize(Dimension d);**
  - **Dimension getSize();**
  - **setLocation(int x, int y);**
  - **setLocation(Point p);**
  - **Point getLocation();**

24

Containers e Componentes

- Outros componentes disponíveis:
  - Button (JButton)
  - Menu (JMenu)
  - Text Component (JTextComponent)
  - List (JList)
  - Table (JTable)
  - Container

25

Uma janela simples...

- Execute a classe abaixo e veja o que acontece...

```
import javax.swing.*;
class SimpleFrame
{
 public static void main(String args[ ])
 {
  JFrame frame = new JFrame("Swing Application");
  JButton but = new JButton("I am a Swing button");
  JLabel texto = new JLabel("Number of button clicks: 0");
  JPanel painel = new JPanel( );
  painel.add(but);
  painel.add(texto);
  frame.getContentPane( ).add(painel);
  frame.pack( );
  frame.show( );
 }
}
```

26

Uma janela simples…

- A aplicação do exercício anterior possui os seguintes elementos:
  - JFrame: armazena os demais componentes
  - JPanel: painel, serve para facilitar o posicionamento do botão e do label
  - JButton: o botão "I am a Swing button"
  - JLabel: o texto "Number of button clicks: 0"
- JFrames são top-level containers: sempre estão presentes
- JPanels são intermediate containers: podem estar ou não presentes (mas geralmente estão)
- JButton e JLabel são componentes atômicos: não podem ser usados para conter e normalmente respondem ao usuário

27

Uma janela simples...

- Questões ainda não respondidas:
  - Como organizar os componentes em um JPanel?
  - Java oferece diversos layouts para estruturação de componentes
  - Por exemplo, para JPanel o layout default é FlowLayout, que distribui os componentes na horizontal
- Mas, e se quisermos distribuir de outro modo?
  - Basta trocar o layout por outro!

28

Entendendo Layouts

- **Já foi visto que interfaces em JAVA são construídas na medida em que adicionamos Components a Containers.**
- **Os Containers são responsáveis então por manter os componentes visíveis, repassar os eventos, etc.**
- **Como a filosofia da linguagem JAVA é de que os programas sejam extremamente portáveis, a filosofia da interface visa também ser extremamente adaptável.**
- **Por essa razão a disposição dos Components sobre o Container não é indicada por um par ordenado (x,y) como na maioria das bibliotecas de construção de interface com o usuário (MFC - Microsoft, OWL - Borland etc).**

29

Entendendo Layouts

- **É possível definir seus próprios Layouts, mas a linguagem oferece um conjunto de Layouts básicos que simplificam o trabalho.**
- **O arranjo dos componentes no container é gerenciado por um LayoutManager**
  - **A vantagem da existência de um LayoutManager é que a apresentação dos componentes se adapta quando do redimensionamento da janela**
  - **A desvantagem é o pouco domínio que o programador tem da posição dos componentes com alguns LayoutManagers**

30

FlowLayout

- Os components são distribuídos da esquerda para a direita e de cima para baixo

```
Panel c =new Panel( );
c.add(new Button("1"));
c.add(new TextField(9));
c.add(new Button("dois"));
c.add(new Button("três"));
```

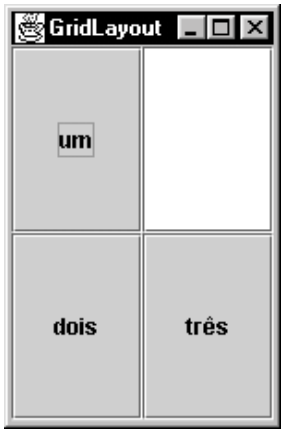
- Respeita o tamanho preferido dos componentes mesmo quando não houver espaço suficiente no container

31

GridLayout

- Similar ao FlowLayout, mas cada component é alocado em uma célula de igual tamanho. Permite definir um vetor ou matriz de células nas quais os componentes são alocados.

```
Panel c =new Panel();
c .setLayout(new GridLayout(2,2));
c.add(new Button("um"));
c.add(new TextField(5));
c.add(new Button("dois"));
c.add(new Button("três"));
```



- Divide a área em uma grade
- Dispõe os elementos da esquerda para a direita e de cima para baixo
- Todos tem o mesmo tamanho

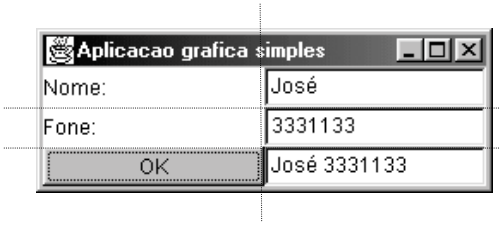
32

Exemplo

```
public class AplicacaoGrafica extends Frame{
  private Button butOK;
  private TextField campo1,campo2,resp;
  private Label texto1,texto2;

 public AplicacaoGrafica( ){
   super("Aplicacao grafica simples");
   addWindowListener(new AppListener( ));
   texto1  = new Label("Nome:");    campo1  = new TextField(15);
   texto2  = new Label("Fone:");    campo2  = new TextField(15);
   butOK   = new Button("OK");      resp    = new TextField(15);
   setLayout(new GridLayout(3,2));
   add(texto1); add(campo1);
   add(texto2); add(campo2);
   add(butOK);  add(resp);
   butOK.addActionListener(new ActionListener(){
     public void actionPerformed(ActionEvent e){
       resp.setText(campo1.getText()+" "+campo2.getText());
     }
   });
   pack( );
  }
}
```

33

Resultado



34

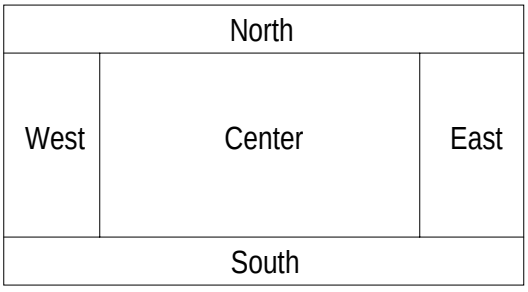
GridBagLayout

- **Semelhante ao GridLayout, porém as divisões podem ter tamanhos diferentes**
- **Utiliza a classe GridBagConstraints para dimensionar e posicionar os componentes**
- **Um layout flexível e complicado usado quando nenhum dos outros se aplica**

35

BorderLayout

- **Layout default para a maioria das aplicações gráficas. Quando se adiciona um componente, é necessário especificar em qual das áreas ele deve ser adicionado. Ex: add(butOK, BorderLayout.WEST);**
- **Divide um container em cinco regiões**
  - **BorderLayout.CENTER**
  - **BorderLayout.NORTH**
  - **BorderLayout.EAST**
  - **BorderLayout.SOUTH**
  - **BorderLayout. WEST**



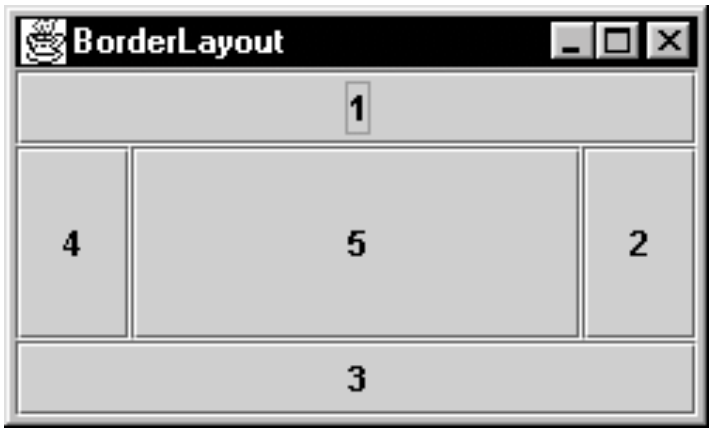
36

BorderLayout - exemplo

```
JPanel jpanel=new JPanel(new BorderLayout( ));
JButton b=new JButton("Botão");
jpanel.add(b,BorderLayout.CENTER);

JFrame f=new JFrame( );

f.getContentPane( ).add(jpanel);
f.pack( );
f.setVisible(true);
```

37

BorderLayout



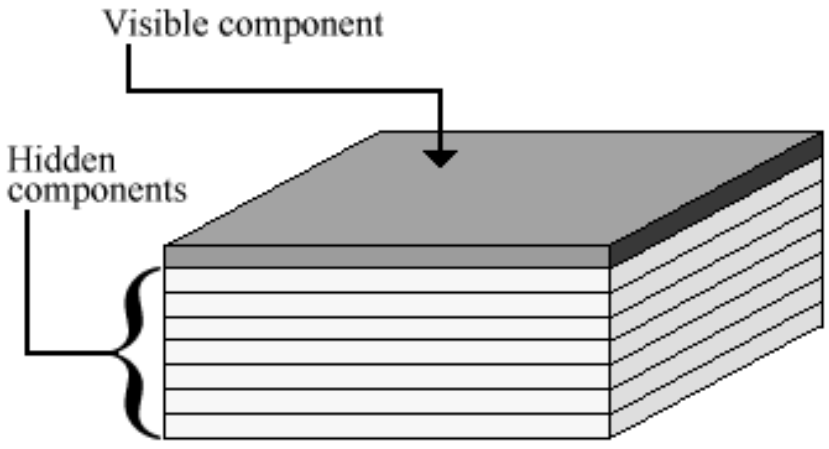
38

CardLayout

- **Usado para exibir um componente de cada vez como em uma pilha de cartas**
  - **somente o objeto que estiver no topo será visível**
- **Métodos:**
  - **first(Container), last(Container),**
  - **next(Container), previous(Container),**
  - **show(Container, String)**

39

CardLayout



40

BoxLayout

- Respeita o tamanho preferido dos componentes
- Coloca os componentes em uma linha ou coluna
  - BoxLayout.X_AXIS para componentes em linha
  - BoxLayout.Y_AXIS para componentes em coluna

```
JPanel c =new JPanel();

// trocando o layout do JPanel
c.setLayout(new BoxLayout(c,BoxLayout.Y_AXIS));

//adicionando os JButtons nas posições desejada
c.add(new JButton("um"));
c.add(new JButton("dois"));
c.add(new JButton("três"));
c.add(new JButton("quatro"));
```

41

BoxLayout

42

GridLayout

```
JPanel c =new JPanel( );
c .setLayout(new GridLayout(2,2));
c.add(new JButton("um"));
c.add(new JTextField(5));
c.add(new JButton("dois"));
c.add(new JButton("três"));
```

- Divide a área em uma grade
- Dispõe os elementos da esquerda para a direita e de cima para baixo
- Todos tem mesmo tamanho

43

Compondo Layouts usando Panels

- A classe Panel é derivada de Container e pode ser usada para agrupar Components de maneira a criar Layouts compostos.
- Por ser também um Container, um Panel pode ter seu Layout próprio.
- Como um Container pode conter outros Containers, podemos agrupar vários Components em um Panel e passar a considerá-los como um único Component para efeitos de layout.
- Veja a seguir o código que exibe o seguinte diálogo:

44

Exemplo 1:
Compondo Layouts usando a AWT

```
public class AplicacaoGrafica extends Frame{
  private Button butOK;
  private TextField campo1,campo2,campoR;
  private Label texto1,texto2,textoR;
  private Panel p1 = new Panel( );
  private Panel p2 = new Panel( );

  public AplicacaoGrafica( ){
    super("Aplicacao grafica simples");
    // Cria os componentes
    texto1  = new Label("Nome:");   campo1  = new TextField(15);
    texto2  = new Label("Fone:");    campo2  = new TextField(15);
    butOK   = new Button("OK");
    textoR  = new Label("Resp:");    campoR  = new TextField(20);

    // Define o layout do container básico
    setLayout(new GridLayout(2,1));
    // Define o layout dos Panels
    p1.setLayout(new GridLayout(2,2));
    p2.setLayout(new FlowLayout(FlowLayout.CENTER));
    // Adiciona os componentes aos panels
    p1.add(texto1); p1.add(campo1);
    p1.add(texto2); p1.add(campo2);
    p2.add(butOK);
    p2.add(textoR); p2.add(campoR);
    // Adiciona os panels ao container básico
    add(p1); add(p2);
                .
                .
                .
```

45

Exemplo 2:
Compondo Layouts usando a Swing

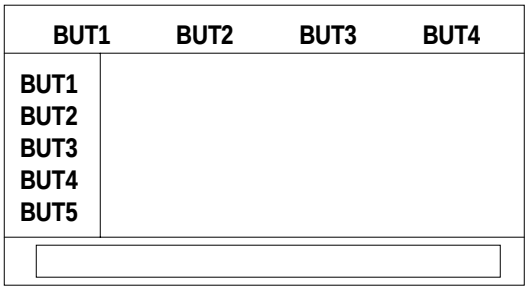
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class PanelDemo extends JFrame {
  private JPanel buttonPanel;
  private JButton buttons[ ];

  public PanelDemo( )
  {
    super( "Panel Demo" );
    Container c = getContentPane();
    buttonPanel = new JPanel();
    buttons = new JButton[ 5 ];
    buttonPanel.setLayout( new GridLayout( 1, buttons.length ) );

    for ( int i = 0; i < buttons.length; i++ ) {
      buttons[ i ] = new JButton( "Button " + (i + 1) );
      buttonPanel.add( buttons[ i ] );
    }

    c.add( buttonPanel, BorderLayout.SOUTH );
    setSize( 425, 150 );
    show( );
  }

  public static void main( String args[] )
  {
    PanelDemo app = new PanelDemo( );
    app.addWindowListener(
      new WindowAdapter() {
        public void windowClosing( WindowEvent e )
        {          System.exit( 0 ); } } );
  }
}
```

46

Exercícios

6) Altere o layout da janela do último exemplo de maneira que o botão de OK fique centrado na parte inferior da janela juntamente com um botão de CANCEL.

7) Crie o layout de uma janela que tenha 4 botões na parte superior, 5 botões no lado esquerdo, um campo de entrada de dados na parte inferior e deixa a área central livre para a edição de gráficos como no esquema abaixo:



47