

# EducaCiência FastCode

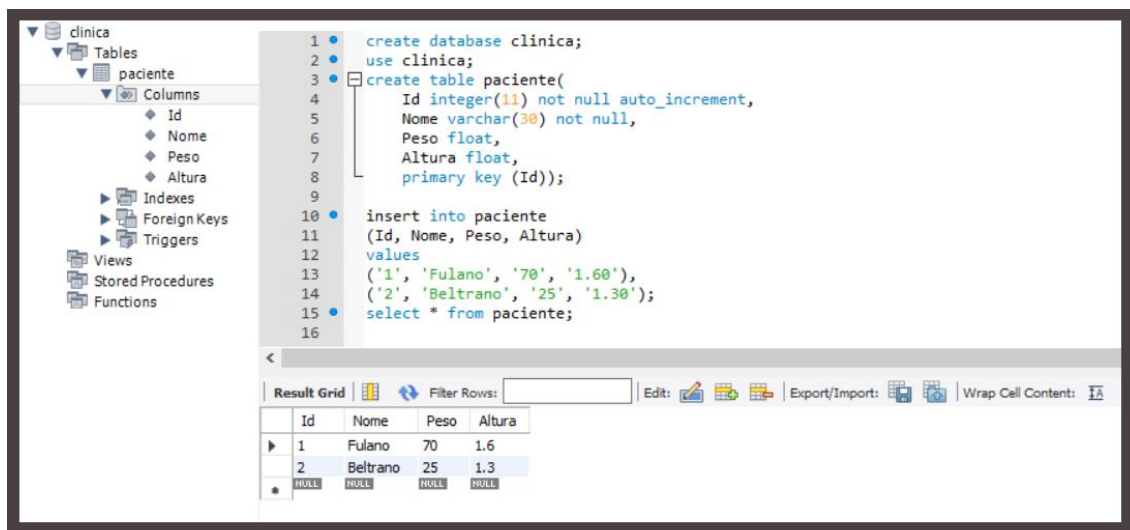
Fala Galera,

- Artigo: 20/2020 Data: Maio/2020
- Público Alvo: Desenvolvedores – Iniciantes
- Tecnologia: Java
- Tema: CRUD – criando Método Inserir
- Link: <https://github.com/perucello/DevFP>

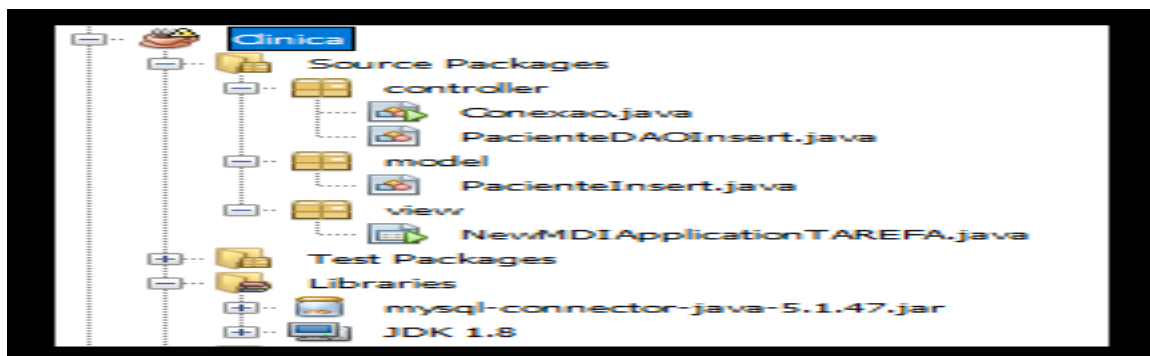
Criaremos uma série de artigos para explanarmos o CRUD em um projeto Desktop. Neste artigo 20/2020 daremos inicio no nosso proposito, criando o MÉTODO INSERIR.

Para este ambiente , já temos nosso Banco de Dados criado e chamado “clinica”. Nosso ambiente consiste em:

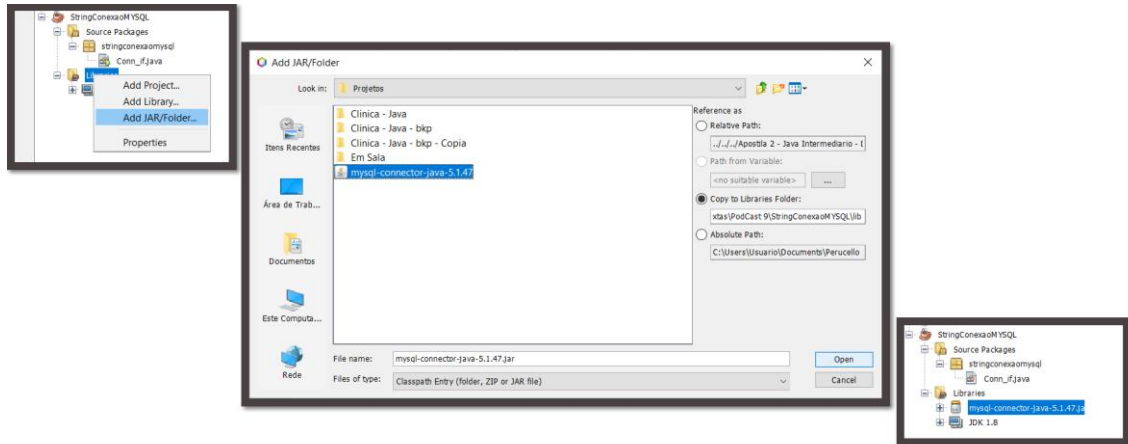
⇒ Banco de Dados MySql



⇒ Padrão MVC de arquitetura



Vale lembrar que temos que adicionar nossa biblioteca “jar” de Conexão. Vamos lá.

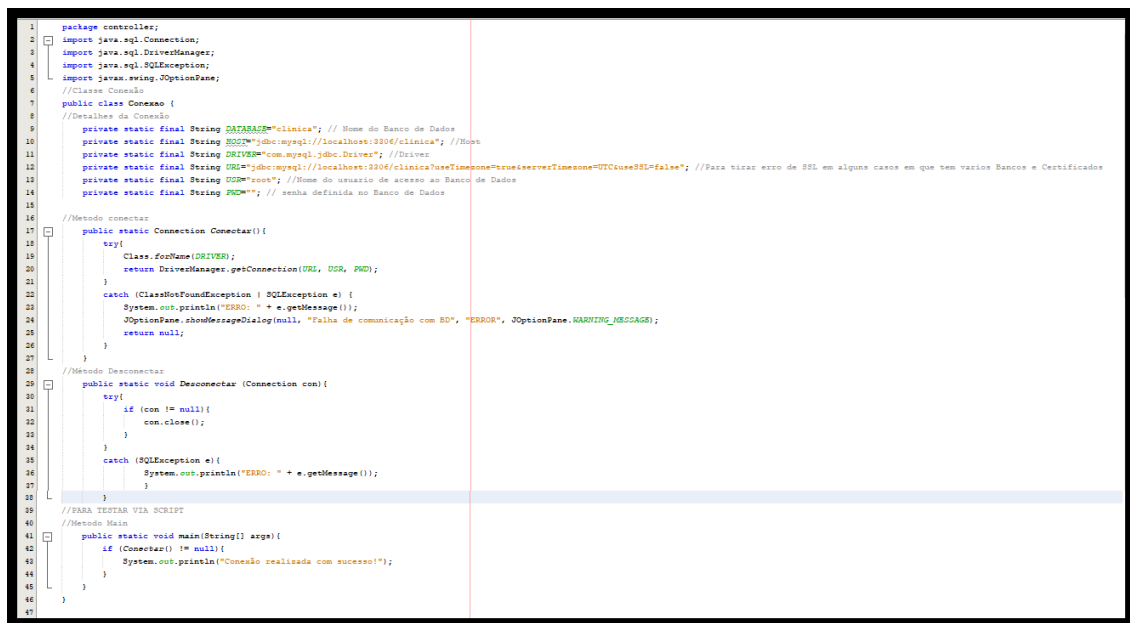


# CRUD - Insert

Vou tentar detalhar o máximo possível os passos para que não fique nenhuma dúvida , ou que possa ser replicado sem maiores problemas o nosso Projeto.

**Pacote Controller** – neste pacote, teremos duas classes, a nossa Classe Conexão e nossa Classe PacienteDAOInserir.

a) Classe conexão – nesta classe, teremos nossa String de Conexão com o Banco de Dados



- b) Classe PacienteDAOInserir – nesta classe, adicionaremos as Strings Sql que será executada junto ao nosso Banco de Dados, vale ressaltar que temos alguns passos pré definidos no nosso código, onde estou utilizando do recurso *//comentário* para explicar melhor.

```

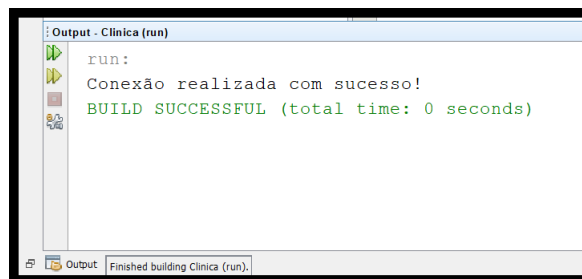
1 package controller;
2 import java.sql.Connection;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.sql.Statement;
7 import model.PacienteInsert;
8 //Classe PacienteDAOInserir
9 public class PacienteDAOInserir {
10     //Abrindo nossa conexão com Banco de Dados
11     private final Connection con;
12     private PreparedStatement cmd;
13     //Método para abrir nossa Conexão
14     public PacienteDAOInserir() {
15         this.con = Conexao.Conectar();
16     }
17     //Criando nosso Método para Inserir - insert do Banco de Dados
18     public int inserir(PacienteInsert p) {
19         try {
20             String sql = "insert into paciente (nome, peso, altura) values (?, ?, ?)";
21             cmd = con.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);
22             cmd.setString(1, p.getNome());
23             cmd.setFloat(2, p.getPeso());
24             cmd.setFloat(3, p.getAltura());
25             if (cmd.executeUpdate() > 0) {
26                 ResultSet rs = cmd.getGeneratedKeys();
27                 return (rs.next() ? rs.getInt(1) : -1);
28             } else {
29                 return -1;
30             }
31         } catch (SQLException e) {
32             System.out.println("ERRO SQL: " + e.getMessage());
33             return -1;
34         } finally {
35             Conexao.Desconectar(con);
36         }
37     }
38 }
39
40

```

Com isso, temos nosso Pacote Controller já pronto, temos nossa String de Conexão com o Banco de Dados OK e temos nossa Classe que será responsável por deter a manipulação com o Banco de Dados Ok.

Vamos apenas testar nossa conexão com o Banco de Dados.

- Subir WampServer
- Clicar com Botão na classe Conexão e pedir para executar (run)



```

Output - Clínica (run)
run:
Conexão realizada com sucesso!
BUILD SUCCESSFUL (total time: 0 seconds)

```

**Pacote Model** – nesta classe, teremos nossa Regra de Negócio, porém, neste Projeto em questão, vamos ter nossos Métodos Getters e Setters – lembrando que no Netbeans, temos um recurso para gerar estes Métodos automáticos, onde estamos informando o momento em nosso Script.

```

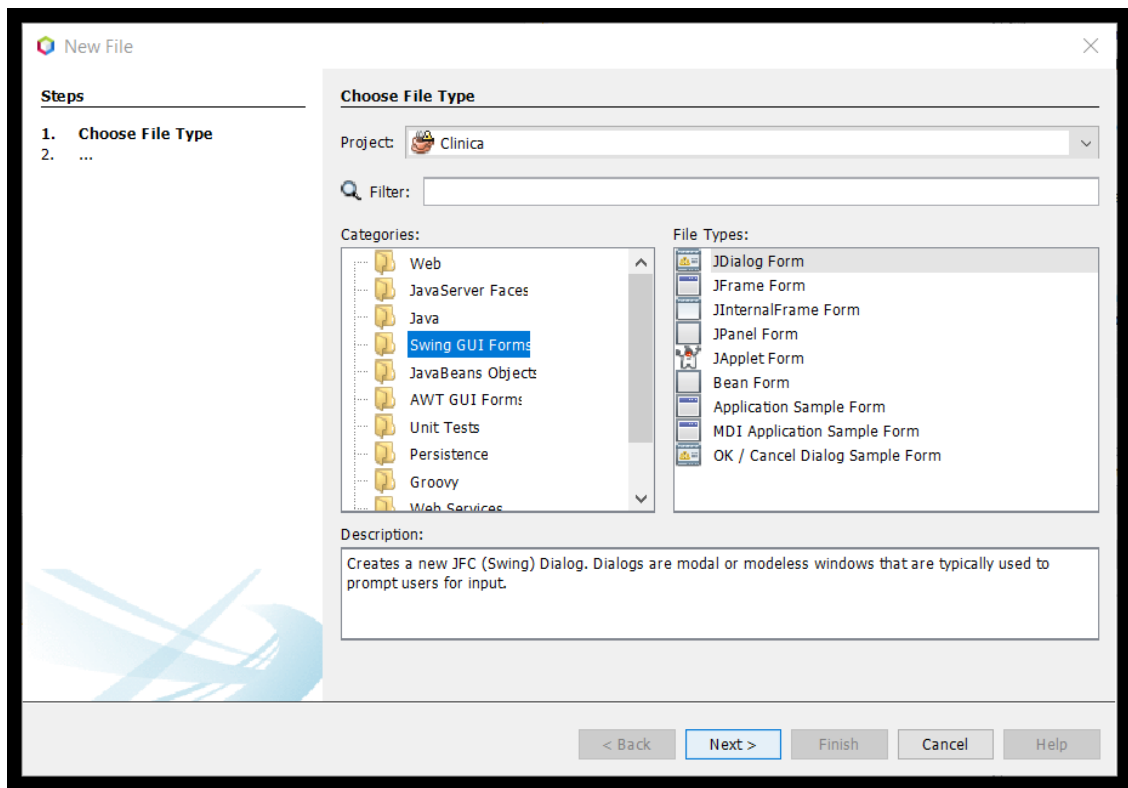
1 package model;
2 public class PacienteInsert {
3     //Criando nossos atributos - variaveis
4     private int id;
5     private String nome;
6     private float peso;
7     private float altura;
8
9     public PacienteInsert() {
10    }
11
12    public PacienteInsert(int id, String nome, float peso, float altura) {
13        this.id = id; //referenciando ao id do Banco de Dados
14        this.nome = nome; //referenciando ao nome no Banco de Dados
15        this.peso = peso; //referenciando ao peso no Banco de Dados
16        this.altura = altura; //referenciando à altura no Banco de Dados
17    }
18
19    //Metodos Getters e Setters - para este momento, podemos clicar com botão direito, irmos em INSERTCODE e pedirmos para gerar automaticamente os Metodos.
20    public int getId() {
21        return id;
22    }
23
24    public void setId(int id) {
25        this.id = id;
26    }
27
28    public String getNome() {
29        return nome;
30    }
31
32    public void setNome(String nome) {
33        this.nome = nome;
34    }
35
36    public float getPeso() {
37        return peso;
38    }
39
40    public void setPeso(float peso) {
41        this.peso = peso;
42    }
43
44    public float getAltura() {
45        return altura;
46    }
47
48    public void setAltura(float altura) {
49        this.altura = altura;
50    }
51
52    @Override
53    public String toString() {
54        return nome;
55    }
56
57

```

**Pacote View** – Nesse momento, após criarmos as classes nos Pacotes Controller e Model vamos criar a iteração do Projeto com o Usuario, ou seja, a Tela em que o usuário irá ver e manipular os dados para que seja inserido no Banco de Dados as informações como é nossa proposta neste artigo – CRUD Método Inserir.

Não é minha intenção deixar a Tela com o visual bem apresentado, mas, sim explorarmos os códigos do lado do BackEnd, sendo assim, vamos utilizar de um artefato visual que o próprio netbeans nos fornece.

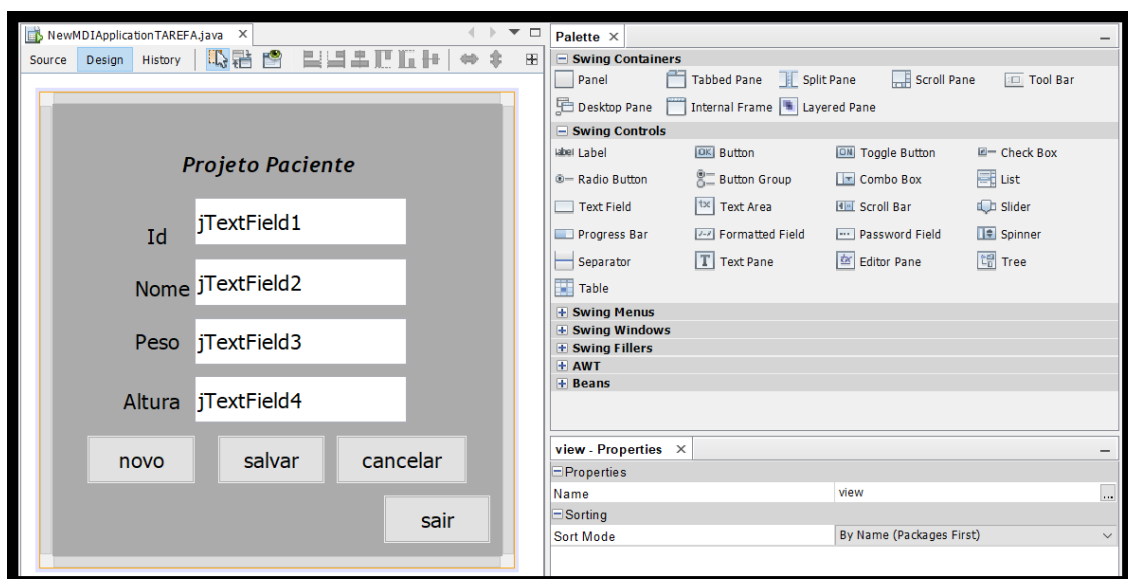
Vamos clicar com Botão direito no pacote View irmos em **New/Other** e selecionar **Swing Gui Forms**, nesta categoria vamos selecionar **MDI Application Sampe Form**.



Vamos nomear nosso Form criado MDI e vamos manipulá-lo agora.

Visto que temos nossa tabela no Banco de Dados já criada, vamos definir a tela do nosso Form MDI e criarmos a interface.

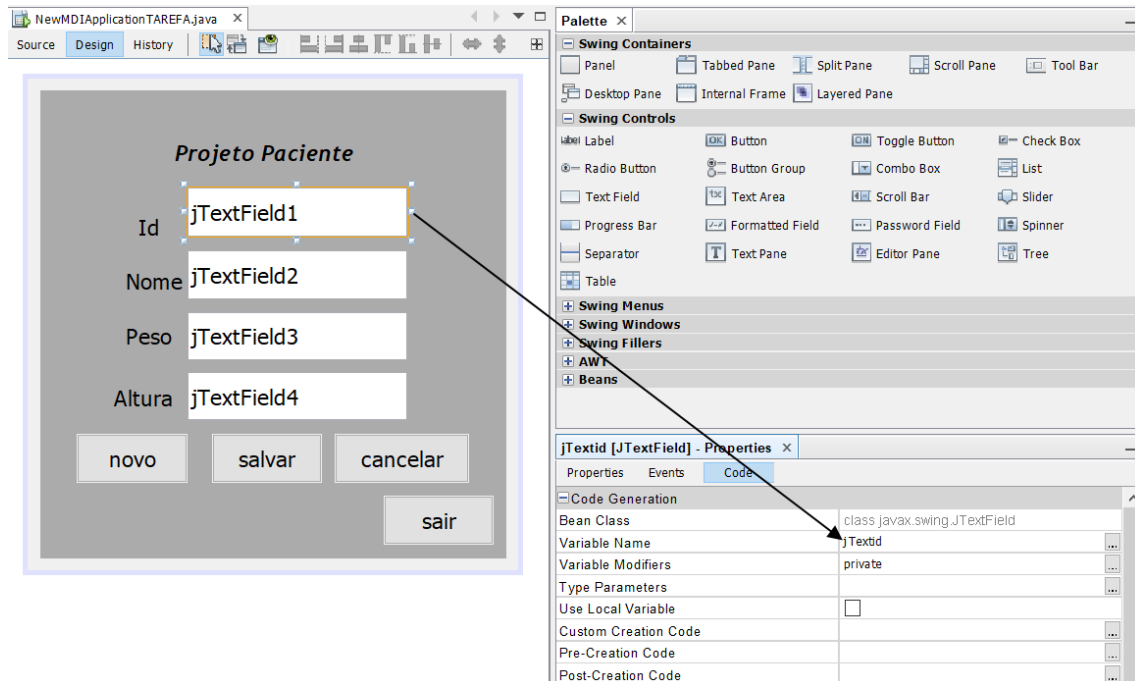
Para maior familiaridade, vou por a tela criada com os recursos do Palette utilizado para fins didáticos.



Com estes passos criados, precisamos nomear nossas variáveis, neste momento, basta você clicar no Palette inserido, ir em Propriedades, CODE e nomear nossa variável.

Para este artigo, estou nomeando as variáveis para serem didáticas na exemplificação.

Faremos este processo para todos os campos.



O netbeans, se formos na seção de códigos (source) , veremos que o Netbeans nos fornece todas estas informações que manipulamos.

```
// Variables declaration - do not modify
private javax.swing.JDesktopPane desktopPane;
private javax.swing.JButton jButtonCANCELAR;
private javax.swing.JButton jButtonNOVO;
private javax.swing.JButton jButtonSAIR;
private javax.swing.JButton jButtonSALVAR;
private javax.swing.JLabel jLabel3NOME;
private javax.swing.JLabel jLabelALTURA;
private javax.swing.JLabel jLabelID;
private javax.swing.JLabel jLabelPESO;
private javax.swing.JLabel jLabelTITULO;
private javax.swing.JTextField jTextALTURA;
private javax.swing.JTextField jTextNOME;
private javax.swing.JTextField jTextPESO;
private javax.swing.JTextField jTextid;
// End of variables declaration
```

Agora com nosso Layout da tela pronto, nossas variáveis já nomeadas , vamos aos códigos.

Na Opção Source, iremos manipular os códigos, estarei aqui colocando um Passo a Passo para que acompanhem a linha de raciocínio.

Vamos até o final do nosso código, após as declarações das variáveis, vamos identificar a última "Chave" e antes dela, vamos incluir os seguintes passos:

```

wMDIApplicationTAREFA.java
Design History
private javax.swing.JTextField jTextFieldNOME;
private javax.swing.JTextField jTextFieldPESO;
private javax.swing.JTextField jTextFieldid;
// End of variables declaration

//Passo 1 - criando metodo para configurar o Formulario
private void configurarformulario() {
    this.setTitle("Java Intermediario"); //titulo
    this.setResizable(false);
    this.setLocationRelativeTo(null);
    estadocontrole(true); //estado controle adicionado somente depois que criarmos o Metodo estadocontrole
    jTextFieldid.setEnabled(false); //nesse momento, estamos definindo que o campo jTextFieldid nao será manipulado (desabilitado)
}

//Passo 2 - Criando metodo para manipularmos o estado Controle da nossa tela - mais performatico.
private void estadocontrole(boolean e) {
    jButtonNOVO.setEnabled(e); //estamos setando que nosso Botao Novo ficará Habilitado
    jButtonSALVAR.setEnabled(!e); // estamos definindo que nosso botao Salvar nao estará habilitado
    jButtonCANCELAR.setEnabled(!e); // estamos definindo que nosso botao Cancelar nao estará habilitado

    jTextFieldNOME.setEnabled(!e); //Estamos definindo que o campo jTextFieldNOME nao estará habilitado
    jTextFieldPESO.setEnabled(!e); //Estamos definindo que o campo jTextFieldPESO nao estará habilitado
    jTextFieldALTURA.setEnabled(!e); //Estamos definindo que o campo jTextFieldALTURA nao estará habilitado
}

//Passo 3 - criando metodo para limpar os dados que estiverem no campo texto.
private void limpar() {
    jTextFieldid.setText(""); //estamos definindo que o campo jTextFieldid será limpo após ação
    jTextFieldNOME.setText(""); //estamos definindo que o campo jTextFieldNOME será limpo após ação
    jTextFieldPESO.setText(""); //estamos definindo que o campo jTextFieldPESO será limpo após ação
    jTextFieldALTURA.setText(""); //estamos definindo que o campo jTextFieldALTURA será limpo após ação
}

//Chave que fecha a classe

```

Agora que criamos nossos métodos, vamos aproveitar, e instanciar o método **configurarformulario** para iniciar junto à aplicação.

```

public class NewMDIApplicationTAREFA extends javax.swing.JFrame {

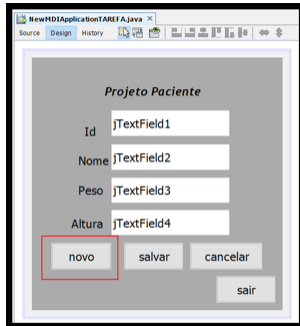
    public NewMDIApplicationTAREFA() {
        initComponents();
        //Passo 4 - vamos adicionar nosso Configurarformulario na inicialização.
        configurarformulario();
    }
}

```

Feito isso, vamos manipular ou melhor, colocar nossos códigos em nossos botões.

### Volte no Design do Projeto e dê um duplo click BOTAO NOVO

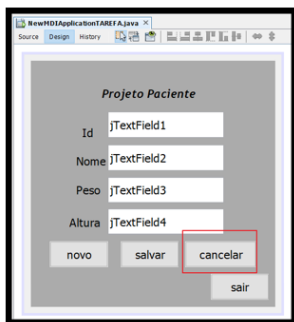
Aqui definimos que o nosso **estadocontrole** ao ter o Botão NOVO pressionado, habilitará os campos que até então temos ele bloqueado e chamamos o Método Limpar para limpar todos os textos que estiverem disponíveis.



```
private void jButtonNOVOActionPerformed(java.awt.event.ActionEvent evt) {  
    estadocontrole(false);  
    limpar();  
}
```

### Volte no Design do Projeto e dê um duplo click BOTAO CANCELAR

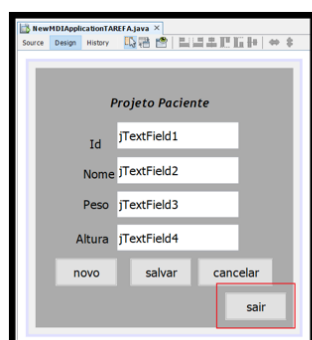
Neste botão, vamos fazer o inverso do estadocontrole do botão novo, pois ao clicar em CANCELAR, queremos que os campos textos e os botões SALVAR e CANCELAR sejam bloqueados, no entanto, basta neste momento definirmos o estadocontrole como (true).



```
private void jButtonCANCELARActionPerformed(java.awt.event.ActionEvent evt) {  
    estadocontrole(true);  
    limpar();  
}
```

### Volte no Design do Projeto e dê um duplo click BOTAO SAIR

Aqui, vamos dar um “dispose” que além de fechar nossa tela, tem como proposito finalizar o processo Java.



```
private void jButtonSAIRActionPerformed(java.awt.event.ActionEvent evt) {  
    dispose();  
}
```

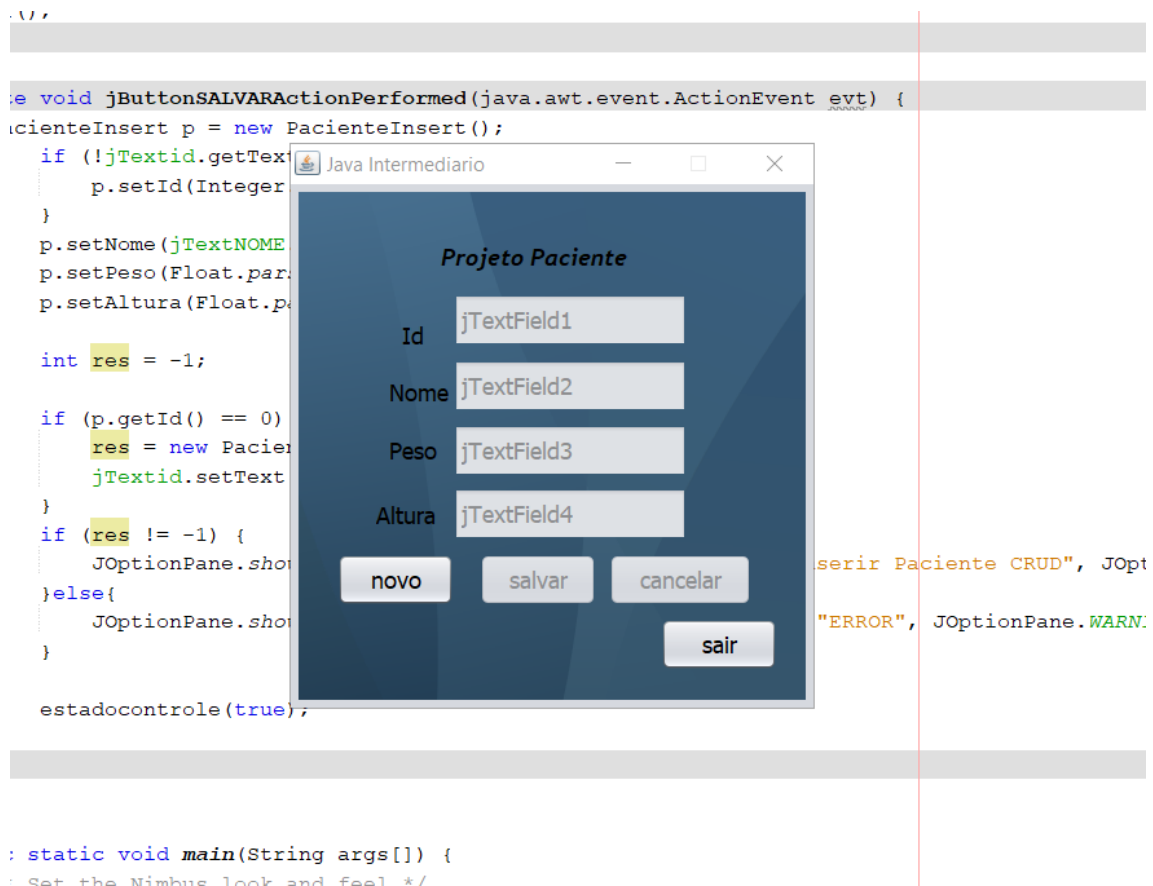
Agora está faltando apenas nosso botão SALVAR, então vamos manipular ele, pois é um código um pouco maior.



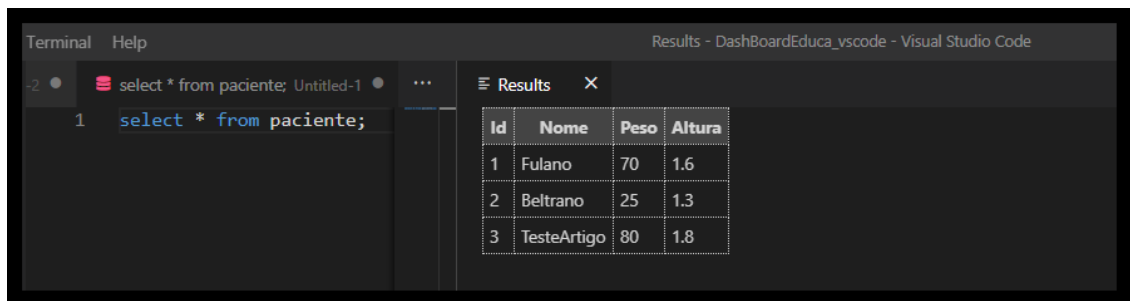
## Volte no Design do Projeto e dê um duplo click BOTAO SALVAR



Agora, que finalizamos nosso projeto, chega o momento mais interessante, vamos rodar. Vamos executar nosso Form – clique com botão direito sobre o MDI do pacote VIEW e vamos selecionar “run”.



Vimos que iniciou, basta agora manipularmos os dados, clicando no Botão Novo, inserindo dados e clicando no Botão SALVAR.



Os códigos estarão disponíveis no Git.

Até mais !  
Espero ter ajudado !