Apostila de Java – Interface Gráfica

1 – Introdução a interface gráficas em Java

Agora vamos falar sobre o pacote javax.swing, sua relação com o pacote java.awt e introduziremos a classe JFrame, componente principal dos aplicativos de interface gráfica em Java.

1.1 – Entendo o Swing

O Swing é um grande grupo de componentes visuais (caixas de texto, rótulos-

labels, botões – radio, checbox, de ação) escritos em Java puro.

Um componente do Swing é reconhecido pela letra “J” antecedendo o nome do mesmo componente na hierarquia AWT. Assim, se no AWT temos o componente Button, em Swing esse componente é chamado de JButton. Label passa a ser chamado de JLabel, Frame de JFrame e assim por diante.

1.2 – Uso do JFrame na construção de janelas no aplicativo

A janela do aplicativo é a parte mais importante da interface, além de ser o fundamento sobre o qual os demais componentes serão construídos, conhecidas como IDEs visuais (Ambientes de Desenvolvimento Integrado).

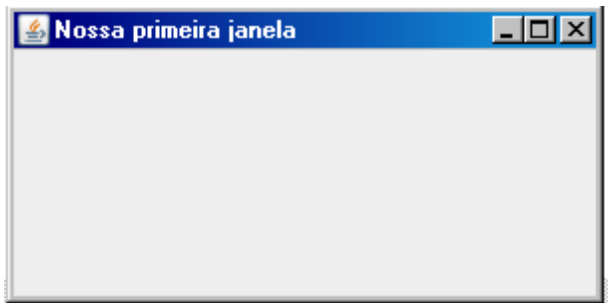
A classe JFrame é uma versão melhorada de Frame que traz consigo suporte aos componentes Swing. Essa classe fornece todas as propriedades, métodos e eventos que precisamos para construir janela, tais como as que nossos usuários estão acostumados a visualizar em seus sistemas. Ora, o usuário sabe que a janela de um aplicativo pode ser minimizada, maximizada, redimensionada, e etc. Então vamos demonstrar a classe JFrame serve aos nossos propósitos.

A janela principal de um aplicativo de interface gráfica em Java é criada como uma instância da classe JFrame.

1.3 - Janela normal

```
import javax.swing.*;
public class PrimeiraJanela  extends JFrame{
  public PrimeiraJanela(){
    super("Nossa primeira janela");
    setSize(300, 150); // largura (comprimento) e altura
    setVisible(true);  }

  public static void main(String args[]){
          PrimeiraJanela  app = new PrimeiraJanela();
          app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);   }    }
```



Vamos analisar cada linha do aplicativo.

Tenha a certeza de importar esse pacote em todos os aplicativos de interface gráfica que estiver escrevendo: import javax.swing.*;

Em seguida temos a definição da classe principal:

```
public class PrimeiraJanela extends JFrame
{
// a implementação da classe vai aqui.
}
```

Observe como a classe PrimeiraJanela, que será a janela do aplicativo, herda da classe JFrame. A partir desse ponto, nossa classe possui todas as propriedades, métodos e eventos que precisamos para que o aplicativo possa ser minimizado, maximizado e fechado por meios dos botões na barra de títulos. Dentro da classe temos o construtor padrão, que será invocado quando uma instância dessa classe for criada (o que acontecerá quando a janela for exibida na tela):

```
import javax.swing.*;
public class PrimeiraJanela  extends JFrame
{
  public PrimeiraJanela(){
    super("Nossa primeira janela");
    setSize(300, 150); // largura (comprimento) e altura
    setVisible(true);  }
```

Uma classe JFrame pode ser instanciada de duas maneiras:
app = new() ::que cria uma instância da classe que é inicialmente invisível, mas sem um título na barra de títulos,
ou:
app = new(String Título) :que cria uma instância da classe inicialmente invisível e com um objeto String representando o texto da barra de títulos.

Assim, no construtor da classe PrimeiraJanela, invocamos o segundo construtor da superclasse JFrame fornecendo o texto que queremos como título da janela.

Após a definição do título da janela, temos uma chamada ao método setSize:

setSize(300, 150); (300 = largura (comprimento) e 150 = altura)

Finalmente exibimos a janela efetuando uma chamada ao método setVisible o qual apresenta a seguinte sintaxe: setVisible(true);

Para finalizar o código para a janela, temos a definição do método main():

```
public static void main(String args[])
{
    PrimeiraJanela app = new PrimeiraJanela();
    app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

Na primeira linha criamos uma instância da classe PrimeiraJanela e a atribuímos à referência app, então acessamos o método setDefaultCloseOperation e fornecemos o valor EXIT_ON_CLOSE, que é uma das constantes da classe JFrame que define como o aplicativo vai se comportar quando o usuário tentar fechar a janela, seja clicando no botão fechar na barra de títulos ou pressionando ALT + F1. Veja no quadro outros valores que você pode usar:

Valor da Constante	Resultado
HIDE_ON_CLOSE	Faz com que a janela seja apenas ocultada quando o usuário tentar fechar o aplicativo, ou seja, o programa continua sua execução. O uso desse valor só é justificado quando queremos que a janela ou algum de seus componentes esteja disponível para acesso e manipulação, mas não queremos que isso seja visível aos olhos do usuário.
DO_NOTHING_ON_CLOSE	Faz com que o usuário não seja capaz de fechar a janela. Neste caso você deve efetuar algum processamento e fechar a janela, baseado em alguma ação específica no seu código.
DISPOSE_ON_CLOSE	Libera, ou seja, retira da memória, tanto a janela do aplicativo quanto seus componentes. A diferença entre este valor e HIDE_ON_CLOSE é que o primeiro oculta a janela e só então libera da memória.

1.4 - Janela Maximizada: setExtendedState(MAXIMIZED_BOTH);

```
import javax.swing.*;
public class JanelaMaximizada extends JFrame{
  public JanelaMaximizada(){
    super("Como exibir a janela maximizada");
    setSize(300, 150);
    setVisible(true);
    setExtendedState(MAXIMIZED_BOTH);
  }

  public static void main(String args[]){
    JanelaMaximizada app = new JanelaMaximizada();
    app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

1.5 - Janela minimizada: setExtendedState(ICONIFIED);

```
import javax.swing.*;
public class JanelaMinimizada extends JFrame{
  public JanelaMinimizada(){
    super("Como exibir a janela minimizada");
    setSize(300, 150);
    setVisible(true);
    setExtendedState(ICONIFIED);
  }

  public static void main(String args[]){
    JanelaMinimizada app = new JanelaMinimizada();
    app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

1.6 – Janela que não pode ser redimensionada: setResizable(false);

Por padrão, as janelas dos aplicativos em Java podem ser redimensionadas em tempo de execução, ou seja, durante a execução do programa. Algumas vezes não queremos que o usuário seja capaz de alterar o tamanho da janela. Com a lista abaixo podemos fazer com que a janela não seja redimensionada:

```
import javax.swing.*;
public class TamanhoFixo extends JFrame{
  public TamanhoFixo(){
    super("Uma janela não dimensionável");
    setResizable(false);
    setSize(300, 150);
    setVisible(true);
  }
```

```
  public static void main(String args[]){
    TamanhoFixo app = new TamanhoFixo();
    app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

1.7 – Janela Centralizada: setLocationRelativeTo(null);

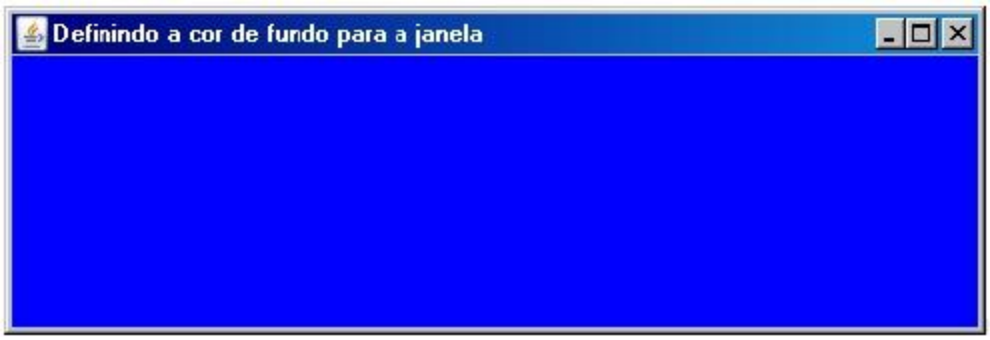
```
import javax.swing.*;
public class JanelaCentralizada extends JFrame{
  public JanelaCentralizada (){
    super("Janela Centralizada");
    setSize(300, 150);
    setVisible(true);
    setLocationRelativeTo(null);
  }
  public static void main(String args[]){
    JanelaCentralizada app = new JanelaCentralizada ();
    app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

1.8 – Trocando a cor do fundo da janela

Embora as propriedades mais importantes da janela do aplicativo já tenham sido vista a possibilidade de alterar a cor de fundo da janela só pôde ser apresentada agora. Isso se deve ao fato de a cor do fundo ser definida não para a classe JFrame, mas para o painel de conteúdo, ou seja o objeto Container que contém os controles da janela.

O aplicativo seguinte exibe uma janela com a cor azul definida como cor de fundo. Observado que a cor de fundo para painel não afetará os demais controles da janela. Veja o aplicativo abaixo:

```
import javax.swing.*;
import java.awt.*;
public class CorDeFundo extends JFrame{
  public CorDeFundo(){
    super("Definindo a cor de fundo para a janela");
    Container tela = getContentPane();
    tela.setBackground(Color.blue);
    setSize(500, 100);
    setVisible(true);
  }
  public static void main(String args[]){
    CorDeFundo app = new CorDeFundo();
    app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```



Temos que importar o pacote import java.awt.*;, pois a classe Container está nele. Veja como uma instância dessa classe é criada e atribuída à referência tela: Container tela = getContentPane();

Após essa declaração o tela é um objeto Container que representa a janela do aplicativo. Por meio dessa referência podemos adicionar e manipular os novos componentes. Comando que troca a cor de fundo da janela:
tela.setBackground(Color.blue);

Você pode usar: red, white, green, silver

1.9 – Usando cores personalizadas para suas janelas

Você sabia que você pode editar cores personalizadas para suas janelas de aplicativos Java? Com o comando setBackground (), isso é possível.
Veja o exemplo:

```
import javax.swing.*;
import java.awt.*;
public class CorDeFundo2 extends JFrame{
  public CorDeFundo2(){
    super("Definindo a cor de fundo para a janela");
    Container tela = getContentPane();
    tela.setBackground(new  Color(255,128,128));
    setSize(500, 100);
    setVisible(true);
  }
  public static void main(String args[]){
    CorDeFundo2 app = new CorDeFundo2();
    app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

tela.setBackground(new Color(255,128,128));

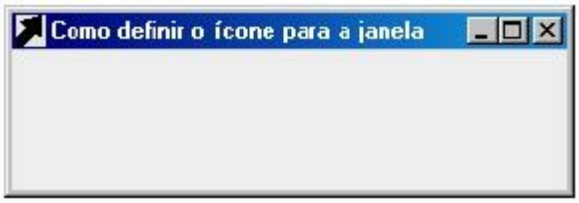
tela.setBackground(new Color(255,228,228));

tela.setBackground(new Color(255,255,201));

1.10 – Definindo um ícone para a janela do aplicativo

Nada contra o ícone padrão das janelas dos aplicativos Java. Mas, seria bem interessante se pudéssemos definir um ícone que lembra nossa marca, ou talvez um ícone que represente melhor a funcionalidade do aplicativo.

```
import javax.swing.*;
public class DefinirIcone extends JFrame{
  public DefinirIcone(){
    super("Como definir o ícone para a janela");
    ImageIcon icone = new ImageIcon("teste.gif");
    setIconImage(icone.getImage());
    setSize(300, 150);
    setVisible(true);
  }
  public static void main(String args[]){
    DefinirIcone app = new DefinirIcone();
    app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```



Após executar o aplicativo, observe como o ícone padrão (a xícara de café) foi alterada, substituída pela imagem que foi adicionada no aplicativo.

A definição de um ícone para a janela pode ser realizada com apenas duas linhas de código:

```
ImageIcon icone = new ImageIcon("teste.gif");
setIconImage(icone.getImage());
```

Na primeira linha criamos uma instância da classe ImageIcon e a atribuímos à referência ícone. Objetos desta classe podem ser criados por meio de nove construtores diferentes. Optamos por aquele que recebe o caminho e/ou nome da imagem como argumento. Veja sua sintaxe:

```
ImagemIcon(String caminho_e_nome_da_imagem);
ImageIcon("teste.gif");
```

Só precisamos fornecer o nome da imagem ou o caminho e o nome da imagem. É importante observar que esse caminho é sempre transformado em uma URL antes de ser passado para o construtor da classe. Assim, você pode fornecer uma imagem usando apenas o nome da imagem:

```
ImageIcon icone = new ImageIcon("teste.gif");
```

O Caminho e o nome da imagem:

```
ImageIcon icone = new ImageIcon("imagens/teste.gif");
```

Após iniciamos a classe ImageIcon, fazemos uso do método getImage dessa classe para obter a imagem e a definimos como ícone da janela com uma chamada a setIconImage da classe JFrame. setIconImage(icone.getImage());