Exercícios Práticos de Java Swing

1. 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( );
 }
}
```

2.

```
import javax.swing.*;
public class TestaJanela {  // Objeto Janela
    static JFrame janela = new JFrame("Título da janela");

    public static void main (String args[]) {
      janela.setBounds(50, 100, 400, 150); // Seta posição e tamanho
      janela.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      janela.setVisible(true);  // Exibe a janela  }
 }

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

a. Teste a classe exemplo com os diferentes argumentos para o método `setDefaultCloseOperation()`.
b. Faça um trecho de programa que anime uma janela, variando sua posição e tamanho.

3.

```
import javax.swing.*; import java.awt.*;
public class TestaContainer {   // Objeto Janela
static JFrame janela = new JFrame("Título da janela");
 public static void main (String args[ ]) {
int i;
 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 do container
 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
 }
}
```

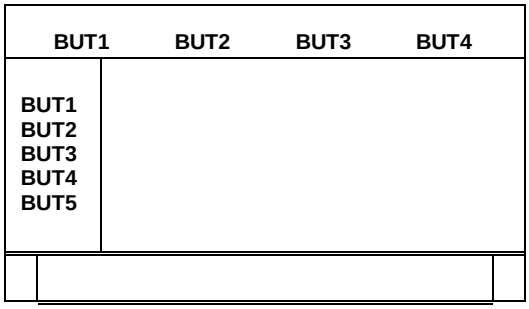
a. Redimensione interativamente a janela da aplicação e observe o comportamento dos botões da interface.
b. Troque o argumento de `FlowLayout()`, para `FlowLayout (FlowLayout.LEFT)` e observe.
c. Adicione no programa acima, os seguintes componentes:
   i. `JLabel label = new JLabel("Exemplo de texto:");`
   ii. `caixa.add(label);`
   iii. `JTextField campo = new JTextField(15);`
   iv. `caixa.add(campo);`
   v. `janela.pack(); // Redimensiona a janela`

4.

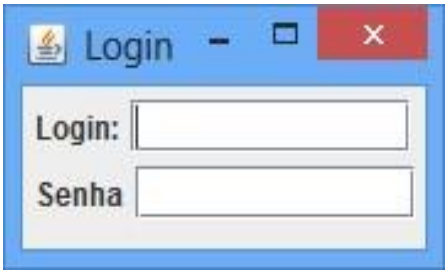
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class PanelDemo extends JFrame {
   private JPanel buttonPanel;
   private JButton buttons[ ];
   public PanelDemo( ) {
      super( "Panel Demo" );
      Container c = getContentPane();
      buttonPanel = new JPanel();
      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 );
                }
       }  );
      }
}
```

a. 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.

b. 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:

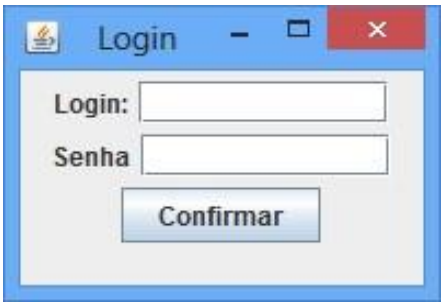


5. Geralmente aplicações comerciais possuem uma área restrita, onde apenas usuários privilegiados possuem acesso. Supõe-se que você é um engenheiro de software *frontend* e uma de suas CR's (Requisitos) é desenvolver uma Interface Gráfica para efetuar o login em uma área restrita de um determinado sistema em desenvolvimento. O Engenheiro de Usabilidade enviou o modelo que ele gostaria que você desenvolvesse. Não é necessário implementar nenhuma funcionalidade pois o engenheiro gostaria de ver como ficou a tela, antes de implementar ações



6. Ao enviar a tela para o Engenheiro, ele percebeu que estava faltando algo. Um botão para confirmar a operação. Ele abriu outra CR para você.

CR	Origem	Destino	Descrição
~~CR01~~	Engenheiro de Usabilidade	Engenheiro de Sistemas (Você)	Desenvolver uma tela de login sem funcionalidades.
CR02	Engenheiro de Usabilidade	Engenheiro de Sistemas (Você)	A partir da CR01, criar um botão na tela e implementar a funcionalidade de verificação de login e senha. Na Verificação, se a senha correta, mostrar uma janela de login realizado com sucesso. Mostrar tela de erro caso contrário. Para melhorar a usabilidade do usuário, o tamanho da tela tem que ser 200x140. Segue em anexo a tela a ser desenvolvida.



Dica: Para validar o login e a senha do usuário use um Mapa. Mapa são estruturas que guardam chave e valor.

```
//Declaração
Map<String, String> chaveiro;
//Instaciação chaveiro = new HashMap<String, String>();
chaveiro.put("adriano", "123"); //Populando o mapa com login e senha
chaveiro.put("fulano01", "4445"); //Populando o mapa com login e senha

//Verificar se um mapa contém uma chave.
if(chaveiro.containsKey(login) = = true ) //Recuperar um valor de uma determinada chave:
chaveiro.get(login)
```

7. Foi requisitado para desenvolver um programa pra fazer uma pesquisa dos salários de sua empresa. O usuário poderá escolher apenas uma opção de salário. Toda vez que o usuário clicar em alguma opção você deve inserir uma frase sobre o salário.

Dicas:

i) Use como layout o **GridLayout** com 5 linhas e 1 coluna.
ii) Crie 4 **RadioButtons** e um JLabel para exibir a mensagem.
iii) Para facilitar crie um array de **JRadioButton**, ao invés de criar cada um separadamente.
iv) Crie um **ButtonGroup** e adicion os J**RadioButtons** nesse grupo, de forma que o usuário não consiga clicar em mais de um **RadioButton**.
v) Crie um tratador de eventos que implementa o **ItemListener**
vi) Crie um array de String com as mensagens de cada RadioButton. De modo que a verificação fique na forma

```
// jrbOpcao é um array de radio buttons
// jlMsg é um JLabel
//msg é um array de String com as mensagens para cada RadioButton
for(int i =0 ; i < jrbOpcao.length; i++)
   if (e.getSource() == jrbOpcao[i])
   jlMsg.setText(msg[i]);
```

8. A empresa onde você trabalha está desenvolvendo um sistema de gestão de petshop e abriram a seguinte CR para você

CR	Origem	Destino	Descrição
CR81	Engenheiro de Usabilidade	Engenheiro de Sistemas (Você)	Desenvolver uma tela que escolhe um nome de um cão com um combobox e para cada cão escolhido apresenta a respectiva foto na tela. Para o sistema ser extensível, o combobox será preenchido de acordo com o conteúdo de uma pasta de fotos dos cães.

[Imagem em Anexo]

[Dicas] Você pode obter a pasta atual usando a classe File.

```
File[] files = new File("pasta/imagens").listFiles();
for (File file : files) {
     if (file.isFile())
          String nomeArquivo = file.getName();
```

i) O ComboBox é preenchido com array de Strings. Os nomes dos arquivos podem ser colocados em um ArrayList e depois convertido em um array de String.

ii) Para converter um `ArrayList<String>` em array de String pode-se usar o comando.

```
String a[] = nome_arraylist.toArray(new String[nome_arraylist.size()]);
```

iii) A foto pode ser colocada em um JLabel. E a imagem "setada" com o método setIcon();

9. Analise o código abaixo. O que ele faz? Modifique este código e crie um Menu chamada Color, neste menu deverá existir três itens Blue, Green e Red. Quando o usuário clicar em alguns desses itens, o background da tela principal deverá mudar de cor para a cor indicada.

```
public class Tela06 extends JFrame {
 private JMenuBar barramenu;
private JMenu menuArquivo, menuEditar;
 private JMenuItem miabrir, misalvar, misair, micopiar, micolar;

public Tela06() {
  super("Menu Simples - GUI");
  criaMenu();
 }
 public void criaMenu() {
              setSize(400, 200);
              setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
              setLayout(new BorderLayout());
              menuArquivo = new JMenu("Arquivo"); //Itens do MENU menuArquivo
              miabrir = new JMenuItem("Abrir");
              menuArquivo.add(miabrir);
              misalvar = new JMenuItem("Salvar");
              menuArquivo.add(misalvar);
              misair = new JMenuItem("Sair");
              menuArquivo.add(misair);
              menuEditar = new JMenu("Editar"); //Itens do MENU menuEditar
              micopiar = new JMenuItem("Copiar");
              menuEditar.add(micopiar);
              micolar = new JMenuItem("Colar");
              menuEditar.add(micolar);
              barramenu = new JMenuBar();
              barramenu.add(menuArquivo);
              barramenu.add(menuEditar);
              setJMenuBar(barramenu);
              setVisible(true);
 }

 public static void main(String[] args) {
           Tela06 janela = new Tela06();
 }
}
```

[Dicas]

(a) Você trata eventos de um JItem da mesma forma que eventos de botão.

(b) É possível mudar a cor do painel principal (ex. para Azul) usando o comando:

```
getContentPane().setBackground(new Color(0,0,255));
```