



IDE (Integrated Development Environment) / RAD (Rapid Application Development)

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Agosto de 2012



Introdução

- Uma “interface gráfica” ou GUI (*Graphical User Interface*) apresenta um mecanismo amigável ao usuário para interagir com um aplicativo;
- **Objetivo:** tornar o *software* cada vez mais amigável para facilitar a interação do usuário com este.



Comentário

- Interfaces consistentes permitem que o usuário aprenda mais rápido novos aplicativos.



Exemplo Interface Gráfica



- Comparação com ambiente caractere.



Pergunta

- Você já se imaginou acessando a Internet num computador onde o *mouse* não funciona? E fora do ambiente gráfico?



Importante

- A interface gráfica deve facilitar o entendimento da aplicação, ou seja, deve ser o mais amigável possível;
- Não abuse nas cores:
 - O uso de cores fortes não é recomendável para aplicativos comerciais;
 - O uso de uma única cor também (o tradicional cinza) às vezes torna a tela desagradável ou de difícil entendimento.
- Faça uso de imagens significativas para melhorar o entendimento do *software*. Lembre-se: “uma imagem vale mais que mil palavras!!!”



Pré-requisitos

- Conhecimento de Programação Orientada a Objetos;
- Cada componente (formulário, botão, *menu*, barra de rolagem e etc.) será um objeto que deverá ser instanciado em seu aplicativo;
- Conhecimento de Programação Orientada a Eventos (que nós abordaremos nesta disciplina);
- Conhecimento de uma linguagem de programação que possua todos estes quesitos;
- Nesta disciplina usaremos linguagem Java e a IDE Netbeans.



Ferramentas RAD

- São *softwares* que nos proporcionam uma maneira rápida de desenvolvermos outros *softwares*;
- Nos fornecem componentes prontos para serem usados (*drag and drop*):
 - janelas,
 - rótulos,
 - botões,
 - campos de texto,
 - campos de imagem e
 - etc.



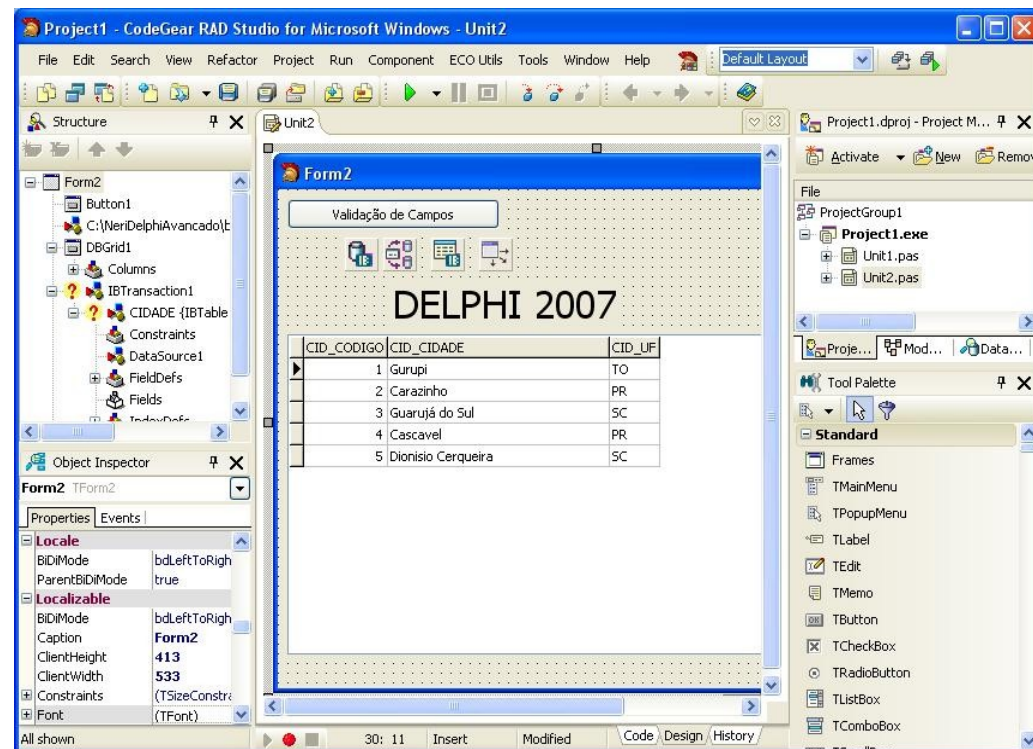
Exemplos de Ferramentas RAD

- Delphi,
- Netbeans,
- Dreamweaver,
- Visual Studio .Net e
- Etc.



Delphi

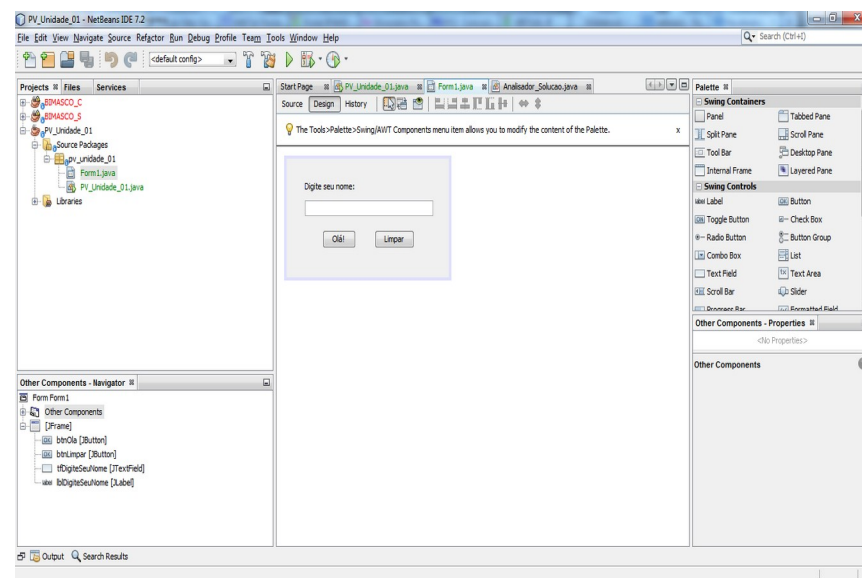
- Empresa: Embarcadero
- Linguagens: Object Pascal
- S.O.: Windows
- Programação Estruturada, Orientada a Objetos e Eventos.
- Versão: Paga.





Netbeans

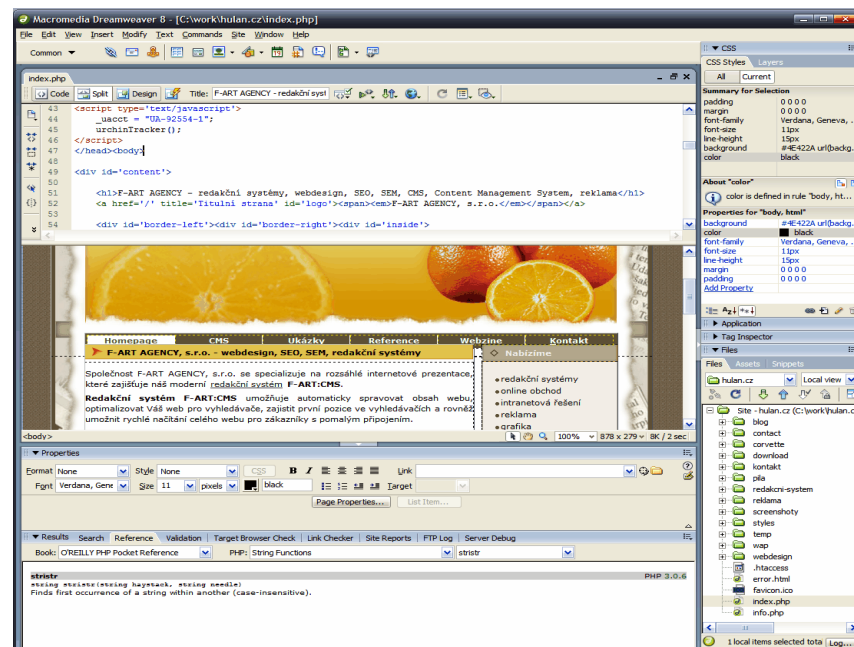
- Empresa: Oracle
- Linguagens: Java, PHP, C++, HTML, Java Script e etc.
- S.O.: Windows, GNU/Linux, FreeBSD, Unix, Mac OS e etc.
- Programação Orientada a Objetos e Eventos.
- Versão: Gratuita.





Dreamweaver

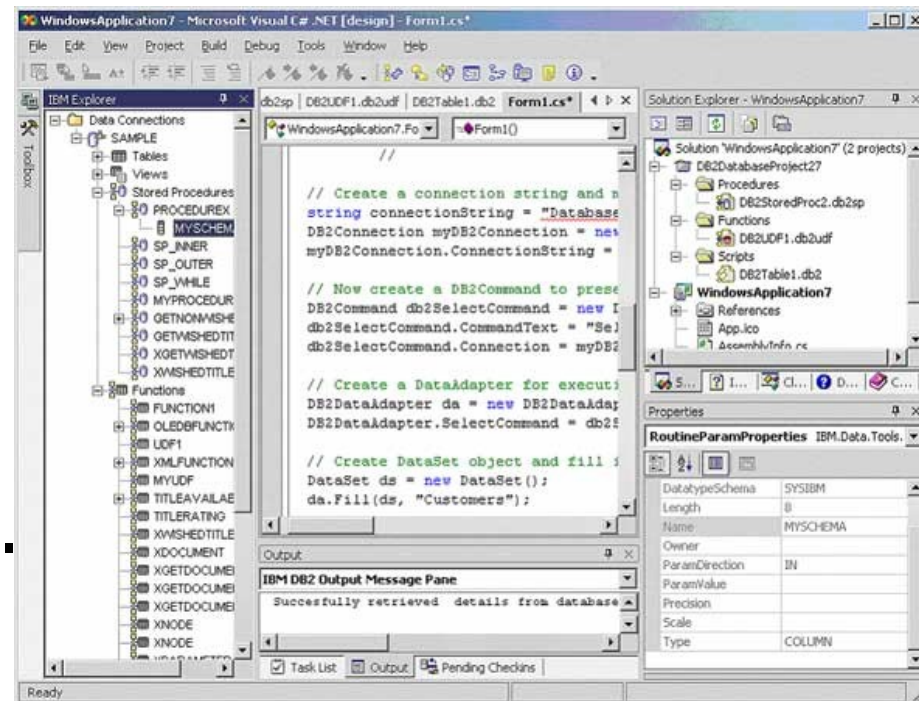
- Empresa: Adobe
- Linguagens: HTML, Flash, Java Script e etc.
- S.O.: Windows.
- Programação Estruturada, Orientada a Objetos e Eventos.
- Versão: Paga.





Visual Studio .Net

- Empresa: Microsoft
- Linguagens: C#, J#, VB# e etc.
- S.O.: Windows.
- Programação Orientada a Objetos e Eventos.
- Versões: Paga e Gratuita(recursos reduzidos).





Nesta Disciplina Usaremos a IDE NetBeans

- Inicializar a ferramenta;
- E vamos a nossa primeira atividade.



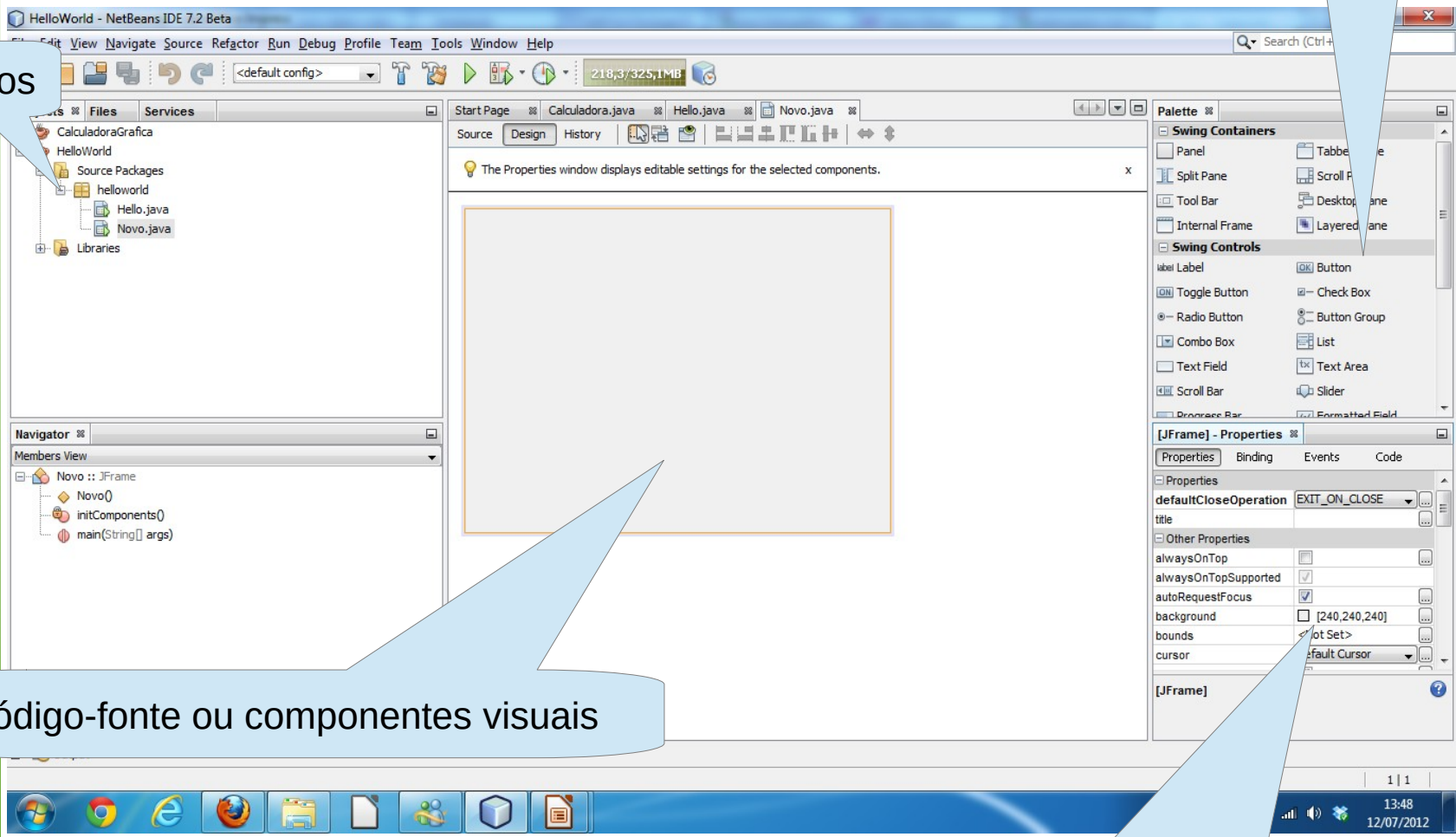
IDE NetBeans

- Desenvolvida pela Sun Microsystems, empresa comprada pela Oracle;
- Uma das melhores existentes no mercado;
- Proporciona extrema facilidade na construção de aplicativos gráficos no estilo *drag and drop*;
- Proporciona também uma maneira fácil de se acessar e alterar as propriedades e métodos de cada componente visual, através da paleta “*Properties*”;
- Para isto, utilizaremos um pacote de classes conhecido como SWING, onde poderemos encontrar diversos componentes visuais;
- Vamos começar!



Entendendo a IDE NetBeans

Projetos

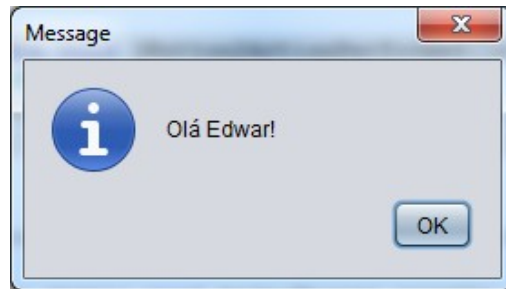
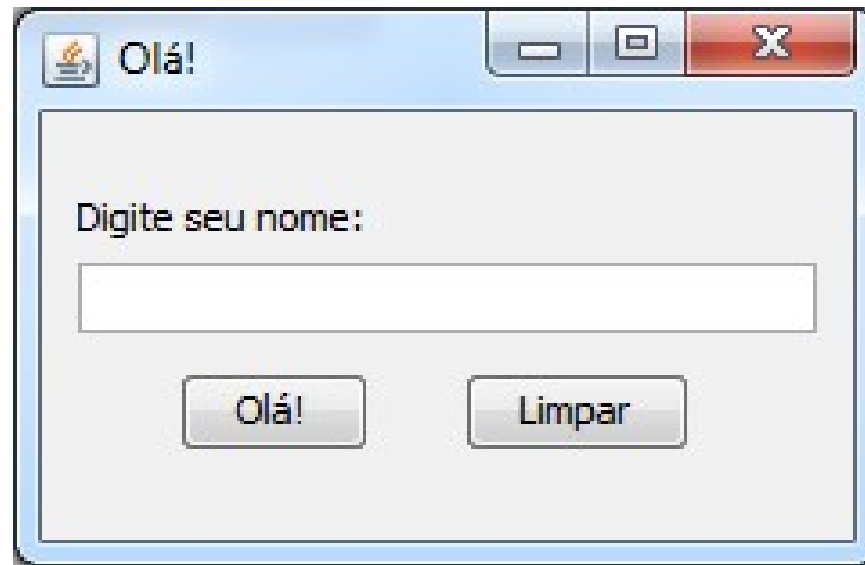


Código-fonte ou componentes visuais

Propriedades dos componentes visuais



Primeiro Aplicativo Gráfico



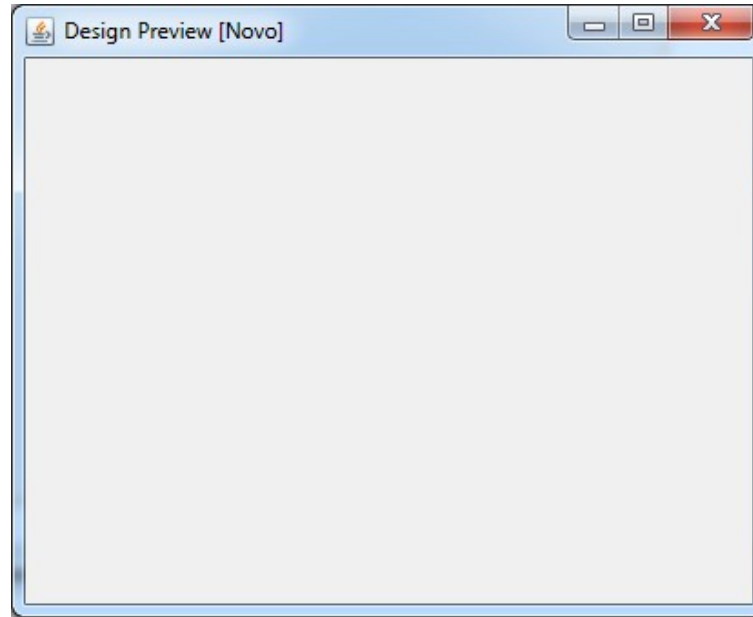


JFrame

- Formulário:
 - A maioria das janelas é uma instância ou subclasse dessa classe;
 - Fornece a barra de título;
 - Fornece botões para minimizar, maximizar e fechar a aplicação.
- Propriedades:
 - *title* - define o título da janela;
 - *background* - define a cor de fundo da janela.



Componente JFrame



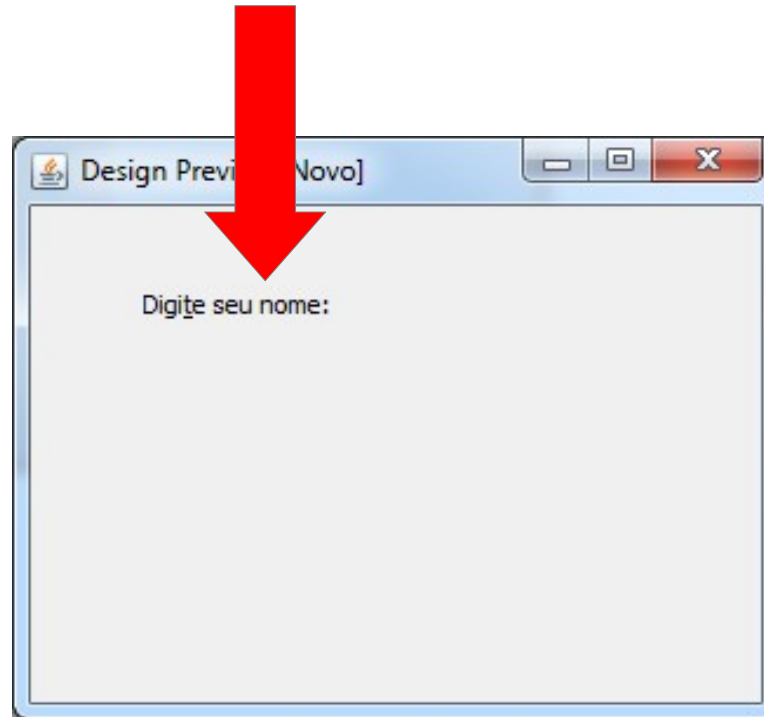


JLabel

- Rótulo:
 - Instruções de texto ou informações que declaram o propósito de cada componente.
- Propriedades:
 - *text* - altera o texto apresentado pelo componente;
 - *displayedMnemonic* - permite a criação de uma tecla de atalho para o componente;
 - *labelFor* - faz a ligação do rótulo a um outro componente.



Componente JLabel



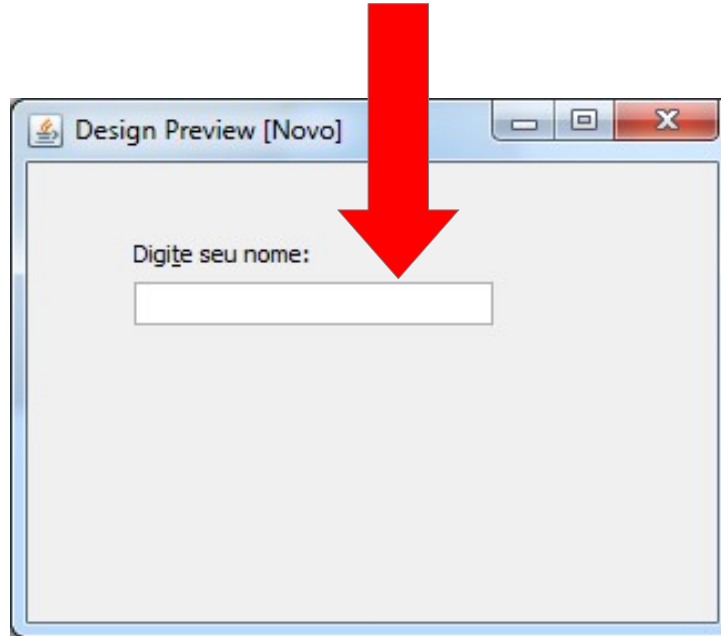


JTextField

- Campo de Texto:
 - Utilizado para entrada de dados simples em formulários;
 - Ideal para entradas que não ultrapassem o tamanho de uma linha.
- Propriedades:
 - *text* – propriedade que armazenará o texto digitado pelo usuário do *software*.



Componente JTextField



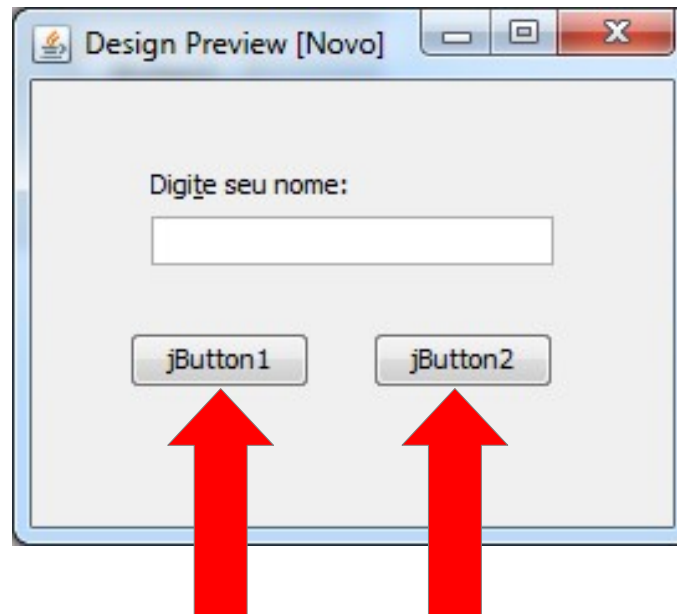


JButton

- Botão:
 - Ao ser apertado desencadeia uma ação específica previamente programada.
- Propriedades:
 - *text* – define o texto que será apresentado no botão;
 - *mnemonic* – cria uma tecla de atalho para o botão;
 - *icon* – define um ícone para o botão.
- Evento (o que é?):
 - *ActionPerformed* – evento disparado quando se aperta o botão. É neste local que será colocada a funcionalidade do botão.



Componentes JButton





JOptionPane

- Caixa de Diálogo:
 - Componente não visual;
 - Utilizadas pelas aplicações para interagir com o usuário;
 - Contém diálogos de entrada e diálogos de mensagem.



Componente JOptionPane





- Componente não visual;
- Deve ser instanciado no código-fonte, conforme exemplo a seguir:



```
JOptionPane.showMessageDialog(null, "Olá!");
```



Mensagens ao Usuário

Tipo de diálogo de mensagem	Ícone	Descrição
ERROR MESSAGE		Um diálogo que indica um erro para o usuário.
INFORMATION MESSAGE		Um diálogo com uma mensagem informativa para o usuário.
WARNING MESSAGE		Um diálogo que adverte o usuário de um problema potencial.
QUESTION MESSAGE		Um diálogo que impõe uma pergunta ao usuário. Normalmente, esse diálogo exige uma resposta, como clicar em um botão Yes ou No .
PICASSO MESSAGE	Nenhum ícone	Um diálogo que contém uma mensagem, mas nenhum ícone..



Programando os Eventos dos Botões

- A parte de programação visual foi concluída;
- Vamos programar a parte não visual para concluirmos o *software*;
- Clique no botão “Olá!”:
 - vá a paleta “Properties” do NetBeans, clique na aba “Events”;
 - serão mostrados todos os eventos do componente;
 - clique no JComboBox que aparece a frente do evento “ActionPerformed” grafado com a palavra “none”;
 - Escolha a única opção existente e logo a tela do código fonte aparecerá para você já com toda a classe do formulário criada e com o cursor posicionado no método referente ao evento “ActionPerformed”.



Programação dos Botões

- Olá!

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    if(jTextField1.getText().isEmpty()) {  
        JOptionPane.showMessageDialog(null, "Olá!");  
    }  
    else {  
        JOptionPane.showMessageDialog(null, "Olá " + jTextField1.getText() + "!");  
    }  
}
```

- Limpar

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    jTextField1.setText(null);  
}
```



Pronto!

- Agora é só executar o *software* e testar suas funcionalidades!



Observação!

- Todo componente visual possui uma “propriedade”, na aba “Code” da paleta “Properties” chamada: “Variable Name”;
- Esta propriedade permite você dar um nome específico para cada componente visual que você utilizar no seu aplicativo;
- Dar nomes significativos a cada componente é considerada uma ótima prática de programação.



Sabendo disto,

- Vamos alterar o nome dos seguintes componentes:
 - `jTextField1` → `tfDigiteSeuNome`
 - `jButton1` → `btnOk`
 - `JButton2` → `btnLimpar`
 - `NewFrame1` → `Form1`



Programação dos Botões

- Olá!

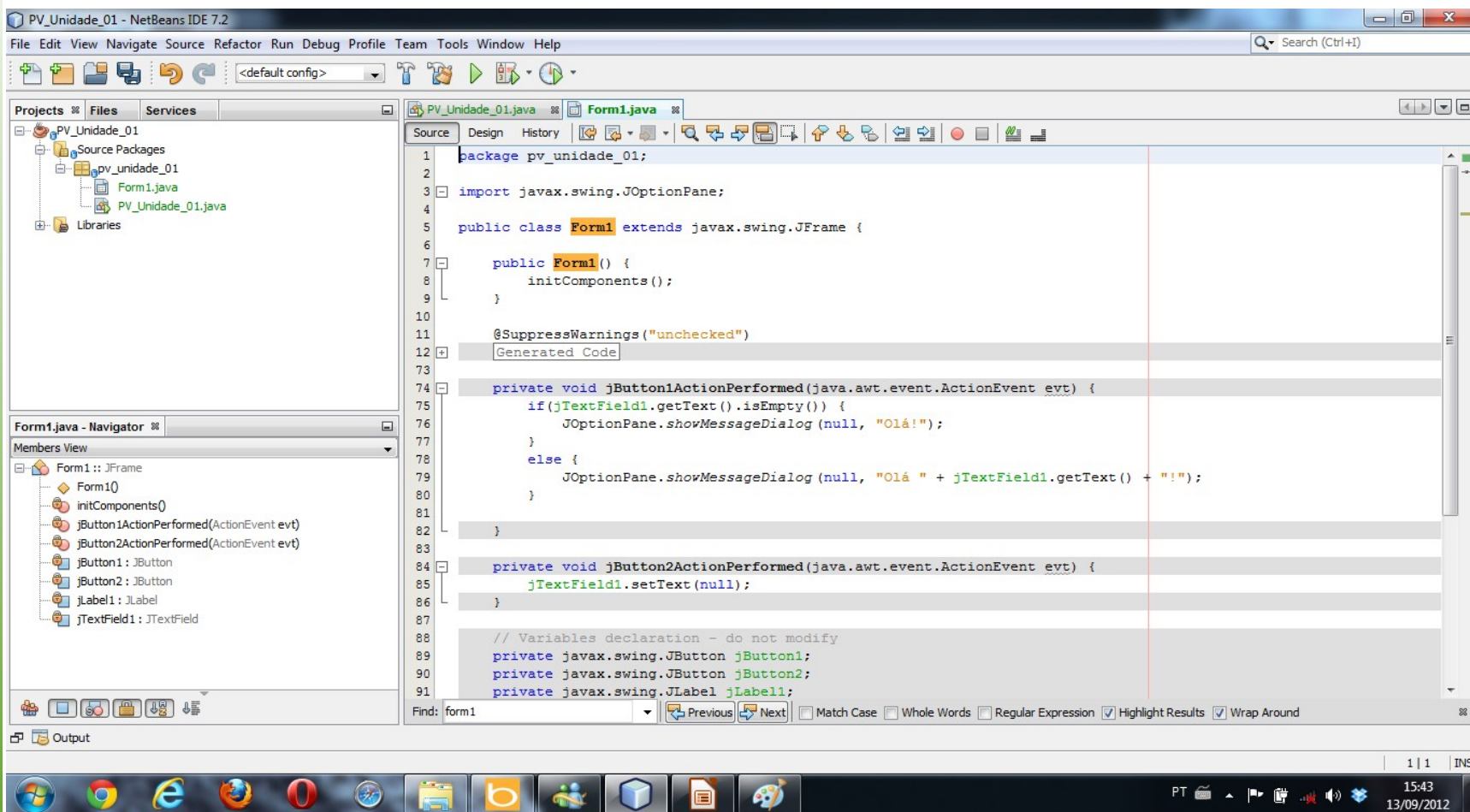
```
private void btnOlaActionPerformed(java.awt.event.ActionEvent evt) {  
    if(tfDigiteSeuNome.getText().isEmpty()) {  
        JOptionPane.showMessageDialog(null, "Olá!");  
    }  
    else {  
        JOptionPane.showMessageDialog(null, "Olá " + tfDigiteSeuNome.getText() + "!");  
    }  
}
```

- Limpar

```
private void btnLimparActionPerformed(java.awt.event.ActionEvent evt) {  
    tfDigiteSeuNome.setText(null);  
}
```



Nosso IDE está assim





Entendendo o Ambiente

The screenshot shows the NetBeans IDE interface for a project named 'PV_Unidade_01'. The 'Projects' window on the left displays the project hierarchy: 'PV_Unidade_01' (Projeto) contains 'Source Packages' (Package (Pacote)), which includes 'pv_unidade_01' (Package (Pacote)). Inside 'pv_unidade_01', there are 'Form1.java' (Classe Main (Principal)) and 'PV_Unidade_01.java' (Classe Main (Principal)). Below the source packages are 'Libraries' (Bibliotecas da Linguagem). The 'Form1.java - Navigator' window shows the class structure: 'Form1:: JFrame' (JFrame) containing 'Form1()' (Classe Main (Principal)), 'btnLimparActionPerformed(ActionEvent evt)', 'btnOlaActionPerformed(ActionEvent evt)', 'initComponents()', and 'btnLimpar : JButton'. The 'Source' window shows the code for 'Form1.java', including imports, class declaration, and method implementations.

```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
PV_Unidade_01
Source Packages
pv_unidade_01
Form1.java
PV_Unidade_01.java
Libraries
PV_Unidade_01.java
Form1.java
Source Design History
import javax.swing.JOptionPane;
class Form1 extends javax.swing.
public Form1() {
    initComponents();
}
@SuppressWarnings("unchecked")
Generated Code
73
74 private void btnOlaActionPerformed
75     if(tfDigiteSeuNome.getText()
76         JOptionPane.showMessageDialog
77     }
78     else {
79         JOptionPane.showMessageDialog
80     }
81 }
82
83 private void btnLimparActionPerformed
84     tfDigiteSeuNome.setText(null
```

JFrame

Projeto

Package (Pacote)

Classe Main (Principal)

Bibliotecas da Linguagem



Entendendo o Código

- Classe Principal

```
PV_Unidade_01.java Form1.java
Source History
1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5  package pv_unidade_01;
6
7  /**
8   *
9   * @author Edwar Saliba Júnior - http://www.esj.eti.br
10  */
11  public class PV_Unidade_01 {
12
13  /**
14   * @param args the command line arguments
15   */
16  public static void main(String[] args) {
17      Form1 form1 = new Form1();
18      form1.setVisible(true);
19  }
20 }
```

Comentário

Declaração de Pacote

Comentário

Declaração de Classe

Comentário

Declaração do método "main" (Principal)

Chamada do método "setVisible", que faz o JFrame aparecer na tela do computador

Declaração e instanciação da classe "Form1" como objeto "form1"



Entendendo o Código

- Classe Form1

```
package pv_unidade_01;

import javax.swing.JOptionPane;

public class Form1 extends javax.swing.JFrame {

    public Form1() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void btnOlaActionPerformed(java.awt.event.ActionEvent evt) {
        if(tfDigiteSeuNome.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "Olá!");
        }
        else {
            JOptionPane.showMessageDialog(null, "Olá " + tfDigiteSeuNome.getText() + "!");
        }
    }

    private void btnLimparActionPerformed(java.awt.event.ActionEvent evt) {
        tfDigiteSeuNome.setText(null);
    }

    // Variables declaration - do not modify
    private javax.swing.JButton btnLimpar;
    private javax.swing.JButton btnOla;
    private javax.swing.JLabel lblDigiteSeuNome;
```

Pacote no qual a Classe está inserida

Importe de outra classe java, de outro pacote

Declaração da classe e implementação de herança

Método construtor

Código gerado automaticamente para os componentes colocados no JFrame

Evento disparado ao se pressionar o botão "Olá"

Evento disparado ao se pressionar o botão "Limpar"

Variáveis criadas para os componentes colocados no JFrame



Perguntas???



Tarefa

- Leiam o tutorial “**NetBeans – Conhecendo um pouco da IDE**”
 - pode ser encontrado em: <http://www.esj.eti.br>
| [Educação](#) | [IFTM](#) | [Campus Paracatu](#) |
[Tecnologia em Análise e Desenvolvimento de Sistemas](#) | [Programação Visual](#) | [Tutoriais](#)



Bibliografia

- DEITEL, H. M.; DEITEL, P. J. **Java Como Programar**; 6. ed., São Paulo: Pearson, 2005.



Nivelamento

- Utilizando o conhecimento que você possui das disciplinas de POO, faça um programa em “Linguagem Java” que resolva o seguinte problema:
 - crie um software que permita a um usuário *cadastrar, excluir, alterar, consultar e imprimir* quantos veículos ele quiser em m ArrayList. Cada veículo será referenciado por sua posição no ArrayList. Para cada veículo deverá ser possível guardar as seguintes informações:
 - String marca
 - String modelo
 - int anoFabricacao
 - Int anoModelo
 - float preco
 - seu software deverá, obrigatoriamente, fazer uso de uma classe “gerencia” ou “fichário”.
- Envie sua solução, apenas o projeto compactado (anexado), para o e-mail:
 - eddiesaliba3@gmail.com
 - no campo assunto coloque: **PV - Nivelamento**
 - e no corpo do e-mail coloque seu **nome completo**.