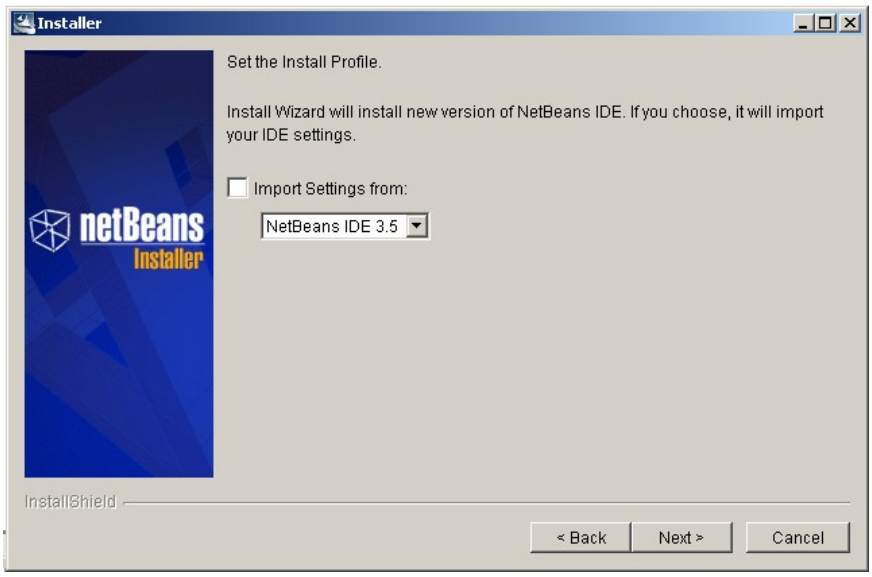
O Netbeans

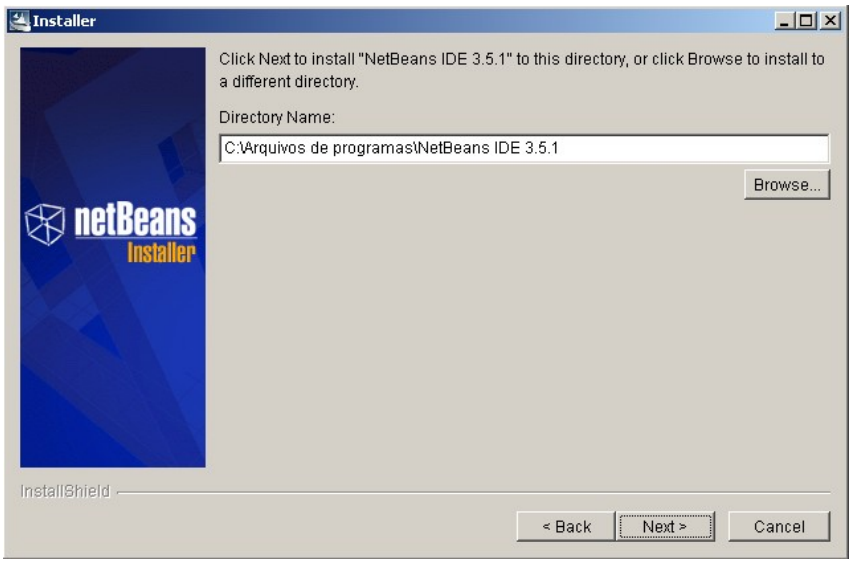
O Netbeans é um ambiente integrado de desenvolvimento (IDE) que permite ao programador criar programas utilizando recursos gráficos.

Para trabalhar com o NetBeans é necessário ter instalado, anteriormente em sua máquina uma das versões do JDK (Java), preferencialmente uma versão igual ou superior a J2SDK1.3.1.

Instalando o Netbeans

- No mesmo site http://java.sun.com/ , faça também o download do NetBeans, caso não tenha um CD de instalação.
- Após o download, dê um duplo clique no ícone criado e siga os passos mantendo os padrões de instalação.
- Segue como exemplo algumas ilustrações da instalação da versão 3.5.1, a versão 4.0 segue o mesmo padrão:



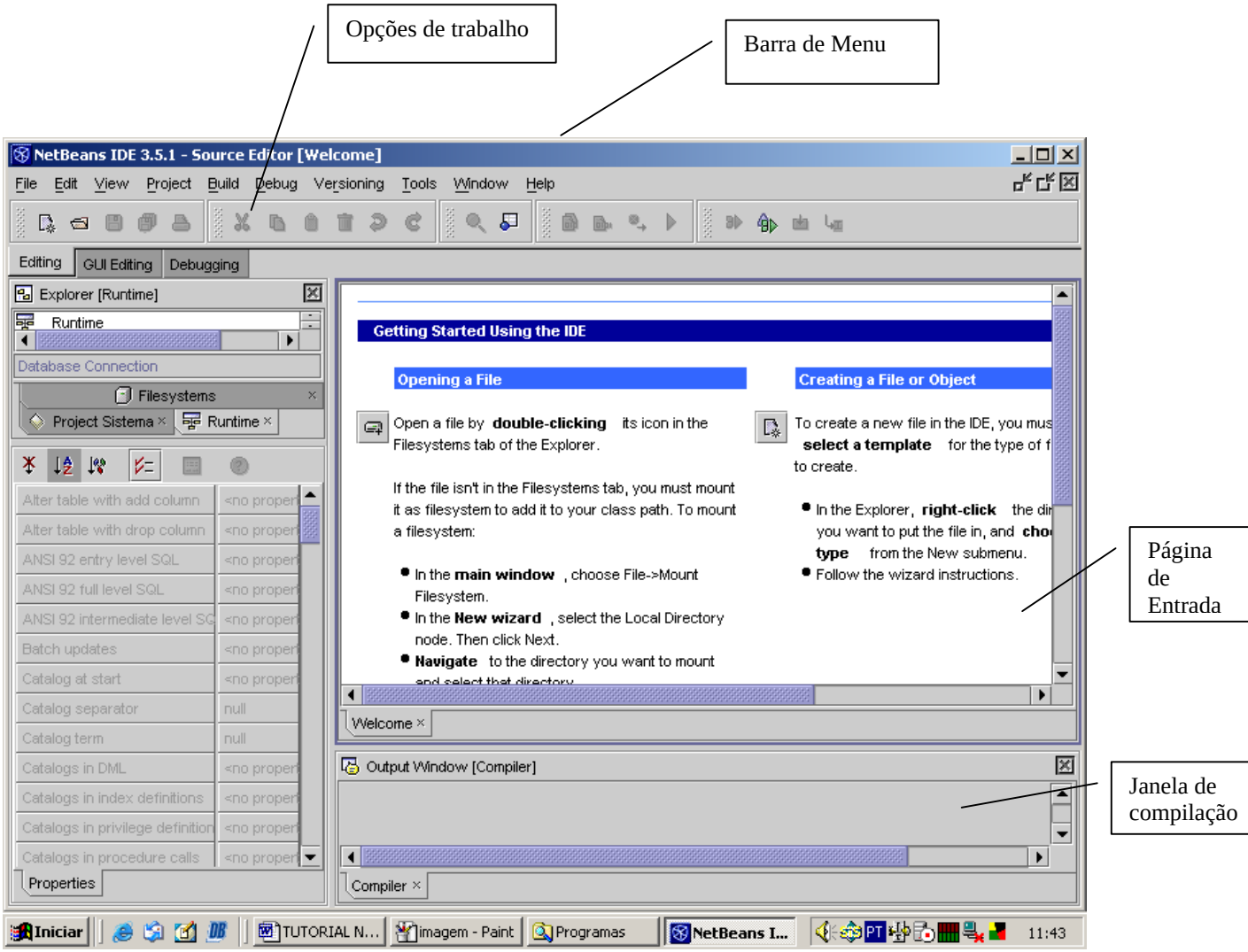


NOTA: Antes de iniciar a construção dos exemplos é interessante conhecer um pouco da sintaxe básica da linguagem para facilitar o seu entendimento.

3. Conhecendo a IDE

- Depois de instalado, execute o NetBeans clicando no ícone na área de trabalho ou clicando no menu iniciar→ Programas→ NetBeans → NetBeans

- O mesmo deverá exibir uma tela como a mostrada a seguir ou algo parecido, isto vai depender do tipo da instalação/ SO/ Número de vezes que o mesmo já tenha sido executado:



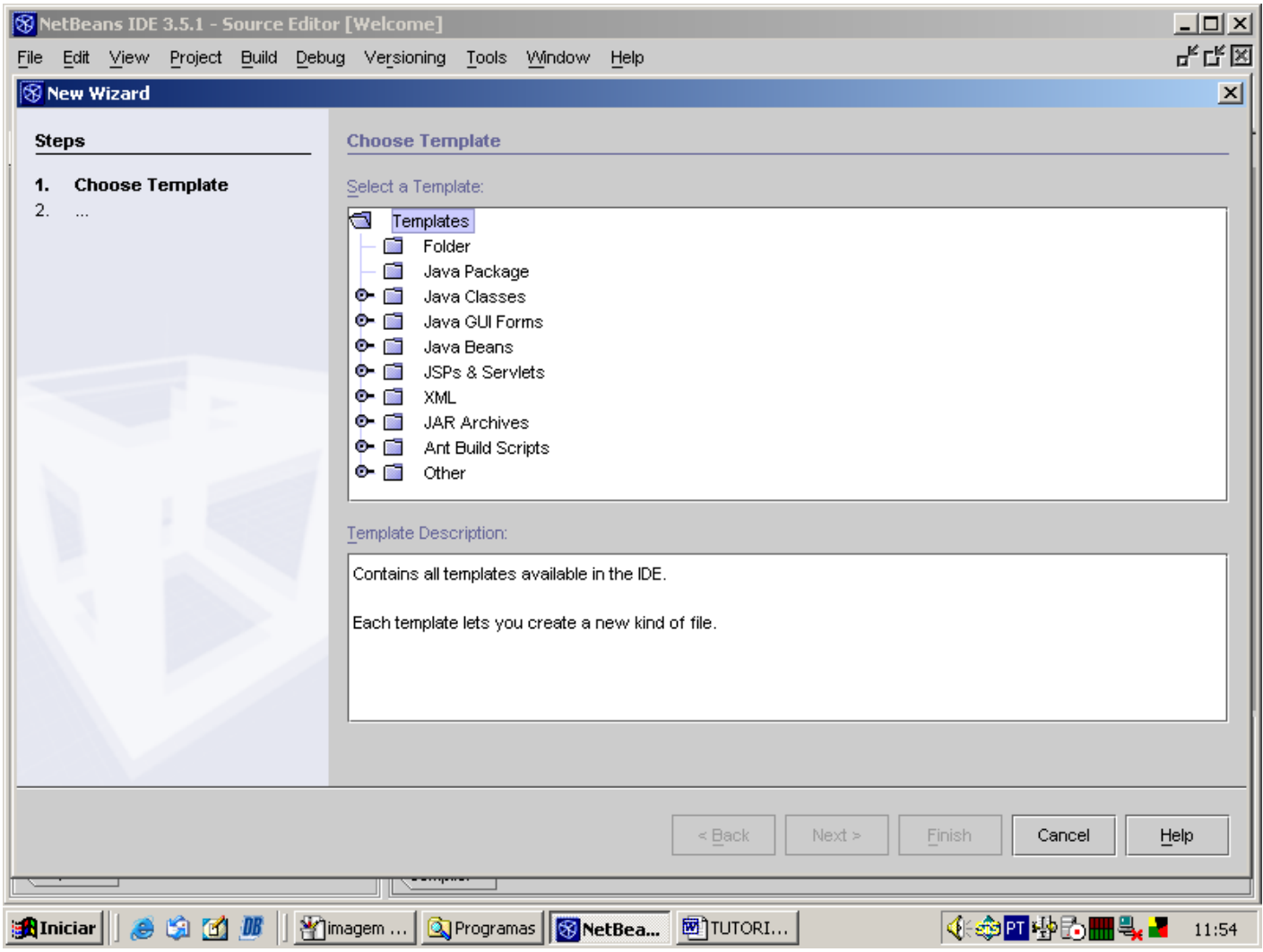
3.1. Criando uma Aplicação MDI:

- Composta de:

1. Um Formulário principal
2. Um Menu

- Passos:

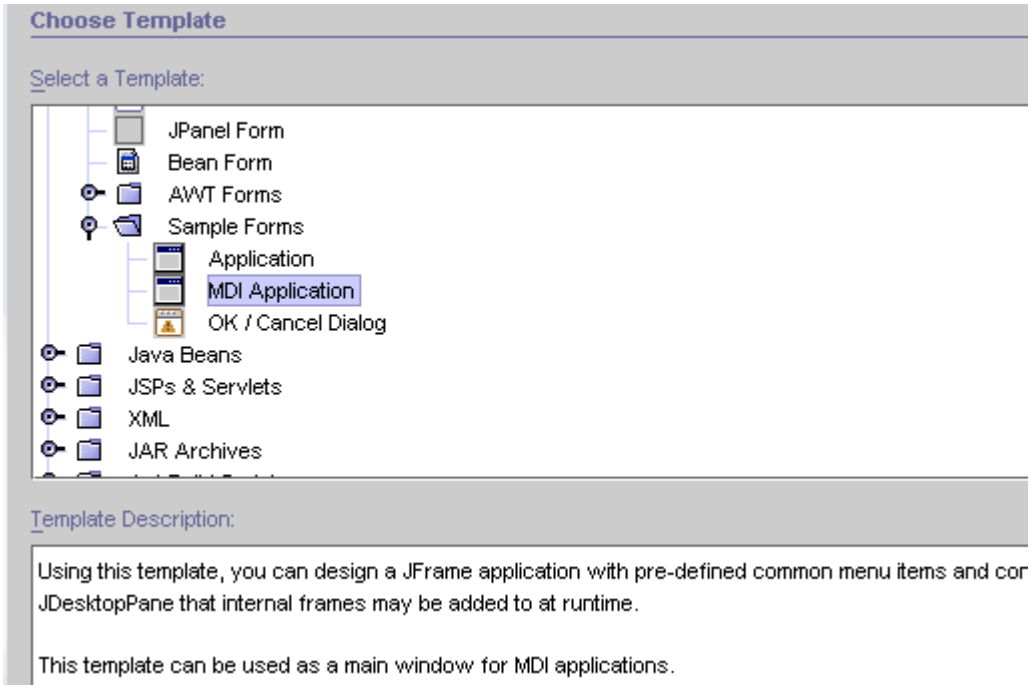
1º - Clique no menu File→ New: será exibida uma tela como esta:



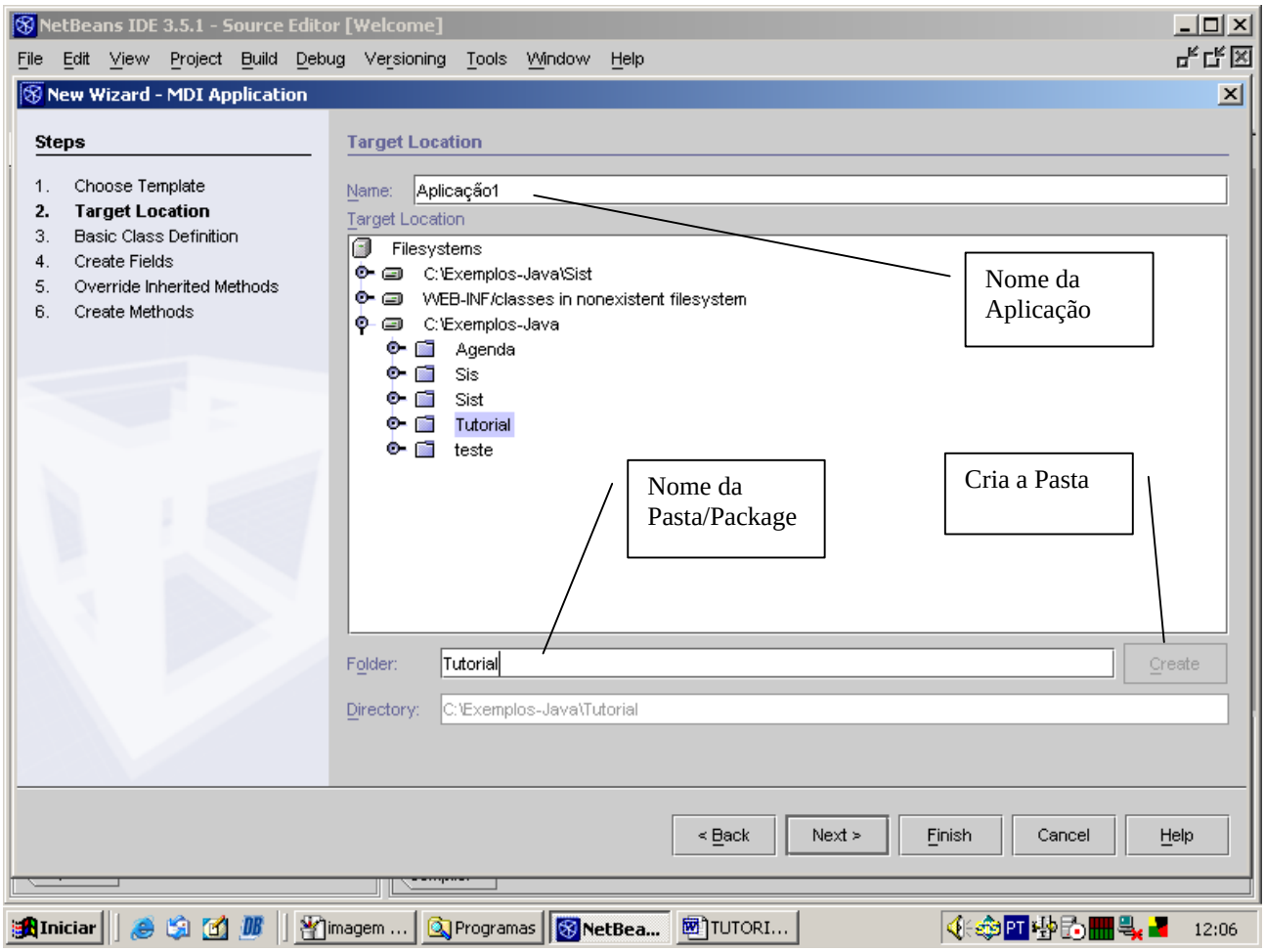
2º - Dê um duplo clique em Java GUI Forms ou clique na respectiva Lupa:

3º - Selecione Sample Form → MDI Aplication – com isto criaremos uma aplicação composta de um formulário com um menu e que aceita a abertura de outros formulários

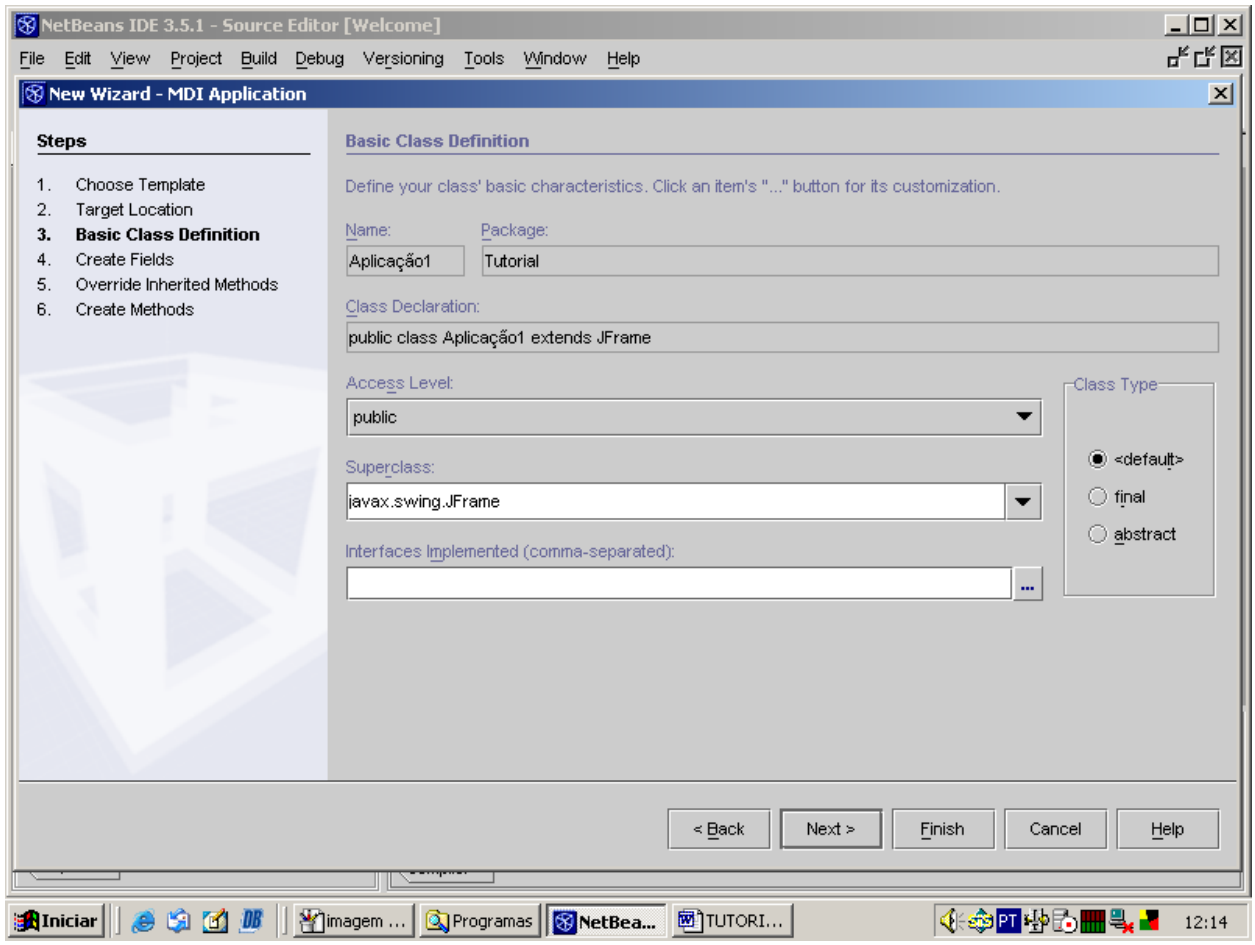
4º - Clique no botão NEXT:



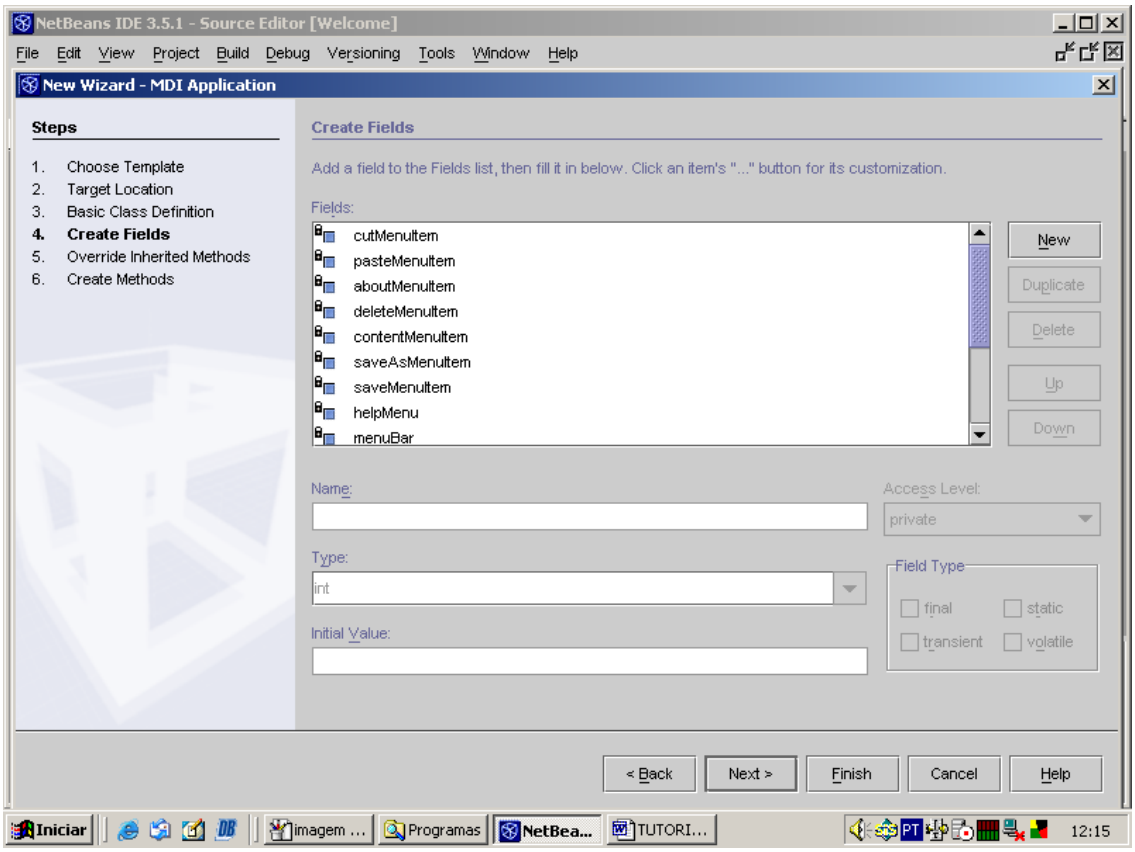
5º - Dê um nome para o seu Aplicativo e no Campo Folder entre com um nome que será a pasta onde a aplicação ficará armazenada - Package. Clique no botão Create → Next:



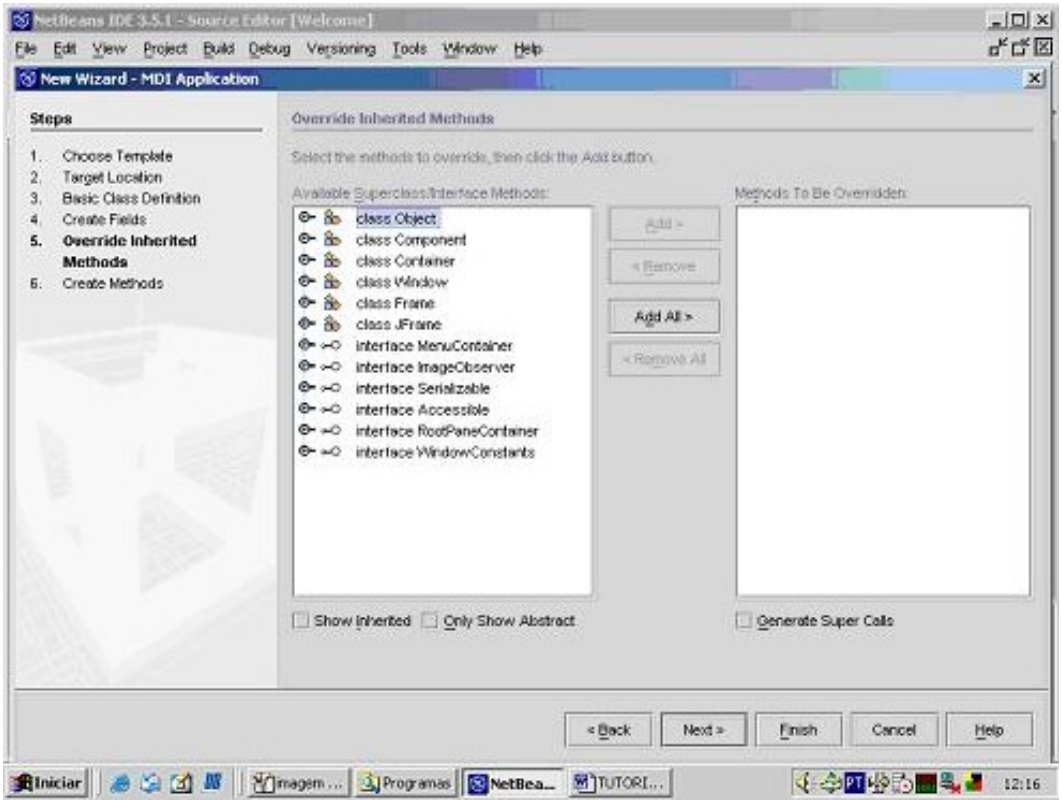
6º - Será mostrada uma tela informando o nome da Aplicação e o nome do Package criado, a superclasse que será extendida e as classes a serem implementadas. Não mude as opções padrão, clique no botão Next:



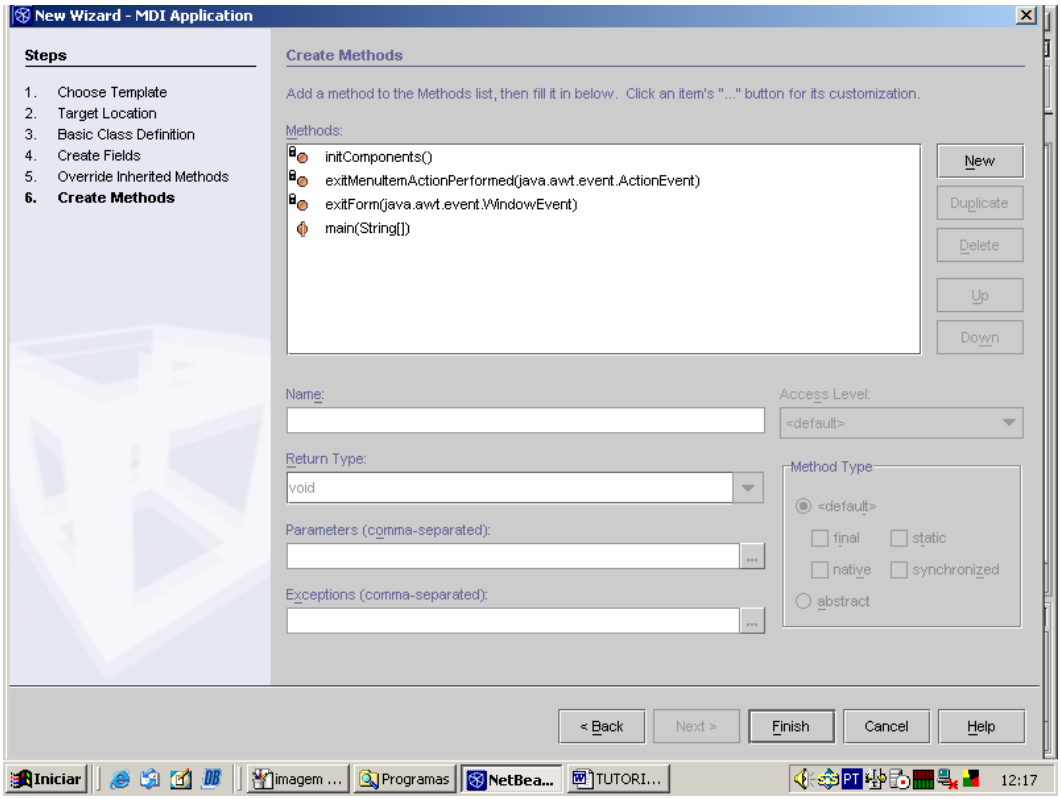
7º - Clique no botão Next:



8º - Clique no botão Next:



9º - Clique no botão Finish:



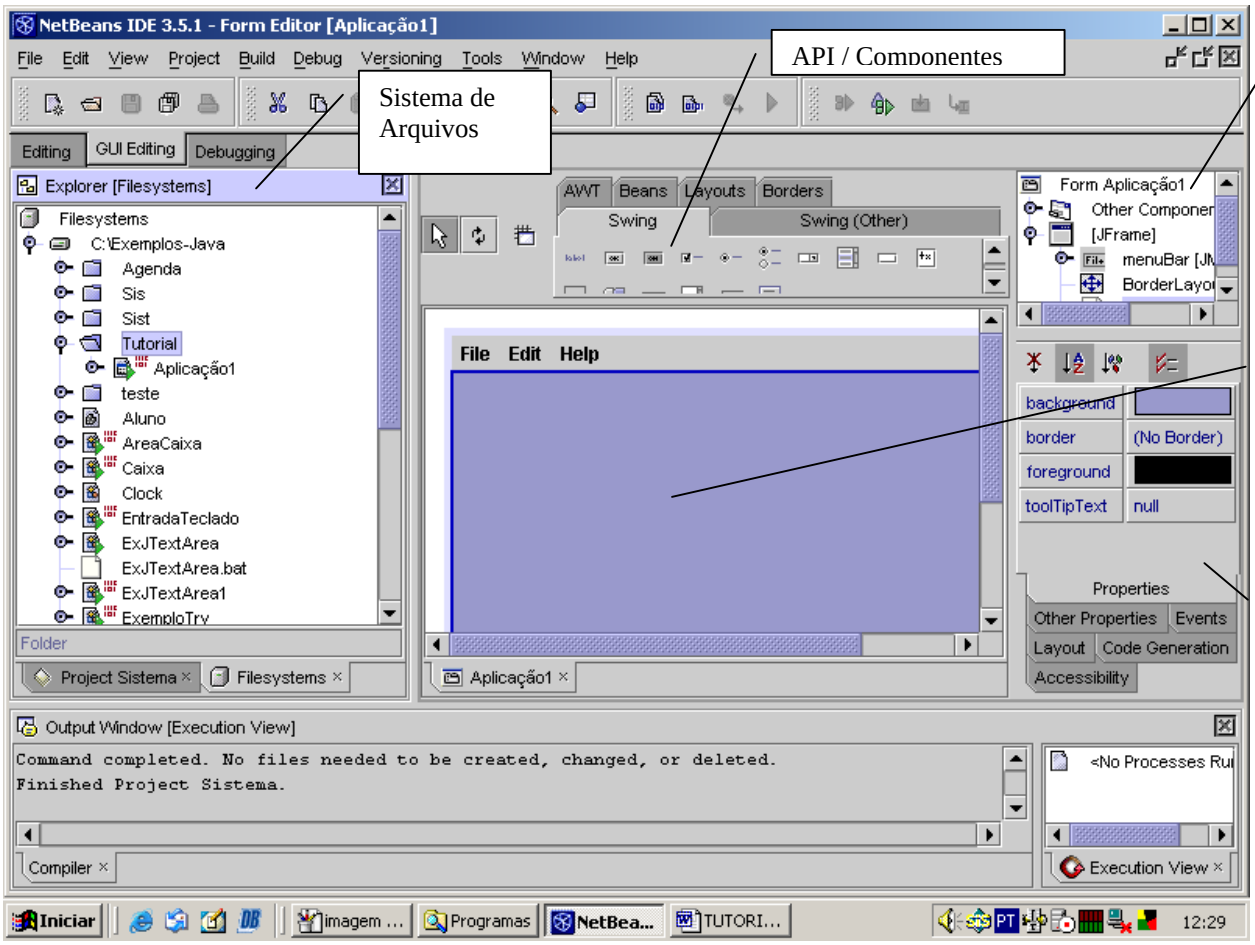
A IDE abrirá uma tela como a que se segue. O NetBeans trabalha com varias mini-telas com finalidades especificas, todas de forma integradas a janela principal, a barra de menu.

- Caso queira aumentar ou diminuir a área de visualização de uma destas basta passar o mouse e redimensiona-las de acordo com a necessidade do programa.

- Para visualizar uma destas janelas clique no Menu View e selecione a Janela desejada. Ex: Source Editor ou Form Editor

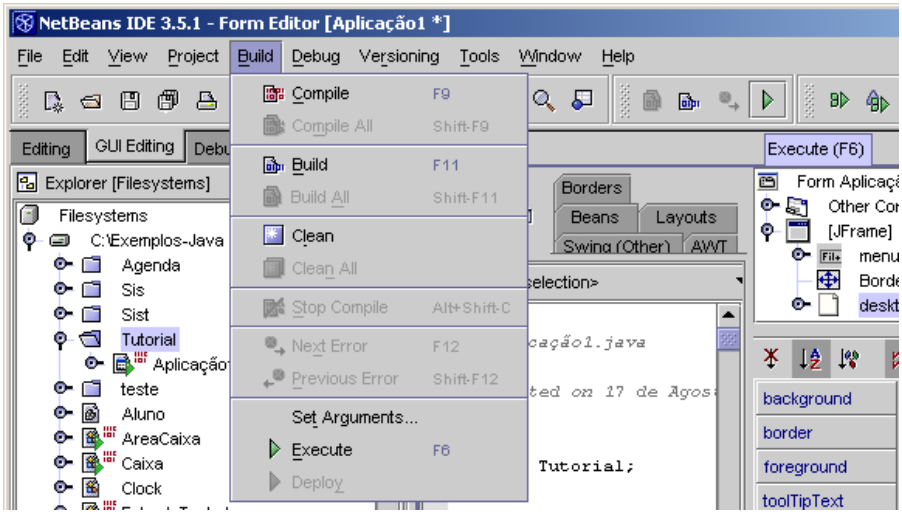
10º - Feche a Janela Explorer (File System) – clique no X, para aumentar a área do formulário de Trabalho:

3.2 Componentes da Janela em modo de Trabalho

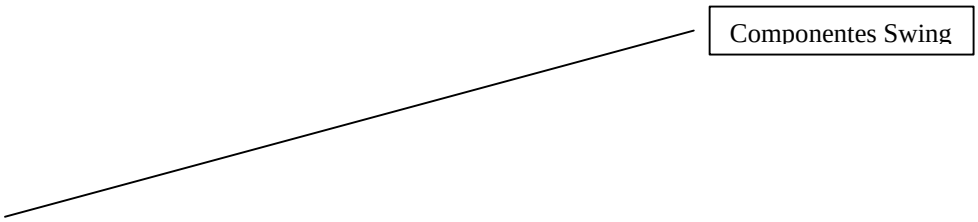


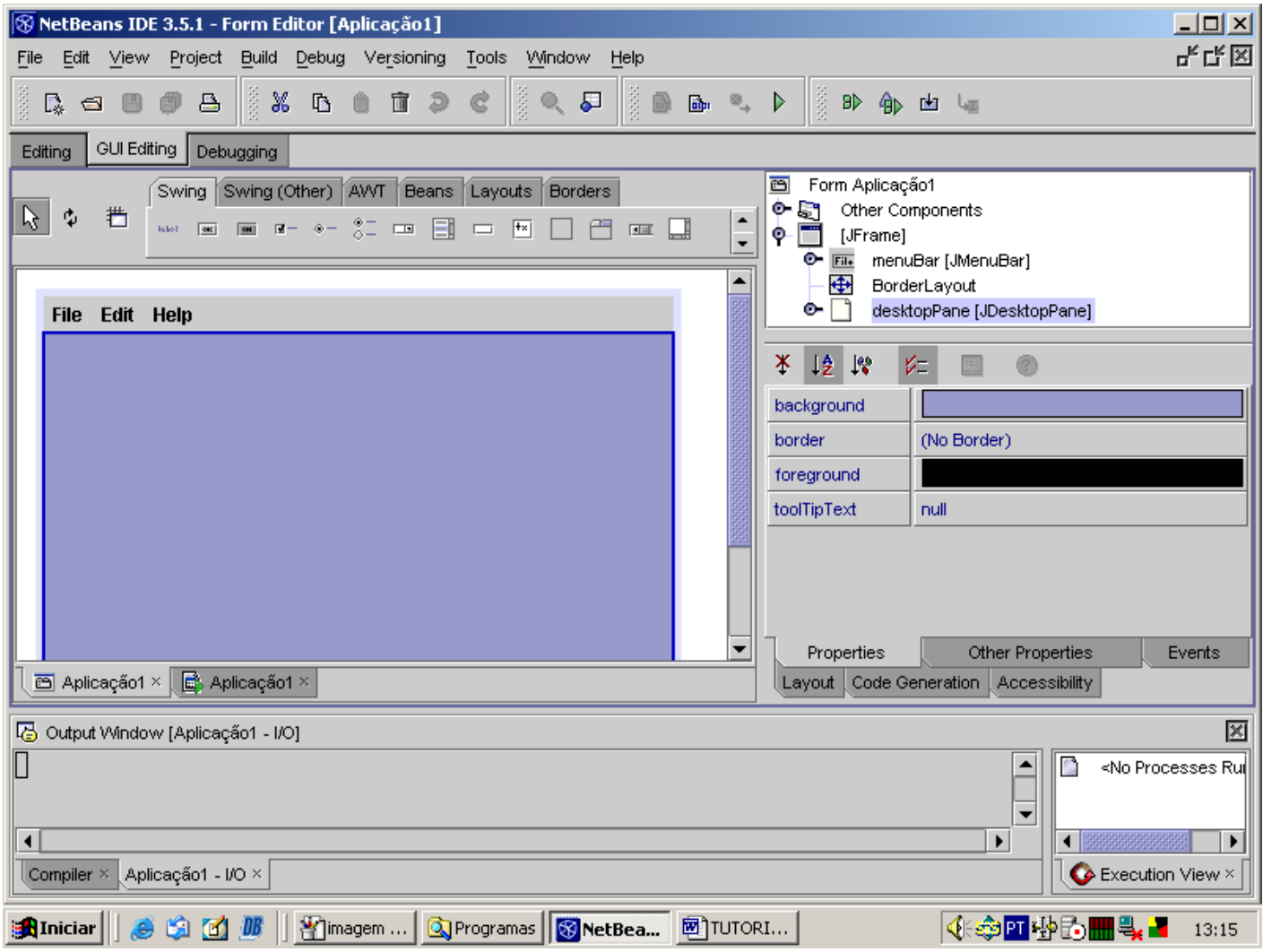
3.3 Compilando: Clique no Menu Build→ Compile ou Pressione F9

3.4 Executando: Clique no Menu Build → Execute ou Pressione F6



Na execução da aplicação o formulário / janela abre exibindo apenas o menu, isto porque não tem nenhum componente, verifique o menu sair, observe que o mesmo já foi implementado pelo NetBeans. Agora vamos inserir alguns componentes, procure deixar a sua aplicação parecida com a seguinte Tela no modo formulário. (Lembre-se que é possível alternar entre as telas de código e formulário clicando no menu View e selecione Form Editor/Source Editor):





3.5 Trabalhando com componentes:

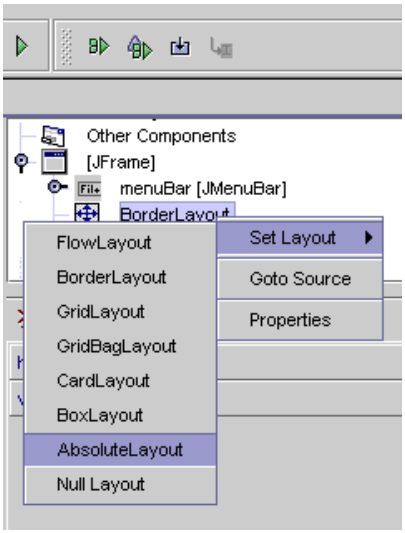
- Na janela no canto superior direito, que mostra os componentes da aplicação, clique o Botão direito do mouse sobre BorderLayout e selecione Set Layout →AbsolutLayout

---

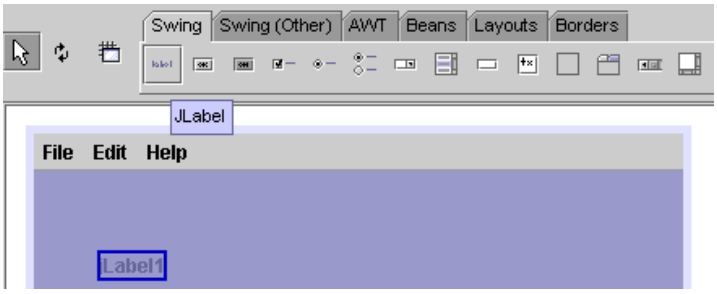
NOTA: - Esta mesma operação pode ser efetuada clicando-se com o botão direito do mouse sobre a área do Formulário e selecionando SetLayout → AbsolutLayout.

- É necessário modificar o Layout para que se possa colocar os componentes (botões, Labels, etc) na posição desejada, isto porque o Formulário (JFrame/Frame) quando é criado tem como padrão o Layout BorderLayout que

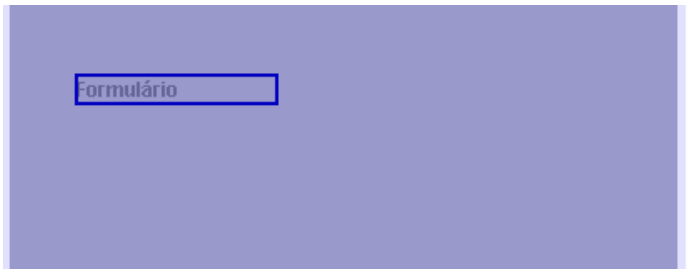
trabalha de forma diferente. Mais a frente será abordado de uma forma melhor sobre os Layout.

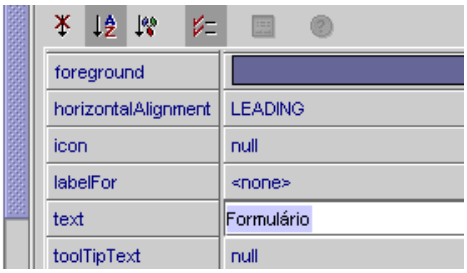


- Passe o mouse sobre os componentes da Aba Swing e observe que os mesmos são selecionados, clique no correspondente ao JLabel e clique no Formulário:



- Na Janela Propriedades clique em Text e insira o texto que deseja que apareça no JLabel:

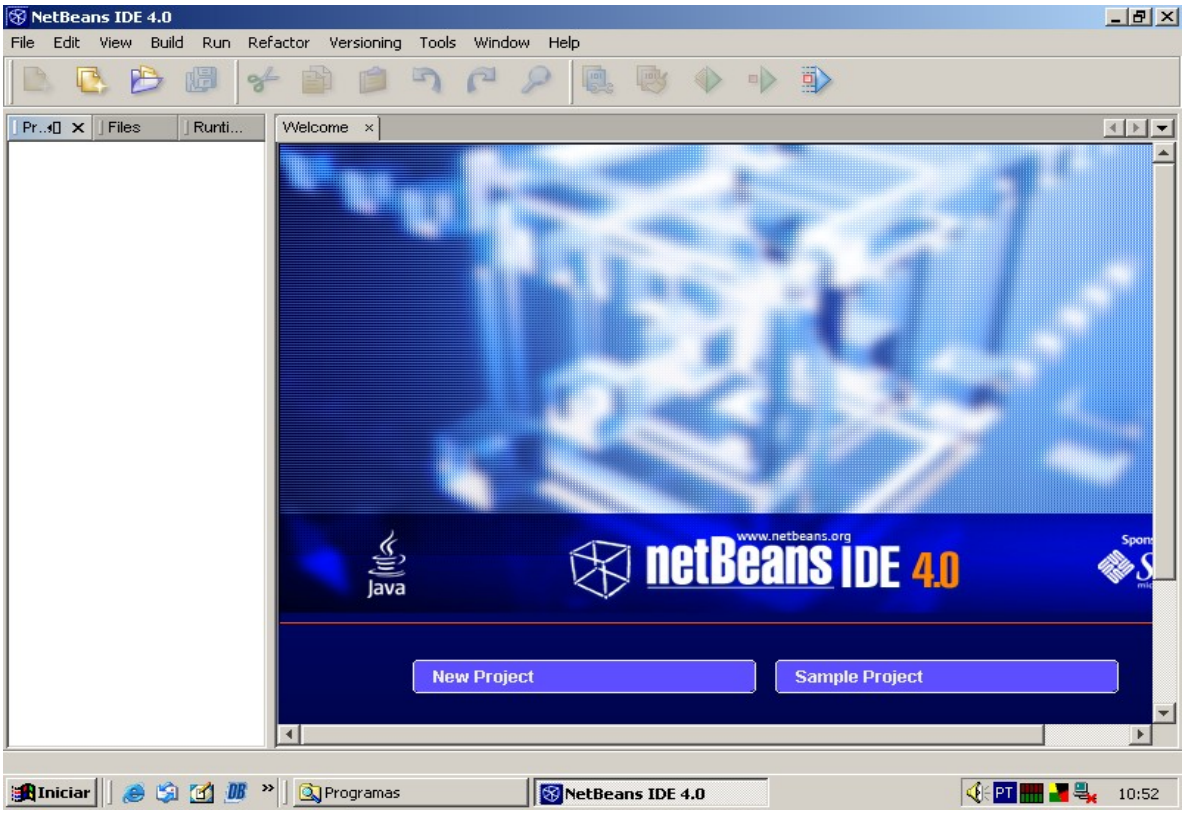




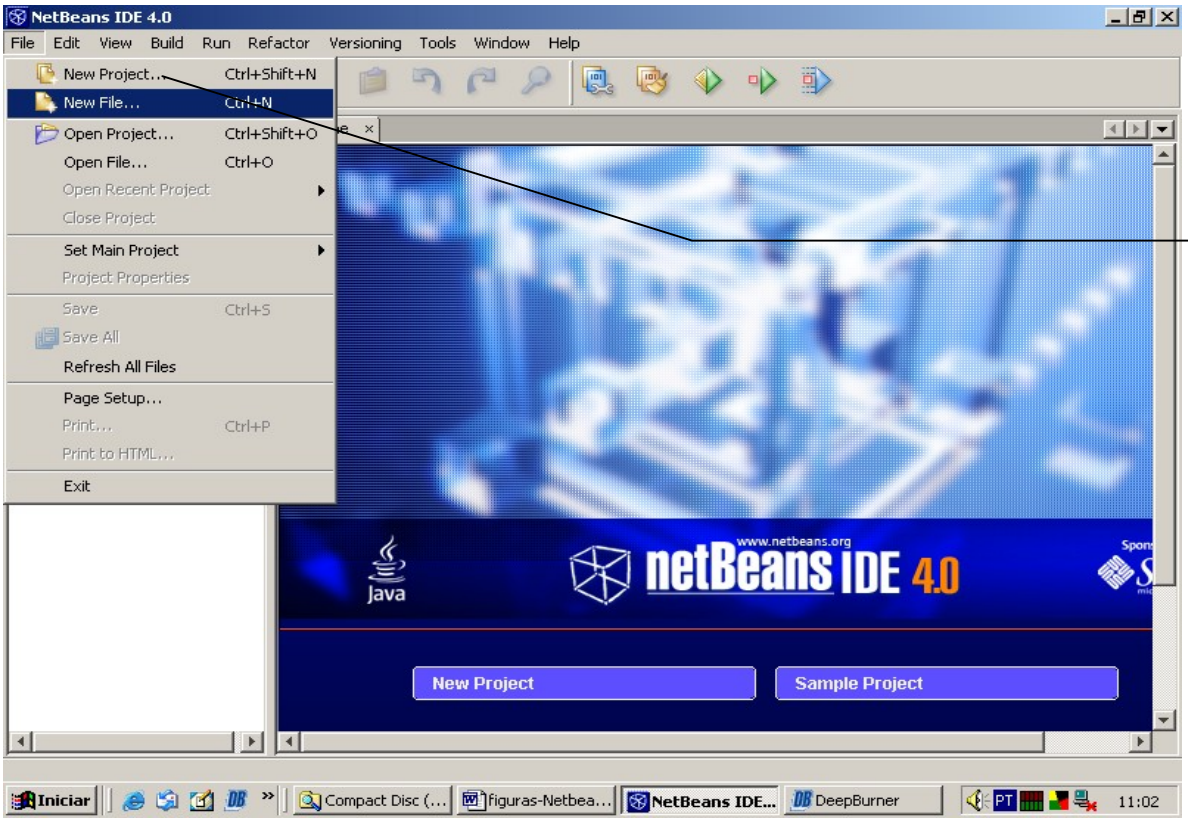
- Insira outros componentes e mude suas propriedades realizando experiências, compilando e executando, para visualizar os resultados.

Telas do NetBeans - Versão 4.0

- Tela de Inicialização/Abertura

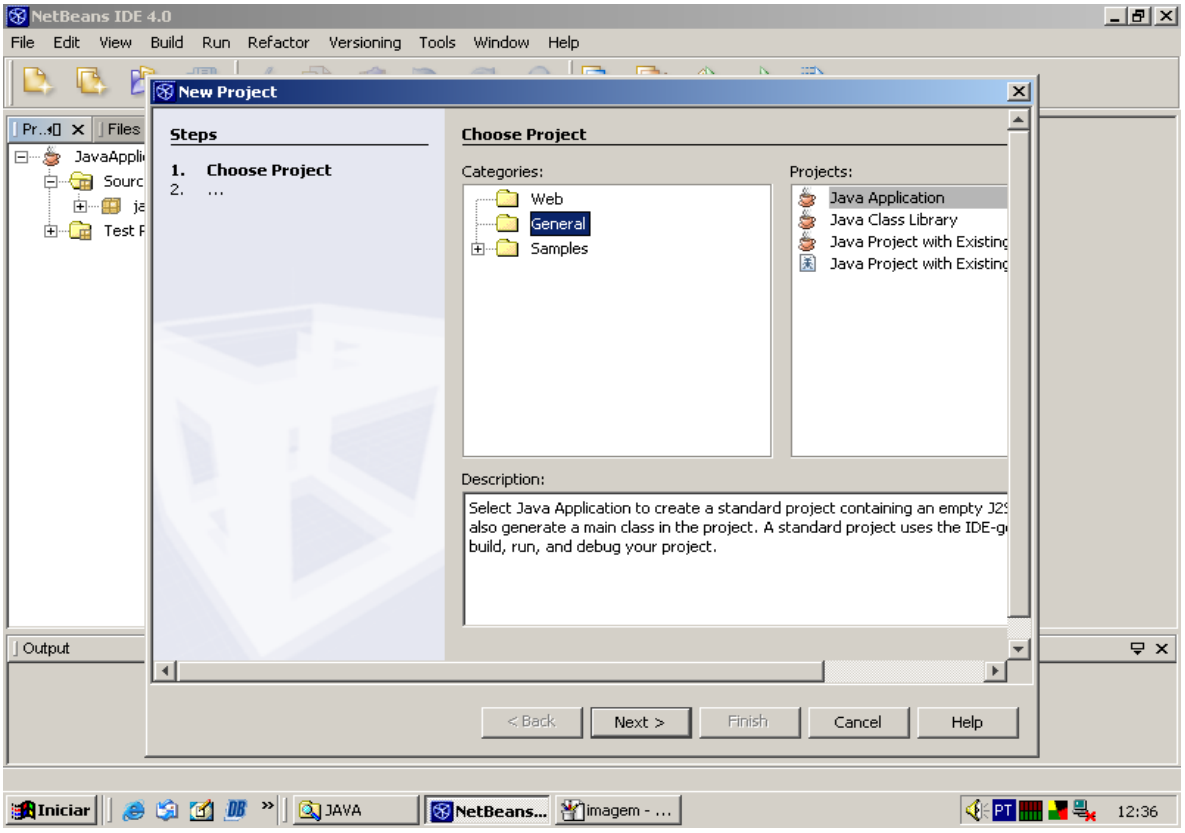


- Criando uma Aplicação: - primeiro é necessário criar um projeto

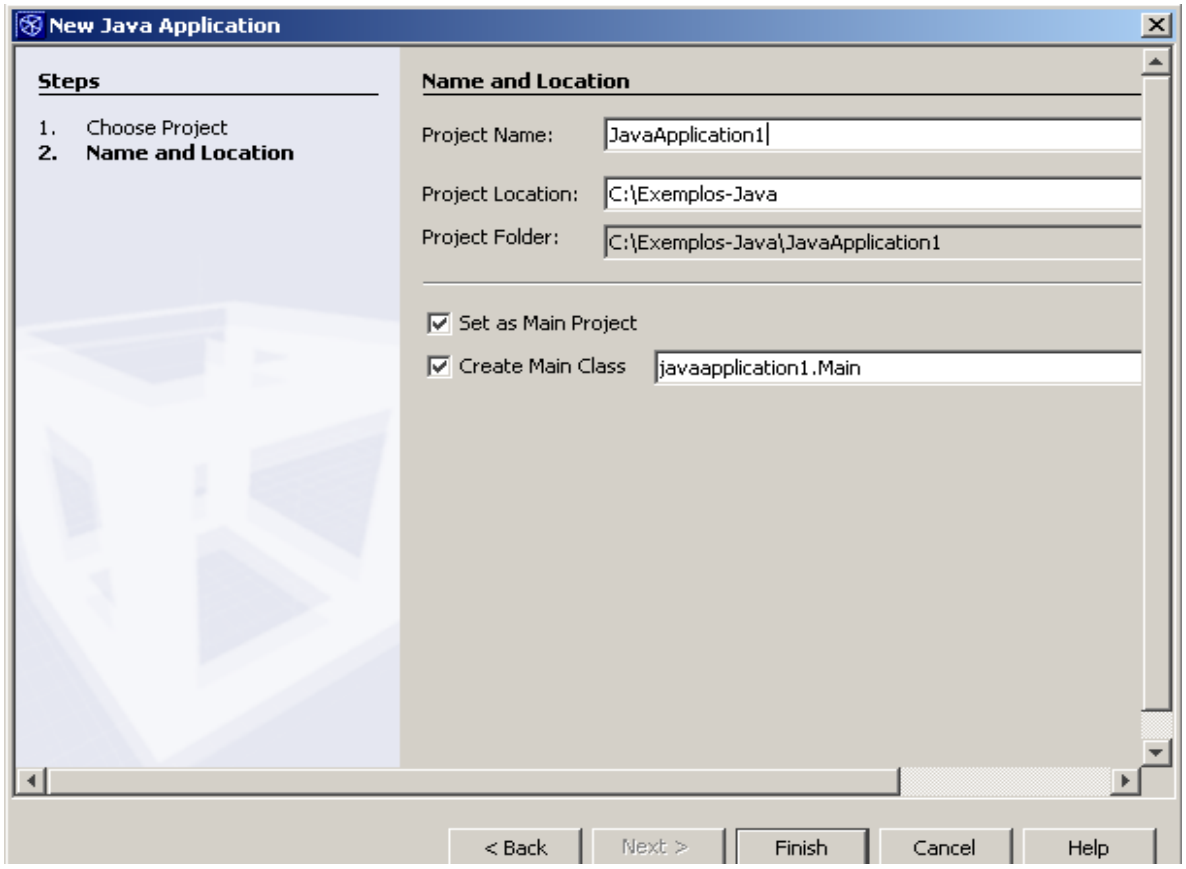


Cria um Projeto

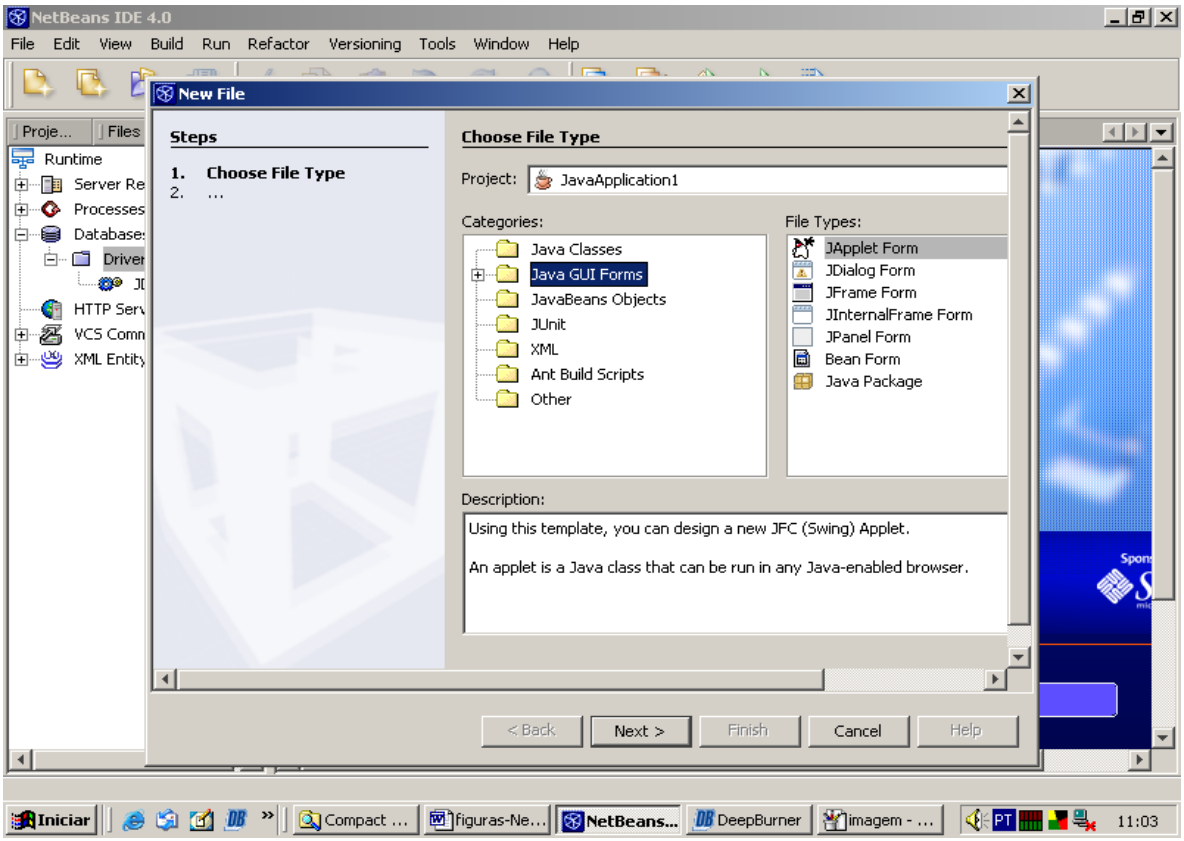
- Selecione General→ Java Application → Next



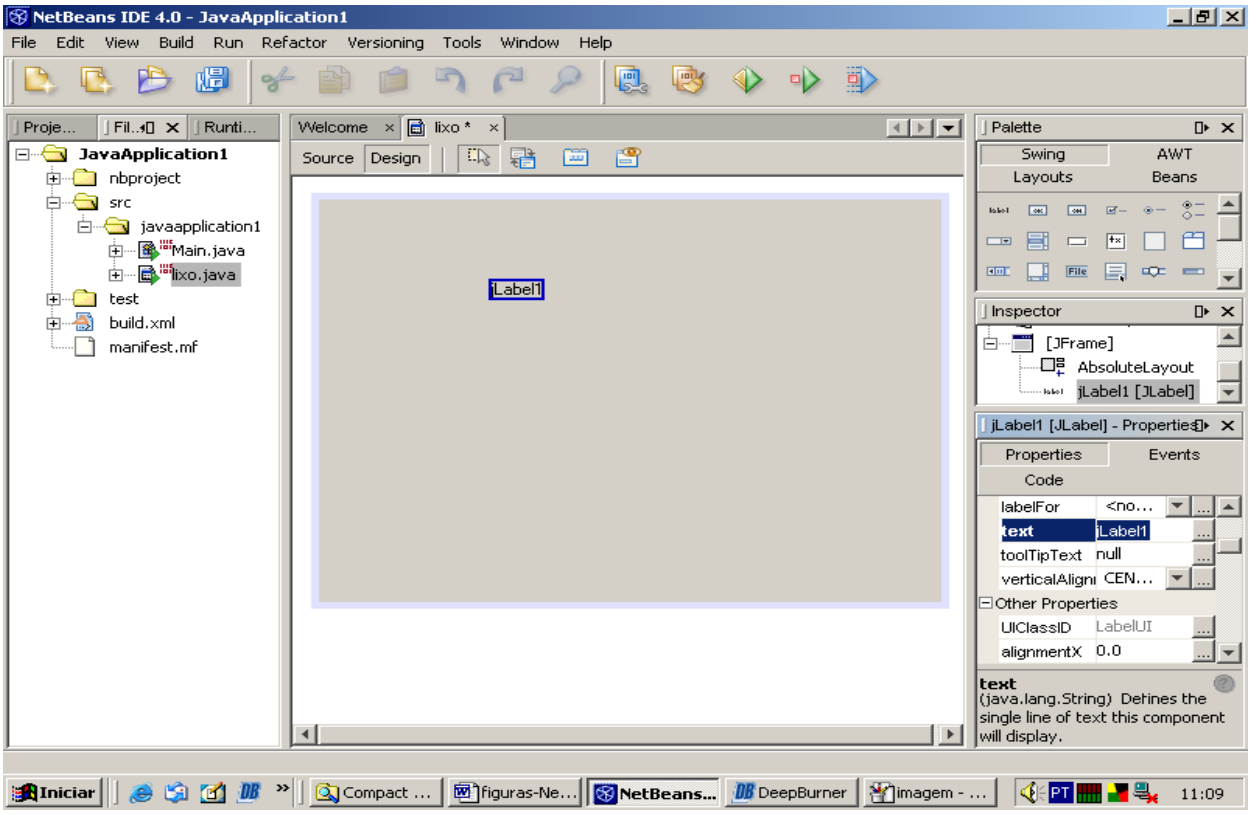
- Em “Project” digite o nome Desejado



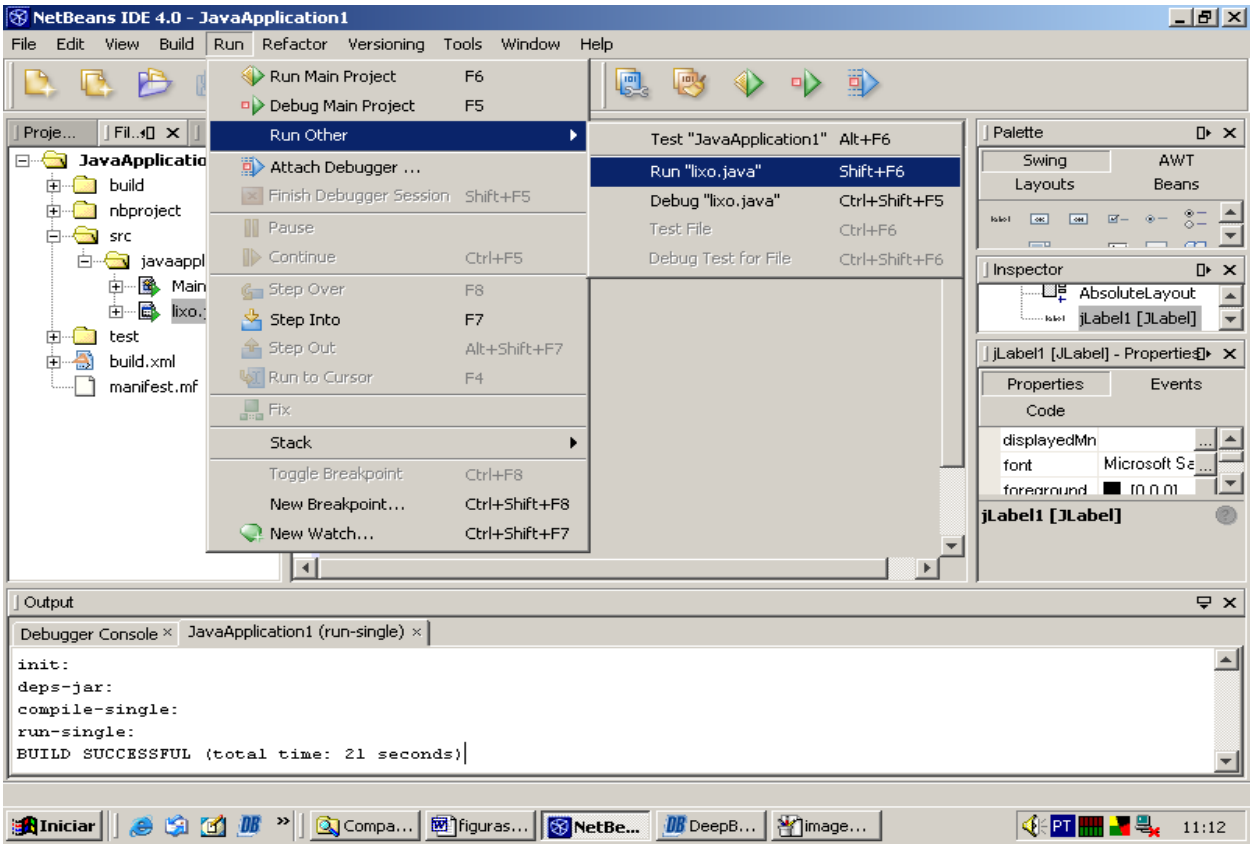
- Criando um Programa: File→ New File→ Java GUI Form→ JFrame→ Next



- Na nova janela que se abre digite o nome do Programa e clique no Botão Finish
- Insira os componentes desejados



- Compile e Execute: Menu “Build”→ Compile; Para executar Menu “Run”→Run Other



NOTA: Os exemplos que serão apresentados foram implementados utilizando o NetBeans 3.5.1, que se diferencia visualmente muito pouco em relação a versão “4.0”, posto que serão exemplos simples. Se for utilizar a versão “4.0” os passos a serem executados são os mesmos, salvo exceção quanto à criação do Projeto que já foi demonstrado.

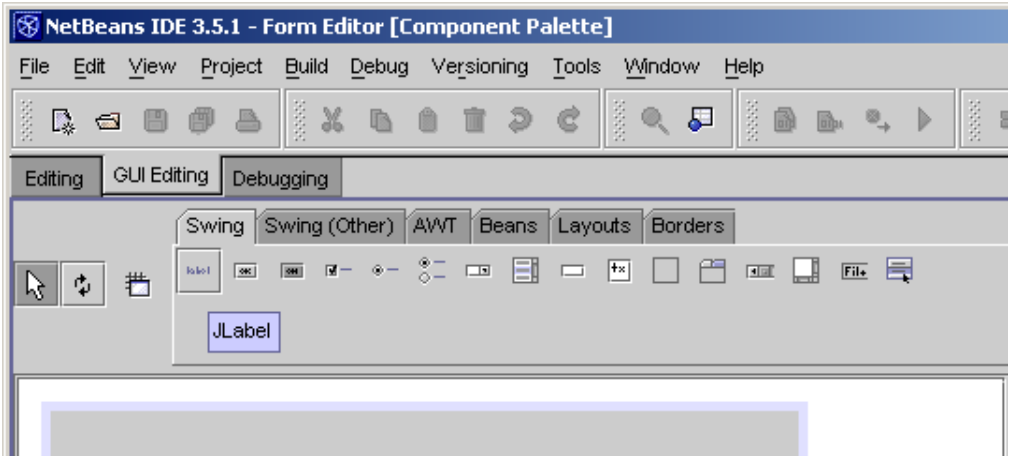
4. Aplicações Gráficas com Swing

- Assunto: Aplicações Gráficas com Swing
- Objetivo: criar aplicações Gráficas utilizando Swing

Aplicações Gráficas com Swing

- Nova família de componentes com funcionalidade ampliada e grande capacidade de configuração
- O seu propósito é permitir a criação de interfaces versáteis e sofisticadas
- Surgiu em 1997 com a JFC (Java Foundation Classes), com o Java 2, que engloba:
    - Componentes Swing
    - Compatibilidade com múltiplas ´Look and Feel` - “peles”
    - Biblioteca de acessibilidade – monitores e teclados especiais
    - Biblioteca Java 2D
    - Compatibilidade com `Drag and Drop`

Principais Componentes



- Os seus componentes são semelhantes ao da AWT (Abstract Window Toolkit), pois o modelo de eventos é o mesmo.

- Diferenças: Componentes Swing são mais flexíveis e mais numerosos
- Não utilizam código nativo, seus nomes iniciam com um **J**
- A base para uma aplicação é o JFrame ou JWindow (janela) ou a classe JApplet (miniapplicativos).
- JFrame não é um mero container, mas um painel especial que agrega três outros componentes em camadas
- Destes o que nos interessa é o ContentPane (painel de conteúdo), onde vamos inserir os demais componentes:

  Container conteudo= getContePane();

a) JLabel: rótulo de texto.

Métodos específicos:

String getText()⇒ retorna o texto do label
void setText(String lbl) ⇒ ajusta o texto do label para lbl

b) JButton: é um botão simples que pode ser criado com ou sem rótulo.

Métodos específicos:

String getText()⇒ retorna o label(etiqueta) do botão
void setText(String etiq) ⇒ ajusta label do botão para o conteúdo de etiq

c) JTexField e JTextArea

- TextField: caixa de entrada de texto, possibilita a entrada e a edição de uma linha de texto.
- TextArea: caixa de entrada de texto com múltiplas linhas. Exibe barra de rolagem horizontal e vertical.

- principais métodos:

  String getText()⇒ retorna o texto contido no TextField
  void setText(String txt) ⇒ ajusta o texto da TextField para txt

d) JList e JCombobox

- JList: caixa de lista que permite a exibição de uma lista de itens que não podem ser editados diretamente pelo usuário.
- JCombobox: implementa uma lista de itens em que um único item selecionado é exibido.
- principais métodos:

  int getSelectedIndex();⇒ retorna índice do item selecionado
  String getSelectedItem();⇒ retorna o nome do item selecionado
  void select(String str); ⇒ ajusta o item selecionado para str

→ MÉTODOS COMUNS A TODOS OS COMPONENTES

void resize(int width, int height) ⇒ Tamanho do componente

void move(int x, int y) ⇒ Mover componente

void setForeground(Color x) ⇒ Cor do componente

void setBackground(Color y) ⇒ Cor de Fundo do componente

void disable() ⇒ Desabilitando componente

void enable() ⇒ Habilitando componente

Gerenciadores de Layout

Gerenciamento de layout (Layout Management) é o processo de determinar o tamanho e a posição dos componentes na janela gráfica do programa, ou seja determinar onde os componentes irá ficar guiando a maneira como os elementos de interface serão dispostos dentro do container (Frame, Panel,Window).

Existe basicamente os seguintes tipos de layout:

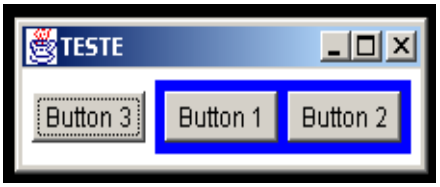
a) FlowLayout
b) BorderLayout
c) CardLayout
d) GridLayout
e) GridBagLayout

A escolha do gerenciador de layout depende muito das necessidades do programa.

a) FlowLayout

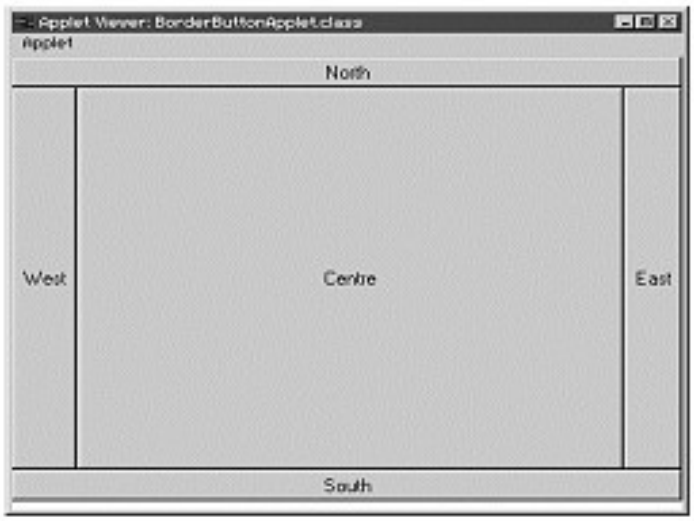
Simplesmente posiciona os componentes da esquerda para a direita, criando novas linhas se necessário.

EX:



b) BorderLayout

É o Padrão: Divide a janela em cinco áreas nas quais os componentes podem ser exibidos: norte, sul, leste, oeste e centro.



c) CardLayout

Permite apresentar dois ou mais componentes (geralmente painéis) compartilhando o mesmo espaço de apresentação. Funciona como uma pilha de cartas onde apenas uma fica visível.

- Cria-se um Painel fixo de Controle e um genérico (CardLayout) para conter outros paineis

d) GridLayout

Deixa todos os componentes com igual tamanho, exibindo-os como uma tabela (linhas e colunas).

EX: substitua a linha 9 do Programa Panel.java por:

```
setLayout( new GridLayout(3,1));
```

e) GridBagLayout

É o mais flexível dos gerenciadores de layout, permite colocar componentes em grades de colunas, sendo possível um componente ocupar mais de uma coluna ao mesmo tempo. As linhas também não precisam necessariamente ter os mesmos tamanhos, ou seja, você pode configurar diferentes larguras e alturas de acordo com a necessidade. No entanto, é o mais difícil de ser implementado.

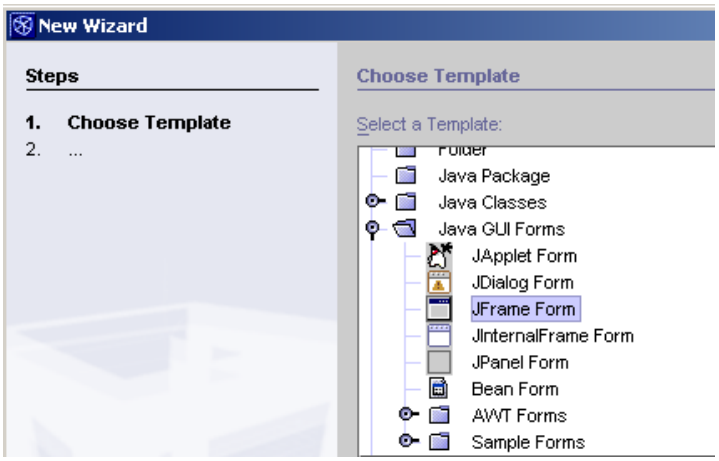
5. Aplicações Gráficas com Swing – Construindo Aplicações com o NetBeans

- Assunto: Aplicações Gráficas com Swing – Construindo aplicações com o NetBeans
- Objetivo: criar aplicações Gráficas utilizando Swing

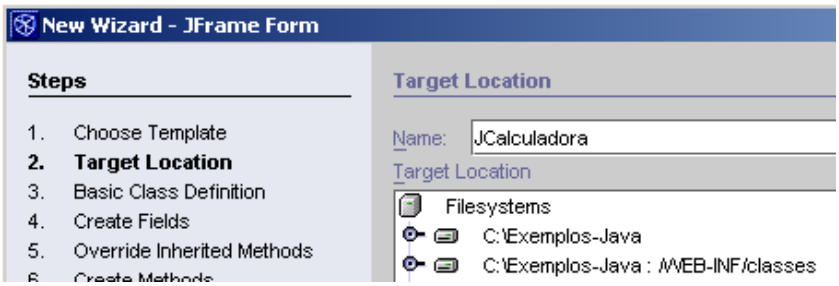
Utilizando JLabel, JTextField e JButton com o NetBeans

1. Crie uma aplicação para somar os valores de duas Notas Fiscais a serem informados pelo usuário:

- Nesta Aplicação utilizaremos dois JTextField (onde o usuário irá digitar os valores) um JLabel (mostrar o resultado da soma) e um botão a ser clicado para efetuar a soma
- Abra o NetBeans → Clique no Menu File → New → Dê um Duplo clique em Java GUI Forms → Selecione JFrame Form → Clique no Botão Next



- Na nova janela que se abre Digite “JCalculadora" → Clique no Botão Finish, seu projeto será iniciado



- Antes de inserir os componentes, devemos modificar o Layout para AbsoluteLayout ou para NullLayout, para podemos criar o nosso Layout.

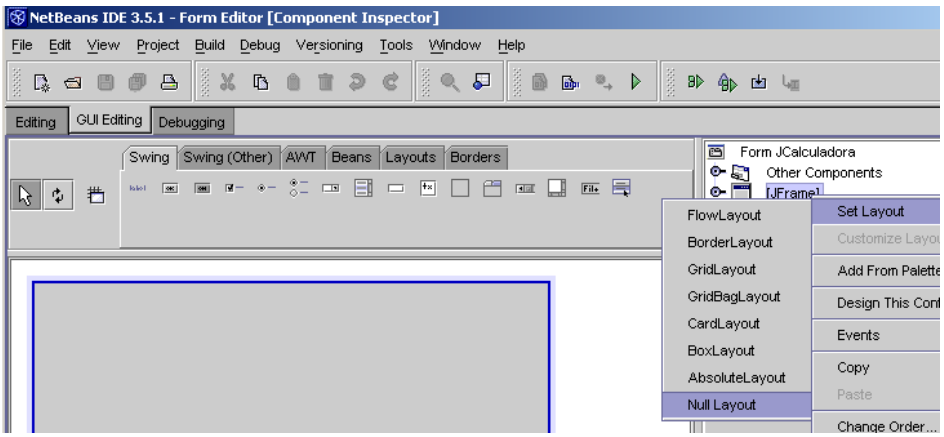
NOTA: - O AbsoluteLayout é um gerenciador de Layout criado especificamente para o NetBeans, portanto se for utiliza-lo sem o NetBeans terá de acrescentar o respectivo pacote e distribuí-lo juntamente com o seu programa. Caso contrário o mesmo não executará corretamente.

- O Null Layout não é um gerenciador de Layout, mas corresponde à situação em é desativado uso dos gerenciadores, neste caso a posição dos componentes é explicitamente definida através de métodos específicos que a ferramenta se encarrega de manipular, mas os mesmo não vale para o Formulário (Frame/JFrame), onde se faz necessário a utilização do método setBounds:

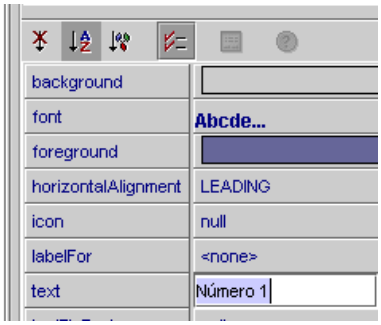
Ex: setBounds(10,10,300,400);

// abre o tela na posição largura 10, altura 10, com um largura de 300 e altura 400.

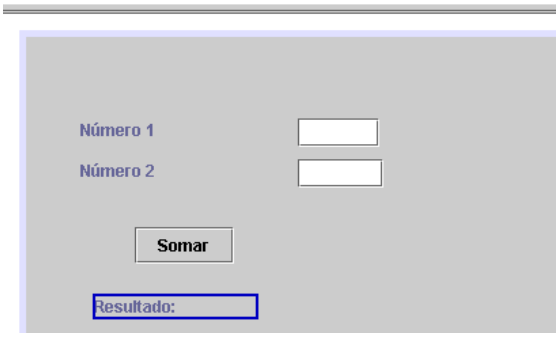
- Na janela, no canto superior direito, Clique com o botão direito do mouse sobre "JFrame" → setLayout → clique em NullLayout



- Na janela do Centro do vídeo, embaixo da Aba “Swing”, existem vários ícones representando os respectivos componentes, passe o mouse sobre os mesmos e aparecerá o nome, clique no primeiro, “JLabel", e clique no Form, Clique no segundo, “JButton”, e clique no Form, Clique no nono, “JTextField”, e clique no Form (dois JTextField), insira mais dois JLabel e outro JtextField.
- Na janela Propriedade modifique as propriedades dos componentes de acordo com a tabela:



Componente	Propriedade	Valor
JFrame	Title	Calculadora
JLabel1	Text	Número 1
JLabel2	Text	Número 2
JLabel3	Text	Resultado
JTextField1	Text	“ “
JtextField2	Text	“ “
JButton1	Text	Somar



- A parte visual já foi criada precisamos implementar o código no botão para efetuar a soma (“pegar” os números e somá-los)
- Dê um Duplo clique no botão e na janela de código que aparece digite:

double num1=Double.parseDouble(jTextField1.getText()); // converte texto para double
double num2=Double.parseDouble(jTextField2.getText());
double result = num1 + num2;
String R= String.valueOf(result); //Converte o Resultado para String
jLabel3.setText("Resultado: "+R); //exibe no JLabel o Resultado

- Após digitar os comandos o arquivo de código deverá ficar da seguinte forma:

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
double num1=Double.parseDouble(jTextField1.getText()); // converte texto par
double num2=Double.parseDouble(jTextField2.getText());
double result = num1+num2;
String R= String.valueOf(result);    //Converte o Resultado para String
jLabel3.setText("Resultado: "+R); //exibe no JLabel o Resultado
        // Add your handling code here:
    }
```

- Observe que os nomes dos componentes (jLabel3) começam com letras minúsculas.
- Parte do Código com a declaração dos componentes criados:

```
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    // End of variables declaration
```

- A parte do Código que estiver na cor azul não aceita modificações.

- Se for utilizado o Layout AbsoluteLayout basta compilar e executar o programa para o mesmo funcionar normalmente.
- Mas, como o Layout que está sendo utilizado é o Null Layout, quando o programa for executado a janela aparecerá mostrando apenas a barra de titulo.
- Para a janela abrir e mostrar todos os componentes, devemos acrescentar o comando com a localização e tamanho da janela (setBounds(posX,posY,Largura,Altura)), procure a seguinte linha de código:

```
/** Creates new form JCalculadora */
public JCalculadora() {
    initComponents();
}
```

- Após initComponents(), insira o seguinte comando:

setBounds(10,10,300,400);

- Após o comando o arquivo de código ficará desta forma:

```
/** Creates new form JCalculadora */
public JCalculadora() {
    initComponents();
    setBounds(50,50, 450, 300);
}
```

- Se precisar voltar para o modo Form Editor clique no Menu View → Form Editor ou “Crt+8”
- Para ir para o código clique no menu View→ Source Editor ou “Crt+3”.
- Compile Menu Build → Compile (F9), execute Menu Build → Execute (F5)

Criando uma Calculadora

- Utilize o programa anterior e acrescente mais 4 (quatro) JButtons:

Componente	Propriedade	Valor	Propriedade	Valor

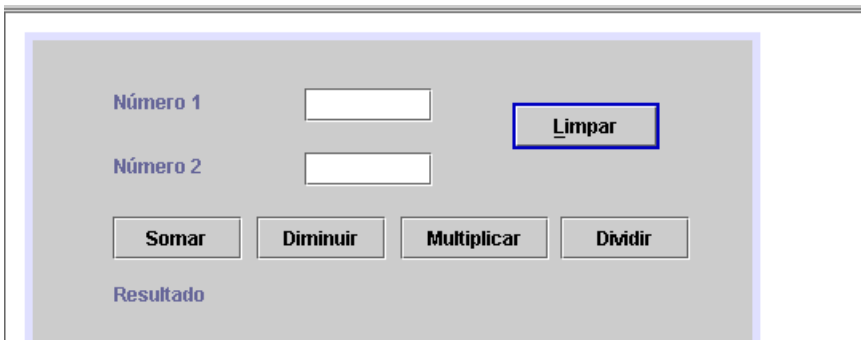
JButton2	Text	Diminuir	**Mnemonic**	D
JButton3	Text	Multiplicar	Mnemonic	M
JButton4	Text	Dividir	Mnemonic	V
JButton5	Text	Limpar	**TooltipText**	Limpar as caixas de Texto

- Dê um duplo clique no botão Diminuir e insira o seguinte código:

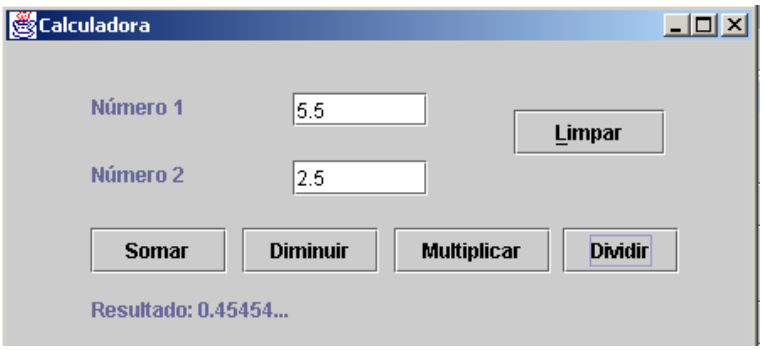
double num1=Double.parseDouble(jTextField1.getText()); // converte texto para double
double num2=Double.parseDouble(jTextField2.getText());
double result = num1- num2;
String R= String.valueOf(result); //Converte o Resultado para String
jLabel3.setText("Resultado: "+R); //exibe no JLabel o Resultado

- Repita o processo nos outros botões modificando apenas a operação matemática.
- Dê um duplo clique no botão “Limpar” e digite:

jTextField1.setText(" "); // limpa a caixa de texto
jTextField2.setText(" ");
jTextField1.requestFocus(); //muda o foco para a caixa 1

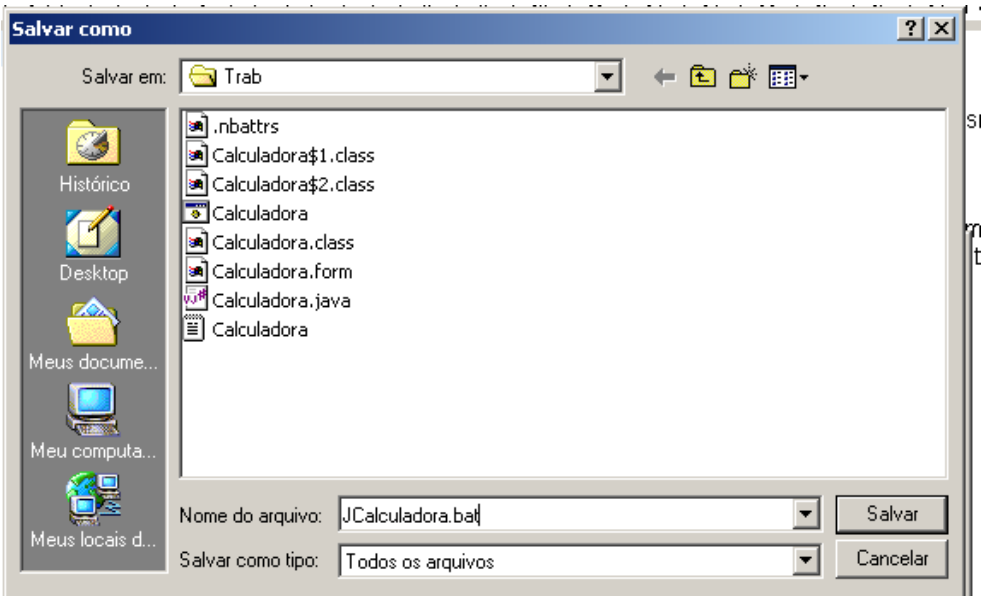


- Compile (F9) e execute (F6).



- Para criar um arquivo que é executado diretamente pelo Windows basta criar um arquivo “.Bat”:

    - Abra o Bloco de Notas e digite: java JCalculadora
    - Salve com um nome qualquer e com a extensão “Calculadora.Bat”, na mesma pasta do programa:



OBS: Lembre-se que o Layout “AbsoluteLayout” é EXCLUSIVO do NetBeans, sendo que para utiliza-lo efetivamente na sua aplicação, você deverá incluir no seu projeto o respectivo pacote, senão na execução ocorrerá erros, ou modifique o Layout para “Null Layout” e defina o tamanho de seu Frame/JFrame para que o

mesmo possa ser visualizado no tamanho desejado (setBounds()), caso contrário o mesmo aparecerá mostrando somente a barra de titulo.

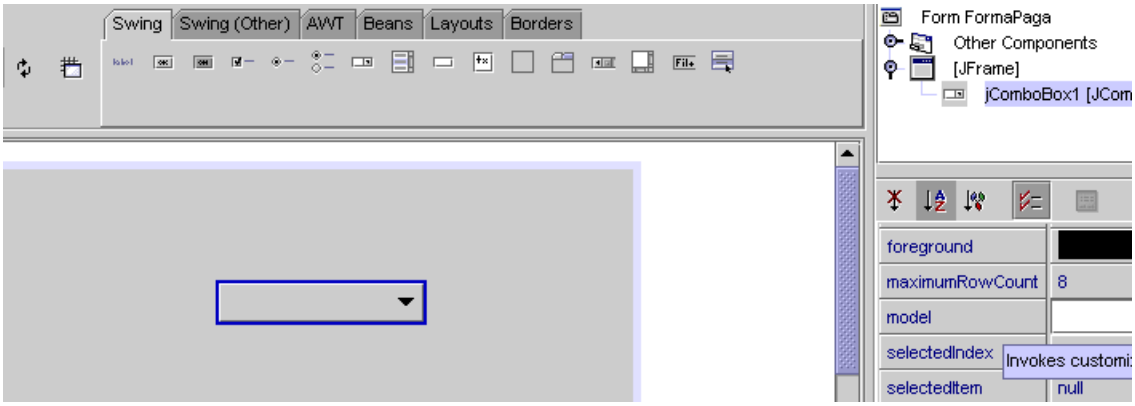
OBS2: Para fins de agilizar a construção dos exemplos será utilizado o Layout “AbsoluteLayout”, mas se for construir comercialmente um programa, o mesmo deverá ter o Layout “Null Layout”.

6. Utilizando JCombobox/Jlist e JRadioButton

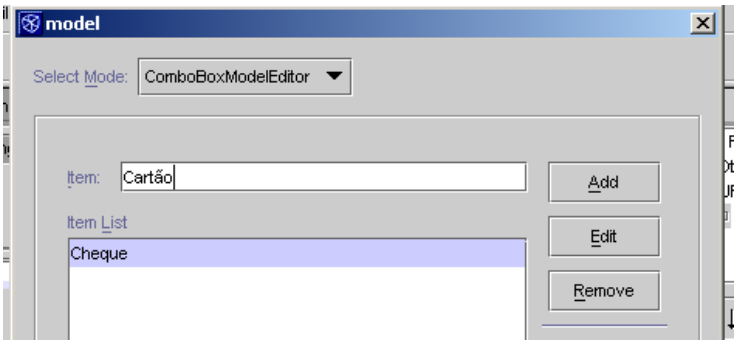
JCombobox/JList

a) Criar uma aplicação para informar qual a forma de pagamento selecionada pelo usuário: Cheque, Cartão ou Dinheiro:

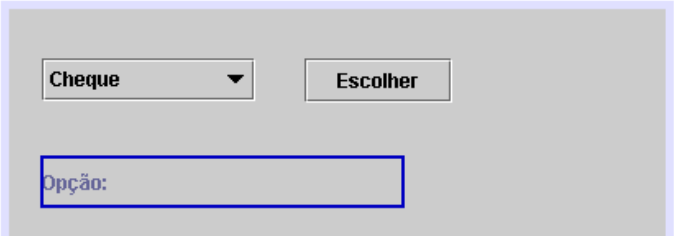
- Clique no Menu File → New → Java GUI Forms → clique em JFrame Form → clique em Next
- Na Janela que se abre digite FormaPaga → clique no botão Finish
- Modifique o Layout para AbsoluteLayout
- Clique no Form e modifique a propriedade “Title” do Form para “Forma de Pagamento”
- Clique no sétimo ícone “JCombobox" e clique no Form, na Janela Propriedades clique em Model – clique no Botão “**...**”.



- Na janela que se abre em “Item” digite: **cheque** e clique no botão “Add”, digite: **cartão** e clique no botão “Add”, digite dinheiro e clique no botão “Add”, clique no botão “OK”



- Insira um JLabel: clique no primeiro ícone (JLabel) e clique no Form.
- Insira um JButton: clique no segundo ícone (JButton) e clique no Form



- A parte visual foi criada, falta o código. Para saber qual a opção selecionada utilize o Método:

  Object getSelectedItem().

- Dê um duplo clique no botão “Escolher”:

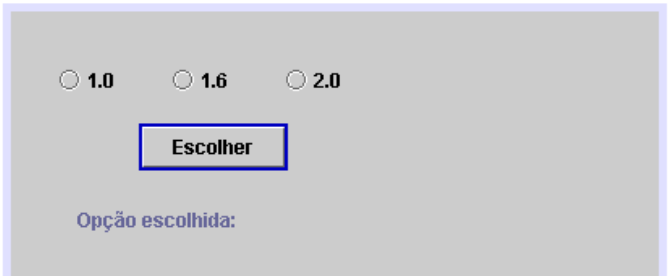
  String S= (String) jComboBox1.getSelectedItem();//converte em String
  jLabel1.setText("A opção escolhida foi: " + S); //exibe opção no Jlabel

- Compile (F9) e Execute (F6).
- Se desejar também é possível saber o índice: **getSelectedIndex()**
- A utilização do JList é idêntica a vista a cima, basta substituir o método **getSelectedItem()**, por **getSelectedValue().**

Utilizando JRadioButton com o NetBeans

b) Criar uma aplicação para mostrar o tipo de motor (1.0/1.6/2.0) escolhido pelo usuário

- Crie uma nova Template JFrame Form “Motor”, clique no Menu File→ New **...**
- No Form que se abre Clique no sexto ícone “ButtonGroup” e clique no Form para criar um grupo de radio e permitir a seleção de apenas uma opção
- Clique no quinto ícone “JRadioButton" e clique no Form, na janela Propriedades selecione “Text” e digite “Motor 1.0”. Selecione a propriedade “buttonGroup” e clique em “buttonGroup1”, repita todo o processo por duas vezes para criar a opção “Motor 1.6” e “Motor 2.0”
- Insira um JLabel, Text: “Opção Escolhida”, no Form e um JButton, “Escolher”.

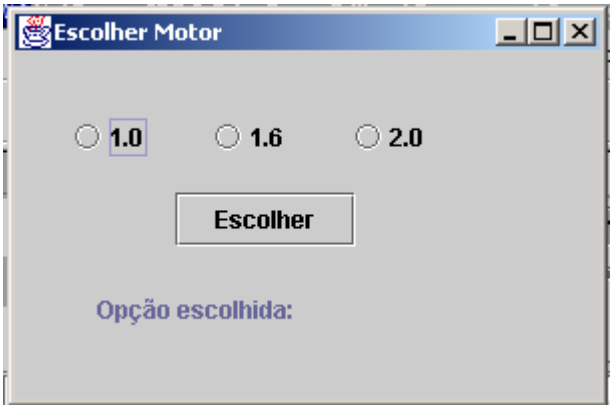


- Para saber qual a opção esta marcada devemos utilizar o Método **boolean isSelected()**, que devolve true ou false.
- Dê um duplo clique no botão para inserir o código:

```
if (jRadioButton1.isSelected())  //verifica se a opção esta marcada -true
        jLabel1.setText("Motor escolhido: 1.0"); //exibe opção no JLabel
if (jRadioButton2.isSelected())  //verifica se a opção esta marcada -true
      jLabel1.setText("Motor escolhido: 1.6");
if (jRadioButton3.isSelected())  //verifica se a opção esta marcada -true
     jLabel1.setText("Motor escolhido: 2.0");
```

- Para saber qual é o Texto exibido pelo JRadioButton basta utilizar o Método: String getText().

Ex: **String s=jRadioButton1.getText();**



7. Aplicações Gráficas com Swing – Componentes Especializados

- Assunto: Aplicações Gráficas com Swing – Componentes Especializados
- Objetivo: Criar aplicações utilizando componentes especializados Swing.

JOptionPane

- Janelas de dialogo para informar ou solicitar confirmação de operações ou efetuar a entrada direta de valores:
  - Mensagens
  - Entrada de Texto
  - Janelas de Confirmação

Exemplos:

//Mensagem

JOptionPane.**showMessageDialog**(this, “mensagem”);

// Confirmação

```
int x = JOptionPane.showConfirmDialog( this, “Mensagem1”);
```

// Entrada de Texto

```
String s= JOptionPane.showInputDialog(“Mensagem 2”);
```

```
int x = JOptionPane.showConfirmDialog(this, "Confirmar?“ , "Aplicação",
        JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE);
```

- Nas IDE como NetBeans é possível inserir um objeto visual, mas não é possível controlar as ações dos respectivos botões.
- Se quiser utiliza-los no Netbeans, insira-os dentro de um método actionPerformed de um JButton.
- Exemplo: - Criar um Programa contendo um botão que ao ser pressionado exiba informações sobre o programa:

- Crie uma nova Template JFrame Form “Mensagem”, Menu File→New ...
- Mude o Layout para AbsoluteLayuot
- Insira um JButton “Mensagem”, dê um duplo clique para inserir o código:

JOptionPane.showMessageDialog(this,"Programa criado \n utilizando o Netbeans");

// o “\n” foi utilizado para mostrar como inserir uma quebra de linha.

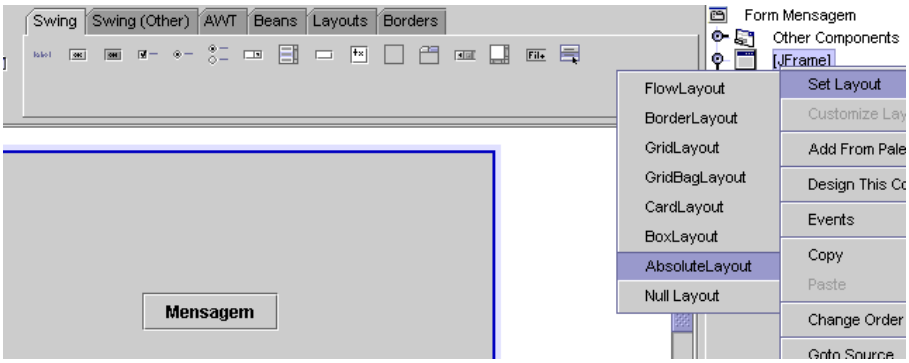
- Este componente pertence ao pacote Swing que deve ser importado através do “import”.

- Procure no código o comando que cria a classe, que está no início do programa:

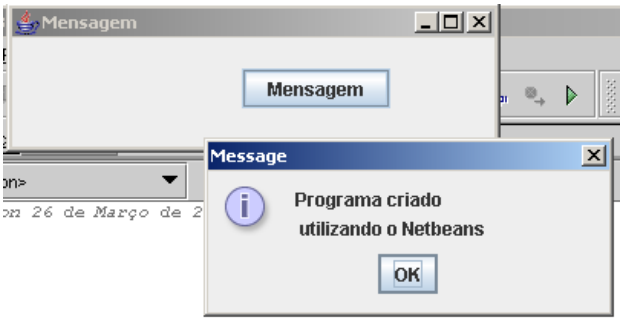
public class Mensagem extends javax.swing.JFrame {

- Antes deste comando insira o código:

import javax.swing.*; //importa os componentes do pacote swing.



- Compile e Execute.



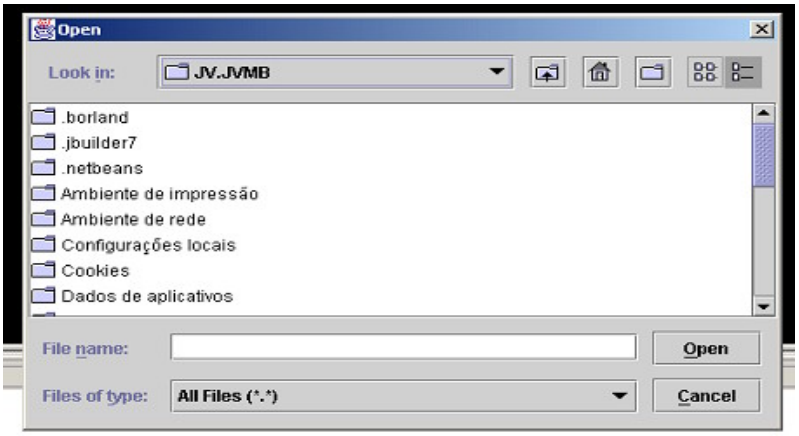
- Todos os exemplos anteriores poderiam ter utilizado o JOptionPane.showMessageDialog.

- **ToolTipText:** texto explicativo exibido quando o mouse passa sobre o componente:

■ No Netbeans basta inserir o texto desejado na propriedade ToolTipText dos Componetes (JLabel, JTextField, JButton ...).

JFileChooser

- Janelas de dialogo para seleção de arquivos:
  - Abrir (Open)
  - Salvar (Save)



Exemplo: Código a ser inserido na ação de um botão “Abrir Arquivo”:

```
JFileChooser arq= new JFileChooser();
int Result=arq.showOpenDialog(this);
if(Result==JFileChooser.APPROVE_OPTION){
        File arquivo= arq.getSelectedFile(); //Classe para
Arquivos
```

```
                    System.out.println(arquivo);        //Imprime nome do
Arquivo
                    }
        }
    }
```

■**Classe File**: suporte para tratamento de arquivos:

- FileReader/FileWrite – FileInputStream / FileOutputStream

- são usados para ler ou gravar arquivos no sistema:

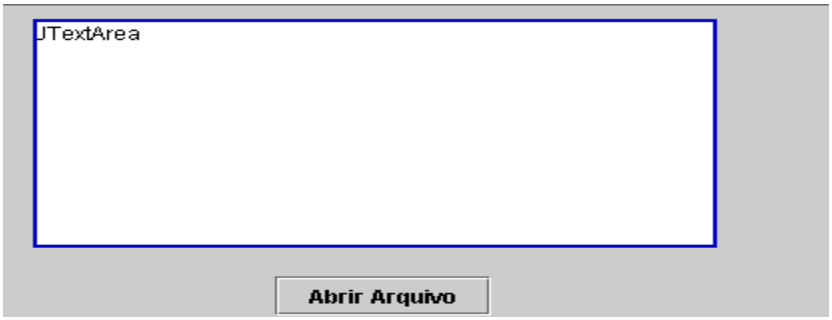
```
FileReader in = new FileReader("Entrada.txt");

FileWriter out = new FileWriter ("Saida.txt");
```

■Comando para ler e escrever arquivos (Byte e não String/char): read() e write(String s)

```
while ((int c=arqin.read()) !=-1){ // lê Entrada.txt
     out.write(c); // grava em Saida.txt
     System.out.write(c); // direciona para o vídeo
     }
    out.write(p); // gravar a String no final do arquivo
    in.close();  // fecha os arquivos
    out.close();
   }
}
```

- Exemplo: criar um programa que permita selecionar um arquivo de texto, ".txt" e mostrar o conteúdo em um JTextArea:

→ Crie uma nova Template, Java Gui Form → JFrame Form



→ Insira um JTextArea e um JButton, “Abrir Arquivo”, e dê um duplo clique no botão, digite:

```
try{       // Blco de Tratamento de erros/exceções nos arquivos
   File arquivo;          //Objeto para arquivos
   JFileChooser arq= new JFileChooser(); // objetos de seleção de arquivo
   int Result=arq.showOpenDialog(this);  // mostra a janela de seleção de arquivo
   if(Result==JFileChooser.APPROVE_OPTION){  //verifica se foi seleciona um arquivo
            arquivo = arq.getSelectedFile(); //verifica o arquivo selecionado
             int c;  //variável para receber os Bytes do arquivo
            String Texto="";      //variável para receber os dados do arquivo
            FileReader inArq = new FileReader(arquivo.getPath()); //abre o arquivo para leitura
             while((c = inArq.read())!=-1){      //lê Byte por Byte até o final do arquivo (-1)
                     Texto=Texto+(char)c; // transforma o Byte lido em um char
                    } //final do while
            jTextArea1.setText(Texto); // exibe no JTextArea o conteúdo lido do arquivo
```

```
                inArq.close();
               }  // final do if
        } catch (IOException ioe) { // verifica qual foi o erro/exceção
            JOptionPane.showMessageDialog(this,"erro ao abrir o arquivo"); //
mensagem de erro
            }
```

→ Insira, antes da declaração da classe “public class”, no começo do programa:

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

- É possível acrescentar um outro botão para salvar novos texto que o usuário pode inserir no JTextArea:

→ Acrescente um novo JButton “Salvar”, dê um duplo clique e digite:

```
try{
        File arquivo;
        JFileChooser arq= new JFileChooser();
        int Result=arq.showSaveDialog(this);
        if(Result==JFileChooser.APPROVE_OPTION){
                arquivo = arq.getSelectedFile(); //Classe para Arquivos
                        FileWriter inArq = new FileWriter(arquivo.getPath());
                        inArq.write(jTextArea1.getText()); // lê o arquivo
                 inArq.close();
                        }
        }catch(IOException ioe) {
            JOptionPane.showMessageDialog(this,"erro ao abriri o arquivo");
            }   // Add your handling code here:
    }
```

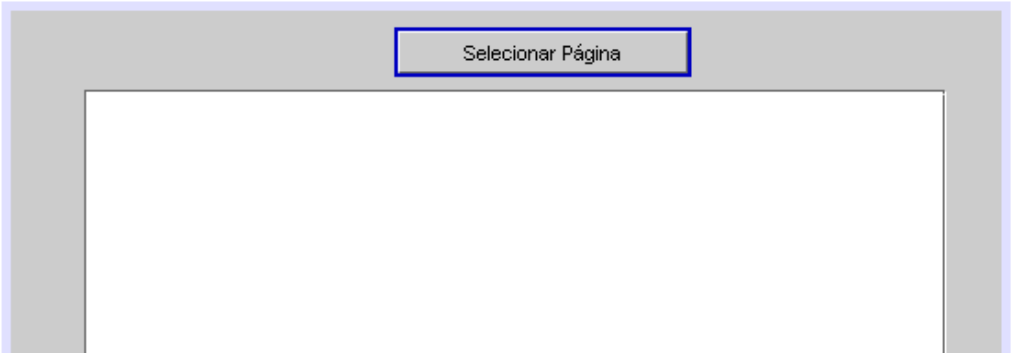
- Estes comandos podem ser inseridos em um Menu.
- A primeira aplicação criada como exemplo, representa a criação de editor de texto já composto de um Menu, mas que não possui os respectivos códigos para efetuar as operações de abrir um documento e salvá-lo. Experimente implementar os respectivos códigos nessa para efetivamente criar um editor de texto.

JEditorPane

- Painel de Conteúdo é uma área de texto especializada na exibição e edição de vários tipos de conteúdo: texto simples(text/plain), HTML(text/html) e RTF – Rich Text Format(text/rtf).
- Exemplo: Criar um Programa que permita exibir um arquivo com extensão ".html" ou ".htm":

→ Crie uma nova Template Java GUI Form → JFrame Form

→ Insira um objeto JScrollPane no Form e insira dentro deste um objeto JEditorPane, da Aba Swing (Other)



→ Insira um objeto JButton, "Selecionar Página":

JFileChooser arq = new JFileChooser();

```
int result=arq.showOpenDialog(this);
if(result==JFileChooser.APPROVE_OPTION){
   try{
      File arquivo= arq.getSelectedFile();
      URL pagina= new URL("file:"+arquivo.getPath());
      jEditorPane1.setPage(pagina);
   } catch(MalformedURLException mue) {
      JOptionPane.showMessageDialog(this,"Erro na página");
   }catch( IOException ioe){
      JOptionPane.showMessageDialog(this,"Erro no arquivo");
   }
}
```

→ Insira a importação dos pacotes, antes da declaração da classe “public class”:

```
import javax.swing.*;
import java.net.*;
import java.io.*;
```

→ Compile e Execute.

8. Trabalhando Múltiplos Formulários - Menus.

- Assunto: Aplicações Gráficas com Swing – Múltiplos Formulários.
- Objetivo: Criar aplicações com Vários Formulários

SUMÁRIO

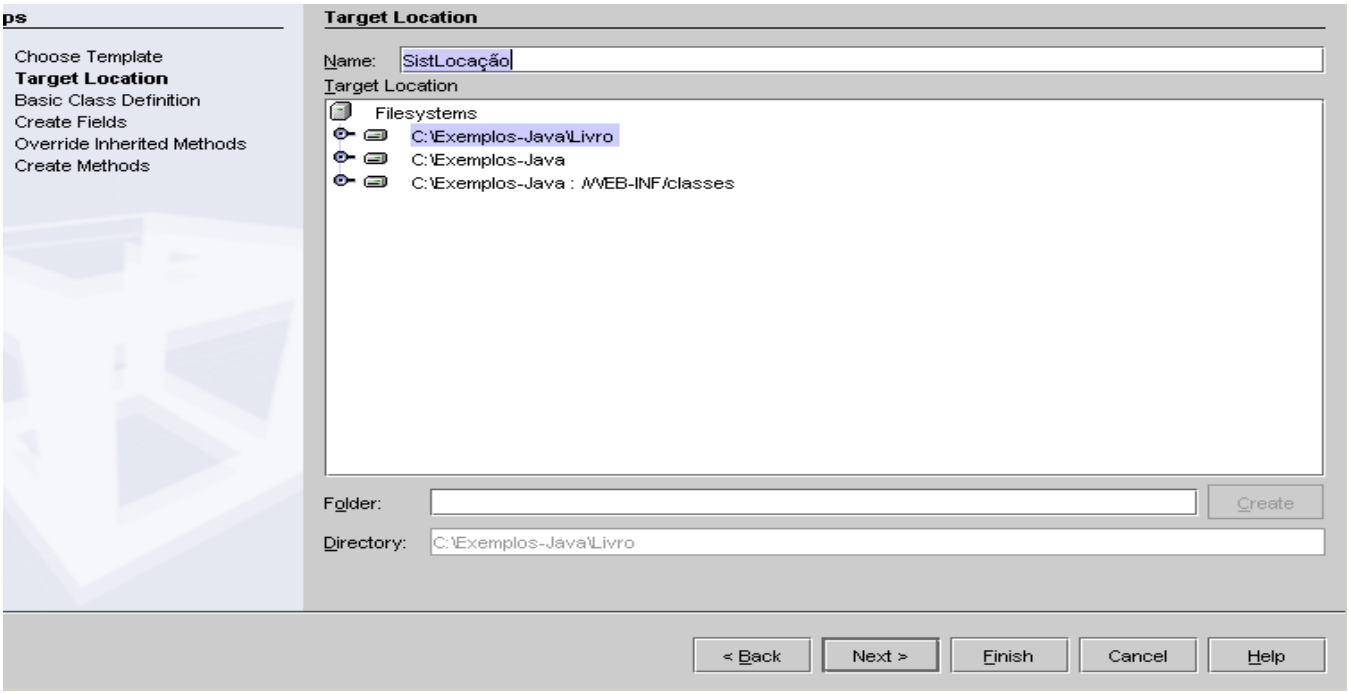
Criando Múltiplos Formulários - Menus

- Para se construir uma Aplicação contendo vários Formulários é preciso criá-los em separados e chamá-los em um Formulário Principal através de Botões ou de Menus.
- No Formulário Principal a chamada dos outros formulários consiste em se criar instâncias dos respectivos formulários que se deseja exibir, utilizando o método show(), ou o método setVisible(boolean).
- Na criação de Formulários utilizando o NetBeans (e outras IDE), a ferramenta cria os comandos necessários para o fechamento do Formulários, desta forma se o Formulário Principal chamá-lo e este após, ser aberto, for fechado provocará o fechamento da aplicação.
- Para evitar este “problema” devemos modificar “estes comandos” para podermos fechar o formulário sem fechar a Aplicação.

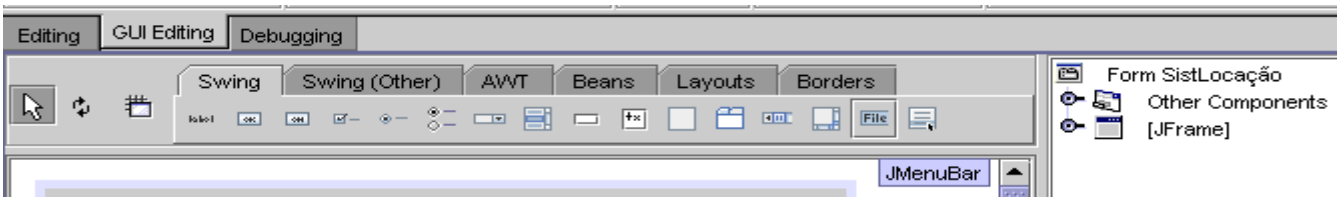
Exemplo

- Criar uma Aplicação contendo um Menu que exibi os Formulários de Cadastro de Clientes e de Veículos

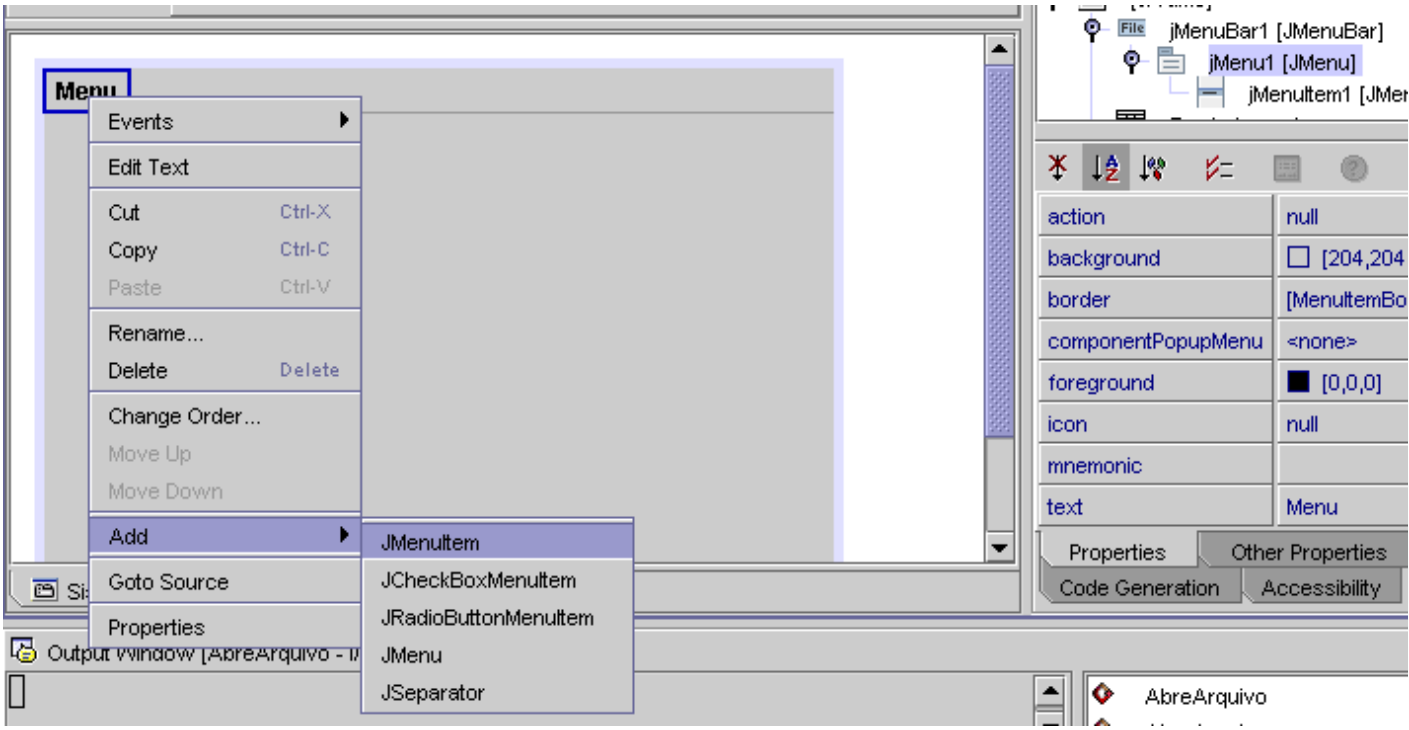
■ Clique no Menu File → New → Java GUI Forms → Botão Next → em Name digite “SistLocação” → Clique no Botão Finish



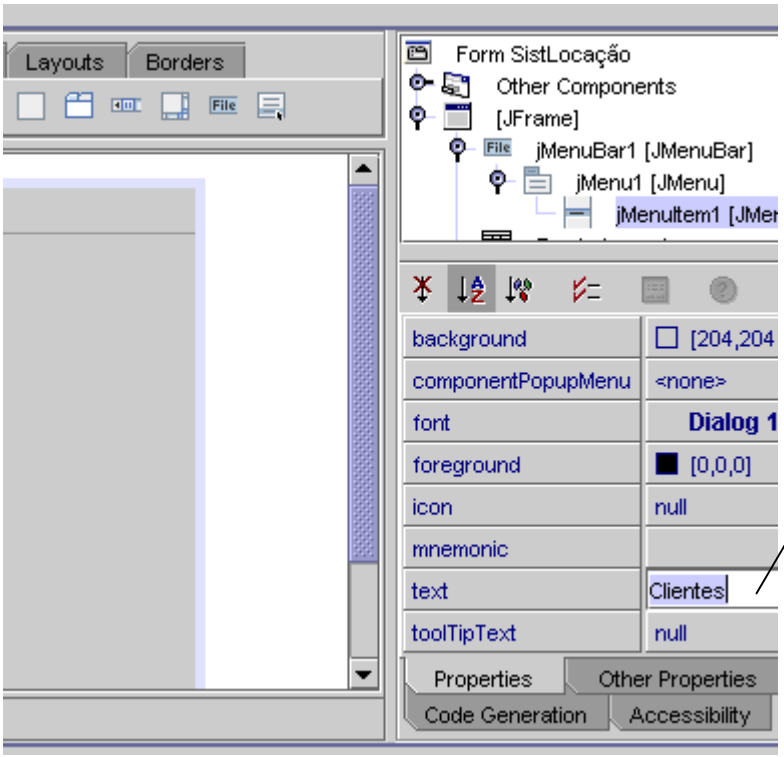
■ É preciso criar o Menu.



■ Na Aba Swing, clique no ícone JMenuBar e clique no Form → na Janela Propriedades clique em Text e digite “Cadastro” (texto que será exibido pelo Menu)→ Clique com o botão direito do mouse sobre o Menu criado → selecione Add JMenuItem → na janela propriedade clique em Text → digite Clientes → clique novamente com o botão direito do mouse sobre o Menu criado → selecione Add → JMenuItem → clique em Text e digite Veículos.



■ Observe que os SubMenus não aparecem no Form este serão exibido na execução do programa, mas apara o objeto criado no Form.



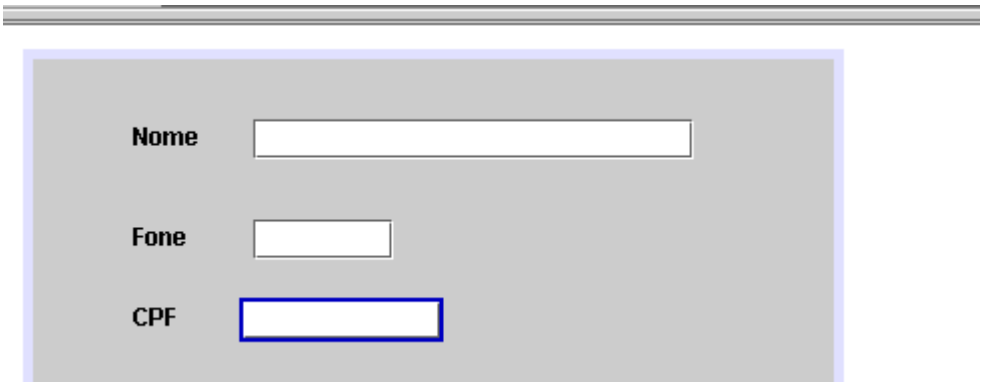
- Objeto JMenuItem criado no Form, o mesmo não aparece no Form durante a construção do programa. Só aparece na Execução.

- Texto do submenu que será exibido para o usuário na execução do programa.
- Se desejar modificar o texto de um submenu, basta clicar, na janela acima, no objeto desejado (Ex: jMenuItem1) e na propriedade Text digitar o novo texto.

■ Os Menus foram criados faltam os Formulários Clientes e Veículos.

■ Crie um Novo Form→Clique no Menu File → New → clique no Botão Next → em Name digite Clientes → clique no botão Finish

■ Modifique o Layout para AbsoluteLayout e insira três JLabels para exibir os rótulos “Nome”, “Fone”, “CPF” e três JTextFields para podermos digitar/exibir os dados.



■ Devemos modificar o comando responsável pelo fechamento do Form, para que não feche a Aplicação.

■ Clique com o botão direito do mouse no Form e clique em “Goto Source” ou presiione “Ctr-3”, procure o comando, que deve estar antes do método main:

“private void exitForm(java.awt.event.WindowEvent evt) {“

- Apague o comando:

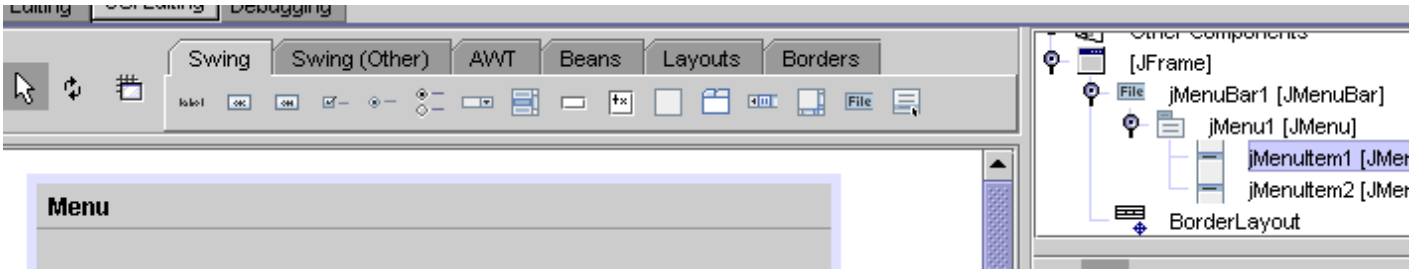
System.exit(0);

- E digite no seu lugar:

setVisible(false);

■ Volte para o primeiro Form (SistLocação), para isto clique na Aba SistLocação, localizada na janela principal do Form

■ Dê um duplo clique em JMenuItem1, se o mesmo não estiver aparecendo clique no menu criado, dê um duplo clique no JMenuItem1, para inserir o evento responsável pela chamada do Form Clientes.



■ Na janela de código que se abre, procure o seguinte comando:

"private void jMenuItem1ActionPerformed"

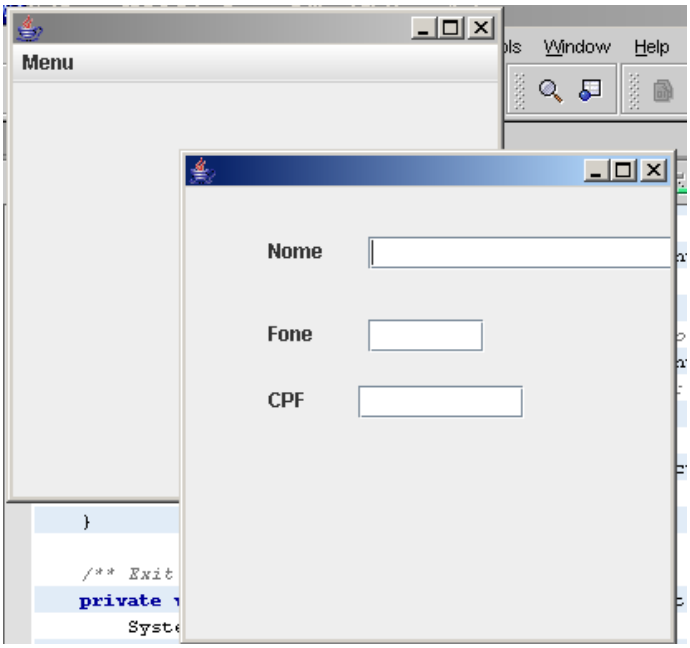
- Antes deste comando digite:

Clientes FormCliente = new Clientes(); //instanciação do Formulario Clientes

- Se tiver criado a Template com outro nome substitua o nome "Clientes" por este nome.
- E dentro do método, depois de "**// Add your handling code here:**", digite:

FormCliente.show(); //exibição do Formulário Clientes

- Compile e Execute.



- Repita os mesmos passos para criar o Form Veículos e para exibi-lo.
- Clique no Menu File → New → Next → digite o Nome “Veículos” → clique no botão Finish
- Mude o Layout e insira os componentes (JLabel e JTextField), vá para o código e modifique o comando **System.exit(0)** para **setVisible(false)**
- Volte para o Form SistLocação e dê um duplo clique em JMenuItem2, antes do respectivo método actionPerformed instancie o Form:

Veículos FormVeiculo = new Veículos(); //instanciação do Form

- Dentro do método digite o comando para exibição:
  FormVeiculo.show();

- Compile e Execute. Seu Pequeno Sistema foi Criado

9. Applets – Funcionamento e Estrutura

- Assunto: Applets: Funcionamento e Estrutura
- Objetivo: escrever mini-aplicativos java que rodam na internet e intranet.

Applets

- São pequenos programas Java que podem ser inseridos dentro de páginas HTML.
- interagir com o usuário que a consulte
- pode executar tarefas complexas, como realizar cálculos e apresentar gráficos, sons e imagens em movimento.

Applets: Funcionamento

- Para inserir uma applet numa página HTML, usamos a diretiva **<applet>**, que deve apresentar pelo menos três parametros: **code**, **width** e **height**. Assim, a especificação mais simples tem a forma:

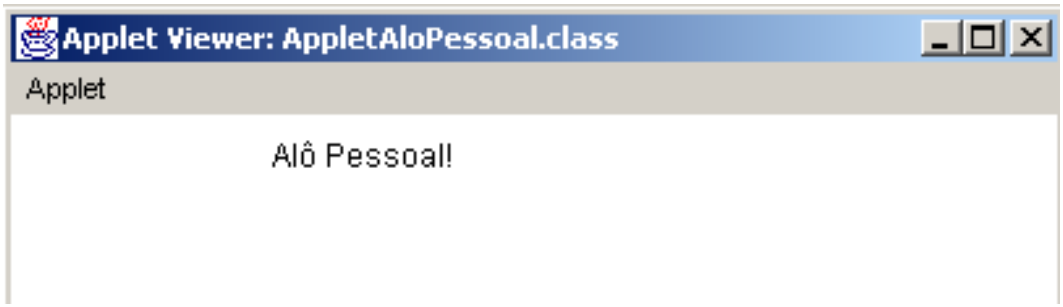
<applet code=[java applet] width=[largura] height=[altura]>
</applet>

- **[java applet]:** nome da classe principal do programa Java – AloPessoal.class
- **[largura]** e **[altura]:**indicam a largura e a altura em pixels da área dentro da página reservada para a apresentação da applet.
- uma applet deve conter ao menos uma classe pública, e essa classe pública deve estender a classe **Applet**.
- A classe **Applet** faz parte do package applet:

```
import java.awt.*;  //importação dos componentes
import java.applet.*;       //importação da classe Applet
public class AloPessoal extends Applet {
   public void paint(Graphics g){       //método para desenho
     g.drawString("Alô Pessoal!", 100, 20);   // desenha um texto
                }
}
```

- Este programa deve ser compilado, e após a depuração dos erros é informado o arquivo gerado: .class, no arquivo html através do parâmetro code (**code = ArquivoJava.class)** é informado qual o arquivo Java a ser utilizado na página HTML.
- Arquivo Html (com um nome qualquer, Ex: Alo.html):

```
<applet code= AloPessoal.class width=120 height=120>
 </applet>
```



- Parâmetros adicionais:
  - **codebase**: URL do diretório que contém as classes compiladas (bytecodes) do applet
  - **name**: Um nome que identifica uma particular instância de um applet dentro de uma página HTML (útil para comunicação entre applets).
  - **align**: especifica um alinhamento para a área da applet dentro da página. Análogo ao alinhamento de imagens.

- **alt**: Texto a ser exibido na área da applet durante o carregamento, ou na impossibilidade de carregar a applet.

Applets: Estrutura

A classe Applet define quatro métodos básicos para controlar a execução e interação com o browser:

a) init(): executado quando a applet e carregada ou recarregada **–** adição de componentes, recebimento de parâmetros de execução e preparo da applet

b) start(): executado após o método init ou ao ser recarregada **–** deve ser utilizado quando é necessário assegurar alguma condição especial.

c) stop(): executada quando o usuário muda de página permanecendo na memória, ou fecha o browser.

d) destroy(): executada quando a applet for removida da memória.

paint(), é definido pela classe Component da AWT: atualiza a exibição da applet

A maneira mais simples de se construir uma applet é por meio do método paint **–** o conteúdo apresentado só depende desse método.

A combinação destes métodos representa o ciclo de vida de uma applet.

Criando Applets no NetBeans

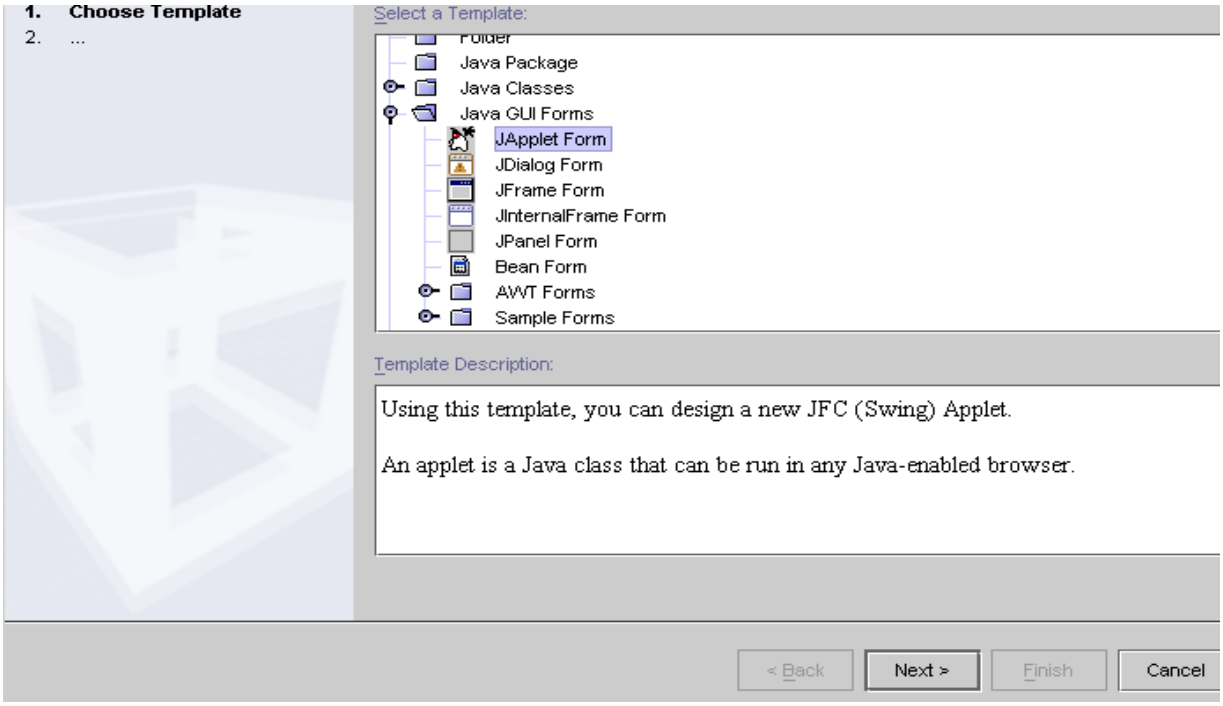
A criação de Applet utilizando o Netbeans é uma tarefa simples, devendo ser criado apenas o programa Java, o Applet, porque a ferramenta cria automaticamente o corresponde arquivo HTML, com o mesmo nome do seu programa, contendo os códigos para executar a página web.

1) Criar um Applet que permita ao usuário digitar dois números reais e mostrar o resultado da soma destes:

■ A criação de Applet no Netbeans é feita através de Templates, é possível trabalhar com a Classe Applet do pacote AWT ou com a classe JApplet do pacote Swing.

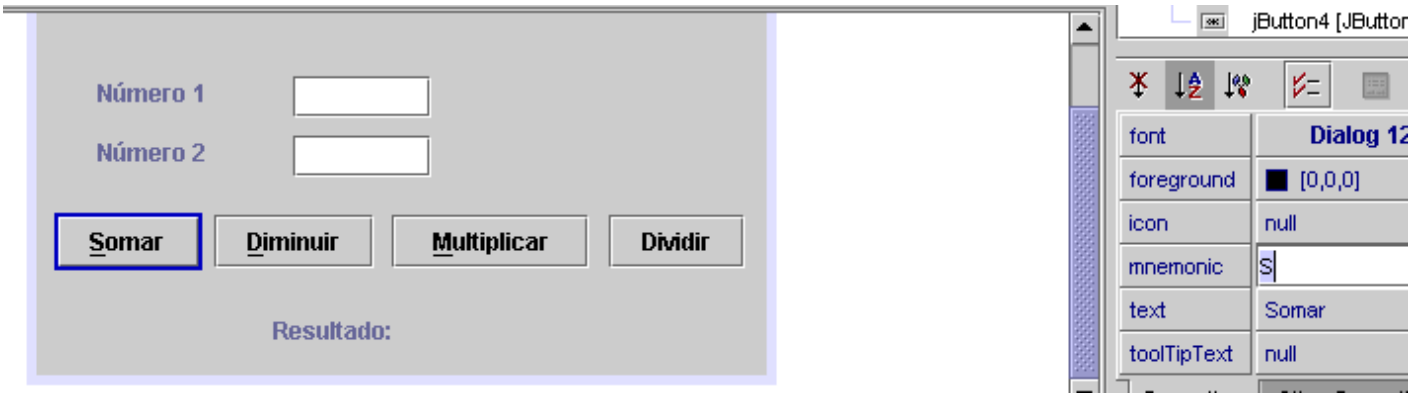
■ O Exemplo que se segue, será implementado utilizando o JApplet.

■ Clique no menu File → New → Java GUI Form →JApplet Form → clique no Botão Next → digite o nome “Calculadora” e clique no botão Finish.



■ Antes de inserir os componentes modifique o Layout para “NullI Layut”, para poder organizar os componentes da maneira desejada. Clique com o botão direito do mouse, na janela no canto superior direito, em JApplet e selecione “Null Layout”

■ Insira três JLabel (“Número 1: ” e “Número 2: “), dois JTextField, quatro JButton(“Somar”, “Diminuir”, “Multiplicar”, “Dividir”), organize-os da seguinte forma:



■ Para inserir um atalho para o botão, digite a letra desejada na propriedade "Mnemonic"

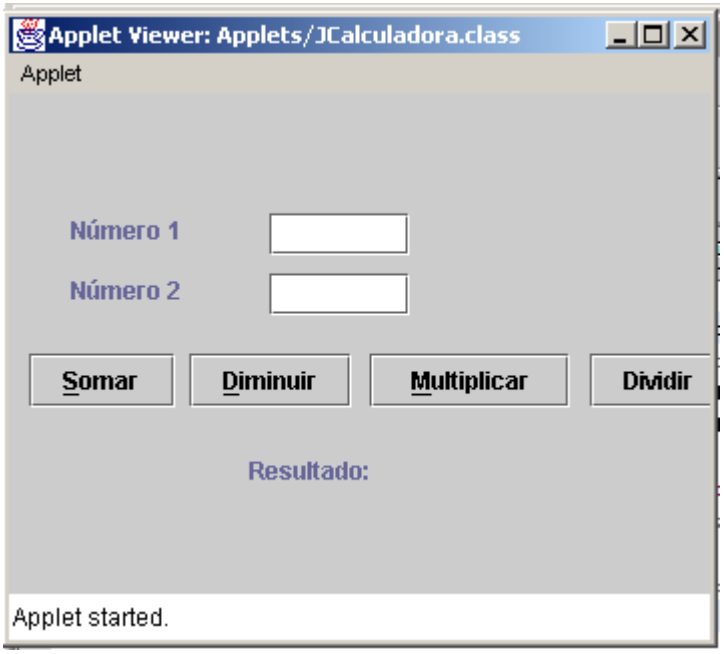
■ A parte visual foi criada, falta os eventos dos botões.

■Dê um duplo clique botão "Somar" e digite:

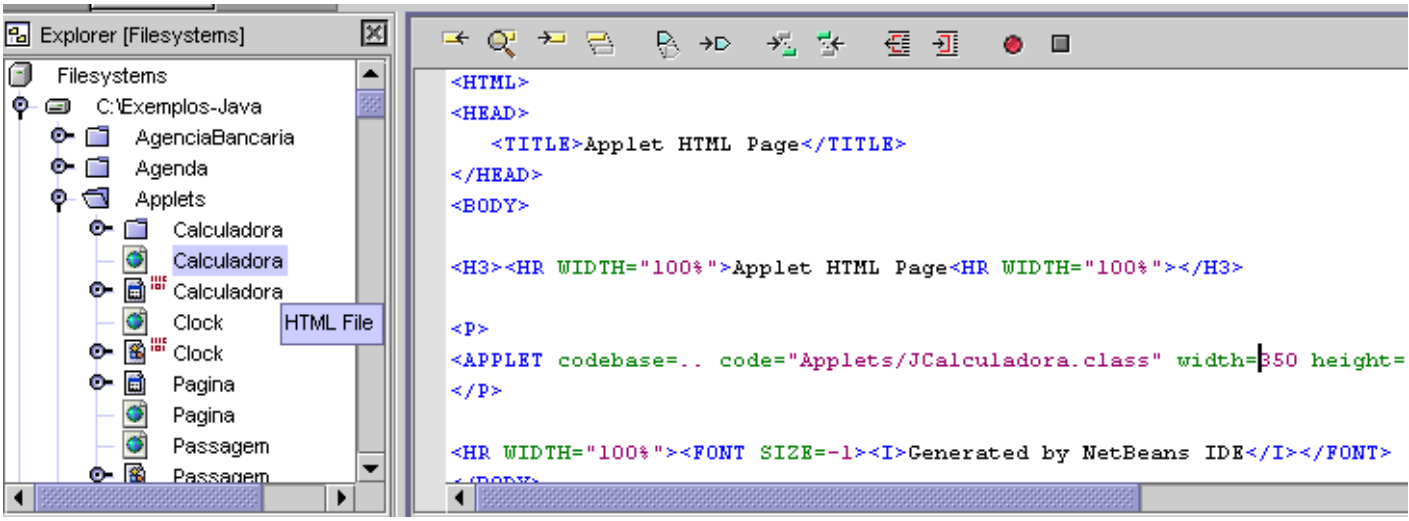
```
float op1,op2, result; // variáveis auxiliares
//"pega" o número digitado e o converte para float
op1=Float.parseFloat(jTextField1.getText());
op2=Float.parseFloat(jTextField2.getText());
result=op1+op2;
//converte o resultado em String e exibe
jLabel3.setText(String.valueOf("Resultado: " + result));
 jTextField1.setText(" ");          //Limpar o JTextField
 jTextField2.setText(" ");
 jTextField1.requestFocus();     //muda o foco para o JTextField 1
```

■ Compile e Execute.

■ Observe que ao executar o programa o mesmo é mostrado no Applet Viewer, um aplicativo do JDK construído para auxiliar na implementação dos applets, e gera automaticamente o arquivo HTML com o mesmo nome do programa applet.



■ Para visualisar o arquivo HTML, clique no menu View→FileSystem, e dê um duplo clique no arquivo com o mesmo nome do programa, mas do tipo HTML

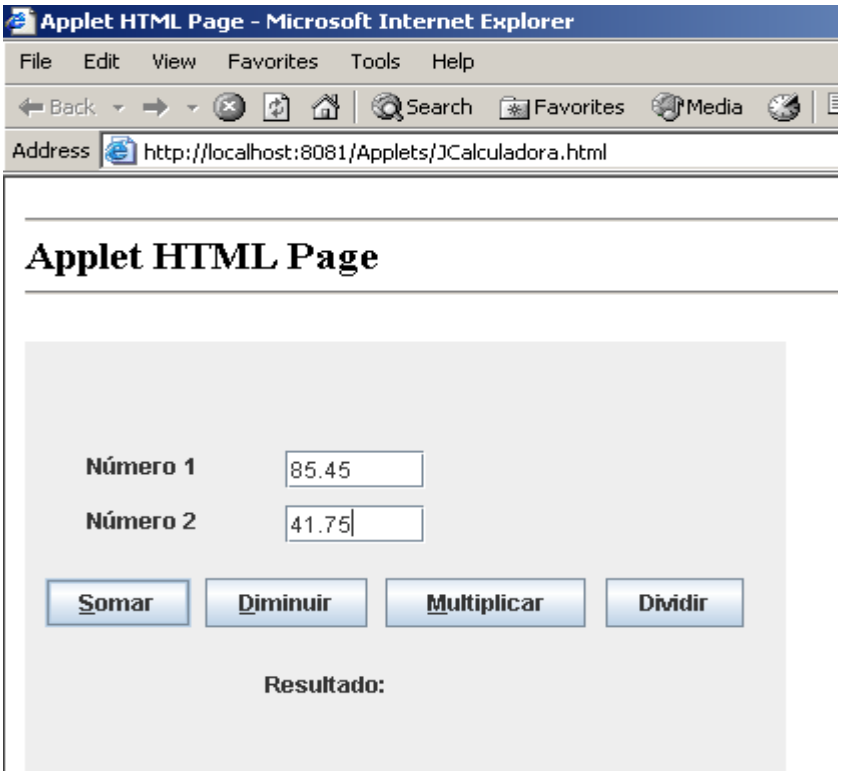


■ Se desejar modifique a área destinada a mostrar o applet no browser, modifique no arquivo HTML os valores dos códigos WIDTH (Largura do Apllet) e HEIGHT(Altura). Ex:

<APPLET codebase=.. code="Applets/JCalculadora.class" width=350 height=250>

</APPLET>

- Para executar o arquivo, ou seja, abrir no browser, basta executar (F6):



- Para implementar as operações matemáticas (eventos) dos outros botões, dê um duplo clique no botão e digite o mesmo código do botão soma e modificando apenas o sinal de soma para a operação desejada. Exemplo par ao botão “Subtrair”

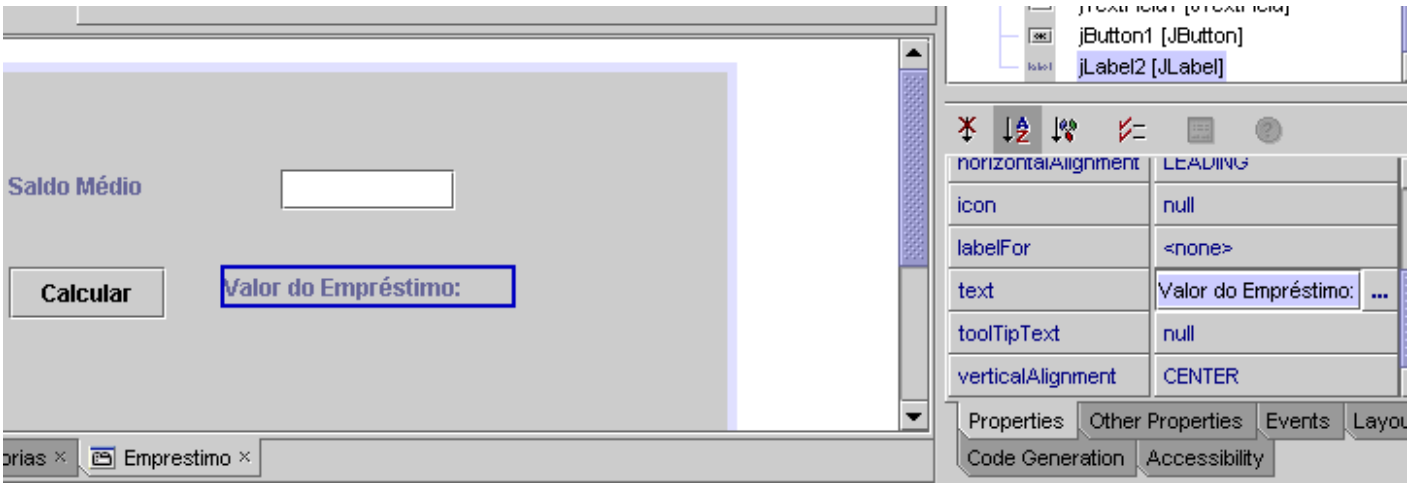
```
float op1, op2, result; // variáveis auxiliares
 // "pega" o número digitado e o converte para float
 op1 = Float.parseFloat( jTextField1.getText() );
 op2 = Float.parseFloat( jTextField2.getText() );
 result = op1- op2;
//converte o resultado em String e exibe
```

```
jLabel3.setText( String.valueOf( "Resultado: " + result) );
jTextField1.setText(" ");          //Limpar o JTextField
jTextField2.setText(" ");
jTextField1.requestFocus();     //muda o foco para o JTextField 1
```

- Também é possível exibir os resultados das operações utilizando um JOptionPane.

2) Crie um Applet que informa o valor do empréstimo, que é calculado conforme o valor do saldo médio digitado pelo usuário (até R$ 1000,00 → zero, de R$ 1001 até 2500→30% do Saldo Médio, acima de R$ 2500→ 50% do Saldo Médio):

- Crie um Novo Programa: File→New→Java GUI Form→JApplet→digite o nome ”Empréstimo” → Finish
- Clique com o Botão direito do mouse no Form, selecione “setLayout” e clique em “Null Layout”.
- Insira dois JLabel no Form, modificando as propriedade “Text” para “Saldo Médio” e “Valor do Empréstimo”, insira um JTextField e um JButton, modifique a propriedade “Text” para “Calcular”.



- Dê um duplo clique no botão e digite os comandos:

```
float saldo;       //variável auxiliar
 try{              //tratamento de erros do valores
    saldo = Float.parseFloat(jTextField1.getText()); //”pega” o valor digitado e converte
//verificação dos valor digitado e do empréstimo que pode ser concedido
   if (saldo<1000)
      jLabel2.setText("Valor do Empréstimo: Zero");
   if ((saldo>=1000) & (saldo<=2500))
         jLabel2.setText("Valor do Empréstimo: " + saldo * 0.3);
   if (saldo>2500)
      jLabel2.setText("Valor do Empréstimo: " + saldo * 0.5);
    }catch (Exception e){        //exibe mensagem caso não seja digitado um valor válido
      jLabel2.setText("Não é um Valor Válido");
    }
```

- Compile e Execute.

10. Banco de Dados No NetBeans

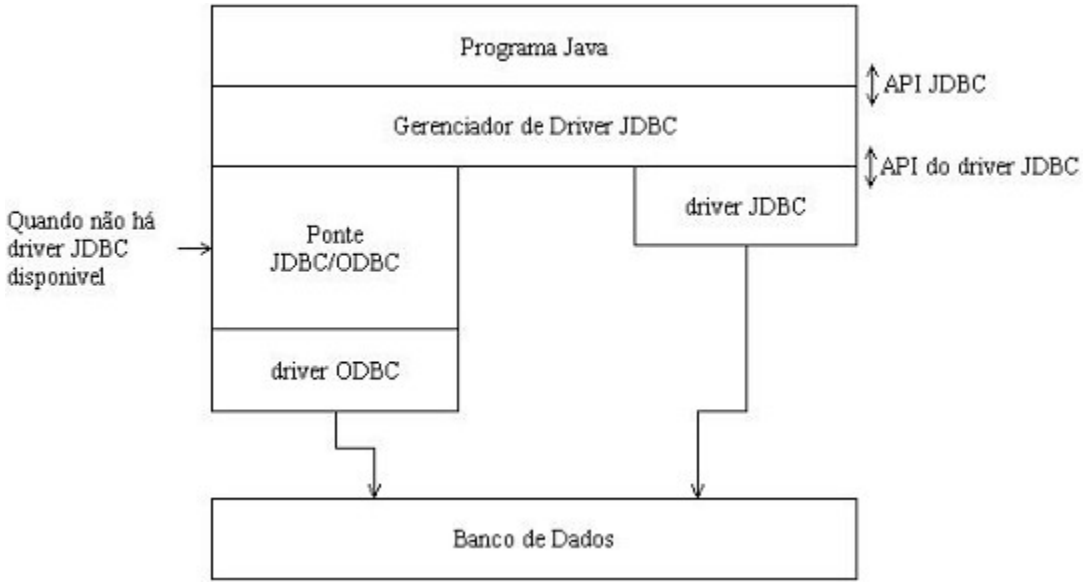
- Assunto: Acessando Bancos de Dados.
- Objetivo: criar programas que acessam Banco de Dados.

Acessando Bancos de Dados

- A comunicação de código Java puro com Banco de dados é uma tarefa muito difícil, devido a enorme quantidade de bancos de dados, existente no mercado, com linguagens proprietárias.
- A plataforma Java permite o acesso a banco de dados por meio de uma API (Application Programming Interface) chamada JDBC (Java Database Connectivity).
- É uma interface entre a linguagem Java e outra linguagem que todos os bancos de dados suportam.
- O Objetivo da JDBC é possibilitar o uso de dados existentes em SGBDR remotos. Utiliza o SQL (Structured Query Language), linguagem destinada às operações dos SGBDR.

Arquitetura

- É muito parecida com o padrão ODBC (Open DataBase Connectivity).
- A Aplicação para acessar um BD deve utilizar a API JDBC, contida no pacote java.sql.
- Por meio da classe DriverManager, seleciona-se o driver de acesso ao BD que será utilizado.O driver passa a oferece uma interface padronizada para a Aplicação, implementando em Java ou de forma nativa o acesso ao BD.



- Instalação de uma fonte JDBC/ODBC em plataforma Windows:

1. Acessar o painel de controle e ativar o aplicativo Fonte de Dados ODBC;
2. Clicar no botão adicionar (DNS do usuário) para criar uma nova ponte de um banco de dados;
3. Selecionar Driver adequado (Driver do Microsoft Access, por exemplo)
4. Preencher a caixa de diálogo com um nome para a Fonte de Dados do BD e a localização do servidor.
5. Clicar no botão “Selecionar” para definir o BD a ser usado.

- Podemos construir uma aplicação usando o seguinte roteiro:

1º) Importar o pacote java.sql:

import java.sql.*;

2º) Carregar o driver de acesso ao BD : (ponte JDBC-OBDC)

Class.forName(“sun.jdbc.odbc.JdbcObdcDriver”);

4º) Obter uma conexão com o BD: (BD Access: IJPlanetas)

Connection conexão =

DriverManager.getConnection(“jdbc:odbc:IJPlanetas”);

5º) Por meio da conexão realizada, devemos obter uma instância de um objeto Statement, PreparedStatement ou CallableStatement, para passar comandos SQL ao BD:

Statement stmt= conexão.createStatement();

6. Criar objetos para executar operações no BD (execute (String), executeQuery (String), ExecuteUpdate(String), executeBatch()):

int reg =stmt.executeUpdate(“delete from planeta where nome=’Maria’”);
ResultSet rs = stmt.executeQuery(“select nome from planeta”);

7. Algumas operações retornam resultados do BD (um objeto Resultset – conjunto de registros), os quais devemos exibir ou processar:

while(rs.next()) { //move o curso de registros

String Nome=rs.getString(“nome”); // obtém o valor do campo “nome” da tabela

System.out.println(Nome);

}

8. Quando não precisar mais dos objetos obtidos, libere-os:

rs.close();
stmt.close();

9. Após todas as operações encerre a conexão com BD:

conexão.close();

• Exemplos de métodos de ResultSet:

rs.absolute(3); //move cursor para linha
rs.updateString(”Nome”, ”Maria”); //atualiza nome
rs.updateRow(): //atualiza linha na tabela
rs.moveToInserRow(); //insere nova linha
rs.updateInt(2, 3535); //atualiza coluna 2

Principais Comandos SQL

Segue exemplos dos principais comandos que serão utilizando em uma aplicação que realizar operações em um Banco de Dados:

Tabela Brasil			
idade	Estado	Pop	Sit
Brasília	DF	2.000.000	Cheio

- SELECT: - realiza consultas

Select cidade, estado FROM Brasil WHERE pop > 1000;

- INSERT: - Insere um registro

  INSERT INTO Brasil (cidade, estado) VALUES (Cid, Est)

- UPDATE: - Modifica um registro

  UPDATE Brasil SET Sit = “Cheio” WHERE Pop > 1000;

- DELETE: - Excluir um registro

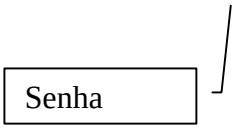
  DELETE FROM Brasil WHERE cidade =“BSB”

Construção de Aplicações - Acessando um Banco de Dados Access

Este exemplo realiza uma consulta na tabela denominada Relacao, com os campos Nome e Id. Vamos partir do princípio de que a “ponte” JDBC-OBDC já foi criada com o nome BDAccess.

```
import java.sql.*;      //pacote para trabalhar com BD

public class LeBDAccess{

    public static void main(String args[]){

        try{ //tratamento de erros
```

```
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");//seleciona Driver

        //conecta com BD
        Connection con=DriverManager.getConnection("jdbc:odbc:PBD-
Nomes","","");

        //objeto comandos SQL
        Statement stmt= con.createStatement();

        //Comando SQL
        ResultSet rs=stmt.executeQuery("select * from Relacao");

        //verifica se existem registros
        while(rs.next()){

                // obtém o valor armazenado no campo “Nome” da tabela
Relacao
                   String nome=rs.getString("nome");
                // obtém o valor arqmazenado no campo da Id
                   int id=rs.getInt("id");
                // imprime o conteúdo da variável
                  System.out.println("Nome:"+nome+ " Ident:"+id);
                }
                rs.close(); //fecha "ponte"
                con.close(); //fecha conexão com BD

        } catch(SQLException e){ //trata os erros
                System.out.println(“erro de conexão”+ e.getMessage()); //
mensagem de erro
```



```
            } catch(ClassNotFoundException e){
            System.out.println("Driver não encontrado");
            }
        }
}
```

JDBC no NetBeans

O NetBeans não possui ferramentas que permitam criar visualmente componentes, em um programa, para manipular os dados de uma tabela, a exemplo dos existentes em outras ferramentas, em outras Linguagens de programação.

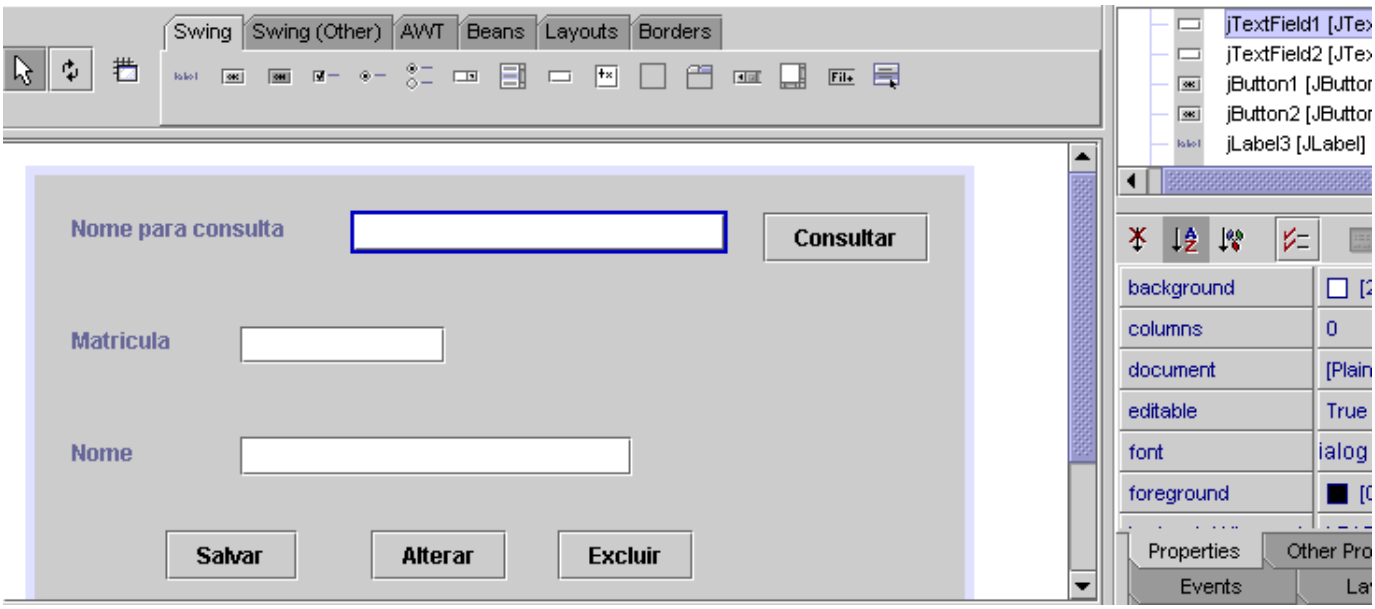
Para trabalhar com Banco de Dados é preciso fazê-lo via código, que deverá ser inserido dentro do método “actionPerformed” de um JButton.

Cria-se o “Formulário” contendo os componentes visuais para mostrar, e manipular, os dados da tabela do Banco de Dados. Dentre estes componentes estão os botões ou menus, os quais serão os responsáveis por executar os códigos para realizar uma determinada operação, consulta, inserção, alteração ou exclusão em um BD.

Depois de criada a “parte visual”, devem ser implementados os códigos nos botões/menus:

OBS: - Os exemplos que se seguem utilizam uma Fonte de Dados ODBC com o Nome: “PBD_Nomes” que representa uma conexão com um Banco de Dados Access, contendo uma Tabela com o nome: “TabFicha”, contendo os campos (Matricula – Tipo Número/Chave Primária; Nome – Tipo Texto)

- Parte Visual do Programa:



a) <u>Códigos para Inserir:</u>

```
try{ //tratamento de erros
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");//busca Driver
       //conecta no BD
      Connection con = DriverManager.getConnection("jdbc:odbc:PBD-Nomes","","");
      Statement stmt = con.createStatement();       //objeto comdo sql
      String CadNome = jTextField1.getText();  //obtém nome digitado
      int CadMat=Integer.parseInt( jTextField2.getText() );
      stmt.executeUpdate("insert into TabFicha (Matricula, Nome) values (' " +
CadMat + " ',' " +
                                    CadNome +  " ' ) " );
      JOptionPane.showMessageDialog( this, " Dados Salvos! ");
      con.close();  // fecha conexão com BD
        }  catch( SQLException e){ //trata os erros
         JOptionPane.showMessageDialog(this,  "Erro  Cmdo  SQL  " +
e.getMessage() );
```

```
        } catch( ClassNotFoundException e){
         JOptionPane.showMessageDialog( this, " Driver não encontrado " );
}
```

b) <u>Códigos para Consultar que devem ser inseridos em um JButton/Menu:</u>

```
boolean consulta=true; //variável auxiliar utilizada para informar se houve sucesso na consulta
try{ //tratamento de erros
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");//busca Driver
        //conecta no BD
        Connection con=DriverManager.getConnection("jdbc:odbc:PBD-Nomes","","");
        Statement stmt= con.createStatement();//objeto comdo sql
        String ConsNome= jTextField1.getText();
        ResultSet RS= stmt.executeQuery(
                        "Select * from TabFicha where nome=' " + ConsNome + " ' ");
        while (RS.next()){
                int Mat= RS.getInt("Matricula");
                jTextField2.setText(String.valueOf(“Matricula”));
                 jTextField3.setText(RS.getString("Nome"));
                consulta=false;
                JOptionPane.showMessageDialog(this,"Dados Encontrados!");
         }
      if (consulta) JOptionPane.showMessageDialog(this,"Dados Não Encontrados!");
      RS.close ();
      stmt.close();
      con.close(); //fecha conexão com BD
        } catch(SQLException e){ //trata os erros
```

```
          JOptionPane.showMessageDialog(this,"Erro Cmdo SQL
"+e.getMessage());
          } catch(ClassNotFoundException e){
           JOptionPane.showMessageDialog(this,"Driver não encontrado");
          }
```

c) <u>Códigos para Alterar:</u>

```
try{ //tratamento de erros
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");//busca Driver
        //conecta no BD
        Connection con=DriverManager.getConnection("jdbc:odbc:PBD-
Nomes","","");
        Statement stmt= con.createStatement();//objeto comdo sql
      String ConsNome= jTextField1.getText();
      int CadMat=Integer.parseInt(jTextField2.getText());
      String CadNome=jTextField3.getText();
      int registro = stmt.executeUpdate("
                        update TabFicha set Nome=' " +CadNome+ " ', Matricula='
                        "+CadMat+ " 'where Nome=' " + ConsNome + " ' ");
      if (registro!=0) JOptionPane.showMessageDialog(this,"Dados Alterados!");
         else JOptionPane.showMessageDialog(this,"Dados Não Alterados!");
      stmt.close();
      con.close(); //fecha conexão com BD
      } catch(SQLException e){ //trata os erros
          JOptionPane.showMessageDialog(this,"Erro Cmdo SQL
"+e.getMessage());
      } catch(ClassNotFoundException e){
          JOptionPane.showMessageDialog(this,"Driver não encontrado");
          }
```

d) <u>Códigos para Excluir:</u>

```
try{ //tratamento de erros

        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");//busca Driver

        //conecta no BD

        Connection con=DriverManager.getConnection("jdbc:odbc:PBD-Nomes","","");

        Statement stmt= con.createStatement();//objeto comdo sql

        String ExcNome= jTextField1.getText();


        int registro=stmt.executeUpdate("delete from TabFicha where Nome=' " + ExcNome + " ' ");

        if (registro!=0) JOptionPane.showMessageDialog(this,"Dados Excluidos!");

                else JOptionPane.showMessageDialog(this,"Dados não Excluidos!");

        stmt.close();

        con.close(); //fecha conexão com BD
```

```
        } catch(SQLException e){ //trata os erros

                JOptionPane.showMessageDialog(this,"Erro Cmdo SQL
"+e.getMessage());

         } catch(ClassNotFoundException e){

                JOptionPane.showMessageDialog(this,"Driver não encontrado");

         }
```

NOTA:***Antes de inserir os códigos é preciso importar os pacotes para trabalhar com Banco de Dados e exibir caixa de mensagens:**

→ Localize o código de criação da classe. Ex:

Public Class ...

→ Antes deste comando insira os seguintes códigos:

import java.sql.*;

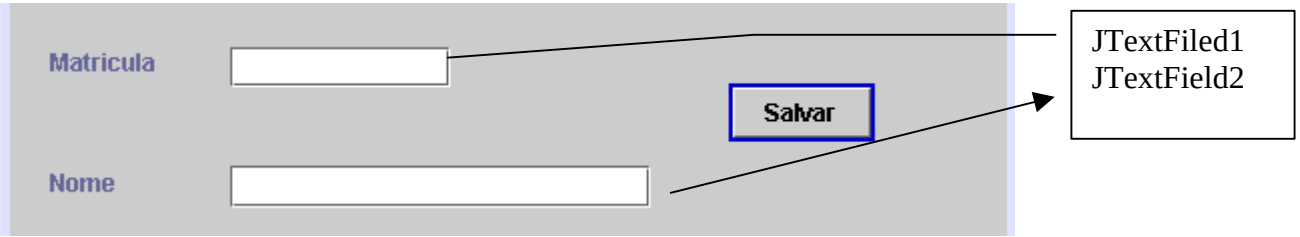
import javax.swing.*;

Exemplo Completo

- Crie uma aplicação para cadastrar a Matricula e os Nomes dos Funcionários de uma Empresa.

1. Clique no Menu File → New → Selecione Java GUI Forms → JFrame Forms→ clique no botão Next. Na janela que se abre, no campo Name, digite “SistCadFunc” e clique no Botão Finish.

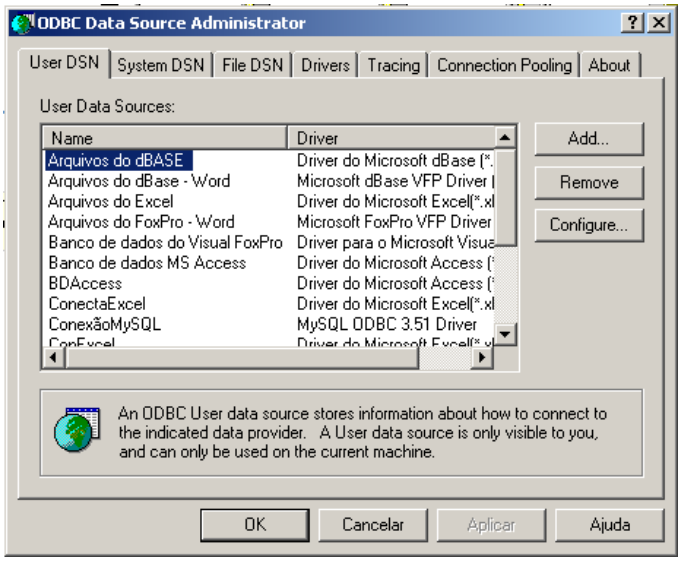
2. Modifique o Layout do JFrame para AbsoluteLayout e insira os seguintes componentes: dois JLabel (Nome e Matricula), dois JTextField e um JButton (Salvar)

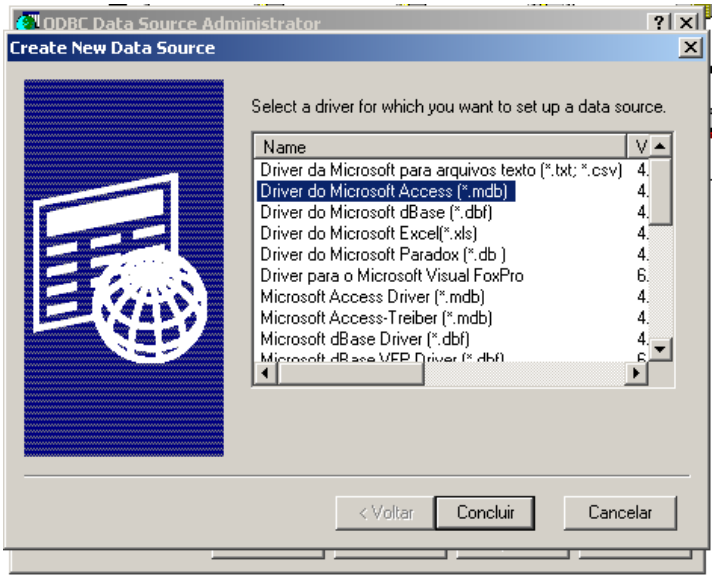


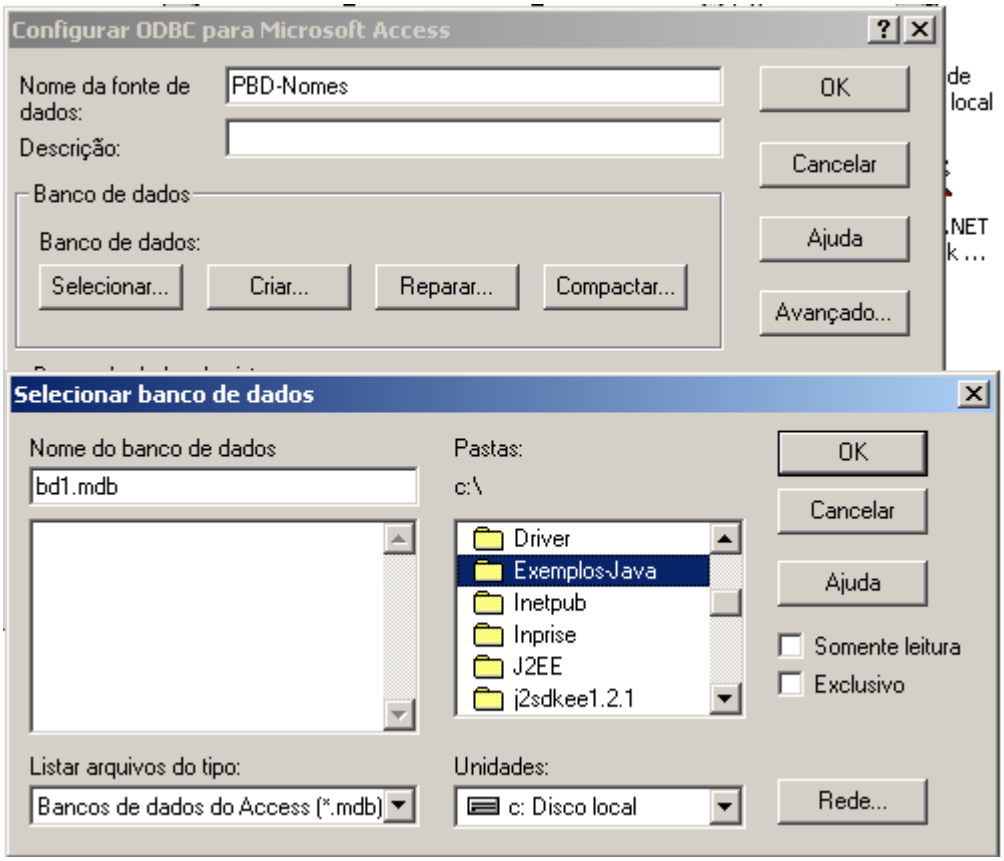
3. A parte visual esta construída, falta o código para salvar os dados.

4. Abra o Access e crie um BD (BDCadastro) com uma Tabela (TabFicha) com os campos: Matricula (Tipo Numérico/Chave Primária) e Nome – Tipo Texto.

5. Devemos criar a "Ponte" JDBC/ODBC: abra o Painel de Controle → Ferramentas Administrativas→ Fontes de Dados ODBC→ clique no Botão Add → Selecione o Driver p/ BD Access → clique no botão Concluir → no campo "Nome da Fonte de Dados" digite "PBD-Nomes" → clique no botão Selecionar → Procure o BD criado (BDCadastro) e clique no Botão OK →OK → OK


ODBC Data Source Administrator
User DSN
System DSN
File DSN
Drivers
Tracing
Connection Pooling
About
User Data Sources:
Name
Driver
Arquivos do dBASE
Driver do Microsoft dBase (*.
Arquivos do dBase - Word
Microsoft dBase VFP Driver
Arquivos do Excel
Driver do Microsoft Excel(*.xl
Arquivos do FoxPro - Word
Microsoft FoxPro VFP Driver
Banco de dados do Visual FoxPro
Driver para o Microsoft Visua
Banco de dados MS Access
Driver do Microsoft Access (
BDAccess
Driver do Microsoft Access (
ConectaExcel
Driver do Microsoft Excel(*.xl
ConexãoMySQL
MySQL ODBC 3.51 Driver
Add...
Remove
Configure...
An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.
OK
Cancelar
Aplicar
Ajuda


ODBC Data Source Administrator
Create New Data Source
Select a driver for which you want to set up a data source.
Name
Driver da Microsoft para arquivos texto (*.txt; *.csv)
Driver do Microsoft Access (*.mdb)
Driver do Microsoft dBase (*.dbf)
Driver do Microsoft Excel(*.xls)
Driver do Microsoft Paradox (*.db)
Driver para o Microsoft Visual FoxPro
Microsoft Access Driver (*.mdb)
Microsoft Access-Treiber (*.mdb)
Microsoft dBase Driver (*.dbf)
< Voltar
Concluir
Cancelar



6. Criado o BD e a Ponte ODBC, volte para a sua aplicação no NetBeans e dê um duplo clique no JButton (Salvar) para codificar a conexão:

Depois de "**private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {**", digite:

```
try{ //tratamento de erros
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");//busca Driver
    //conecta no BD
    Connection con = DriverManager.getConnection("jdbc:odbc:PBD-Nomes","","");
    Statement stmt = con.createStatement();      //objeto comdo sql
    String CadNome = jTextField1.getText();  //obtém nome digitado
    int CadMat=Integer.parseInt( jTextField2.getText() );
```

Aspas Simples e Aspas Duplas

```
      stmt.executeUpdate("insert into TabFicha (Matricula,Nome) values ('" +
CadMat + "','" + CadNome +  "')" );
      JOptionPane.showMessageDialog( this, " Dados Salvos! ");
      con.close();  // fecha conexão com BD
        }  catch( SQLException e){ //trata os erros
         JOptionPane.showMessageDialog(this, "Erro Cmdo SQL " +
e.getMessage() );
        } catch( ClassNotFoundException e){
         JOptionPane.showMessageDialog( this, " Driver não encontrado " );
}
```

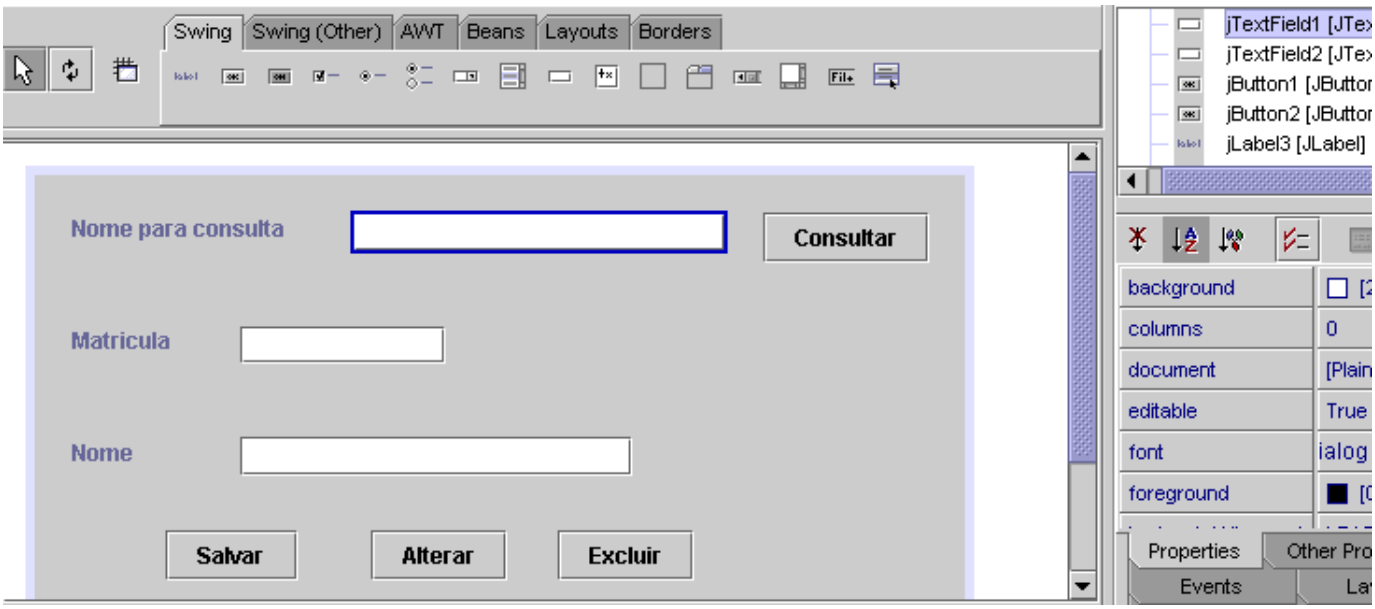
Aspas Duplas e Aspas Simples

7. Compile Execute.

8. Realizando operações de Consulta, Alteração e Exclusão em BD:

Insira três Botões para executar as operações restantes no Banco de Dados (Consulta, Alteração e Exclusão), um JLabel, para exibir o Texto “Nome para Consulta/Exclusão”, supondo que estas operações serão feitas através do campo “Nome”, e insira também um JTextField para p usuário poder digitar o “Nome”, ou crie um novo Form contendo os JLabel, JTextField , e JButton, sendo que o mesmo deverá ser chamado por outro Form através de um JMenurBar.

Supondo que foram inseridos os novos componentes da seguinte forma:



Basta dar um duplo clique no botão desejado e inserir o respectivo código para a respectiva operação:

Consultar

```
boolean consulta=true;
try{ //tratamento de erros
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");//busca Driver
        //conecta no BD
        Connection con=DriverManager.getConnection("jdbc:odbc:PBD-
Nomes","","");
        Statement stmt= con.createStatement();//objeto comdo sql
      String ConsNome= jTextField1.getText();
      ResultSet RS= stmt.executeQuery("Select * from TabFicha where nome='" +
ConsNome + "' ");
      while (RS.next()){
         int Mat= RS.getInt("Matricula");
         jTextField2.setText(String.valueOf(Mat));
         jTextField3.setText(RS.getString("Nome"));
```

Aspas Simples e Aspas Duplas

Aspas Duplas e Aspas Simples

```
            consulta=false;
            JOptionPane.showMessageDialog(this,"Dados Encontrados!");
            }
        if (consulta) JOptionPane.showMessageDialog(this,"Dados Não
Encontrados!");
        RS.close ();
        stmt.close();
        con.close(); //fecha conexão com BD
          } catch(SQLException e){ //trata os erros
            JOptionPane.showMessageDialog(this,"Erro Cmdo SQL
"+e.getMessage());
            } catch(ClassNotFoundException e){
             JOptionPane.showMessageDialog(this,"Driver não encontrado");
            }
```

Alterar:

```
try{ //tratamento de erros
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");//busca Driver
          //conecta no BD
          Connection con=DriverManager.getConnection("jdbc:odbc:PBD-
Nomes","","");
          Statement stmt= con.createStatement();//objeto comdo sql
        String ConsNome= jTextField1.getText();
        int CadMat=Integer.parseInt(jTextField2.getText());
        String CadNome=jTextField3.getText();
        int registro = stmt.executeUpdate("update TabFicha set Nome='" +CadNome+
"', Matricula='"+CadMat+ "'where Nome='" + ConsNome + "' ");
        if (registro!=0) JOptionPane.showMessageDialog(this,"Dados Alterados!");
            else JOptionPane.showMessageDialog(this,"Dados Não Alterados!");
        stmt.close();
```

```
        con.close(); //fecha conexão com BD
          } catch(SQLException e){ //trata os erros
           JOptionPane.showMessageDialog(this,"Erro Cmdo SQL
"+e.getMessage());
           } catch(ClassNotFoundException e){
            JOptionPane.showMessageDialog(this,"Driver não encontrado");
           }
```

Excluir:

```
try{ //tratamento de erros
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");//busca Driver
          //conecta no BD
          Connection con=DriverManager.getConnection("jdbc:odbc:PBD-
Nomes","","");
          Statement stmt= con.createStatement();//objeto comdo sql
           String ExcNome= jTextField1.getText();
           int registro=stmt.executeUpdate("delete from TabFicha where Nome='" +
ExcNome + "'");
          if (registro!=0)
                JOptionPane.showMessageDialog(this,"Dados Excluidos!");
                 else
                    JOptionPane.showMessageDialog(this,"Dados não Excluidos!");
            stmt.close();
            con.close(); //fecha conexão com BD
          } catch(SQLException e){ //trata os erros
           JOptionPane.showMessageDialog(this,"Erro Cmdo SQL
"+e.getMessage());
           } catch(ClassNotFoundException e){
            JOptionPane.showMessageDialog(this,"Driver não encontrado");
           }
```

9. Compile Execute.

OBS: Como os comando de conexão com o Banco de Dados estão se repetido, os mesmos poderiam ser implementados dentro de uma função e serem executados através de uma chamada desta função quando se desejar estabelecer uma conexão com o banco de dados para realizar uma determinada operação em suas tabelas, diminuindo a quantidade de código a ser digitado.