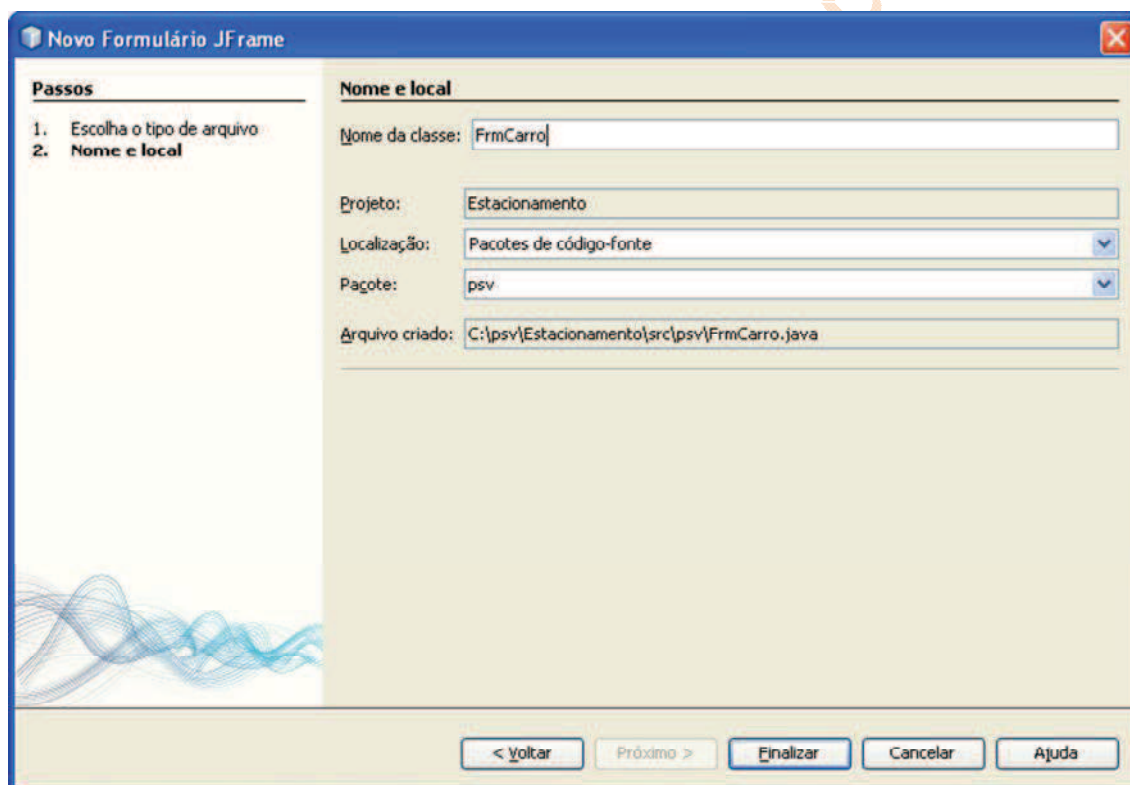


## Introdução

Para acompanhar este tutorial é recomendável que você tenha seguido todas as etapas da primeira parte deste que se encontra em: <http://www.pusivus.com.br> na seção de tutoriais.

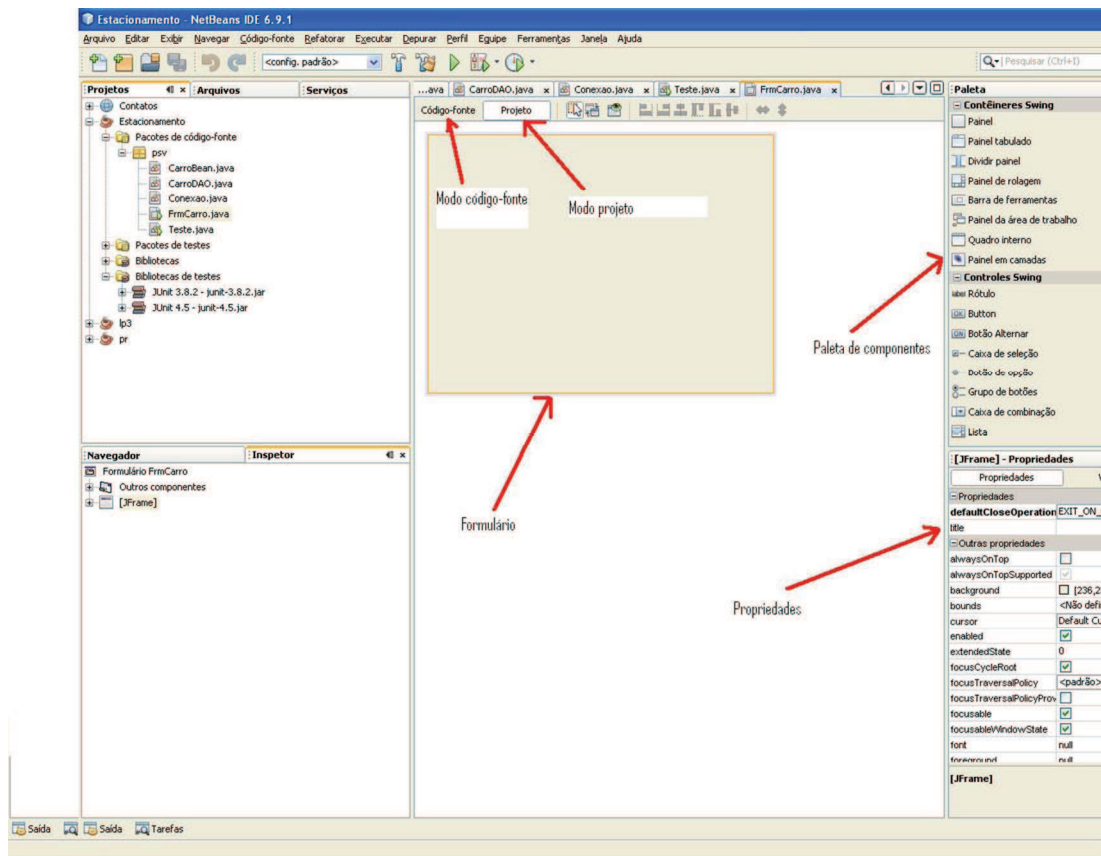
Supondo que você tenha acompanhado a primeira parte deste tutorial, você já deve ter um banco de dados (MySQL) chamado estacionamento com uma tabela chamada carro e as seguintes classes: CarroBean, CarroDAO e Conexao. Com isso, só precisamos agora criar a interface e empacotar a nossa aplicação. Para realizar este trabalho, este tutorial está dividido em duas etapas, a primeira é a criação da interface para comunicação com o usuário e segunda, a criação do arquivo .jar. Cada etapa foi dividida em passos, siga-os.

1 – Crie um formulário dentro do pacote criado na parte I deste tutorial. Para isso, click com o botão direito sobre o pacote e escolha novo->formulário JFrame, em seguida preencha o nome do formulário conforme figura abaixo e click em finalizar.



Feito isso, você verá o ambiente sofreu mudanças, novas componentes aparecem agora, conforme pode ser observado na figura abaixo. Como se pode ver, temos um formulário pronto para ser modelado de acordo com o nosso desejo e necessidade.

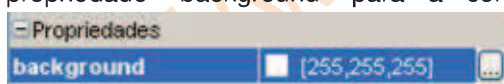
Nesse formulário você pode colocar qualquer um dos componentes (botão, rótulo, painel,..) da paleta que se encontra à direita acima. Para fazer isso, basta clicar, arrastar e soltar o componente dentro do formulário. Você também tem as propriedades dos componentes, que podem ser alterados, em foco (selecionado), à direita abaixo da paleta de componentes. Além disso, você pode alternar entre o modo desenho e o modo código-fonte, bastando, para isso, selecionar uma das opções. Click em “código-fonte”, você verá que já existe algum código gerado automaticamente.



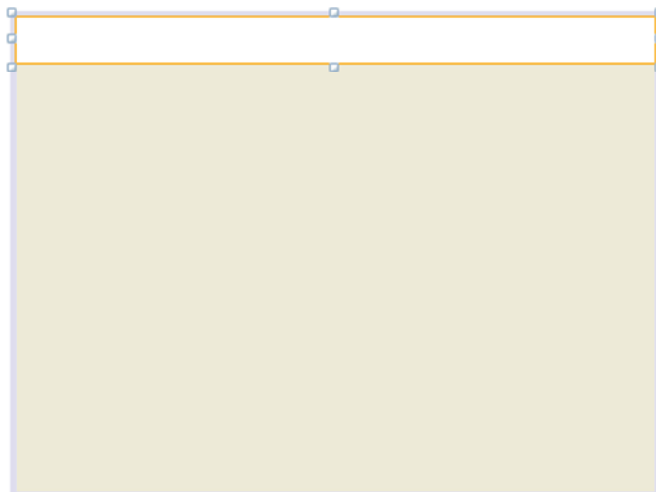
2 – Já que temos o nosso formulário, vamos colocar os componentes necessários para manutenção do nosso banco de dados, ou seja, colocaremos os componentes de entrada de dados, botões, e uma tabela para exibir dados de pesquisa.

Então, vamos começar a desenhar o nossa tela.

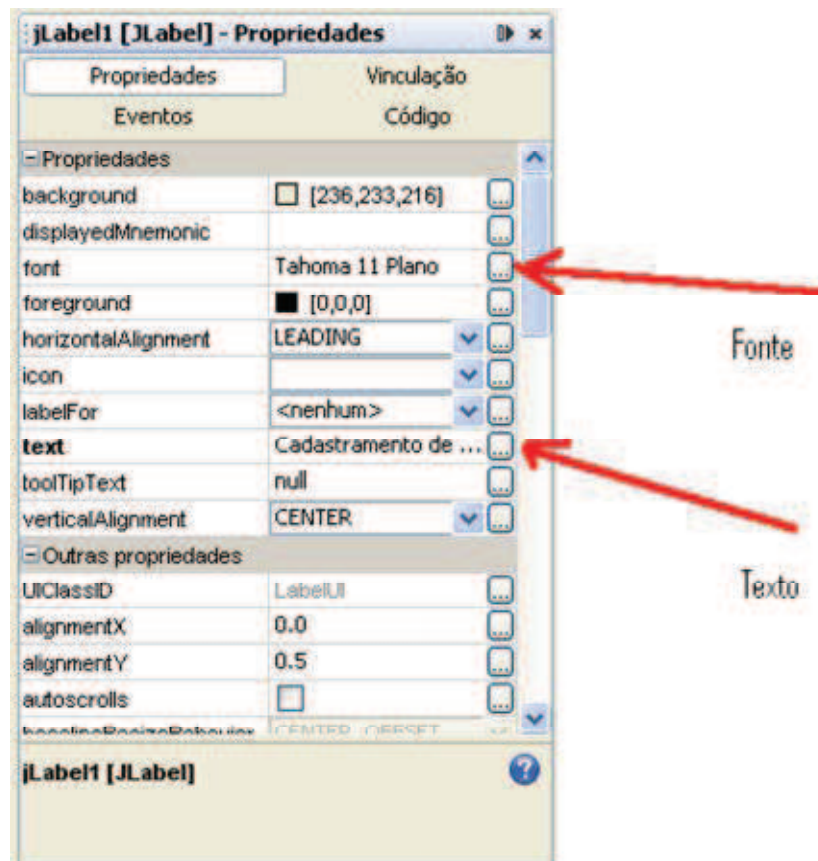
- Procure o componente **Panel** na paleta de componente, click nele e o arraste para o formulário, redimensione-o de maneira que como uma faixa superior no formulário. Mude a propriedade "background" para a cor branca. Para isso, localize a propriedade



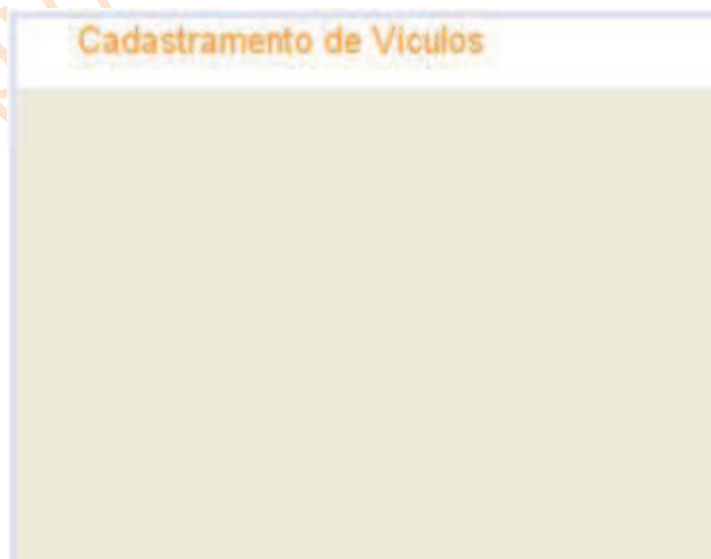
e click no botão ao lado. O seu formulário deve se parecer com a figura abaixo.



- b) Coloque um componente **Rótulo (Label)** dentro do componente **Panel** que você acaba de colocar no formulário e mude as propriedades text (escreva **Cadastramento de veículos**) e font (a fonte e o tamanho desejado e click em OK). Você pode acessar as propriedades dos componentes clicando com o botão direito sobre eles e selecionando propriedades, uma janela como a figura abaixo aparecerá. Para mudar a cor da fonte, acesse a propriedade "foreground".



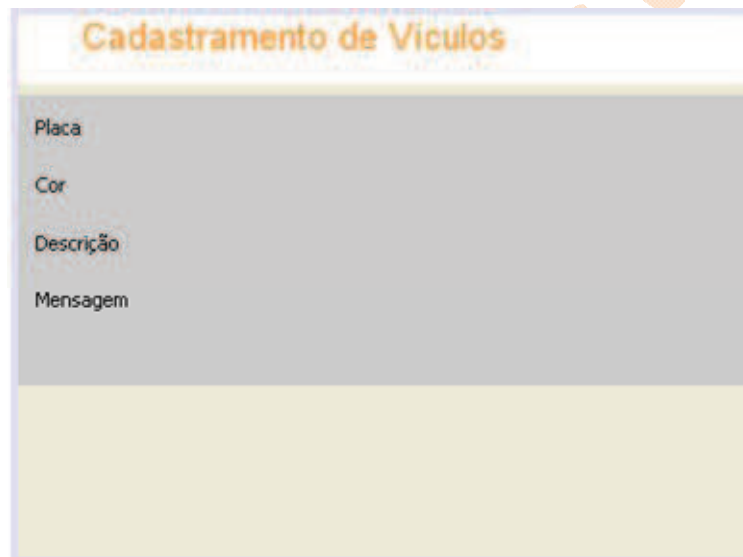
Até aqui, a sua tela deverá estar como a da figura abaixo.



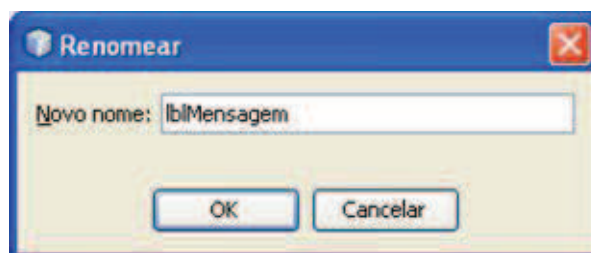
- c) Coloque outro Panel, logo abaixo do anterior e mude a propriedade “background” para uma cor diferente da que foi colocada no primeiro, como mostra a figura abaixo.




- d) Coloque quatro rótulos no segundo Panel e mude a propriedade text deles para ficar como mostra a figura abaixo.



Para o quarto rótulo, precisamos mudar seu nome interno, ou seja, o nome ao qual iremos nos referir a ele. É neste rótulo que colocaremos as mensagens devolvidas pela classe CarroDAO, por isso precisamos dar um nome que nos seja familiar (em vez JLabel5). Para isso, click com o botão direito do mouse sobre o rótulo “Mensagem” e click na opção “Alterar nome da variável”; coloque o nome **lblMensagem**, conforme figura abaixo, e click em OK. Procure colocar os nomes dos componentes com 3 letras iniciais que lembrem do que se trata.



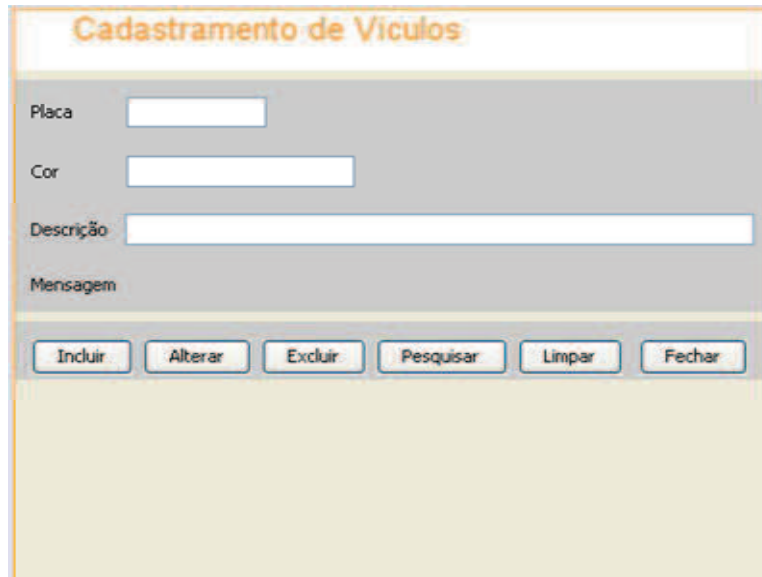
- e) Coloque 3 componentes “Caixa de texto” e mude as propriedades text de todos para vazio, ou seja, limpe o texto que estiver lá (conforme figura abaixo). Altere o nome da variável (da mesma forma como foi feito com o rótulo “Mensagem”) de cada um da seguinte forma: o primeiro deverá se chamar txtPlaca, o segundo, txtCor e o terceiro, txtDescricao.



Altere para o código-fonte (click em “código-fonte”) e você verá que os seus componentes já estão com os nomes que você colocou, como pode visto na figura abaixo. Não é possível alterar qualquer valor na parte cinza do código.

```
18 public class FrmCarro extends javax.swing.JFrame {
19
20     /** Creates new form FrmCarro */
21     public FrmCarro() {
22         initComponents();
23     }
24
25     /** This method is called from within the constructor to
26      * initialize the form.
27      * WARNING: Do NOT modify this code. The content of this method is
28      * always regenerated by the Form Editor.
29      */
30     @SuppressWarnings("unchecked")
31     Generated Code
32
33
34
35
36
37
38
39
40
41     /**
42      * @param args the command line arguments
43      */
44     public static void main(String args[]) {
45         java.awt.EventQueue.invokeLater(new Runnable() {
46             public void run() {
47                 new FrmCarro().setVisible(true);
48             }
49         });
50     }
51
52     // Variables declaration - do not modify
53     private javax.swing.JLabel jLabel1;
54     private javax.swing.JLabel jLabel2;
55     private javax.swing.JLabel jLabel3;
56     private javax.swing.JLabel jLabel4;
57     private javax.swing.JPanel jPanel1;
58     private javax.swing.JPanel jPanel2;
59     private javax.swing.JLabel lblMensagem;
60     private javax.swing.JTextField txtCor;
61     private javax.swing.JTextField txtDescricao;
62     private javax.swing.JTextField txtPlaca;
63     // End of variables declaration
64
65 }
```

- f) Coloque um novo **Panel** abaixo do segundo, mude seu “background” para a mesma cor do segundo e coloque 6 botões, mude a propriedade text deles, conforme mostra a figura abaixo. Mude também o nome interno (botão direito sobre ele, escolher “Mudar nome da variável”, colocar o nome e clicar em OK) deles para btnIncluir, btnAlterar, btnExcluir, btnPesquisar, btnLimpar e btnFechar, conforme a ordem que aparece na figura.



Cadastramento de Veículos

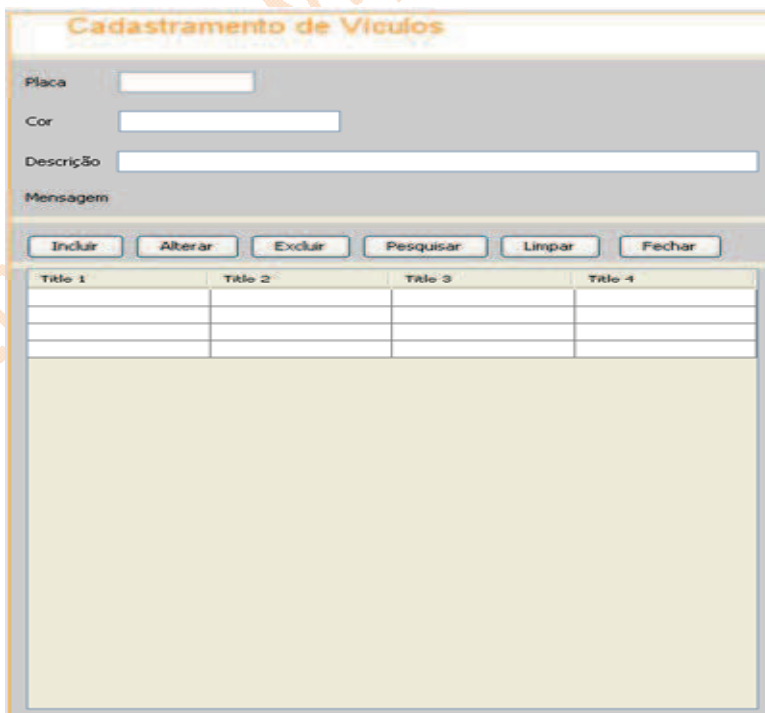
Placa

Cor

Descrição

Mensagem

- g) Vamos colocar agora uma tabela para exibir os dados quando clicarmos em “Pesquisar”. Para isso, coloque um componente **Panel**, mude a cor do seu background para a mesma do último colocado no formulário, coloque um componente **Tabela** nesse **Panel**, conforme mostra figura abaixo.



Cadastramento de Veículos

Placa

Cor

Descrição

Mensagem

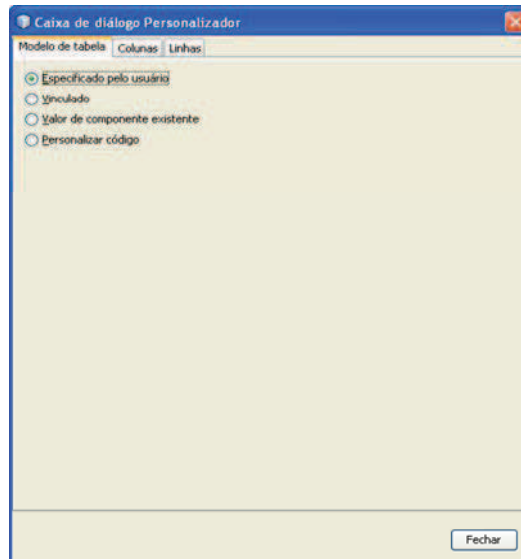
Title 1	Title 2	Title 3	Title 4

Como se pode observar, a tabela já vem pré-definida com 4 linhas e 4 colunas, mas isso para nós não serve porque precisamos de 3 colunas e o número de linha não sabemos, pois

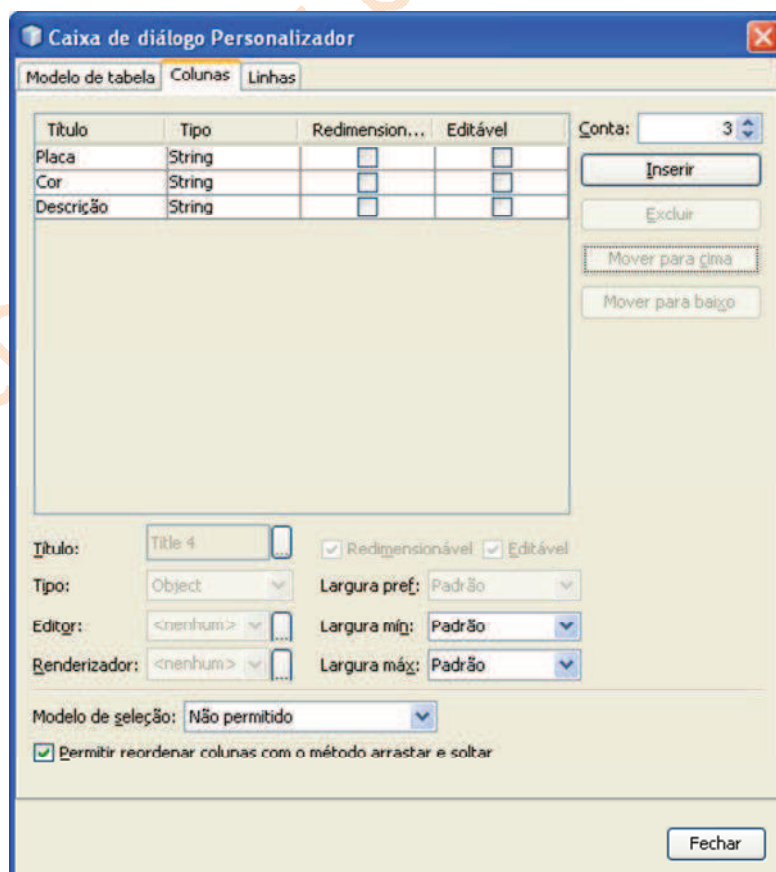


dependerá da quantidade de linhas que a consulta retornará. Por isso, vamos configurar a tabela para ficar do jeito que queremos, ou seja, com 3 colunas, nenhuma linha (acrescentaremos as linhas dinamicamente). Para isso siga o procedimento abaixo.

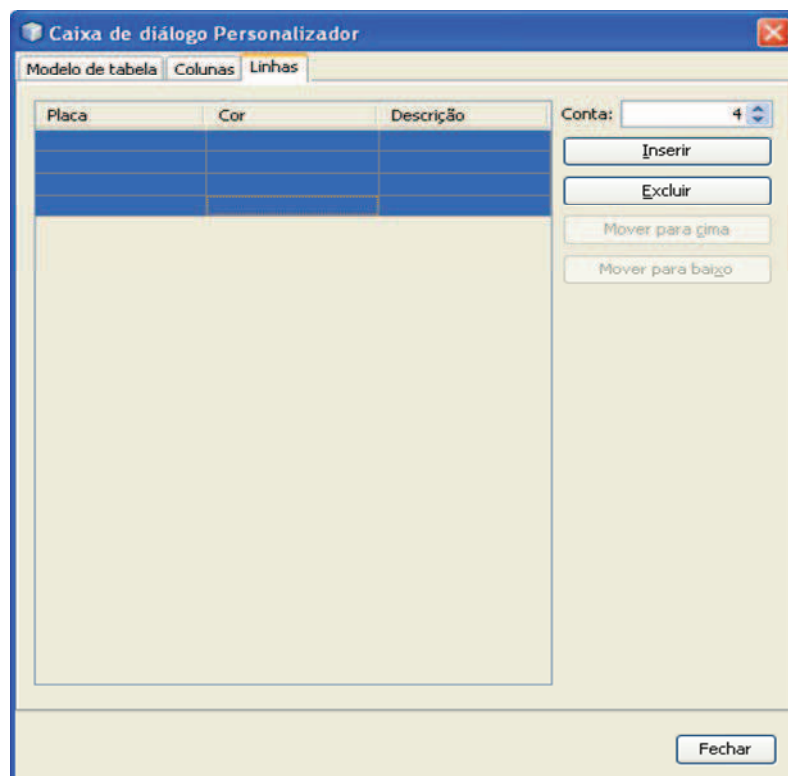
g1) click com o botão direito do mouse sobre a tabela e selecione “Conteúdo da tabela”, com isso, a caixa de diálogo da figura abaixo irá aparecer.



g2) Deixe marcada a opção “Especificado pelo usuário” e selecione a aba “Colunas”. Selecione a última linha e click em excluir, mude os títulos das três linhas restantes, mude o tipo e desmarque os checkbox (conforme figura abaixo) porque não queremos que ela seja editável e nem redimensionada pelo usuário, pois nosso objetivo é usá-la apenas para exibir dados.



g3) Selecione a aba “Linhas”, selecione todas as linhas, conforme figura abaixo, e click em excluir, com isso, temos uma tabela com três colunas e zero linha.



g4) Agora só precisamos mudar o nome da tabela, botão direito -> Alterar nome da variável, coloque tblConsulta. A sua tela deverá estar parecida com a figura abaixo. **Para que ela não seja redimensionável, e estrague seu desenho, desmarque a propriedade “resizeble”.**



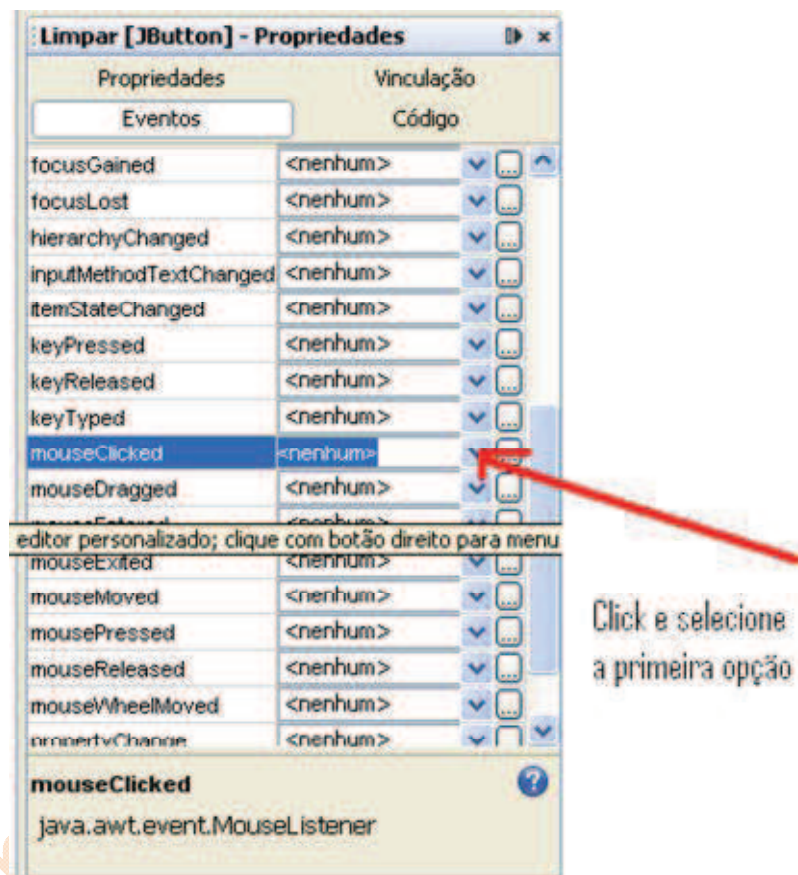


Agora que temos no tela pronta, podemos dar funcionalidade aos botões.

### Botão fechar

Vamos começar com o botão fechar, porém, as explicações iniciais servem para todos os outros. Para colocar funcionalidade em um componente “botão” temos várias formas, mas por questão de simplicidade colocaremos a funcionalidade quando o usuário clicar com o botão esquerdo do mouse sobre ele. Para fazer isso, siga o procedimento abaixo.

Selecione a paleta evento (com o foco no botão “Fechar”, ou seja, dê um click simples – não dê duplo click), abaixo de propriedade, busque pelo evento mouseClicked, selecione a primeira opção da lista e dê um click. Com isso, você será direcionado para dentro do evento poderá colocar seu código.



O evento gerado é apresentado abaixo, o código em vermelho deverá ser colocado por você. Observe que o código que precisamos acrescentar para fechar a janela se resume em apenas uma linha.

```
private void btnFecharMouseClicked(java.awt.event.MouseEvent evt) {  
  
    System.exit(0);  
  
}
```

## Botão Limpar

Crie um evento para o click do mouse, como foi feito para o botão anterior.

Na seção de importação, coloque o seguinte import : `import javax.swing.table.*;`

Vamos usar a classe `DefaultTableModel` desse pacote para redimensionarmos a nossa tabela. Como iremos inserir linhas de forma dinâmica na tabela, quando clicarmos no botão limpar, iremos eliminar todas as linhas da tabela.

O código em vermelho deverá ser acrescentado.

As quatro primeiras linhas estão atribuindo valor vazio para as caixas de texto, bem assim o rótulo de mensagem.

A linha seguinte atribui o modelo do nosso objeto **JTable** para um objeto do tipo `DefaultTableModel` para que possamos utilizar os métodos de remoção de linhas e outros necessários manipulação da tabela. Cabe observar que `tbm` é um objeto da classe `DefaultTableModel` por isso para que ele tenha acesso ao método `getModel` do nosso objeto `JTable`, foi necessário criar um *cast* (`(DefaultTableModel)tblConsulta.getModel()`).

O comando `for` percorre a tabela desde a última linha (`tbm.getRowCount()-1`) até a primeira (zero) e remove todas elas. Cabe dizer que essa não é a única maneira de se fazer, é uma das maneiras de se fazer.

```
private void LimparMouseClicked(java.awt.event.MouseEvent evt) {  
  
    txtPlaca.setText("");  
  
    txtCor.setText("");  
  
    txtDescricao.setText("");  
  
    lblMensagem.setText("");  
  
  
    DefaultTableModel tbm =  
    (DefaultTableModel)tblConsulta.getModel();  
  
    for(int i = tbm.getRowCount()-1; i >= 0; i--){  
  
        tbm.removeRow(i);  
  
    }  
  
}
```

## Botão Inserir

Para que nosso código colocado no método do botão inserir funcione, é necessário fazer a importação do pacote `sql` do `java`. Para isso, acrescente o código `import java.sql.*;` na seção de importação (logo abaixo de package).

Vemos, a seguir, o método de inserção onde muitos dos códigos você já está familiarizado com eles. Dessa forma, só serão explicadas as novidades. O código em vermelho deverá ser acrescentado.

Na primeira linha estamos abrindo uma conexão.

Na segunda estamos criando um objeto da classe CarroBean.

Na terceira estamos criando um objeto da classe CarroDAO.

Nas três linhas seguintes estamos atribuindo os valores aos atributos da classe CarroBean. Para pegar o valor que se encontram nos componentes, usamos o método `getText` e para atribuir valor para esses, usamos o método `setText`. Esses componentes só trabalham com objetos String, por isso quando tivermos campos do Integer, precisaremos converter os valores – de String para Integer, usa-se `Integer.parseInt(String a converter)` e de Integer para String, usa-se `valorInteger.toString()`.

Na penúltima linha é que estamos fazendo a inclusão e exibindo a mensagem de retorno, na `lblMensagem`, ao mesmo tempo.

Na última linha estamos fechando a conexão.

```
private void btnIncluirMouseClicked(java.awt.event.MouseEvent evt) {  
  
    Connection con = Conexao.abrirConexao();  
  
    CarroBean cb = new CarroBean();  
  
    CarroDAO cd = new CarroDAO(con);  
  
  
    cb.setPlaca(txtPlaca.getText());  
  
    cb.setCor(txtCor.getText());  
  
    cb.setDescricao(txtDescricao.getText());  
  
  
    lblMensagem.setText(cd.inserir(cb));  
  
    Conexao.fecharConexao(con);  
  
}
```

### Botão pesquisar

Para o botão pesquisar, precisamos importar o pacote útil do java. Acrescente o código `import java.util.*;` na seção de importação (logo abaixo de `package`). O código em vermelho deverá ser acrescentado.

Na primeira linha estamos abrindo uma conexão.

Na segunda estamos criando um objeto da classe CarroDAO.

Na terceira linha criamos uma lista de carros para receber o retorno do método listarTodos.

Na quarta linha estamos executando o método listarTodos e atribuindo o seu retorno à lista de carros.

Na quinta linha criamos um objeto do tipo DefaultTableModel.

O primeiro for serve para zerar as linhas da tabela para acrescentarmos o resultado da nossa pesquisa.

O segundo for é para pegarmos os objetos da lista de carros para colocarmos na tabela.

A primeira linha dentro do segundo for adiciona uma linha à tabela e os comandos seguintes adicionam os dados em cada célula da tabela, de acordo com o campo da tabela física.

Observe que o i corresponde ao número de linhas que retornaram da pesquisa e 0,1 e 2 são as colunas da tabela.

```
private void btnPesquisarMouseClicked(java.awt.event.MouseEvent evt) {  
  
    Connection con = Conexao.abrirConexao();  
  
    CarroDAO cd = new CarroDAO(con);  
  
    List<CarroBean> listaCarro = new ArrayList<CarroBean>();  
  
    listaCarro = cd.listarTodos();  
  
    DefaultTableModel tbm =  
    (DefaultTableModel)tblConsulta.getModel();  
  
    for(int i = tbm.getRowCount()-1; i >= 0; i--){  
        tbm.removeRow(i);  
    }  
  
    int i = 0;  
  
    for(CarroBean cb : listaCarro){  
        tbm.addRow(new String[1]);  
  
        tblConsulta.setValueAt(cb.getPlaca(), i, 0);  
  
        tblConsulta.setValueAt(cb.getCor(), i, 1);  
  
        tblConsulta.setValueAt(cb.getDescricao(), i, 2);  
  
        i++;  
    }  
  
    Conexao.fecharConexao(con);  
  
}
```

A figura a seguir mostra o resultado da pesquisa.

Placa	Cor	Descrição
JKE2013	Preto	Carro 2
JKK1900	Azul	Carro 1
JUM1890	Branco	Carro sujo

Feita a pesquisa queremos que o usuário possa escolher o registro que deseja alterar ou excluir. Para isso precisamos capturar a linha que ele selecionou. Para isso, vamos criar um método para o click do mouse na tabela, o procedimento é o mesmo feito para os botões. O código em vermelho deverá ser acrescentado.

Na primeira linha estamos capturando a linha que o usuário clicou.

Na segunda, terceira e quarta linha estamos pegando o valor de cada célula da tabela, que corresponde a cada atributo da classe carro. O cast é necessário porque o objeto `JTable` retorna um objeto e nós precisamos de uma `String` para passar para os campos de texto.

Na quarta, quinta e sexta linha estamos atribuindo o valor para os campos de texto do formulário.

O resultado é mostrado na figura a seguir. Como pode ser observado, os dados da linha selecionada pelo usuário são transferidos para os campos do formulário.

```
private void tblConsultaMouseClicked(java.awt.event.MouseEvent evt) {  
  
    Integer linha = tblConsulta.getSelectedRow();  
  
    String placa = (String)tblConsulta.getValueAt(linha, 0);  
  
    String cor = (String)tblConsulta.getValueAt(linha, 1);  
  
    String descricao = (String)tblConsulta.getValueAt(linha, 2);  
}
```

```

txtPlaca.setText(placa);

txtCor.setText(cor);

txtDescricao.setText(descricao);
}

```

Placa	Cor	Descrição
JKE2013	Preto	Carro 2
JKK1900	Azul	Carro 1
JUM1890	Branco	Carro sujo

### Botão Alterar

O botão alterar é semelhante ao botão incluir, por isso cabe as mesmas explicações. O código em vermelho deverá ser acrescentado.

```

private void btnAlterarMouseClicked(java.awt.event.MouseEvent evt) {

    Connection con = Conexao.abrirConexao();

    CarroBean cb = new CarroBean();

    CarroDAO cd = new CarroDAO(con);

    cb.setPlaca(txtPlaca.getText());

    cb.setCor(txtCor.getText());
}

```



```

        cb.setDescricao(txtDescricao.getText());

        lblMensagem.setText(cd.alterar(cb));

        Conexao.fecharConexao(con);
    }

```

## Botão excluir

O botão excluir é semelhante ao botão incluir, por isso cabe as mesmas explicações, exceto no uso do JOptionPane que será explicado a seguir. O código em vermelho deverá ser acrescentado.

Precisamos acrescentar a importação do JOptionPane na seção de import.

```
import javax.swing.JOptionPane;
```

A linha `Object[] opcoes = { "Sim", "Não" };` cria um vetor de objetos para trocarmos o yes e no pelo sim e não. O objetivo é perguntar ao usuário se ele deseja realmente excluir o registro.

O método `showOptionDialog` do JOptionPane é sobrecarregado de forma que possua várias implementações e uma delas é a que estamos usando neste exemplo, que contém 7 argumentos.

O primeiro é nome da classe pai, ou seja, para qual janela eu pretendo associar o a caixa de diálogo, se deixarmos nulo, ele exibe a caixa no meio da tela. Os demais atributos podem ser visto quando ele estiver funcionando.

```

private void btnExcluirMouseClicked(java.awt.event.MouseEvent evt) {

    Connection con = Conexao.abrirConexao();

    CarroBean cb = new CarroBean();

    CarroDAO cd = new CarroDAO(con);

    cb.setPlaca(txtPlaca.getText());

    Object[] opcoes = { "Sim", "Não" };

    int i = JOptionPane.showOptionDialog(null,

        "Deseja excluir esse veículo: "
        +txtPlaca.getText()+"?", "Exclusão",

        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE, null,

        opcoes, opcoes[0]);

    if (i == JOptionPane.YES_OPTION) {

        lblMensagem.setText(cd.excluir(cb));

    }
}

```

```
Conexao . fecharConexao (con) ;
```

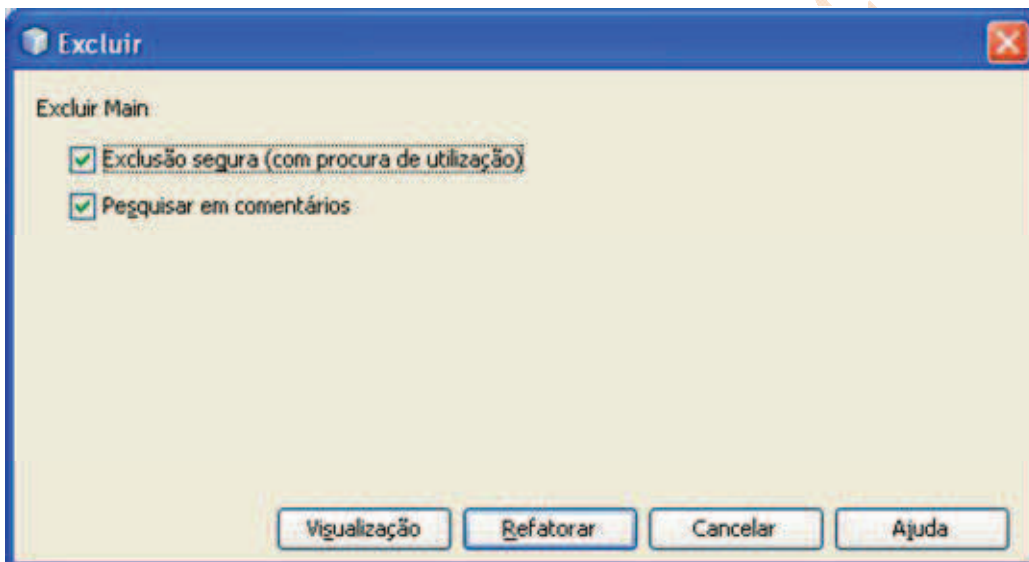
```
}
```

O código completo é apresentado no fim do tutorial.

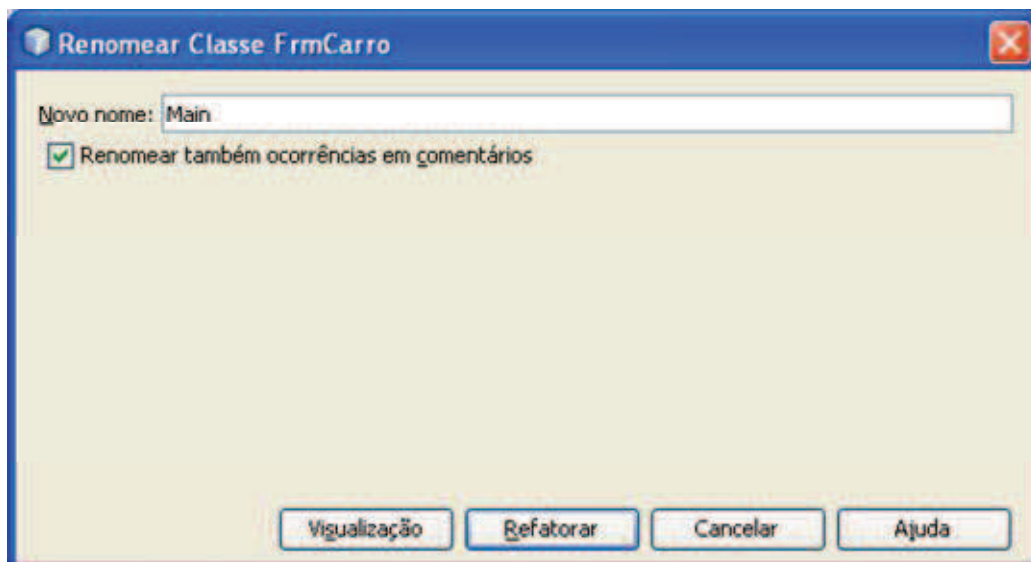
### Criando um pacote de distribuição

Até agora nós executamos nosso aplicativo dentro do ambiente do netbeans, mas se quisermos distribuir esse aplicativo, como faremos? Bom, para isso, podemos criar um pacote executável (.jar) e executá-lo em qualquer máquina que tenha a JVM. Mas, lembre-se de que é necessário levar o banco de dados também. Para criar um pacote de distribuição siga os passos apresentados abaixo.

1 - Exclua a classe Main.java (botão direito sobre selecione a opção “Excluir”). Ao aparecer a janela abaixo, click em exclusão segura e click em Refatorar.



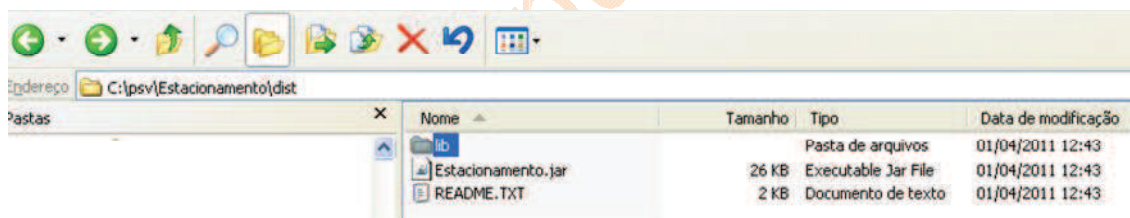
2 - Troque o nome da classe FrmCarro para Main (botão direito sobre ela escolha refatorar->renomear). Preencha os campos conforme figura abaixo e click em refatorar.



3 - Teste a aplicação (F6 – executar projeto principal). Se estiver funcionando perfeitamente, siga em frente.

4 – Selecione o projeto (Estacionamento) e no menu principal escolha Executar->Limpar e Construir Projeto Principal ou Shift+F11. Espere o processo ser construído.

5 – Abra a pasta em que você criou o seu projeto e procure o diretório dist. Você deverá ter as pastas mostradas na figura abaixo. Dentro da pasta lib está o conector do MySQL.



6 – Dê duplo click no arquivo Estacionamento.jar e você verá que o seu aplicativo funciona como funcionava no ambiente do netbeans.

## Considerações finais

Assim nós encerramos a segunda parte do tutorial. Creio que isto serve como ponta-pé inicial para desenvolvimento de aplicação java para desktop. Na parte III do tutorial iniciaremos o desenvolvimento de aplicações para web.

## Código Completo

O código completo é colocado a seguir.

```
package psv;
```