19 – Adicionando menus a um aplicativo Java

Menus são uma parte integrante de qualquer aplicativo de interface gráfica. Seu uso ajuda a agrupar o acesso as funcionalidades do programa em lugares já conhecidos pelo usuário e evita a poluição desnecessárias de componentes na tela do aplicativo.

Menus são criados como objetos das classes JMenurBar, JMenu, JMenuitem, JCheckBoxMenuItem e JRadioButtonMenuItem. Todas essas classes têm sua relevância e uma aplicação de cada um será apresentada em seguida. Por ora, saiba que o menu pode ser adicionados apenas aos objetos de classes que fornecem o método JMenuBar. A classe JMenuBar serve para criar a barra de menus. Já o JMenu adiciona os itens do menu.Já os itens dos menus são colocados com JMenuItem.

O Aplicativo seguinte apresenta uma barra de menus, que contém o menu opções, com os itens limpar, fonte e sair:

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

public class ExemploDeMenu extends JFrame{
        JMenuBar barra;
        JMenu opcoes;
        JMenuItem limpar,fonte,sair;
        public ExemploDeMenu(){
                super("Exemplo de Menus");
                Container tela = getContentPane();
                tela.setLayout(null);
                barra = new JMenuBar();
                setJMenuBar(barra);
                opcoes = new JMenu("Opções");
                barra.add(opcoes);
                limpar = new JMenuItem("Limpar");
                fonte = new JMenuItem("Fonte");
                sair = new JMenuItem("Sair");
                limpar.setMnemonic(KeyEvent.VK_L);
                fonte.setMnemonic(KeyEvent.VK_F);
                sair.setMnemonic(KeyEvent.VK_S);
                opcoes.add(limpar);
                opcoes.add(fonte);
                opcoes.addSeparator();
                opcoes.add(sair);
                setSize(500,300);
                setLocationRelativeTo(null);
                setVisible(true);
        }
public static void main (String args[]){
        ExemploDeMenu app = new ExemploDeMenu();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Vamos analisar as partes mais importantes deste código:

Declarar na área pública a variável que vai receber a barra de menu:
JMenuBar barra;

Criar a barra de menus: barra = new JMenuBar();

Anexam a barra de menus à janela com uma chamada ao método setJMenuBar da classe JFrame (janela): setJMenuBar(barra);
Uma janela pode ter apenas uma barra de menus, disposta horizontalmente na parte superior da janela.

O menu Opções é criado com a seguinte instrução:
opcoes = new JMenu("Opções");

Depois é anexado a barra de menus: barra.add(opcoes);

Observe o o uso do método setMnemonic para definir as teclas de atalho para o menu e todos os itens do menu.

```
limpar.setMnemonic(KeyEvent.VK_L);
fonte.setMnemonic(KeyEvent.VK_F);
sair.setMnemonic(KeyEvent.VK_S);
```

Agora é a hora de criar os itens do menu opções:

```
limpar = new JMenuItem("Limpar");
fonte = new JMenuItem("Fonte");
sair = new JMenuItem("Sair");
```

Lembrando de declarar as variáveis que irão receber os itens do menu como JMenuItem: JMenuItem limpar,fonte,sair;

Agora vamos anexar os itens no menu opções:

```
opcoes.add(limpar);
opcoes.add(fonte);
opcoes.addSeparator();  Adiciona um linha separadora no menu
opcoes.add(sair);
```

Exemplo do aplicativo acima sendo executado

19.1 – Como adicionar submenus a um item do menu

Sbmenus podem ser adicionados a um determinado item de menu para tornar o agrupamento lógico ainda mais eficiente. Essa técnica consiste em adicionar um JMenu a outro JMenu. Veja um exemplo de como isso pode ser feito:

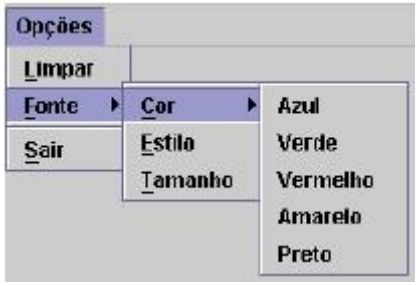
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class ExemploDeMenu2 extends JFrame{
        JMenuBar barra;
        JMenu opcoes,fonte,cor;
        JMenuItem limpar,sair,estilo,tamanho,azul,verde,vermelho,amarelo,preto;
        public ExemploDeMenu2(){
                super("Exemplo de Menus com submenus");
                Container tela = getContentPane();
                tela.setLayout(null);
                barra = new JMenuBar();
                setJMenuBar(barra);
                opcoes = new JMenu("Opções");
                barra.add(opcoes);
                limpar = new JMenuItem("Limpar");
                fonte = new JMenu("Fonte");
                cor = new JMenu("Cor");
                azul = new JMenuItem("Azul");
                verde = new JMenuItem("Verde");
                vermelho = new JMenuItem("Vermelho");
                amarelo = new JMenuItem("Amarelo");
                preto = new JMenuItem("Preto");
                estilo = new JMenuItem("Estilo");
                tamanho = new JMenuItem("Tamanho");
                sair = new JMenuItem("Sair");
                limpar.setMnemonic(KeyEvent.VK_L);
                fonte.setMnemonic(KeyEvent.VK_F);
                sair.setMnemonic(KeyEvent.VK_S);
                cor.setMnemonic(KeyEvent.VK_C);
                estilo.setMnemonic(KeyEvent.VK_E);
                tamanho.setMnemonic(KeyEvent.VK_T);
                opcoes.add(limpar);
                opcoes.add(fonte);
                opcoes.addSeparator();
                opcoes.add(sair);
                fonte.add(cor);
                fonte.add(estilo);
                fonte.add(tamanho);
                cor.add(azul);
                cor.add(verde);
                cor.add(vermelho);
                cor.add(amarelo);
                cor.add(preto);
                setSize(500,300);
                setLocationRelativeTo(null);
                setVisible(true);
        }
public static void main (String args[]){
        ExemploDeMenu2 app = new ExemploDeMenu2();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }}
```

Vamos analisar o aplicativo:

As variáveis que no aplicativo anterior foram declaradas como JMenuItem, para que possam ser anexado a submenus elas terão que mudar de classe e passar a ser da classe JMenu. Veja:

JMenu opcoes,fonte,cor;



Observe como o item Fonte do menu Opções tem uma seta para a direita, indicando que tem nesse item três submenus, Cor, Estilo e Tamanho. O item Cor também possui uma seta para direita, indicando que nesse item possui, cinco subitens.

Agora vamos criar o item fonte e cor como JMenu, e os itens azul, verde, vermelho, amarelo e preto serão criados como JMenuItem, vai indicar que o JMenuItem é um item do JMenu do item Cor.

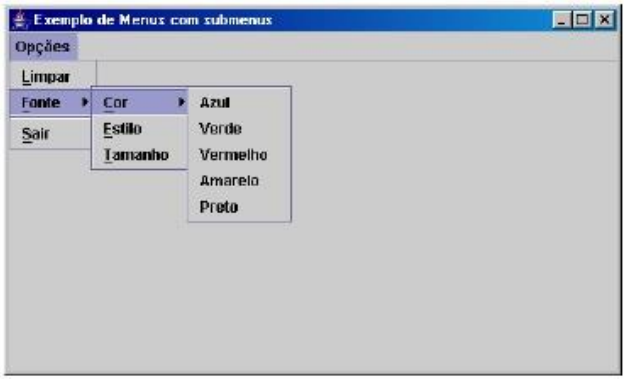
```
fonte = new JMenu("Fonte");
cor = new JMenu("Cor");
azul = new JMenuItem("Azul");
verde = new JMenuItem("Verde");
vermelho = new JMenuItem("Vermelho");
amarelo = new JMenuItem("Amarelo");
preto = new JMenuItem("Preto");
estilo = new JMenuItem("Estilo");
tamanho = new JMenuItem("Tamanho");
```

Agora vamos anexar os itens cor, estilo e tamanho no item fonte:

```
fonte.add(cor);      fonte.add(estilo);   fonte.add(tamanho);
```

E finalmente anexar as cores dentro do item cor:

```
cor.add(azul);       cor.add(verde);      cor.add(vermelho);
cor.add(amarelo);  cor.add(preto);
```



19.2 – Menu executando ações

Quando um determinado item é selecionado, o código do aplicativo deve detectar a seleção e efetuar as ações determinadas. Esse procedimento é fácil. Uma vez que a classe JMenuItem herda abstração de botões e fornece o método addActionListener.
No aplicativos adicionamos ao exemplo um componente JTextArea, para usar as opções dos menus Limpar e Cor.
Use o menu Limpar para apagar todo o conteúdo do JTextArea, e o menu Cor para alterar a cor do texto digitado no textarea.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Item19_2 extends JFrame{
    JMenuBar barra;
    JMenu opcoes,fonte,cor;
    JMenuItem limpar,sair,azul,verde,vermelho,amarelo,preto;
    JTextArea texto;
    JPanel painel;
    JScrollPane rolagem;
    public Item19_2(){
        super("Exemplo de Menus executando acoes");
        Container tela = getContentPane();
        tela.setLayout(null);
        barra = new JMenuBar();
        setJMenuBar(barra);
        opcoes = new JMenu("Opcoes");
        barra.add(opcoes);
        limpar = new JMenuItem("Limpar");
        fonte = new JMenu("Fonte");
        cor = new JMenu("Cor");
        azul = new JMenuItem("Azul");
        verde = new JMenuItem("Verde");
        vermelho = new JMenuItem("Vermelho");
        amarelo = new JMenuItem("Amarelo");
        preto = new JMenuItem("Preto");
        sair = new JMenuItem("Sair");

        opcoes.setMnemonic(KeyEvent.VK_O);
        limpar.setMnemonic(KeyEvent.VK_L);
        fonte.setMnemonic(KeyEvent.VK_F);
        sair.setMnemonic(KeyEvent.VK_S);
        cor.setMnemonic(KeyEvent.VK_C);

        opcoes.add(limpar);
        opcoes.add(fonte);
        opcoes.addSeparator();
        opcoes.add(sair);

        cor.add(azul);
        cor.add(verde);
        cor.add(vermelho);
        cor.add(amarelo);
        cor.add(preto);

        fonte.add(cor);
```

```
texto = new JTextArea(10,20);
rolagem = new JScrollPane(texto);
rolagem.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
rolagem.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
painel = new JPanel();
painel.add(rolagem);
painel.setBounds(30,30,250,250);
tela.add(painel);

limpar.addActionListener(new ActionListener(){
	public void actionPerformed(ActionEvent e){
		texto.setText("");
		texto.requestFocus();
		}
	}
);

azul.addActionListener(new ActionListener(){
	public void actionPerformed(ActionEvent e){
		texto.setForeground(Color.blue);
		repaint();
		}
	}
);

verde.addActionListener(new ActionListener(){
	public void actionPerformed(ActionEvent e){
		texto.setForeground(Color.green);
		repaint();
		}
	}
);

vermelho.addActionListener(new ActionListener(){
	public void actionPerformed(ActionEvent e){
		texto.setForeground(Color.red);
		repaint();
		}
	}
);

amarelo.addActionListener(new ActionListener(){
	public void actionPerformed(ActionEvent e){
		texto.setForeground(Color.yellow);
		repaint();
		}
	}
);

preto.addActionListener(new ActionListener(){
	public void actionPerformed(ActionEvent e){
		texto.setForeground(Color.black);
		repaint();
		}
	}
);

sair.addActionListener(new ActionListener(){
	public void actionPerformed(ActionEvent e){
		System.exit(0);
		}
```

```
            }
        );

        setSize(500,300);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    public static void main (String args[]){
        Item19_2 app = new Item19_2();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Menu Limpar:

```
limpar.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
        texto.setText("");
        texto.requestFocus();}});
```

Quando este item for executado, o componente JTextArea será limpo e receberá o foco.

Menu Sair:

```
sair.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
                System.exit(0);}});
```

Quando este item de menu for selecionado, o aplicativo será fechado, pois ele chama o método System.exit(0), e finaliza o aplicativo.

Menu Fonte -> Cor -> Amarelo

Quando o item correspondente a cor for selecionado e executado o componente JTextArea, receberá a cor escolhida

```
amarelo.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
                texto.setForeground(Color.yellow);
                repaint();}});

vermelho.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
                texto.setForeground(Color.red);
                repaint();}});
```

OBS: O método repaint(), sempre é chamando nas classes, ele serve para atualizar o estilo, fonte ou a cor escolhida.

20 – Criando um menu de contexto – Classe JPopupMenu

Menus de contexto ou menu popup têm a mesma funcionalidade dos menus convecionais. A única diferença é que menus de contexto são exibidos quando clicamos com o botão direito no componente para o qual o menu foi definido. Para ver um exemplo de menu com contexto, abra seu editor de texto favorito e clique com o botão direito sobre a área de edição de texto. As chances de um menu de contexto ser exibido são enormes. Em Java menus de contexto são criados como objetos da classe JPopupMenu. Os itens do menu são criados a partir da classe JMenuItem, de modo que você pode usar todas as técnicas que vimos nos tópicos anteriores.
O aplicativo seguinte cria um menu de contexto para uma área de texto e oferece as opções de recortar, copiar e colar. Veja a listagem para o exemplo:

```
mport javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ExemploDePopupMenu extends JFrame{
        JPopupMenu opcoes;
        JMenuItem recortar,copiar,colar;
        JTextArea texto;
        public ExemploDePopupMenu(){
                super("Exemplo de PopupMenu");
                Container tela = getContentPane();
                tela.setLayout(null);
                texto = new JTextArea(10,20);
                texto.setBounds(30,30,250,250);
                texto.addMouseListener(new MouseAdapter(){
                public void mouseReleased(MouseEvent e){
                    if(e.isPopupTrigger())
                      opcoes.show(e.getComponent(),e.getX(),e.getY());    }});
                Tratador tratmenu = new Tratador();
                opcoes = new JPopupMenu();
                recortar = new JMenuItem("Recortar");
                copiar = new JMenuItem("Copiar");
                colar = new JMenuItem("Colar");
                recortar.setMnemonic(KeyEvent.VK_R);
                copiar.setMnemonic(KeyEvent.VK_C);
                colar.setMnemonic(KeyEvent.VK_L);
                recortar.addActionListener(tratmenu);
                copiar.addActionListener(tratmenu);
                colar.addActionListener(tratmenu);
                opcoes.add(recortar);
                opcoes.add(copiar);
                opcoes.addSeparator();
                opcoes.add(colar);
                tela.add(texto);
                setSize(320,340);
                setLocationRelativeTo(null);
                setVisible(true);
        }
public static void main (String args[]){
        ExemploDePopupMenu app = new ExemploDePopupMenu();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
   }
private class Tratador implements ActionListener{
public void actionPerformed(ActionEvent e){
        if(e.getSource()==recortar)
          texto.cut();
        if(e.getSource()==copiar)
          texto.copy();
        if(e.getSource()==colar)
          texto.paste();
        repaint();  }}
}
```

Quando você executar esse aplicativo, clique com o botão direito na área de texto e o menu de contexto será exibido. Escolha um dos itens do menu e veja o resultado obtido.

O menu de contexto usado para este exemplo é criado na linha:

```
opcoes = new JPopupMenu();
```

Para exibi-lo, criamos uma classe anônima MouseAdapter e a fornecemos como argumento para o método addMouseListener da área de texto:

```
texto.addMouseListener(new MouseAdapter(){
public void mouseReleased(MouseEvent e)  {
        if(e.isPopupTrigger())
        opcoes.show(e.getComponent(),e.getX(),e.getY()); }});
```

Quando clicamos com o botão direito na área de texto e o método mouseReleased é disparado, efetuamos uma chamada ao método isPopupTrigger da classe MouseEvent. Esse método retorna verdadeio se o evento foi disparado com o botão direito do mouse. Satisfeita a condição, o menu de contexto é exibido. Observe os parâmetros para o método show da classe JPopupMenu:

```
show(Componet invoker, int x, int y);
opcoes.show(e.getComponent(),e.getX(),e.getY()); }});
```

O parâmetro invoker é o componente para o qual o menu de contexto esta definido. Os parâmetros x e y são as coordenadas do mouse no momento em que o menu de contexto é chamado.

21 – Uso da classe JToolBar para criar barra de ferramentas

Uma barra de ferramentas é um contêiner para agrupar botões e outros controles usados com maior freqüência no aplicativo. Grande parte dos aplicativos atuais apresenta barras de ferramentas, de modo que você deve analisar e verificar a necessidade de implementar essa funcionalidade em seus programas. Barras de ferramentas em aplicativos Java são criadas como objetos da classe JToolBar. Essa classe oferece algumas funcionalidades bem interessantes, entre elas a capacidade de ser arrastada pela janela do aplicativo (e até fora deste). O aplicativo seguinte mostra como incluir uma barra de ferramentas semelhantes às encontradas em muitos programas atuais. No momento os botões não fornecem nenhuma funcionalidade, mas você será capaz de arrastar a barra para qualquer um dos cantos da janela e, para sua surpresa, arrastá-la para fora do aplicativo. Veja a listagem para o exemplo:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ExemploBarraDeFerramentas extends JFrame{
                JToolBar barra;
                JButton direita,esquerda,centralizar,recortar,copiar,colar;
                ImageIcon imagens[];
                public ExemploBarraDeFerramentas(){
                  super("Exemplo de Barra de Ferramentas");
                  Container tela = getContentPane();
                  tela.setLayout(null);
                  String icones[]={"imagens/direita.gif","imagens/esquerda.gif",
                                   "imagens/centro.gif","imagens/recortar.gif",
                                   "imagens/copiar.gif","imagens/colar.gif"};
                  imagens = new ImageIcon[6];
                  for(int i = 0;i < 6;i++){
                           imagens[i] = new ImageIcon(icones[i]);}
                 direita = new JButton(imagens[0]);
                 esquerda = new JButton(imagens[1]);
                 centralizar = new JButton(imagens[2]);
                 recortar = new JButton(imagens[3]);
                 copiar = new JButton(imagens[4]);
                 colar = new JButton(imagens[5]);
                  direita.setToolTipText("Direita (Ctrl+d)");
                  esquerda.setToolTipText("Esquerda (Ctrl+e)");
                  centralizar.setToolTipText("Centralizar (Ctrl+z)");
                  recortar.setToolTipText("Recortar (Ctrl+x)");
                  copiar.setToolTipText("Copiar (Ctrl+c)");
                  colar.setToolTipText("Colar (Ctrl+v)");
                  barra = new JToolBar("Barra de Ferramentas");
                  UIManager.put("ToolTip.background",SystemColor.info);
                  UIManager.put("ToolTip.foreground",Color.blue);
                 barra.setRollover(true);
                 barra.add(direita);
                 barra.add(esquerda);
                 barra.add(centralizar);
                 barra.addSeparator();
```

```
                barra.add(recortar);
                barra.add(copiar);
                barra.add(colar);
                barra.setBounds(1,1,260,50);
                tela.add(barra);
                setSize(320,340);
                setLocationRelativeTo(null);
                setVisible(true);          }
public static void main (String args[]){
ExemploBarraDeFerramentas app = new ExemploBarraDeFerramentas();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
}
```

Para que este exemplo ficasse bem realista, adicionamos uma barra de menus. Experimente arrastar a barra de ferramentas.

Declarar o vetor que vai receber as imagens:
ImageIcon imagens[];

Para definir as imagens dos botões da barra de ferramentas, usamos o seguinte vetor de objetos com seis posições da classe ImageIcon:
imagens = new ImageIcon[6];

O nome e o caminho das imagens são definidos como itens do vetor ícones:

```
String icones[]={"imagens/direita.gif","imagens/esquerda.gif",
                 "imagens/centro.gif","imagens/recortar.gif",
                 "imagens/copiar.gif","imagens/colar.gif"};
```

O próximo passo é instanciar os objetos da classe ImageIcon com as imagens do vetor ícones. Isso é feito no laço seguinte:

```
          for(int i = 0;i < 6;i++){
                imagens[i] = new ImageIcon(icones[i]);}
```

São criados seis botões com imagens veja seguintes instruções:

```
direita = new JButton(imagens[0]);
esquerda = new JButton(imagens[1]);
centralizar = new JButton(imagens[2]);
recortar = new JButton(imagens[3]);
copiar = new JButton(imagens[4]);
colar = new JButton(imagens[5]);
```

Próximo passo adicionar dicas aos botões da barra de ferramentas:

```
direita.setToolTipText("Direita (Ctrl+d)");
esquerda.setToolTipText("Esquerda (Ctrl+e)");
centralizar.setToolTipText("Centralizar (Ctrl+z)");
recortar.setToolTipText("Recortar (Ctrl+x)");
copiar.setToolTipText("Copiar (Ctrl+c)");
colar.setToolTipText("Colar (Ctrl+v)");
```

A barra de ferramentas é criada na linha:

```
barra = new JToolBar("Barra de Ferramentas");
```

Trocar a cor de fundo da dica dos botões da barra de ferramentas:

```
UIManager.put("ToolTip.background",SystemColor.info);
```

Trocar a cor da letra da dica dos botões da barra de ferramentas:

```
UIManager.put("ToolTip.foreground",Color.blue);
```

Os botões são adicionados à barra de ferramentas pelo método add:

```
barra.add(direita);
barra.add(esquerda);
barra.add(centralizar);
barra.addSeparator();
barra.add(recortar);
barra.add(copiar);
barra.add(colar);
```

Aplicar o efeito Rollover aos botões da barra de ferramentas:

O efeito Rollover modifica algum aspecto do controle quando movemos o ponteiro sobre ele. Geralmente esse efeito aplica cores e imagens diferentes. Em barras de ferramentas a técnica mais comum é aplicar uma borda diferenciada quando o ponteiro adentra ou sai da área do componente.

Java permite aplicar esse efeito a barras de ferramentas por meio de uma chamada ao método setRollover da classe JToolBar. Veja como isso pode ser feito:

```
barra.setRollover(true);
```

Posicionar a barra de ferramentas na janela:

```
barra.setBounds(1,1,260,50);
```

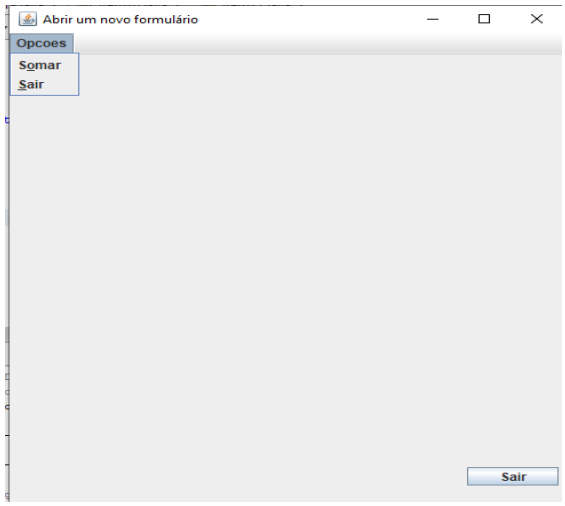
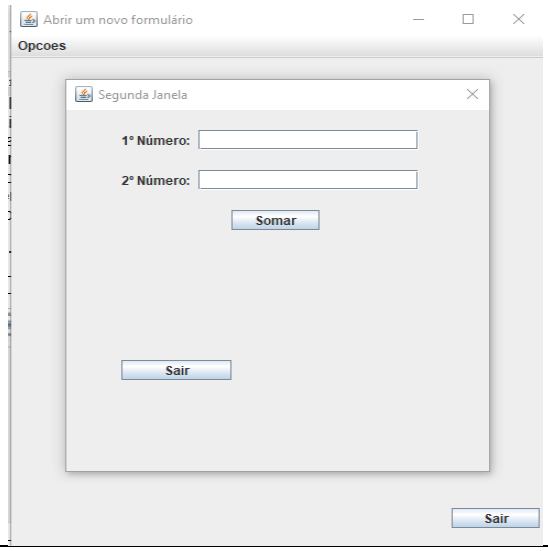
Adicionar a barra de ferramentas na janela:

```
tela.add(barra);
```

22 – Como criar janelas secundárias com o uso da classe JDialog

Janelas secundárias são modais porque dependem da janela principal do aplicativo. Por meio dessas janelas podemos efetuar novas operações acessando até mesmo os componentes da janela principal. Janelas secundárias são criadas como instâncias da classe JDialog, e fornece praticamente todas as funcionalidades da classe JFrame. Isso quer dizer que podemos obter o painel de conteúdo de um objeto dessa classe e incluir componentes, interagir com o usuário, enviando dados para a janela principal.

22.1 – Como criar janelas secundárias modais

Janela Principal	Janela Secundária
	

Janela Modal – Você não pode clicar na janela de traz.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Item22_1 extends JFrame{
	JMenuBar barra;
	JMenu opcoes;
	JMenuItem somar,sair;
	JButton btnSair;
	Inicial segundajanela;
	public Item22_1(){
		super("Abrir um novo formulário");
		Container tela = getContentPane();
		tela.setLayout(null);
		// criando o menu de opções
		barra = new JMenuBar();
		setJMenuBar(barra);
		opcoes = new JMenu("Opcoes");
		barra.add(opcoes);
		somar = new JMenuItem("Somar");
		sair = new JMenuItem("Sair");
```

```
        somar.setMnemonic(KeyEvent.VK_O);
        sair.setMnemonic(KeyEvent.VK_S);
        opcoes.add(somar);
        opcoes.add(sair);

        somar.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
                segundajanela = new Inicial(null,"Segunda Janela",true);
                segundajanela.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                segundajanela.setVisible(true);
                }
        }
        );

        btnSair = new JButton("Sair");
        btnSair.setBounds(400,450,80,20);

        btnSair.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
                int status = JOptionPane.showConfirmDialog(null,"Deseja realmente fechar o
programa?","Mensagem de saida",JOptionPane.YES_NO_OPTION);
                if (status == JOptionPane.YES_OPTION){
                        System.exit(0);}
                }
        }
        );
        tela.add(btnSair);
        setSize(500, 550);
        setVisible(true);
        setLocationRelativeTo(null);

}

public static void main(String args[]){
        Item22_1 app = new Item22_1();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

private class Inicial extends JDialog{
        JButton sair;
        JLabel rotulo1, rotulo2, exibir;
        JTextField texto1, texto2;
        JButton somar2n;
        public Inicial(Frame owner, String title,boolean modal){
                super(owner,title,modal);
                Container tela1 = getContentPane();
                tela1.setLayout(null);
                  // layout do formulário
                  rotulo1 = new JLabel("1º Número: ");
                  rotulo2 = new JLabel("2º Número: ");
                  texto1 = new JTextField(5);
                  texto2 = new JTextField(5);
                  exibir = new JLabel("");
                  somar2n = new JButton("Somar");

                  rotulo1.setBounds(50,20,100,20);
                  rotulo2.setBounds(50,60,100,20);
```

```
            texto1.setBounds(120,20,200,20);
            texto2.setBounds(120,60,200,20);
            exibir.setBounds(50,140,200,20);
            somar2n.setBounds(150,100,80,20);

            somar2n.addActionListener(
                new ActionListener(){
                    public void actionPerformed(ActionEvent e){
                    int numero1, numero2, soma;
                    soma = 0;
                    numero1 = Integer.parseInt(texto1.getText());
                    numero2 = Integer.parseInt(texto2.getText());
                    soma = numero1 + numero2;
                    exibir.setVisible(true);
                    exibir.setText("A soma é: "+soma);
                    }
                }
            );
            exibir.setVisible(false);

            tela1.add(rotulo1);
            tela1.add(rotulo2);
            tela1.add(texto1);
            tela1.add(texto2);
            tela1.add(exibir);
            tela1.add(somar2n);

            sair = new JButton("Sair");
            sair.setBounds(50,250,100,20);
            TBsair tsair = new TBsair();
            sair.addActionListener(tsair);
            tela1.add(sair);
            setSize(400,400);
            setLocationRelativeTo(null);
    }
}

private class TBsair implements ActionListener{
    public void actionPerformed(ActionEvent evento){
                segundajanela.setVisible(false);
                segundajanela.dispose();}
    }
}
```

Vamos analisar o aplicativo:

Após a execução do aplicativo, clique no botão Abrir. Você visualizará a segunda janela com o botão de Sair. O entendimento do código é muito importante para o uso correto de janelas secundárias em um aplicativo Java. É Importante acompanhar atentamente a análise que faremos.

A classe privada Inicial:

```
private class Inicial extends JDialog{
        JButton sair;
        public Inicial(Frame owner, String title,boolean modal){
                super(owner,title,modal);
                Container tela1 = getContentPane();
                tela1.setLayout(null);
                sair = new JButton("Sair");
                sair.setBounds(50,50,100,20);
                TBsair tsair = new TBsair();
                sair.addActionListener(tsair);
                tela1.add(sair);
                setSize(200,200);
                setLocationRelativeTo(null);
      }
}
```

Declaramos a variável que vai fazer referência ao botão sair: JButton sair;

Declaramos a referência segundajanela que receberá a Incial:
Inicial segundajanela;

Classe privada Incial: Esta classe herda JDialog. Observe os parâmetros para seu construtor: public Inicial(Frame owner, String Title, boolean modal);

O parâmetro owner é usado para especificar a janela principal a partir da qual a janela secundária será chamada. O parâmetro title serve para fornece o título para janela, modal especifica se a janela será modal ou não-modal. Uma janela modal tem o foco e deve ser fechada antes que o usuário volte à janela principal do aplicativo.

Na linha linha do construtor temos uma chamada ao construtor da superclasse JDialog: super(owner,title,modal);
O que fazemos aqui é transferir os valores recebidos para a superclasse.

Chamamos os métodos para adicionar componentes na janela:
Container tela1 = getContentPane();
tela1.setLayout(null);

Em seguinda, adicionamos o texto a ser exibido no botão Sair, logo após posicionamos o botão na janela, criamos a variável que vai fazer referência a classe privada TBsair, depois adicionamos essa classe ao botão sair pelo evento addActionListener, e chamamos o método add, e adicionamos o botão

na janela secundária.

```
sair = new JButton("Sair");
sair.setBounds(50,50,100,20);
TBsair tsair = new TBsair();
sair.addActionListener(tsair);
tela1.add(sair);
```

Em seguida definimos o tamanho da janela secundária:

```
setSize(200,200);
```

Por final centralizamos a janela:

```
setLocationRelativeTo(null);
```

Voltando à janela principal agora, para chamar a janela secundária, usamos o seguinte trecho de código:

```
abrir.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
        segundajanela = new Inicial(null,"Segunda Janela",true);
        segundajanela.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        segundajanela.setVisible(true);
        }
}
);
```

Quando clicamos no botão e o método actionPerformed da classe anônima ActionListener é chamado, instanciamos um objeto da janela secundária e o atribuímos à referência segundajanela. Em seguida definimos o modo como a janela vai se comportar quando tentamos fecha-la:

```
segundajanela.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

Ac constate DISPOSE_ON_CLOSE esconde a janela e em seguida a libera da memória. Este é o comportamente mais adequado para janelas modais.

A úttima linha: segundajanela.setVisible(true);

Exibe a janela colocando-a na frente da janela principal. Caso queira, revise o tópico para posicionar a janela secundária no centro da tela quando ela for exibida.

Classe privada TBsair (Tratar Botão Sair)

```
private class TBsair implements ActionListener{
public void actionPerformed(ActionEvent evento){
int status = JOptionPane.showConfirmDialog(null,"Deseja realmente
fechar o programa?","Mensagem de
saída",JOptionPane.YES_NO_OPTION);
        if (status == JOptionPane.YES_OPTION)
                {segundajanela.setVisible(false);
                segundajanela.dispose();
```

```
    }
  }
}
```

Sem muita modificações dos exemplos anteriores, o que tem de diferente é que quando você clicar em Sim para finalizar a janela secundária, o aplicativo coloca a janela invisível com o método setVisible(false); e depois retira da memória com o método dispose().