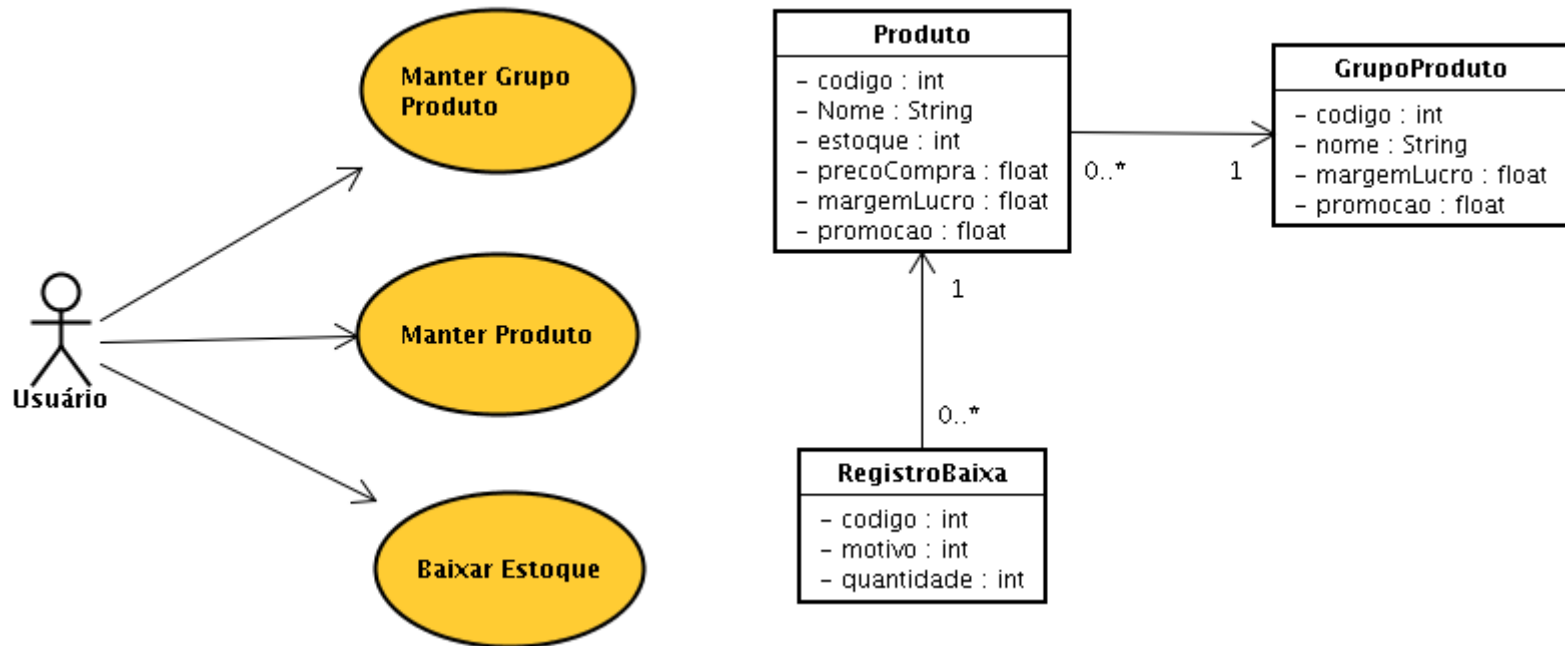


# **Mapeamento Objeto/Relacional JDBC**

**Evandro César Freiburger**

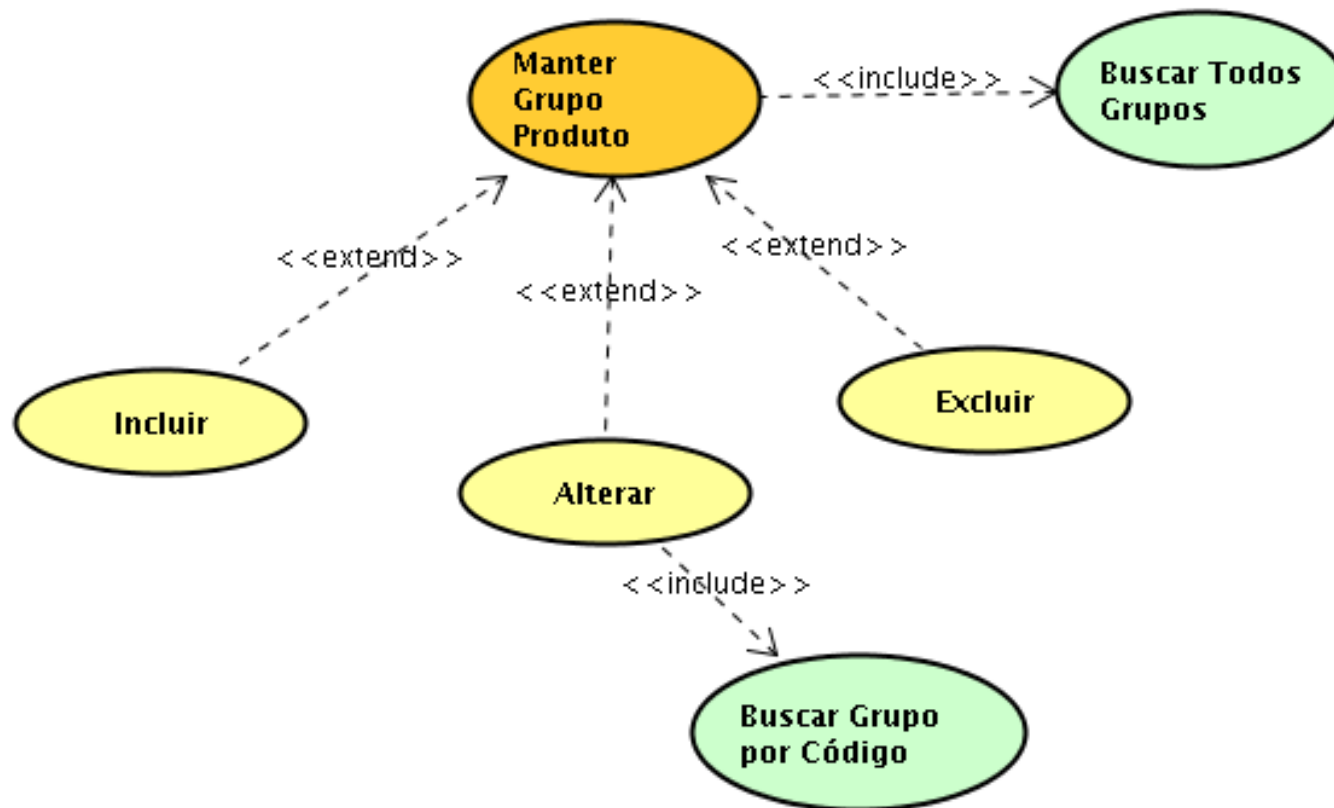
**`evandrofreiberger@gmail.com`**

# Estudo de Caso - Exemplo

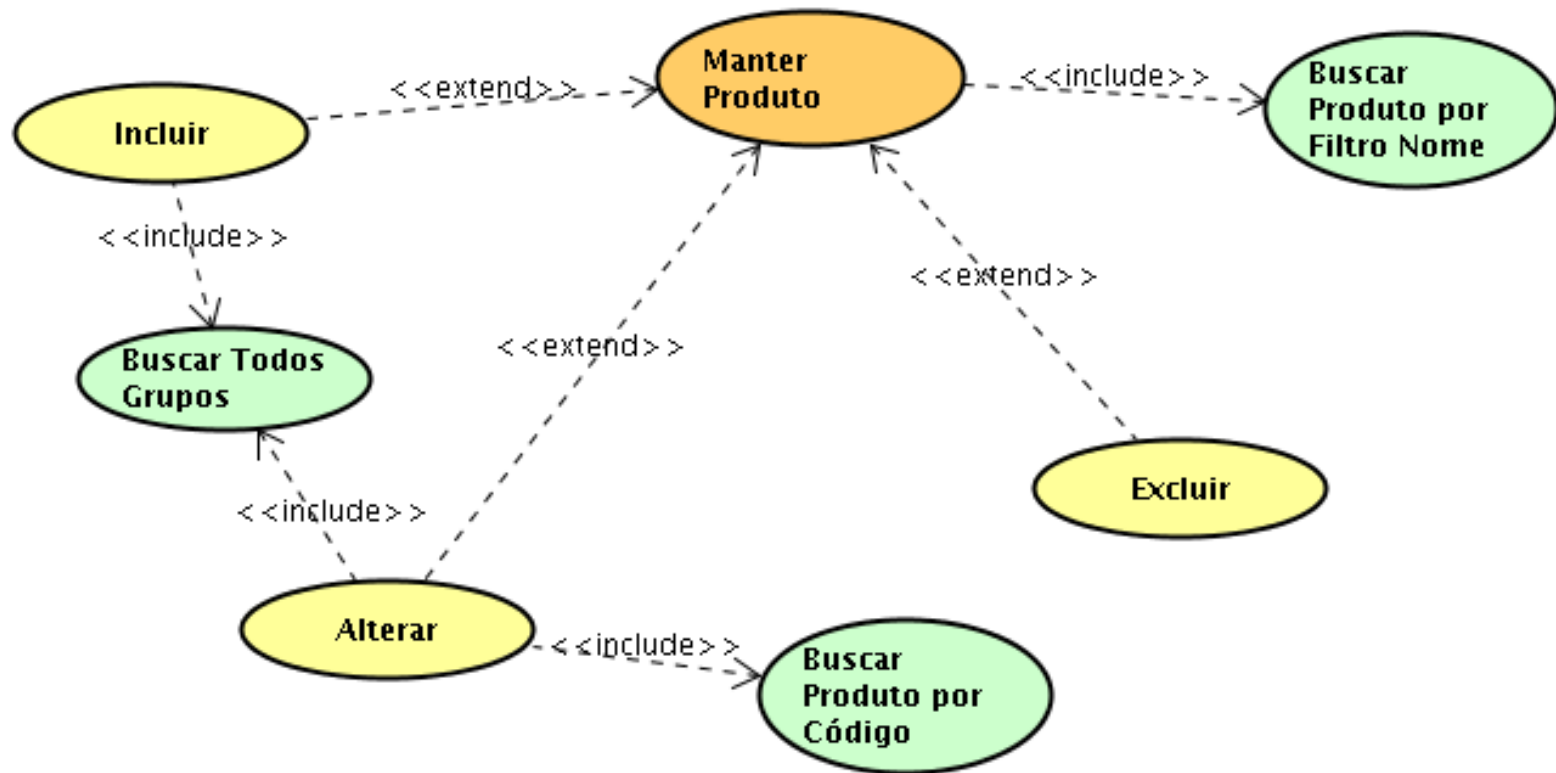


Modelo Conceitual deve ser mapeado para um modelo arquitetural

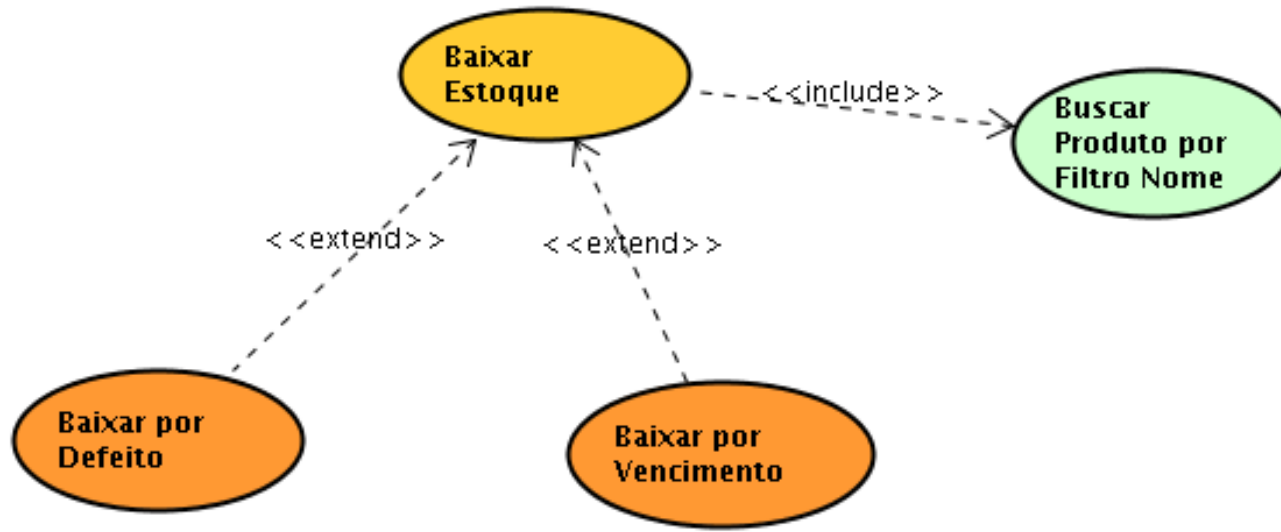
# Estudo de Caso - Exemplo



# Estudo de Caso - Exemplo



# Estudo de Caso - Exemplo



# Estudo de Caso

**Manutenção de Produto e Estoque**

Tabelas   Operações

**Manutenção de Grupo de Produto**

Codigo	Nome	Margem	Promocao
3	Bebidas	10.0	0.0
9	Bebidas destiladas	0.0	0.0
1	Limpeza	0.0	0.0
2	Verduras	20.0	5.0

Incluir   Alterar   Excluir   Sair

**Inclusão de Grupos de Produtos**

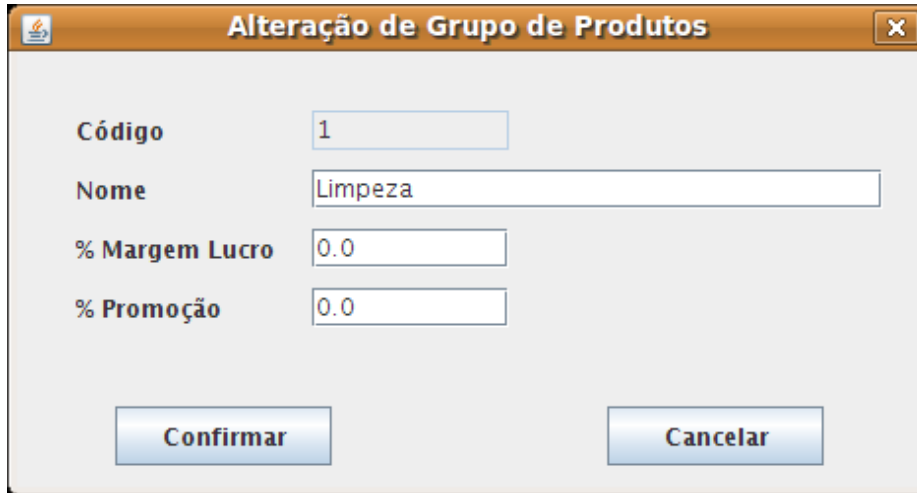
Nome

% Promoção

% Margem Lucro

Confirmar   Cancelar

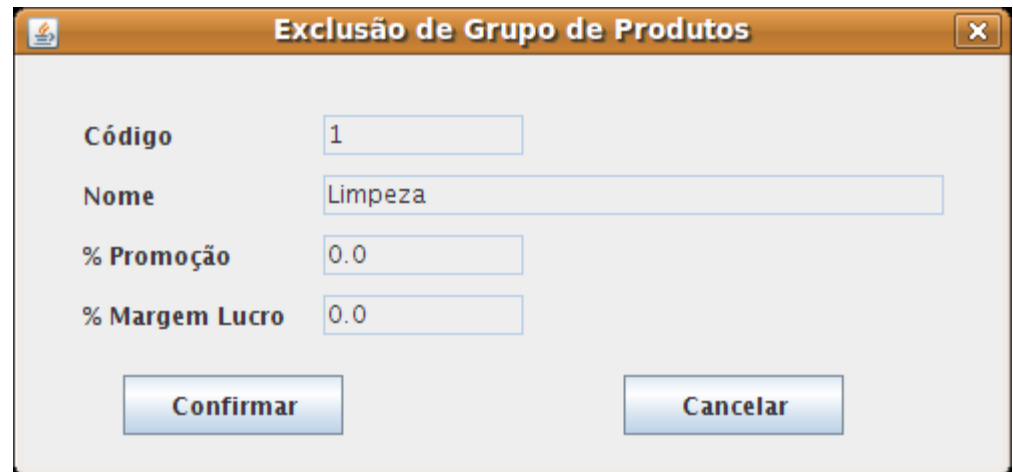
# Estudo de Caso



A dialog box titled "Alteração de Grupo de Produtos" with a close button (X) in the top right corner. It contains four input fields: "Código" with the value "1", "Nome" with the value "Limpeza", "% Margem Lucro" with the value "0.0", and "% Promoção" with the value "0.0". At the bottom, there are two buttons: "Confirmar" and "Cancelar".

Código	1
Nome	Limpeza
% Margem Lucro	0.0
% Promoção	0.0

Confirmar Cancelar



A dialog box titled "Exclusão de Grupo de Produtos" with a close button (X) in the top right corner. It contains four input fields: "Código" with the value "1", "Nome" with the value "Limpeza", "% Promoção" with the value "0.0", and "% Margem Lucro" with the value "0.0". At the bottom, there are two buttons: "Confirmar" and "Cancelar".

Código	1
Nome	Limpeza
% Promoção	0.0
% Margem Lucro	0.0

Confirmar Cancelar

# Estudo de Caso

Manutenção de Produto

Digite parte do nome do produto

Filtrar

Codigo	Nome	Estoque	Compra	Margem	Promoção	Grupo	Venda
3	Detergente neutro22	253	1.5	60.0	20.0	Limpeza	1.9200001
5	Refrigerante 1,6 Lts	93	2.0	10.0	2.0	Bebidas	2.1560001
1	Refrigerante 2l - Coca Cola	92	2.5	20.0	0.0	Bebidas	3.0
7	Sabão em Póxxx	50	5.0	40.0	15.0	Limpeza	5.95
10	teste de inclusao	70					

Incluir

Alterar

Excluir

Alteração de Produtos

Código

7

Nome

Sabão em Póxxx

Valor Compra

5.0

% Margem Lucro

40.0

% Promocao

15.0

Grupo

Limpeza

Confirmar

Cancelar



# Estudo de Caso

The image shows a software interface with two windows. The main window, titled "Manutenção de Produto", contains a search bar with the placeholder text "Digite parte do nome do produto" and a "Filtrar" button. Below this is a table with three columns: "Codigo", "Nome", and "Estoque". The table lists five items: "Detergente neutro22" (stock 253), "Refrigerante 1,6 Lts" (stock 93), "Refrigerante 2l - Coca Cola" (stock 92), "Sabão em Póxxx" (stock 50), and "teste de inclusao" (stock 70). At the bottom of the main window are two buttons: "Baixar (Vencimento)" and "Baixar (Defeito)". A secondary dialog box, titled "Baixa de Estoque por Vencimento Validade", is open in the foreground. It contains four input fields: "Código" (value: 1), "Nome" (value: Refrigerante 2l - Coca Cola), "Quantidade" (empty), and "Data" (value: 01/10/2009). At the bottom of the dialog are "Confirmar" and "Cancelar" buttons.

Codigo	Nome	Estoque
3	Detergente neutro22	253
5	Refrigerante 1,6 Lts	93
1	Refrigerante 2l - Coca Cola	92
7	Sabão em Póxxx	50
10	teste de inclusao	70

**Baixa de Estoque por Vencimento Validade**

Código: 1  
Nome: Refrigerante 2l - Coca Cola  
Quantidade:   
Data: 01/10/2009

Confirmar Cancelar

# Projeto01

- ✓ Manipulação do banco por meio de JDBC
- ✓ Não há separação de camadas
- ✓ Não há mapeamento Objeto-Relacional
- ✓ Redundância de código de manipulação de dados

# Projeto01 - Situações Exemplo



The screenshot shows a software window titled "Manutenção de Grupo de Produto". It contains a table with four columns: "Codigo", "Nome", "Margem", and "Promocao". The table lists four items: "Bebidas" (Codigo 3, Margem 10.0, Promocao 0.0), "Bebidas destiladas" (Codigo 9, Margem 0.0, Promocao 0.0), "Limpeza" (Codigo 1, Margem 0.0, Promocao 0.0), and "Verduras" (Codigo 2, Margem 20.0, Promocao 5.0). Below the table is a large empty rectangular area. At the bottom of the window are four buttons: "Incluir", "Alterar", "Excluir", and "Sair".

Codigo	Nome	Margem	Promocao
3	Bebidas	10.0	0.0
9	Bebidas destiladas	0.0	0.0
1	Limpeza	0.0	0.0
2	Verduras	20.0	5.0

Buttons: Incluir, Alterar, Excluir, Sair

# Projeto01 - Situações Exemplo

```
private void ativaConexaoBD(){  
  
    String url = "jdbc:postgresql://localhost:5432/estoque1";  
    try{  
        Class.forName("org.postgresql.Driver");  
        this.conexao = DriverManager.getConnection(url,"postgres","postgres");  
        this.conexao.setAutoCommit(false);  
    }  
    catch(ClassNotFoundException cnf){  
        JOptionPane.showMessageDialog(this,"Classe do driver do banco não encontrada - "+cnf.getMessage());  
    }  
    catch(SQLException sqle){  
        JOptionPane.showMessageDialog(this,"Banco não encontrado - "+sqle.getMessage());  
    }  
}
```

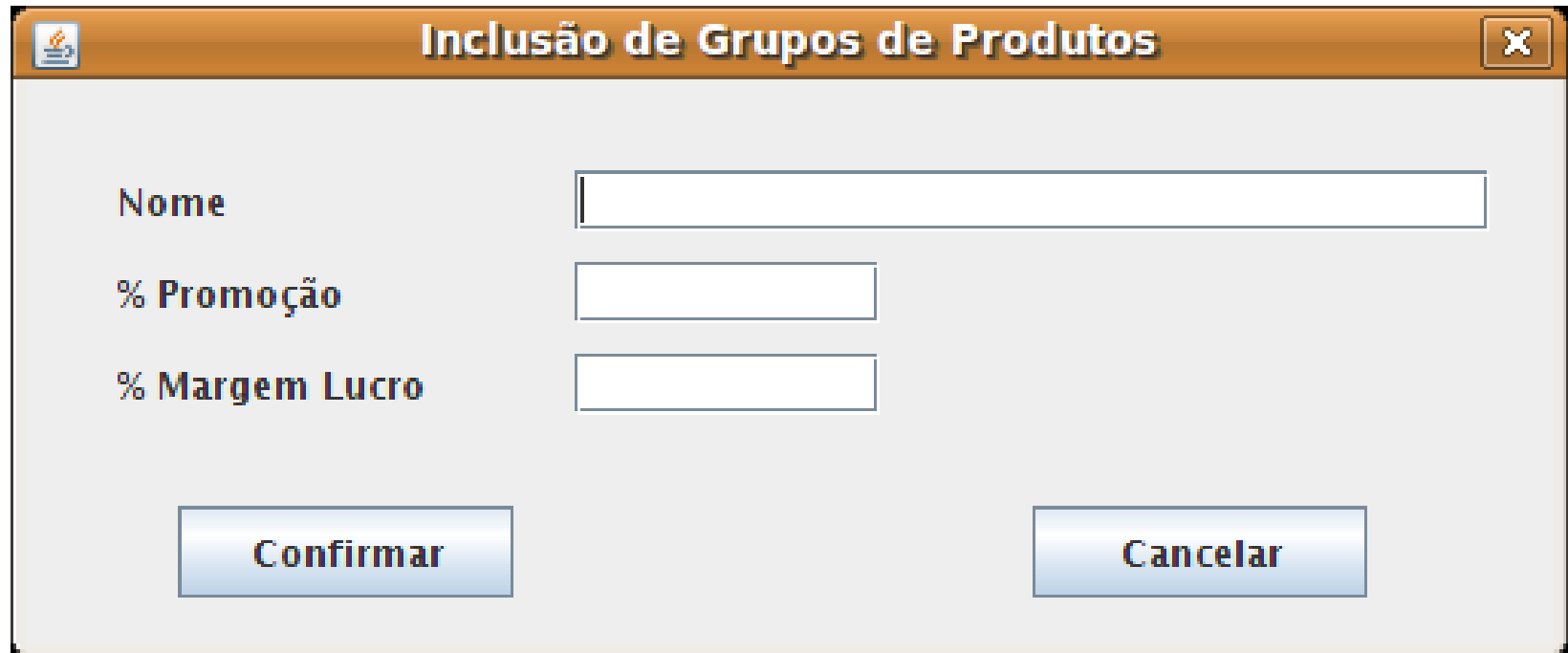
Método para conexão com o banco de dados – presente na classe que cria a interface com o usuário

# Projeto01 - Situações Exemplo

```
public void atualizaTabela() {  
    ArrayList listaLinha = new ArrayList();  
    try{  
        PreparedStatement comando = conexao.prepareStatement("SELECT * FROM GrupoProduto ORDER BY NOME");  
        ResultSet rs = comando.executeQuery();  
        ArrayList linha;  
        while (rs.next()){  
            linha = new ArrayList();  
            linha.add(rs.getInt("codigo"));  
            linha.add(rs.getString("nome").trim());  
            linha.add(rs.getFloat("margemLucro"));  
            linha.add(rs.getFloat("promocao"));  
            listaLinha.add(linha);  
        }  
        rs.close();  
        comando.close();  
    } catch (SQLException ex) {
```

Método para seleção de dados – presente na classe que cria a interface com o usuário

# Projeto01 - Situações Exemplo



The image shows a screenshot of a software window titled "Inclusão de Grupos de Produtos". The window has a standard title bar with a close button (X) in the top right corner. Inside the window, there are three input fields arranged vertically, each preceded by a label: "Nome", "% Promoção", and "% Margem Lucro". Below these fields are two buttons: "Confirmar" on the left and "Cancelar" on the right. The window has a light gray background and a thin border.

Nome	<input type="text"/>
% Promoção	<input type="text"/>
% Margem Lucro	<input type="text"/>
<input type="button" value="Confirmar"/>	
<input type="button" value="Cancelar"/>	

# Projeto01 - Situações Exemplo

```
private void confirmaOperacao() {  
    String nome = null;  
    float promocao = 0;  
    float margem = 0;  
    try {  
        nome = this.campoNome.getText();  
        promocao = Float.parseFloat(this.campoPromocao.getText());  
        margem = Float.parseFloat(this.campoMargem.getText());  
    } catch (Exception ex) {  
        JOptionPane.showMessageDialog(this, "Digitacao inconsistente");  
        return;  
    }  
    if (nome.length() == 0) {  
        JOptionPane.showMessageDialog(this, "Nome não pode ser vazio");  
        campoNome.requestFocus();  
        return;  
    }  
}
```

# Projeto01 - Situações Exemplo

```
if (promocao < 0 || promocao > 100) {
    JOptionPane.showMessageDialog(this, "Percentual de Promoção deve ser entre 0 e 100");
    campoPromocao.requestFocus();
    return;
}
if (margem < 0 ) {
    JOptionPane.showMessageDialog(this, "Percentual de Margem de Lucro deve ser maior que zero");
    campoMargem.requestFocus();
    return;
}
try {
    PreparedStatement comando = conexaoBD.prepareStatement(
        "INSERT INTO GRUPOPRODUTO ( NOME, PROMOCAO, MARGEMLUCRO )VALUES ( ? , ? , ? )");
    comando.setString(1, nome);
    comando.setFloat(2, promocao);
    comando.setFloat(3, margem);
    comando.executeUpdate();
    comando.close();
    conexaoBD.commit();
    JOptionPane.showMessageDialog(this, "Inclusão realizada com sucesso");
    this.limparCampos();
}
```



# Projeto01 - Situações Exemplo

**Alteração de Produtos**

<b>Código</b>	<input type="text" value="7"/>
<b>Nome</b>	<input type="text" value="Sabão em Póxxx"/>
<b>Valor Compra</b>	<input type="text" value="5.0"/>
<b>% Margem Lucro</b>	<input type="text" value="40.0"/>
<b>% Promocao</b>	<input type="text" value="15.0"/>
<b>Grupo</b>	<input type="text" value="Limpeza"/>

# Projeto01 - Situações Exemplo

```
private void preencheCampos(){
    try {

        int codigoGrupo=0;
        int posicaoSelecao=0;
        PreparedStatement comandoProduto = conexaoBD.prepareStatement(
            "SELECT * FROM PRODUTO WHERE CODIGO = ? ORDER BY NOME");
        comandoProduto.setInt(1,this.codigoProduto);
        ResultSet rsProduto = comandoProduto.executeQuery();

        if(rsProduto.next()){
            this.campoCodigo.setText(rsProduto.getString("codigo"));
            this.campoNome.setText(rsProduto.getString("nome").trim());
            this.campoCompra.setText(rsProduto.getString("valorCompra"));
            this.campoMargem.setText(rsProduto.getString("margemLucro"));
            this.campoPercentualPromocao.setText(rsProduto.getString("promocao"));
            codigoGrupo = rsProduto.getInt("grupo");
        }
    }
}
```

# Projeto01 - Situações Exemplo

```
PreparedStatement comandoGrupo = conexaoBD.prepareStatement(
    "SELECT * FROM GRUPOPRODUTO ORDER BY NOME");
ResultSet rs = comandoGrupo.executeQuery();
List listaGrupo = new ArrayList();
int codigo;
String nome;
while(rs.next()){
    codigo = rs.getInt("codigo");
    nome = rs.getString("nome");
    listaGrupo.add(codigo+"-"+nome);
    if(codigoGrupo == codigo){
        posicaoSelecao = listaGrupo.size()-1;
    }
}
this.comboGrupoProduto.setModel(new DefaultComboBoxModel(listaGrupo.toArray()));
this.comboGrupoProduto.setSelectedIndex(posicaoSelecao);
comandoGrupo.close();
```

# JDBC – Sem Mapeamento OR

- ✓ O script SQL é montado a partir de dados fragmentados, e não de objetos

```
public void inserir(String nome, float margem, float promocao) throws NegocioException {  
  
    String mensagemErros = this.validarDados(nome, margem, promocao);  
  
    if (!mensagemErros.isEmpty()) {  
        throw new NegocioException(mensagemErros);  
    }  
  
    try {  
        ConexaoBD conexaoBD = ConexaoBD.getInstancia();  
        PreparedStatement comando = conexaoBD.getConexao().prepareStatement(  
            "INSERT INTO GRUPOPRODUTO ( NOME, MARGEMPLUCRO, PROMOCAO ) VALUES (?, ?, ?)");  
        comando.setString(1, nome);  
        comando.setFloat(2, margem);  
        comando.setFloat(3, promocao);  
        comando.executeUpdate();  
        comando.close();  
    } catch (SQLException ex) {  
        throw new NegocioException("Erro ao incluir novo grupo de produto" + ex.toString());  
    } catch (PersistenciaException ex) {  
        throw new NegocioException("Erro ao conectar no banco de dados" + ex.toString());  
    }  
}
```

# JDBC – Sem Mapeamento OR

- ✓ Retorna um objeto ResultSet – abstração de uma tabela do modelo relacional

```
public ResultSet porCodigo(int codigo) throws NegocioException {  
    ResultSet rs = null;  
    try {  
        ConexaoBD conexaoBD = ConexaoBD.getInstancia();  
        PreparedStatement comandoGrupoProduto = conexaoBD.getConexao().prepareStatement(  
            "SELECT * FROM GRUPOPRODUTO WHERE CODIGO = ? ORDER BY NOME");  
        comandoGrupoProduto.setInt(1,codigo);  
        rs = comandoGrupoProduto.executeQuery();  
    } catch (SQLException ex) {  
        throw new NegocioException("Não foi possível recuperar o grupo de produto - " + ex.toString());  
    } catch (PersistenciaException ex) {  
        throw new NegocioException("Erro ao obter conexão com o banco de dados - " + ex.toString());  
    }  
    return rs;  
}
```

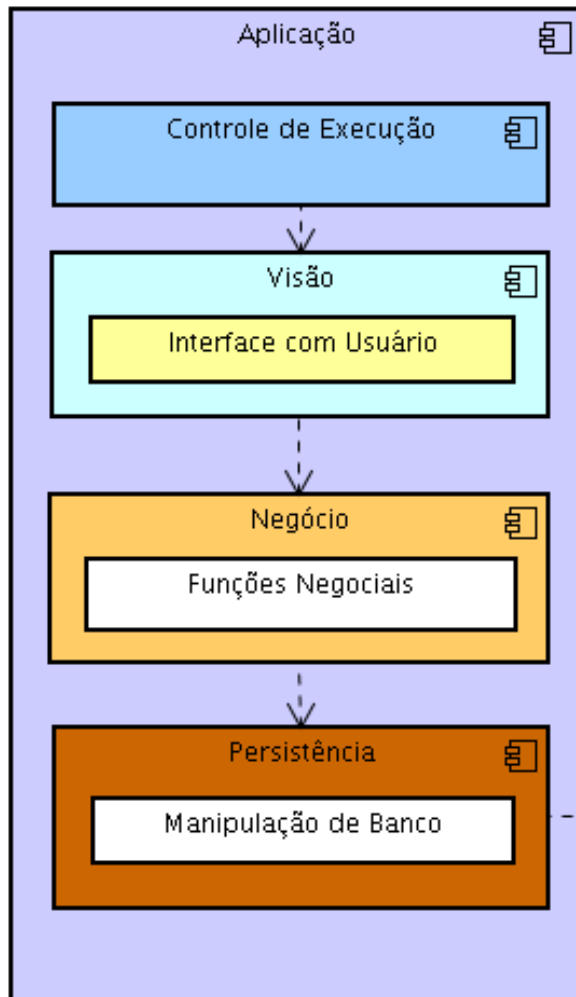
# Mapeamento Objeto-Relacional

- ✓ Os mecanismos de persistência pressupõem que haja separação entre interface do usuário, domínio e persistência
- ✓ As operações e consultas da interface do usuário são realizadas por meio da camada de domínio
- ✓ As camadas de interface e domínio não devem conter expressões SQL
- ✓ A camada de persistência deve receber e devolver apenas objetos do domínio

# Arquitetura em Camadas

- ✓ Arquitetura em Camadas
  - Divisão de Abstrações/Responsabilidades
  - Maior probabilidade de reuso
  - Facilidade de manutenção
  - Facilidade de divisão de trabalho
  - Possibilidade de aquisição/troca

# Arquitetura em Três Camadas



Uma classe de controle

Classes responsáveis pela interação com o usuário (visão)

Classes que representam regras de negócio

Classes de Persistência





# Uso de VO's para Comunicação

- ✓ O uso de parâmetros simples para comunicação entre camadas pode comprometer a legibilidade e a flexibilidade de mudanças
- ✓ Separação da Manipulação de Banco de Dados da camada de negócio
- ✓ Desacoplamento da camada de Negócio e Visão da tecnologia utilizada para acesso ao Banco
- ✓ Uso de Objeto de Transporte de Dados (OTD) através de objetos VO's (Value Object)

# Uso de VO's para Comunicação

## ✓ Parâmetros não encapsulados

```
public void inserir(String nome, int estoque, float compra, float margem, float promocao, int grupo)
{
    String mensagemErros = this.validarDados(nome, estoque, compra, margem, promocao, grupo);
    if (!mensagemErros.isEmpty()) {
        throw new NegocioException(mensagemErros);
    }
}
```

## ✓ Parâmetros encapsulados em VO's

```
public void inserir(ProdutoVO produtoVO) throws NegocioException {
    String mensagemErros = this.validarDados(produtoVO);
    if (!mensagemErros.isEmpty()) {
        throw new NegocioException(mensagemErros);
    }
}
```

# Arquitetura em Três Camadas - Projeto02

## ✓ GrupoProdutoVO – objeto de transporte de dados

```
package vo;
public class GrupoProdutoVO {

    private int codigo;
    private String nome;
    private float margemLucro;
    private float promocao;

    public int getCodigo() {
        return codigo;
    }
    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public float getMargemLucro() {
        return margemLucro;
    }
    public void setMargemLucro(float margemLucro) {
        this.margemLucro = margemLucro;
    }
}
```

```
    public float getPromocao() {
        return promocao;
    }
    public void setPromocao(float promocao) {
        this.promocao = promocao;
    }
    @Override
    public String toString(){
        return this.nome;
    }
    @Override
    public boolean equals(Object obj) {
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final GrupoProdutoVO other = (GrupoProdutoVO) obj;
        if (this.codigo != other.codigo) {
            return false;
        }
        return true;
    }
}
```

# Arquitetura em Três Camadas - Projeto02

## ✓ Camada de visão – inserir grupo de produto

```
private void confirmaOperacao() {
    String nome = null;
    float promocao = 0;
    float margem = 0;

    try {
        nome = this.campoNome.getText();
        promocao = Float.parseFloat(this.campoPromocao.getText());
        margem = Float.parseFloat(this.campoMargem.getText());
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, "Digitacao inconsistente");
        return;
    }

    try {
        GrupoProdutoVO grupoVO = new GrupoProdutoVO();
        grupoVO.setNome(nome);
        grupoVO.setMargemLucro(margem);
        grupoVO.setPromocao(promocao);
        grupo.inserir(grupoVO);
        JOptionPane.showMessageDialog(this, "Inclusão realizada com sucesso");
        this.limparCampos();
    } catch (NegocioException ex) {
        JOptionPane.showMessageDialog(this, ex.getMessage());
    }
}
```

# Arquitetura em Três Camadas - Projeto02

## ✓ Camada de negócio – inserir grupo de produto

```
public class GrupoProduto {

    private GrupoProdutoDAO grupoDAO;

    public GrupoProduto() throws NegocioException {
        try {
            grupoDAO = new GrupoProdutoDAO();
        } catch (PersistenciaException ex) {
            throw new NegocioException("Erro ao iniciar Persistência "+ex.getMessage());
        }
    }

    public void inserir(GrupoProdutoVO grupoVO) throws NegocioException {

        String mensagemErros = this.validarDados(grupoVO);

        if (!mensagemErros.isEmpty()) {
            throw new NegocioException(mensagemErros);
        }

        try {
            grupoDAO.incluir(grupoVO);
        } catch (PersistenciaException ex) {
            throw new NegocioException("Erro ao incluir novo grupo de produto - " + ex.getMessage());
        }
    }
}
```

# Camada de Persistência - Projeto02

## ✓ Superclasse DAO

```
package persistencia;

public class DAO {

    protected ConexaoBD conexao;

    public DAO() throws PersistenciaException {
        conexao = ConexaoBD.getInstancia();
    }

    public void desconectarBD() throws PersistenciaException {
        conexao.desconectar();
    }

}
```

# Camada de Persistência - Projeto02

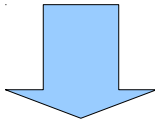
## ✓ Subclasse especializada

```
package persistencia;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import vo.GrupoProdutoVO;
public class GrupoProdutoDAO extends DAO {
    public GrupoProdutoDAO() throws PersistenciaException {
        super();
    }

    public boolean incluir(GrupoProdutoVO grupoProdutoVO) throws PersistenciaException {
        boolean retorno = false;
        try {
            PreparedStatement comando = this.conexao.getConexao().prepareStatement(
                "INSERT INTO GRUPOPRODUTO ( NOME, MARGEMPLUCRO, PROMOCAO ) VALUES (?, ?, ?)");
            comando.setString(1, grupoProdutoVO.getNome());
            comando.setFloat(2, grupoProdutoVO.getMargemLucro());
            comando.setFloat(3, grupoProdutoVO.getPromocao());
            comando.executeUpdate();
            comando.close();
            retorno = true;
        } catch (SQLException ex) {
            throw new PersistenciaException("Erro ao incluir novo grupo de Produto - "+ex.getMessage());
        }
        return retorno;
    }
}
```



# Camada de Persistência - Projeto02

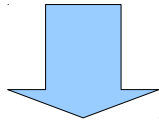


```
public GrupoProdutoVO buscarPorCodigo(int codigo) throws PersistenciaException {  
  
    GrupoProdutoVO gru = null;  
  
    try {  
        PreparedStatement comando = conexao.getConnection().prepareStatement(  
            "SELECT * FROM GRUPOPRODUTO WHERE CODIGO = ?");  
        comando.setInt(1, codigo);  
        ResultSet rs = comando.executeQuery();  
        if (rs.next()) {  
            gru = new GrupoProdutoVO();  
            gru.setCodigo(rs.getInt("codigo"));  
            gru.setNome(rs.getString("Nome").trim());  
            gru.setMargemLucro(rs.getFloat("MargemLucro"));  
            gru.setPromocao(rs.getFloat("Promocao"));  
        }  
        comando.close();  
    } catch (SQLException ex) {  
        throw new PersistenciaException("Erro na seleção por código - "+ex.getMessage());  
    }  
    return gru;  
}
```





# Camada de Persistência - Projeto02

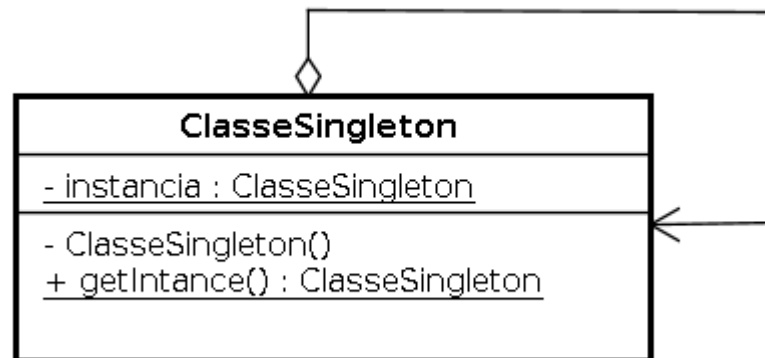


```
public List<GrupoProdutoVO> buscarTodos() throws PersistenciaException {  
    List lista = new ArrayList();  
    GrupoProdutoVO gru = null;  
  
    String comandoSQL = "SELECT * FROM GrupoProduto ORDER BY NOME";  
  
    try {  
        PreparedStatement comando = conexao.getConnection().prepareStatement(comandoSQL);  
        ResultSet rs = comando.executeQuery();  
        while (rs.next()) {  
            gru = new GrupoProdutoVO();  
            gru.setCodigo(rs.getInt("codigo"));  
            gru.setNome(rs.getString("Nome").trim());  
            gru.setMargemLucro(rs.getFloat("MargemLucro"));  
            gru.setPromocao(rs.getFloat("Promocao"));  
            lista.add(gru);  
        }  
        comando.close();  
    } catch (SQLException ex) {  
        throw new PersistenciaException("Erro na seleção - "+ex.getMessage());  
    }  
    return lista;  
}
```



# Gerenciador de Conexão - Singleton

```
1 package persistencia;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class ConexaoBD {
8
9     private Connection con;
10    private static ConexaoBD instancia;
11
12    private ConexaoBD() throws PersistenciaException {
13        try {
14            Class.forName("org.postgresql.Driver");
15            String url = "jdbc:postgresql://localhost:5432/estoque";
16            con = DriverManager.getConnection(url, "postgres", "postgres");
17        } catch (SQLException ex) {
18            throw new PersistenciaException("Erro ao conectar o banco de dados - " +
19                ex.toString());
20        } catch (ClassNotFoundException ex) {
21            throw new PersistenciaException("Driver do banco de dados não localizado - " +
22                ex.toString());
23        }
24    }
25 }
```



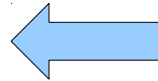
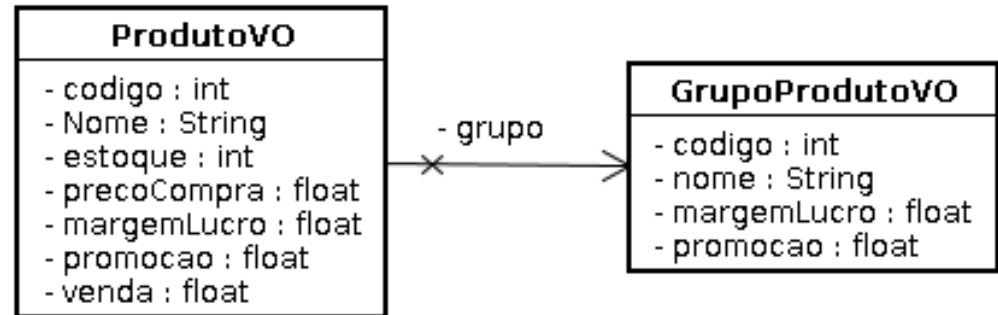
# Gerenciador de Conexão - Singleton

```
26 public static ConexaoBD getInstancia() throws PersistenciaException {  
27     if (instancia == null) {  
28         instancia = new ConexaoBD();  
29     }  
30     return instancia;  
31 }  
32  
33 public Connection getConexao() {  
34     return con;  
35 }  
36  
37 public void desconectar() throws PersistenciaException {  
38     try {  
39         con.close();  
40     } catch (SQLException ex) {  
41         throw new PersistenciaException("Erro ao desconectar o banco de dados - " +  
42             ex.toString());  
43     }  
44 }  
45 }
```

# Mapeamento de Relações - Projeto02

```
package vo;
```

```
public class ProdutoVO {  
    private int codigo;  
    private String Nome;  
    private int estoque;  
    private float precoCompra;  
    private float margemLucro;  
    private float promocao;  
    private float venda;  
    private GrupoProdutoVO grupo;  
  
    public GrupoProdutoVO getGrupo() {  
        return grupo;  
    }  
    public void setGrupo(GruoProdutoVO grupo) {  
        this.grupo = grupo;  
    }  
    public int getCodigo() {...}  
    public void setCodigo(int codigo) {...}  
    public String getNome() {...}  
    public void setNome(String Nome) {...}  
}
```



# Mapeamento de Relações - Projeto02

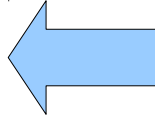
## ✓ No banco de dados

```
CREATE TABLE grupoproduto
(
  codigo serial NOT NULL,
  nome character(40) NOT NULL,
  promocao numeric(5,2) DEFAULT 0,
  margemlucro numeric(5,2) DEFAULT 0,
  CONSTRAINT grupoproduto_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE produto
(
  codigo serial NOT NULL,
  nome character(50) NOT NULL,
  estoque integer DEFAULT 0,
  valorcompra numeric(10,2),
  promocao numeric(5,2) DEFAULT 0,
  margemlucro numeric(5,2) DEFAULT 0,
  grupo integer NOT NULL,
  CONSTRAINT produto_pkey PRIMARY KEY (codigo),
  CONSTRAINT produto_grupo_fkey FOREIGN KEY (grupo)
    REFERENCES grupoproduto (codigo) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

# Mapeamento de Relações - Projeto02

```
public int incluir(ProdutoVO produto) throws PersistenciaException {
    int retorno = 0;
    try {
        PreparedStatement comando = conexao.getConexao().prepareStatement(
            "INSERT INTO PRODUTO ( NOME, ESTOQUE, VALORCOMPRA, MARGEMPLUCRO, PROMOCAO, GRUPO ) VALUES ( ? , ? , ? , ? , ? , ? )";
        comando.setString(1, produto.getNome());
        comando.setInt(2, produto.getEstoque());
        comando.setFloat(3, produto.getPrecoCompra());
        comando.setFloat(4, produto.getMargemLucro());
        comando.setFloat(5, produto.getPromocao());
        comando.setInt(6, produto.getGrupo().getCodigo());
        retorno = comando.executeUpdate();
        comando.close();
    } catch (SQLException ex) {
        throw new PersistenciaException("Erro ao incluir novo produto - "+ex.getMessage());
    }
    return retorno;
}
```



# Mapeamento de Relações - Projeto02

```
public ProdutoVO buscarPorCodigo(int codigo) throws PersistenciaException {

    ProdutoVO pro = null;
    GrupoProdutoDAO grupoDAO = new GrupoProdutoDAO();

    try {
        PreparedStatement comando = conexao.getConexao().prepareStatement(
            "SELECT * FROM PRODUTO WHERE CODIGO = ?");
        comando.setInt(1, codigo);
        ResultSet rs = comando.executeQuery();
        if (rs.next()) {
            pro = new ProdutoVO();
            pro.setCodigo(rs.getInt("codigo"));
            pro.setNome(rs.getString("Nome").trim());
            pro.setEstoque(rs.getInt("Estoque"));
            pro.setPrecoCompra(rs.getFloat("ValorCompra"));
            pro.setMargemLucro(rs.getFloat("MargemLucro"));
            pro.setPromocao(rs.getFloat("promocao"));
            pro.setGrupo(grupoDAO.buscarPorCodigo(rs.getInt("grupo")));
        }
        comando.close();
    } catch (Exception ex) {
        throw new PersistenciaException("Erro na seleção por código - "+ex.getMessage());
    }
    return pro;
}
```

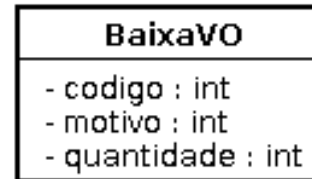


# Mapeamento de Relações - Projeto02

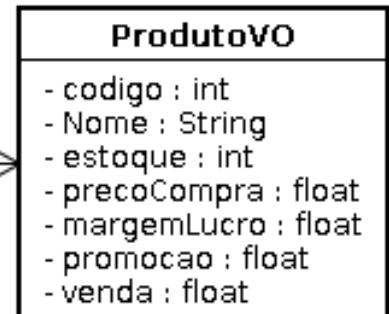
```
package vo;  
import java.sql.Date;  
public class BaixaVO {
```

```
    private int codigo;  
    private TipoMotivoBaixa motivo;  
    private Date data;  
    private ProdutoVO produto;  
    private int quantidade;
```

```
    public ProdutoVO getProduto() {  
        return produto;  
    }  
    public void setProduto(ProdutoVO produto) {  
        this.produto = produto;  
    }  
    public TipoMotivoBaixa getMotivo() {...}  
    public void setMotivo(TipoMotivoBaixa motivo) {...}  
    public Date getData() {...}  
    public void setData(Date data) {...}  
    public int getQuantidade() {...}  
    public void setQuantidade(int quantidade) {...}  
    @Override  
    public boolean equals(Object obj) {...}  
    @Override  
    public int hashCode() {...}
```



- produto



```
}
```



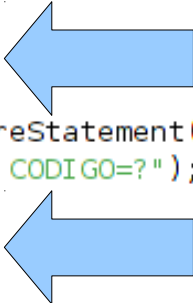
# Mapeamento de Relações - Projeto02

## ✓ No banco de dados

```
CREATE TABLE registrobaixa
(
  codigo serial NOT NULL,
  motivo smallint NOT NULL,
  data date NOT NULL,
  quantidade integer NOT NULL,
  produto integer NOT NULL,
  CONSTRAINT registrobaixa_pkey PRIMARY KEY (codigo),
  CONSTRAINT registrobaixa_produto_fkey FOREIGN KEY (produto)
    REFERENCES produto (codigo) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

# Mapeamento de Relações - Projeto02

```
public void registrarBaixa(BaixaVO baixaVO) throws PersistenciaException {
    try {
        conexao.getConexao().setAutoCommit(false);
        PreparedStatement comando1 = conexao.getConexao().prepareStatement(
            "INSERT INTO REGISTROBAIXA ( MOTIVO, DATA, PRODUTO, QUANTIDADE )VALUES ( ? , ? , ? , ? )");
        comando1.setInt(1, baixaVO.getMotivo().ordinal());
        comando1.setDate(2, baixaVO.getData());
        comando1.setInt(3, baixaVO.getProduto().getCodigo());
        comando1.setInt(4, baixaVO.getQuantidade());
        PreparedStatement comando2 = conexao.getConexao().prepareStatement(
            "UPDATE PRODUTO SET ESTOQUE=(ESTOQUE - ?) WHERE CODIGO=?");
        comando2.setInt(1, baixaVO.getQuantidade());
        comando2.setInt(2, baixaVO.getProduto().getCodigo());
        comando1.executeUpdate();
        comando2.executeUpdate();
        comando1.close();
        comando2.close();
        conexao.getConexao().commit();
    } catch (SQLException ex) {
        try {
            conexao.getConexao().rollback();
        } catch (SQLException ex1) {
            ex1.printStackTrace();
        }
        throw new PersistenciaException("Erro ao baixar estoque do produto" + ex.toString());
    } finally {
        try {
            conexao.getConexao().setAutoCommit(true);
        } catch (SQLException ex1) {
            //
        }
    }
}
```



# Mapeamento de Identidade

- ✓ O ID dos objetos não podem ser usados como ID no banco de dados relacional;
  - Quando recupera-se um registro de uma tabela o ID do objeto que receberá os dados do registro não pode ser definido externamente ao mecanismos de execução dos objetos;
  - O ID do objeto será gerado segundo as regras do mecanismo de execução sem interferência do código gerado pelo programador;
- ✓ Uma possibilidade é usar o ID do banco como identificador da instância que representa o registro;

# Mapeamento de Identidade

## ✓ Exemplo de necessidade de Identidade

**Manutenção de Produto**

Digite parte do nome do produto

Codigo	Nome	Estoque	Compra	Margem	Promoção	Grupo	Venda
42	Ervilha em Lata	50	3.0	40.0	5.0	Enlatados	3.9899998
41	Picanha Nobre	150	20.0	50.0	0.0	Carnes	30.0
40	Presunto Cozido	200	15.0	25.0	0.0	Frios	18.75

**Alteração de Produtos**

Código

Nome

Valor Compra

% Margem Lucro


% Promocao

Grupo

# Mapeamento de Identidade - Projeto02

- ✓ Em uma lista de grupos de produto deve-se localizar um deles

```
private void preencheCampos() {  
    try {  
        ProdutoVO produtoVO = produto.pesquisaCodigo(codigoProduto);  
  
        if (produtoVO != null) {  
            this.campoCodigo.setText(String.valueOf(produtoVO.getCodigo()));  
            this.campoNome.setText(produtoVO.getNome());  
            this.campoCompra.setText(String.valueOf(produtoVO.getPrecoCompra()));  
            this.campoMargem.setText(String.valueOf(produtoVO.getMargemLucro()));  
            this.campoPercentualPromocao.setText(String.valueOf(produtoVO.getPromocao()));  
        }  
        this.comboGrupoProduto.setModel(new DefaultComboBoxModel(grupo.buscarTodos().toArray()));  
        this.comboGrupoProduto.setSelectedItem(produtoVO.getGrupo());  
    } catch (NegocioException ex) {  
        JOptionPane.showMessageDialog(this, "Erro ao recuperar dados de produtos \n" + ex.toString());  
    }  
}
```



- ✓ Como o algoritmo do Combobox sabe comparar/localizar um objeto GrupoProdutoVO?

# Mapeamento de Identidade - Projeto02

- ✓ O Combobox usa o ID do objeto para comparar/localizar um objeto;
- ✓ Em Java, a classe Object, superclasse de todas as classes Java, possui um método equals( Object o) que compara dois objetos;

Sintaxe: objetoA.equals(objetoB)

- ✓ Quando que o objeto apontado por objetoA será igual ao objeto apontado por objetoB?
- ✓ Quando objetoA e objetoB apontarem para a mesma instância de objetos!

# Mapeamento de Identidade

## ✓ Exemplo hipotético

ID: 023478976 <u>Codigo: 30</u> Nome: Enlatados Margem Lucro: 35 promoção: 5	ID: 126477984 <u>Codigo: 31</u> Nome: Carnes Margem Lucro: 40 promoção: 0	ID: 423387593 <u>Codigo: 32</u> Nome: Frios Margem Lucro: 25 promoção: 0
------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------

← No Combobox

ID: 357896871 <u>Codigo: 32</u> Nome: Frios Margem Lucro: 25 promoção: 0
--------------------------------------------------------------------------------------

← No Produto

Usando o algoritmo nativo serão comparados os ID's.  
Embora seja o mesmo grupo de produto os ID's dos objetos são diferentes

# Mapeamento de Identidade

- ✓ Mesma situação com algoritmo nativo de comparação de objetos

The image shows a software interface for product management. The main window, titled "Manutenção de Produto", contains a search bar with the placeholder text "Digite parte do nome do produto" and a "Filtrar" button. Below the search bar is a table with the following data:

Código	Nome	Estoque	Compra	Margem	Promoção	Grupo	Venda
42	Ervilha em Lata	50	3.0	40.0	5.0	Enlatados	3.9899998
41	Picanha Nobre	150	20.0	50.0	0.0	Carnes	30.0
40	Presunto Cozido	200	15.0	25.0	0.0	Frios	18.75

At the bottom of the main window are three buttons: "Incluir", "Alterar", and "Excluir". An "Alteração de Produtos" modal window is open, displaying the details for the selected product (Código: 40). The modal contains the following fields:

- Código: 40
- Nome: Presunto Cozido
- Valor Compra: 15.0
- % Margem Lucro: 25.0
- % Promocao: 0.0
- Grupo: Carnes (selected from a dropdown menu)

At the bottom of the modal are two buttons: "Confirmar" and "Cancelar".



# Mapeamento de Identidade - Projeto02

- ✓ Sobrescrita do método equals na classe GrupoProdutoVO
- ✓ Define um novo critério de igualdade

```
@Override
public boolean equals(Object obj) {
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final GrupoProdutoVO other = (GrupoProdutoVO) obj;
    if (this.codigo != other.codigo) {
        return false;
    }
    return true;
}
```

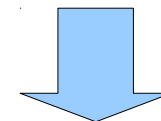
# Mapeamento de Identidade - Projeto02

- ✓ O algoritmo de busca do combobox irá usar o método equals sobrescrito para comparar os objetos

```
private void preencheCampos() {  
    try {  
        ProdutoVO produtoVO = produto.pesquisaCodigo(codigoProduto);  
  
        if (produtoVO != null) {  
            this.campoCodigo.setText(String.valueOf(produtoVO.getCodigo()));  
            this.campoNome.setText(produtoVO.getNome());  
            this.campoCompra.setText(String.valueOf(produtoVO.getPrecoCompra()));  
            this.campoMargem.setText(String.valueOf(produtoVO.getMargemLucro()));  
            this.campoPercentualPromocao.setText(String.valueOf(produtoVO.getPromocao()));  
        }  
        this.comboGrupoProduto.setModel(new DefaultComboBoxModel(grupo.buscarTodos().toArray()));  
        this.comboGrupoProduto.setSelectedItem(produtoVO.getGrupo());  
    } catch (NegocioException ex) {  
        JOptionPane.showMessageDialog(this, "Erro ao recuperar dados de produtos \n" + ex.toString());  
    }  
}
```

# Outra Sobrescrita Interessante

- ✓ Veja os seguintes resultados e os respectivos trechos de código

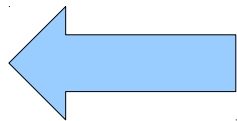


Codigo	Nome	Estoque	Compra	Margem	Promoção	Grupo	Venda
42	Ervilha em Lata	50	3.0	40.0	5.0	Enlatados	3.9899998
41	Picanha Nobre	150	20.0	50.0	0.0	Carnes	30.0
40	Presunto Cozido	200	15.0	25.0	0.0	Frios	18.75

# Método toString() - Projeto02

## ✓ Método da classe TableModelProduto

```
public Object getValueAt(int rowIndex, int columnIndex) {  
    if (columnIndex == 0) {  
        return (tabelaDados.get(rowIndex)).getCodigo();  
    } else if (columnIndex == 1) {  
        return (tabelaDados.get(rowIndex)).getNome();  
    } else if (columnIndex == 2) {  
        return (tabelaDados.get(rowIndex)).getEstoque();  
    } else if (columnIndex == 3) {  
        return (tabelaDados.get(rowIndex)).getPrecoCompra();  
    } else if (columnIndex == 4) {  
        return (tabelaDados.get(rowIndex)).getMargemLucro();  
    } else if (columnIndex == 5) {  
        return (tabelaDados.get(rowIndex)).getPromocao();  
    } else if (columnIndex == 6) {  
        return (tabelaDados.get(rowIndex)).getGrupo();  
    } else if (columnIndex == 7) {  
        return (tabelaDados.get(rowIndex)).getVenda();  
    } else {  
        return null;  
    }  
}
```



# Método toString() - Projeto02

- ✓ Lista de Grupos de Produtos na inclusão de um Produto

The image shows a Java Swing dialog box titled "Inclusão de Produtos". It contains several text input fields for "Nome", "Estoque", "Valor Com...", "% Margem Lucro", and "% Promocao". Below these is a "Grupo" dropdown menu. The dropdown is open, showing a list of options: "Carnes", "Carnes", "Enlatados", and "Frios". A blue arrow points to the dropdown menu. A "Confirmar" button is visible at the bottom left of the dialog.

# Método toString() - Projeto02

- ✓ Apenas é inserida uma lista de objetos GrupoProdutoVO no combobox

```
private void preencheComboGrupoProduto(){  
    try {  
        this.comboGrupoProduto.setModel(new DefaultComboBoxModel(grupo.buscarTodos().toArray()));  
        this.comboGrupoProduto.setSelectedIndex(0);  
    } catch (NegocioException ex) {  
        JOptionPane.showMessageDialog(this, "Erro ao recuperar a lista de grupos de produtos" + ex.getMessage());  
    }  
}
```

# Método toString() - Projeto02

- ✓ Em Java, toda vez que um objeto for enviado para algum dispositivo de saída (escrito) será executado um algoritmo padrão – método toString() do objeto;
- ✓ A classe Object possui uma implementação padrão para o toString(), normalmente não é útil;
- ✓ Uma ideia interessante é fazer a sobrescrita do método toString() nos VO's;
- ✓ Pode-se definir quais dados de um objeto devem ser exibidos, quando esse for escrito em algum lugar.

# Método toString() - Projeto02

- ✓ Sobrescrita do método toString na classe GrupoProdutoVO

```
@Override  
public String toString() {  
    return this.nome.trim();  
}
```



## Método toString() - Projeto02

- ✓ Considere a hipótese de ser necessário apresentar o código – nome do grupo de produto;
- ✓ Caso não tenha sido usado a sobrescrita do método toString(), e sim a escrita explícita do nome do grupo, por exemplo: grupo.getNome(), será necessário alterar todos os locais que o objeto foi escrito;
- ✓ Caso tenha sido usado o recurso de sobrescrita do toString(), bastará alterar o método toString().

# Método toString() - Projeto02

## ✓ Veja alteração e resultados

```
@Override
public String toString() {
    return this.codigo + " - " + this.nome.trim();
}
```

**Digite parte do nome do produto**

Codigo	Nome	Estoque	Compra	Margem	Promoção	Grupo	Venda
42	Ervilha em Lata	50	3.0	40.0	5.0	30 - Enlatados	3.9899998
41	Picanha Nobre	150	20.0	50.0	0.0	31 - Carnes	30.0
40	Presunto Cozido	200	15.0	25.0	0.0	32 - Frios	18.75

% Promocao

Grupo 

32 - Frios

31 - Carnes

30 - Enlatados

32 - Frios