



Universidade Estadual de Maringá (UEM)

Departamento de Informática (DIN)

Curso de Engenharia de Produção



Programação de Sistemas: **ORM, JPA, Hibernate, DAO e JavaDB**

PROF. MS. RICARDO THEIS GERALDI

RICARDOGERALDI@GMAIL.COM

Aula Anterior

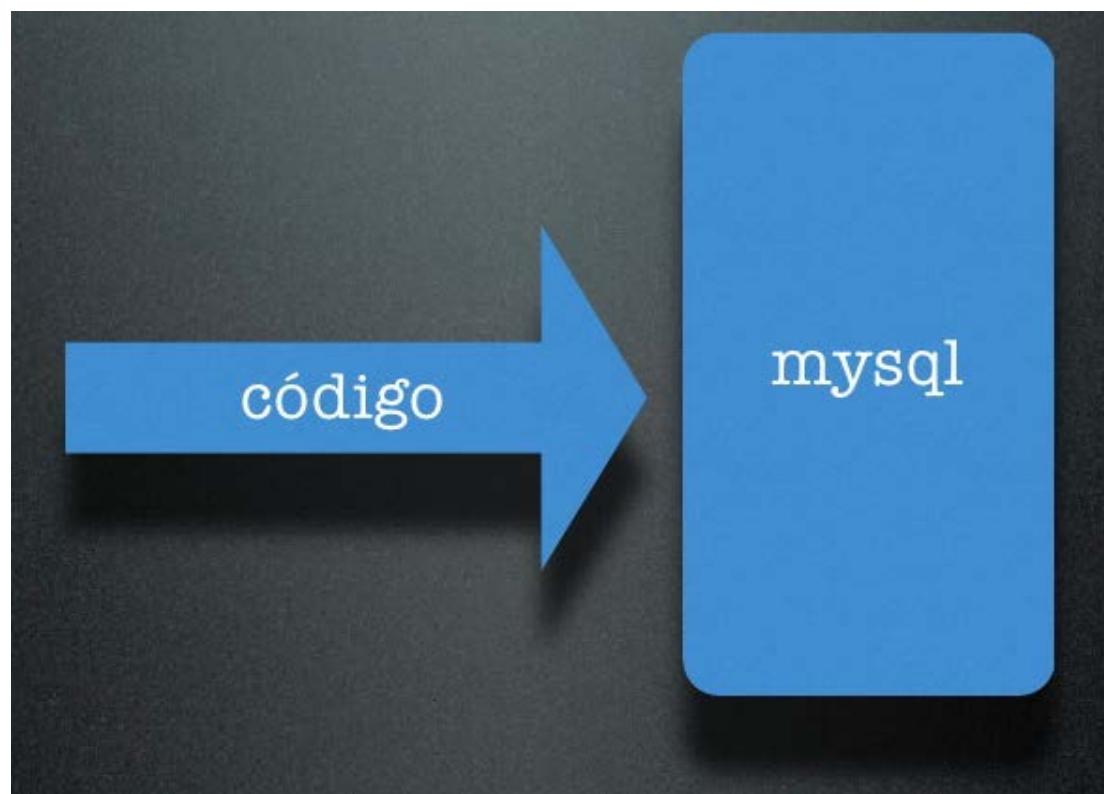
- ▶ Orientação a Objetos
 - ▶ GUI com Swing e AWT

Aula de Hoje

- ▶ Mapeamento Objeto-Relacional (ORM);
- ▶ **JPA - Java Persistence API e Hibernate;**
- ▶ Desenvolvimento de projeto “**ExemploCadastro**” com métodos CRUD (*Create, Read, Update e Delete*);
 - ▶ Usado para exemplificar: **DAO – Data Access Object;**
- ▶ Banco de Dados JavaDB.

Introdução

- ▶ Como você grava no seu banco de dados?



Acessando via C



```
/* Connect to database */
if (!mysql_real_connect(conn, server,
    user, password, database, 0, NULL, 0)) {
    fprintf(stderr, "%s\n", mysql_error(conn));
    exit(1);
}

/* send SQL query */
if (mysql_query(conn, "show tables")) {
    fprintf(stderr, "%s\n", mysql_error(conn));
    exit(1);
}

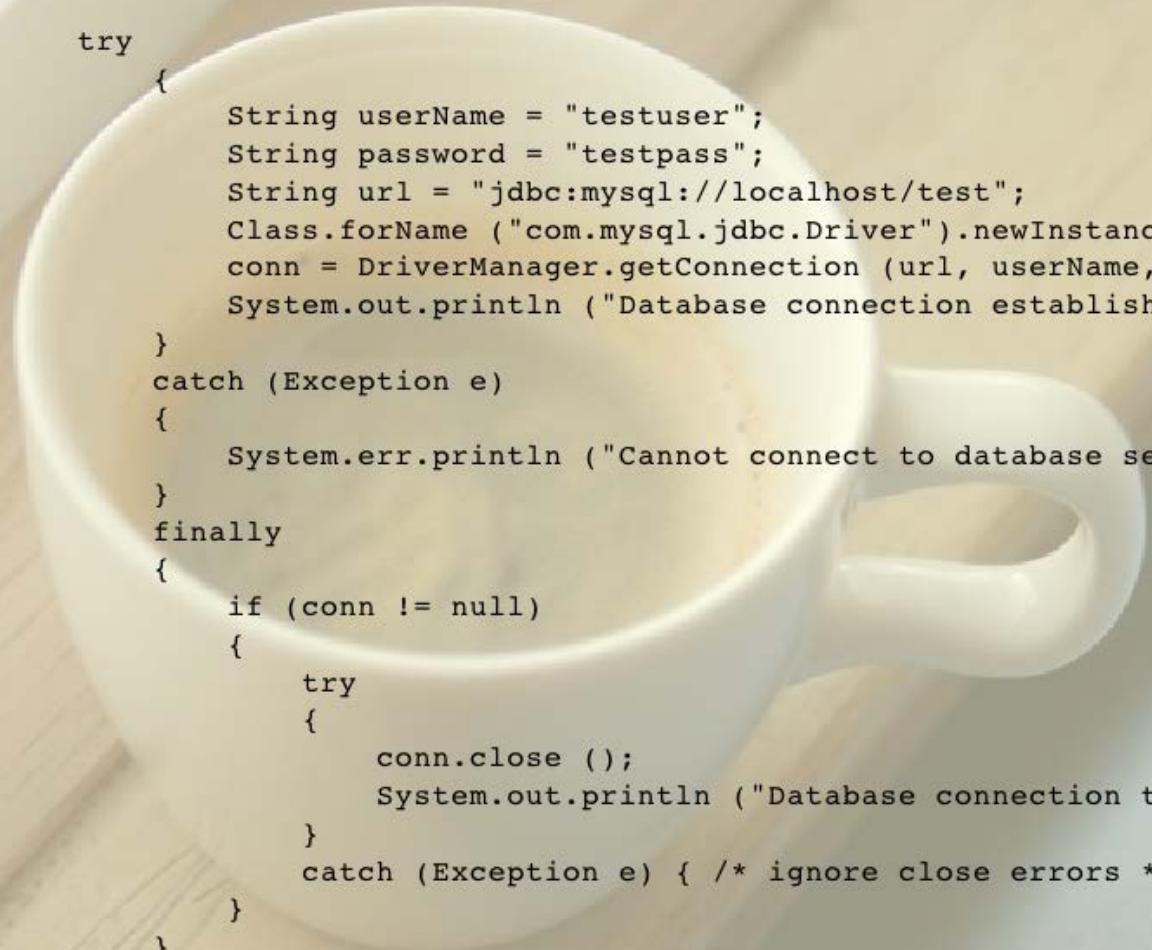
res = mysql_use_result(conn);

/* output table name */
printf("MySQL Tables in mysql database:\n");
while ((row = mysql_fetch_row(res)) != NULL)
    printf("%s \n", row[0]);
```

Acessando com Java

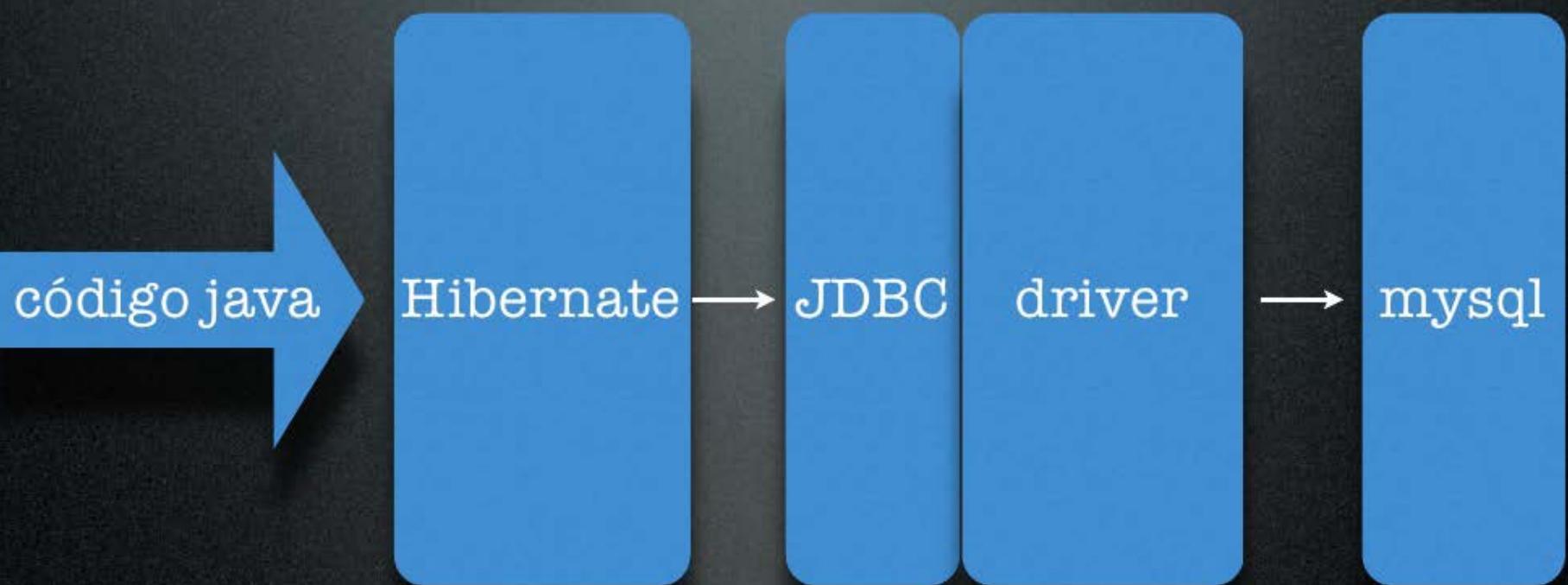


Acessando com Java via JDBC



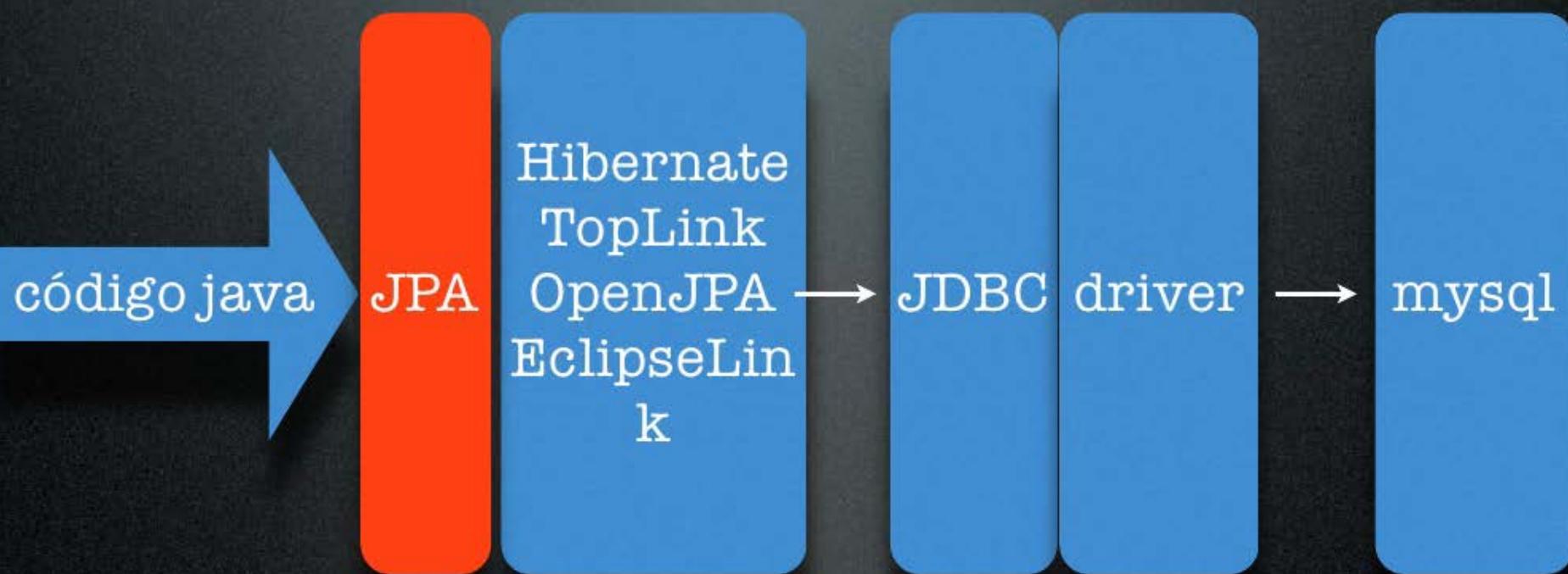
```
try
{
    String userName = "testuser";
    String password = "testpass";
    String url = "jdbc:mysql://localhost/test";
    Class.forName ("com.mysql.jdbc.Driver").newInstance ();
    conn = DriverManager.getConnection (url, userName, password);
    System.out.println ("Database connection established");
}
catch (Exception e)
{
    System.err.println ("Cannot connect to database server");
}
finally
{
    if (conn != null)
    {
        try
        {
            conn.close ();
            System.out.println ("Database connection terminated");
        }
        catch (Exception e) { /* ignore close errors */ }
    }
}
```

Mapeamento Objeto-Relacional (ORM)



Mapeamento Objeto/Relacional (ORM)

Com JPA e Hibernate



Mapeamento Objeto-Relacional (ORM)

- ▶ JPA é um modelo de persistência.
- ▶ Java é utilizada em ambientes corporativos, com grandes bancos de dados (Oracle).
- ▶ Agiliza o desenvolvimento, já que não precisamos escrever comandos SQL.
- ▶ Fácil de mudar de banco de dados, caso necessário.

JPA e Hibernate

- ▶ JPA é a especificação da Oracle para mapeamento objeto-relacional.
- ▶ Hibernate apresenta uma maneira mais elegante de trabalhar com a persistência do que o J2EE (*Java Enterprise Edition*);
- ▶ Hibernate facilita o mapeamento objeto-relacional.

Arquivo de Configuração

- ▶ O arquivo "**persistence.xml**" contém os parâmetros de configuração para acesso ao banco de dados e deve estar no diretório **META-INF**.
- ▶ **Persistence unit name** ("**ExemploCadastroPU**"): indica os parâmetros de acesso à base de dados.
- ▶ **Transaction-type**: **RESOURCE_LOCAL** para servidor web e JTA para servidor de aplicação.
- ▶ **Provider**: qualquer implementação do JPA. Pode ser o Hibernate, OpenJPA ou EclipseLink, que é o produto oficial da Oracle.

Exemplo "Persistence.xml"

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
5 <persistence-unit name="ExemploHibernatePU" transaction-type="RESOURCE_LOCAL">
6     <provider>org.hibernate.ejb.HibernatePersistence</provider>
7     <class>entidades.Cidade</class>
8     <class>entidades.Pessoa</class>
9     <properties>
10         <property name="hibernate.dialect" value="org.hibernate.dialect.DerbyDialect"/>
11         <property name="hibernate.connection.username" value="adminadmin"/>
12         <property name="hibernate.connection.driver_class" value="org.apache.derby.jdbc.ClientDriver"/>
13         <property name="hibernate.connection.password" value="adminadmin"/>
14         <property name="hibernate.connection.url" value="jdbc:derby://localhost:1527/exemplo"/>
15         <property name="hibernate.hbm2ddl.auto" value="update"/>
16         <property name="hibernate.show_sql" value="true"/>
17     </properties>
18 </persistence-unit>
19 </persistence>
```

Entidades

- ▶ Com JPA podemos mapear as classes de entidade diretamente para o banco de dados (JavaDB).
- ▶ Vamos considerar os seguintes exemplos de classes:
 - ▶ Carro;
 - ▶ Cidade;
 - ▶ Livro;
 - ▶ Autor.
- ▶ As **Anotações (Annotations) do JPA** estão no pacote: **javax.persistence**.

JPA - Java Persistence API

- ▶ **Entity**: suporta herança e polimorfismo;
- ▶ **EntityManager**: responsável pelas operações de persistência de objetos;
- ▶ **PersistenceContext**: área da memória que mantém os objetos que estão sendo manipulados pelo EntityManager;
- ▶ **Provedores**: especificação para frameworks de persistência.

JPA - Annotations (Mapeamento)

- ▶ Mapeamento pode ser feito por metadados (.xml) ou **Annotations**:
 - ▶ **@Entity**: Entidade a ser persistida;
 - ▶ **@Table**: especifica propriedades da tabela;
 - ▶ **@Column**: especifica propriedades da coluna;
 - ▶ **@Id**: chave primária (identificador);
 - ▶ **@NamedQuery**: cria consultas (SQL) estáticas;
 - ▶ **@GeneratedValue**: gera id's automaticamente;
 - ▶ **@Temporal**: informações de tempo (Date, Time, etc.);
 - ▶ **@OneToMany**: relacionamento “um-para-muitos”;
 - ▶ **@ManyToOne**: relacionamento “muitos-para-um”;
 - ▶ **@Version**: controle de versão do objeto.

Exemplo Classe "Carro"

```
@Entity
@Table (name = "CARRO")
@NamedQueries({
    @NamedQuery(
        name = "Carro.getAll",
        query = "select c from Carro c"
    ),
    @NamedQuery(
        name = "Carro.getCarro",
        query = "select c from Carro c where c.id = :idCarro"
    )
})
public class Carro implements Serializable {

    /**
     * Serial Version
     */
    private static final long serialVersionUID = 3156919887468366911L;

    @Id
    private Integer id;

    @Column(name = "nome", nullable = false)
    private String nome;

    @OneToMany(mappedBy = "carro", cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private List<Item> itens = new ArrayList<Item>();
```

Exemplo Classe "Carro"

```
@Entity
@Table(name = "ITEM")
@NamedQueries ({
    @NamedQuery(
        name = "Item.getAll",
        query = "select item from Item item"
    ),
    @NamedQuery(
        name = "Item.getItem",
        query = "select item from Item item where item.id = :idItem"
    )
})
public class Item implements Serializable{

    /**
     * Serial Version.
     */
    private static final long serialVersionUID = 4785299543804211500L;

    @Id
    private Integer id;

    @Column(name = "NOME", nullable = false)
    private String nome;

    @Column(name = "DESCRICAO", nullable = true)
    private String descricao;

    @ManyToOne
    private Carro carro;
```

Exemplo "Persistence.xml"

Classes "Carro" e "Item"

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persist
<persistence-unit name="JPA-Metodista" transaction-type="RESOURCE_LOCAL">
  <provider>oracle.toplink.essentials.PersistenceProvider</provider> ← Green arrow
  <class>br.com.rdg.c.Carro</class> ← Yellow arrow
  <class>br.com.rdg.c.Item</class>
  <properties>
    <property name="toplink.logging.level" value="FINE"/>
    <property name="toplink.jdbc.driver" value="org.hsqldb.jdbcDriver"/> ← Red arrow
    <property name="toplink.jdbc.url" value="jdbc:hsqldb:http://localhost:59999/myorg"/>
    <property name="toplink.jdbc.user" value="rodrigo"/>
    <property name="toplink.jdbc.password" value="123"/>
  </properties>
</persistence-unit>
</persistence>
```

Relacionamentos Classes "Carro" e "Item"

- Tipos:
 - `@MappedSuperClass`
 - `@Embedded`
 - `@Embeddable`
 - `@ManyToMany`
 - `@ManyToOne`
 - `@OneToMany`
 - `@OneToOne`
 - `@JoinTable`
 - `@JoinColumns`
 - `@JoinColumn`
 - `@MapKey`
- Entity: Carro

```
@OneToOne (mappedBy = "carro",
           cascade = CascadeType.ALL,
           fetch = FetchType.EAGER)
private List<Item> itens = new ArrayList<Item>();
```
- Entity: Item

```
@ManyToOne
private Carro carro;
```

Manipulando Entity's

Entity: Carro carro;

```
EntityManagerFactory emf =  
    Persistence.createEntityManagerFactory("apresentacaoJPA");  
EntityManager em = emf.createEntityManager();  
EntityTransaction et = em.getTransaction();  
et.begin();  
    Insert: em.persist(carro);  
    Update: em.merge(carro);  
    Remove: em.remove(carro);  
    Busca: em.find(Carro.class, id);  
et.commit(); ou et.rollback();
```

Projeto "ExemploCadastro"

Classe "Cidade"

INÍCIO PROJETO "EXEMPLOCADASTRO"

```
1 package entidades;  
2  
3 public class Cidade {  
4     private Long id;  
5     private String nome;  
6     private String uf;  
7     private Long populacao;  
8  
9     (...Sets e Gets...)  
10 }
```

Projeto "ExemploCadastro"

Classe "Cidade" com JPA

```
1 package entidades;
2
3 import java.io.Serializable;
4 import javax.persistence.Entity;
5 import javax.persistence.GeneratedValue;
6 import javax.persistence.GenerationType;
7 import javax.persistence.Id;
8
9 @Entity // A Classe é uma Entidade e é armazenada em uma tabela no banco.
10 public class Cidade implements Serializable {
11     private static final long serialVersionUID = 1L;
12     @Id // Campo da Chave Primária
13     @GeneratedValue // A chave-primária é gerada automaticamente.
14     private Long id;
15     private String nome;
16     private String uf;
17     private Long populacao;
18 }
```

Projeto "ExemploCadastro"

Exemplo "Persistence.xml"

Classes "Cidade" e "Pessoa"

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
5 <persistence-unit name="ExemploHibernatePU" transaction-type="RESOURCE_LOCAL">
6     <provider>org.hibernate.ejb.HibernatePersistence</provider>
7     <class>entidades.Cidade</class>
8     <class>entidades.Pessoa</class>
9     <properties>
10         <property name="hibernate.dialect" value="org.hibernate.dialect.DerbyDialect"/>
11         <property name="hibernate.connection.username" value="adminadmin"/>
12         <property name="hibernate.connection.driver_class" value="org.apache.derby.jdbc.ClientDriver"/>
13         <property name="hibernate.connection.password" value="adminadmin"/>
14         <property name="hibernate.connection.url" value="jdbc:derby://localhost:1527/exemplo"/>
15         <property name="hibernate.hbm2ddl.auto" value="update"/>
16         <property name="hibernate.show_sql" value="true"/>
17     </properties>
18 </persistence-unit>
19 </persistence>
```

Projeto "ExemploCadastro"

Classe "Banco"

```
1 package exemplodaaogenerico;
2
3 import javax.persistence.EntityManager;
4 import javax.persistence.Persistence;
5
6 public class Banco {
7     private static Banco instancia;
8
9     private EntityManager em;
10
11    public synchronized static Banco getInstance(){
12        if (instancia==null){
13            instancia=new Banco();
14        }
15        return instancia;
16    }
17
18    private Banco() {
19        em = Persistence.createEntityManagerFactory("ExemploHibernatePU").createEntityManager();
20    }
21
22    public EntityManager getEm() {
23        return em;
24    }
25
26 }
```

Projeto "ExemploCadastro"

Classe “DAO“ Genérico

```
1 package exemplodaogenerico;
2
3 import java.io.Serializable;
4 import java.util.List;
5 import javax.persistence.Query;
6 import javax.persistence.EntityManager;
7 import org.hibernate.HibernateException;
8
9 public class DAO<E> {
10
11     private Class<E> classeEntidade;
12     private Query consulta;
13     private EntityManager em;
14
15     public DAO(Class<E> c) {
16         try {
17             this.classeEntidade = c;
18             em = Banco.getInstancia().getEm();
19             consulta = em.createQuery("from " + classeEntidade.getSimpleName());
20         } catch (Exception e) {
21             e.printStackTrace();
22         }
23     }
24
25     //(...Métodos Genéricos CRUD...)
26 }
```

Projeto "ExemploCadastro"

Classe “DAO” Método "insert"

```
1 public void insert(E obj) {  
2     try {  
3         em.getTransaction().begin();  
4         em.persist(obj);  
5         em.getTransaction().commit();  
6     } catch (Exception e) {  
7         if (em.getTransaction().isActive()) {  
8             em.getTransaction().rollback();  
9         }  
10        e.printStackTrace();  
11    }  
12}
```

Projeto "ExemploCadastro"

Classe “DAO” Método “update”

```
1 public void update(E obj) {  
2     try {  
3         em.getTransaction().begin();  
4         em.merge(obj);  
5         em.getTransaction().commit();  
6     } catch (Exception e) {  
7         if (em.getTransaction().isActive()) {  
8             em.getTransaction().rollback();  
9         }  
10        e.printStackTrace();  
11    }  
12}
```

Projeto "ExemploCadastro"

Classe “DAO” Método “delete”

```
1 public void delete(E obj) {  
2     try {  
3         em.getTransaction().begin();  
4         em.remove(obj);  
5         em.getTransaction().commit();  
6     } catch (Exception e) {  
7         if (em.getTransaction().isActive()) {  
8             em.getTransaction().rollback();  
9         }  
10        e.printStackTrace();  
11    }  
12 }
```

Projeto "ExemploCadastro"

Classe “DAO” Método “list”

```
1 @SuppressWarnings("unchecked")
2     public List<E> list() {
3         List<E> l = null;
4         try {
5             em.getTransaction().begin();
6             l = consulta.getResultList();
7             em.getTransaction().commit();
8         } catch (Exception e) {
9             if (em.getTransaction().isActive()) {
10                 em.getTransaction().rollback();
11             }
12             e.printStackTrace();
13         }
14         return l;
15     }
```

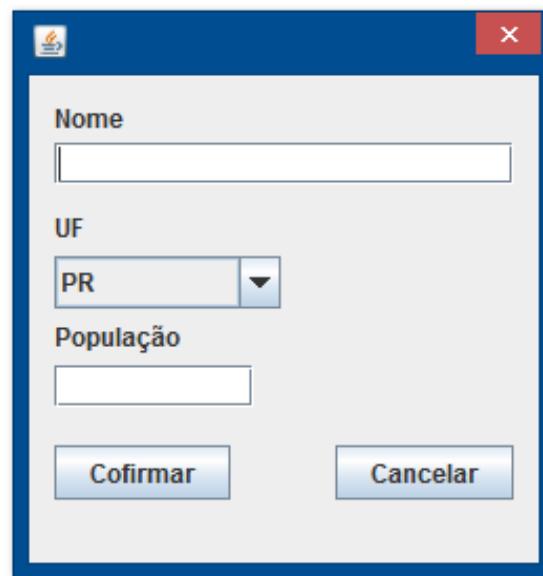
Projeto "ExemploCadastro"

Classe “DAO” Método “find”

```
1 public E find(Serializable id) {  
2     E c = null;  
3     try {  
4         em.getTransaction().begin();  
5         c = (E) em.find(classeEntidade, id);  
6         em.getTransaction().commit();  
7     } catch (Exception e) {  
8         if (em.getTransaction().isActive()) {  
9             em.getTransaction().rollback();  
10        }  
11        e.printStackTrace();  
12    }  
13    return c;  
14 }
```

Projeto "ExemploCadastro"

Classe “DetalhesCidade”



Projeto "ExemploCadastro"

Classe “DetalhesCidade”

```
1 package exemplodagenerico;
2
3 import entidades.Cidade;
4
5 public class DetalhesCidade extends javax.swing.JDialog {
6
7     private Cidade cid;
8
9     private boolean confirmado; //Valor default de um atributo booleano é false
10
11    public void entidadeInterface(){ //Copia os dados que estão na entidade cid para interface
12        jTextField1.setText(cid.getNome());
13        jTextField2.setText(""+cid.getPopulacao());
14        jComboBox1.setSelectedItem(cid.getUf());
15    }
16
17    public void interfaceEntidade(){ //Copia os dados que estão na interface para entidade cid
18        cid.setNome(jTextField1.getText());
19        cid.setUf(""+jComboBox1.getSelectedItem());
20        cid.setPopulacao(Long.parseLong(jTextField2.getText()));
21    }
22
23    /** Creates new form DetalhesCidade */
24    public DetalhesCidade(java.awt.Frame parent, boolean modal) {
25        super(parent, modal);
26        initComponents();
27    }
```

Projeto "ExemploCadastro"

Classe "DetalhesCidade"

JButton "Confirmar e Cancelar" - Event ActionPerformed

```
141 [-] private void jbtnConfirmadoActionPerformed(java.awt.event.ActionEvent evt) {  
142     confirmado = true;  
143     setVisible(false);  
144 }
```

```
136 [-] private void jbtnCancelarActionPerformed(java.awt.event.ActionEvent evt) {  
137  
138     setVisible(false);  
139 }
```

Projeto "ExemploCadastro"

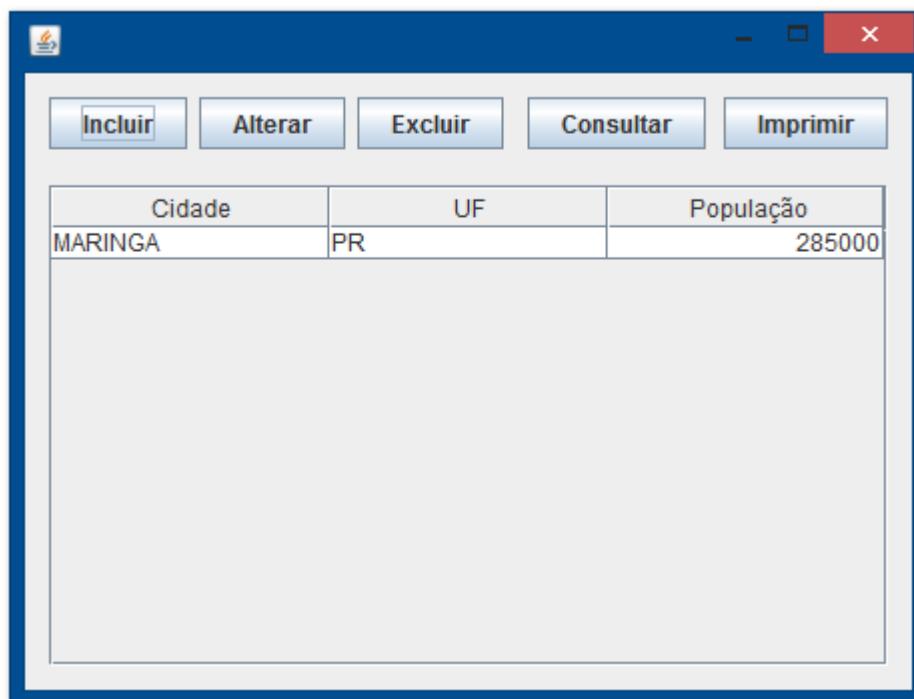
Classe “DetalhesCidade”

The screenshot shows a Java Swing application window. At the top, there is a title bar with the text "Nome". Inside the window, there is a JComboBox containing the items "JF", "PR", and "População". Below the JComboBox is a JTextField. At the bottom of the window are two buttons: "Confirmar" on the left and "Cancelar" on the right. Below this window, there is a properties panel titled "jComboBox1 [JComboBox] - Properties". The "Properties" tab is selected. The properties listed are:

Property	Value
background	[255,255,255]
editable	False
font	Tahoma 11 Plain
foreground	[0,0,0]
maximumRowCount	8
model	PR, SC, RS, SP, RJ, MG, MS, MT
selectedIndex	0
selectedItem	PR
toolTipText	

Projeto "ExemploCadastro"

Classe “CadastroCidades”



Projeto "ExemploCadastro"

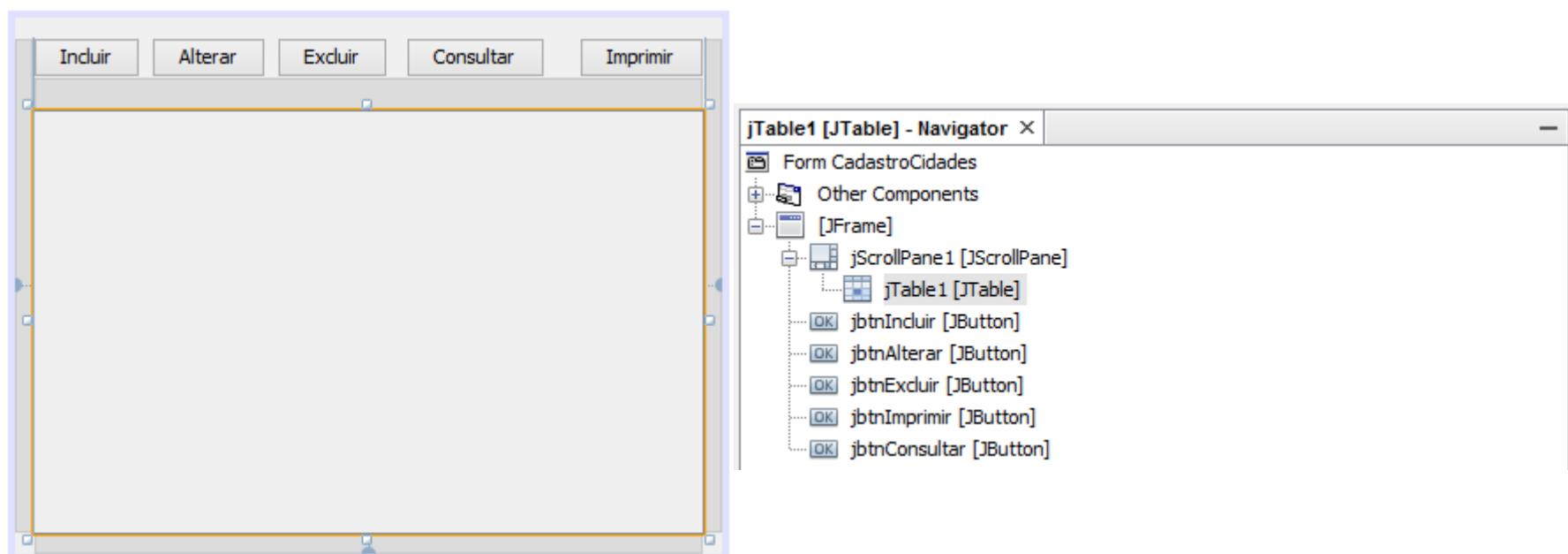
Classe “CadastroCidades”

```
1  package exemplodaogenerico;
2
3  import exemplodaogenerico.DAO;
4  import entidades.Cidade;
5  import javax.swing.JOptionPane;
6
7  public class CadastroCidades extends javax.swing.JFrame {
8
9
10     private DAO<Cidade> dao;
11
12     /** Creates new form CadastroCidades */
13     public CadastroCidades() {
14         dao = new DAO<Cidade>(Cidade.class);
15
16         initComponents();
17     }
18 }
```

Projeto "ExemploCadastro"

Classe "CadastroCidades"

JTable Vinculado a "TMCidade"



Projeto "ExemploCadastro"

Classe "CadastroCidades"

JTable Vinculado a Classe "TMCidade"

39

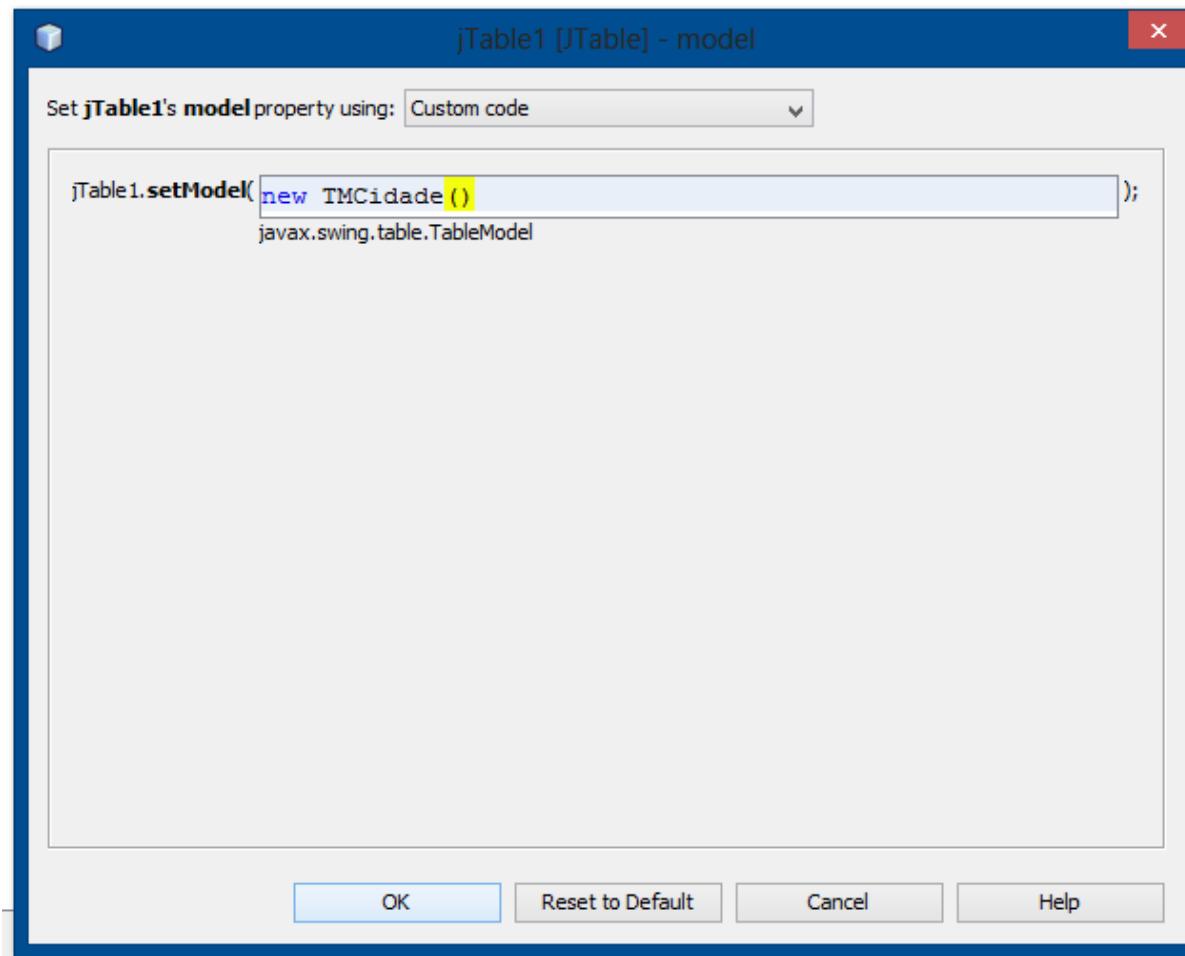
jTable1 [JTable] - Properties X

Properties	Binding	Events	Code
autoCreateColumnsFromModel	<input checked="" type="checkbox"/>		
autoCreateRowSorter	<input type="checkbox"/>		
background	<input type="color"/> [255,255,255]		
border	(No Border)		
font	Tahoma 11 Plain		
foreground	<input type="color"/> [0,0,0]		
model	<User Code>		
toolTipText			
Other Properties			
UIClassID	TableUI		
alignmentX	0.5		
alignmentY	0.5		
autoResizeMode	SUBSEQUENT_COLUMNS	▼	
autoscrolls	<input checked="" type="checkbox"/>		
baselineResizeBehavior	CONSTANT_ASCENT	▼	
cellEditor	<none>	▼	
cellSelectionEnabled	<input type="checkbox"/>		
columnCount	0		
columnModel	[TableColumnModel]		
model	<small>(javax.swing.table.TableModel) The model that is the source of the data for this view.</small>		

Projeto "ExemploCadastro"

Classe “CadastroCidades”

JTable Vinculado a Classe “TMCidade”



Projeto "ExemploCadastro"

Classe TableModel “TMCidade”

```
1  package exemplodaaogenerico;
2
3  import entidades.Cidade;
4  import java.util.List;
5  import javax.swing.event.TableModelListener;
6  import javax.swing.table.TableModel;
7
8  public class TMCidade implements TableModel {
9
10
11     private DAO<Cidade> dao;
12     private List<Cidade> lista;
13
14     public TMCidade() {
15         dao = new DAO<Cidade>(Cidade.class);
16         lista = dao.list();
17     }
18
19     public TMCidade(String filtro){
20         dao = new DAO<Cidade>(Cidade.class);
21         Cidade cid=new Cidade();
22         cid.setNome(filtro);
23         lista = dao.findByExamplePartial(cid);
24     }
25 }
```

Projeto "ExemploCadastro"

Classe TableModel “TMCidade”

Métodos Parte 1

```
① 1  public int getRowCount() { //Retorna quantas linhas tem na tabela
26
27      return lista.size();
28  }
29
① 1  public int getColumnCount() { //Retorna quantas colunas tem na tabela
31
32      return 3; //NOME, UF e POPULACAO
33
34  }
35
① 1  public String getColumnName(int columnIndex) { // Retorna o nome das colunas
37
38      if (columnIndex == 0) {
39          return "Cidade";
40      } else if (columnIndex == 1) {
41          return "UF";
42      } else {
43          return "População";
44      }
45
46  }
```

Projeto "ExemploCadastro"

Classe TableModel “TMCidade”

Métodos Parte 2

```
48     // Retorna a classe das colunas
49     public Class<?> getColumnClass(int columnIndex) {
50         if (columnIndex == 0) {
51             return String.class;
52         } else if (columnIndex == 1) {
53             return String.class;
54         } else {
55             return Long.class;
56         }
57     }
58
59     //Retorna se cada uma das células são ou não editáveis
60     public boolean isCellEditable(int rowIndex, int columnIndex) {
61         return false;
62     }
63
64     //Método mais importante
65     //Retorna qual é o valor ou o conteúdo de cada uma das células
66     public Object getValueAt(int rowIndex, int columnIndex) {
67         if (columnIndex == 0) {
68             return lista.get(rowIndex).getNome();
69         } else if (columnIndex == 1) {
70             return lista.get(rowIndex).getUf();
71         } else {
72             return lista.get(rowIndex).getPopulacao();
73         }
74     }
75
76     //Método que é executado quando o usuário alterar uma célula
77     public void setValueAt(Object aValue, int rowIndex, int columnIndex) {
78     }
```

Projeto "ExemploCadastro"

Classe “CadastroCidades”

JButton “Consultar” – Event ActionPerformed

```
108  [-] private void jbtnConsultarActionPerformed(java.awt.event.ActionEvent evt) {  
109      // TODO add your handling code here:  
110      String f = JOptionPane.showInputDialog ("Filtro");  
111      jTable1.setModel(new TMCidade(f));  
112      jTable1.updateUI();  
113 }
```

Projeto "ExemploCadastro"

Classe "CadastroCidades"

JButton "Incluir" – Event ActionPerformed

```
115  [-] private void jbtnIncluirActionPerformed(java.awt.event.ActionEvent evt) {  
116      // TODO add your handling code here:  
117      DetalhesCidade dc = new DetalhesCidade(this, true);  
118      dc.setCid(new Cidade());  
119      dc.setVisible(true);  
120      if (dc.isConfirmado()) {  
121          dc.interfaceEntidade();  
122          dao.insert(dc.getCid());  
123          jTable1.setModel(new TMCidade());  
124          jTable1.updateUI();  
125      }  
126  }
```

Projeto "ExemploCadastro"

Classe "CadastroCidades"

JButton "Alterar" – Event ActionPerformed

```
128  [-] private void jbtnAlterarActionPerformed(java.awt.event.ActionEvent evt) {  
129      // TODO add your handling code here:  
130      DetalhesCidade dc = new DetalhesCidade(this, true);  
131      int selecionado = jTable1.getSelectedRow();  
132      if (selecionado != -1) {  
133          TMCidade tmc = (TMCidade) jTable1.getModel();  
134          dc.setCid(tmc.getElemento(selecionado));  
135          dc.entidadeInterface();  
136          dc.setVisible(true);  
137          if (dc.isConfirmado()) {  
138              dc.interfaceEntidade();  
139              dao.update(dc.getCid());  
140              jTable1.setModel(new TMCidade());  
141              jTable1.updateUI();  
142          }  
143      }  
144  }  
145 }
```

Projeto "ExemploCadastro"

Classe “CadastroCidades”

JButton “Excluir” – Event ActionPerformed

```
147  private void jbtnExcluirActionPerformed(java.awt.event.ActionEvent evt) {  
148      // TODO add your handling code here:  
149      int selecionado = jTable1.getSelectedRow();  
150      if (selecionado != -1) {  
151          int resposta = JOptionPane.showConfirmDialog(this,  
152              "Deseja excluir o registro selecionado", "Confirmar", JOptionPane.YES_NO_OPTION);  
153  
154          if (resposta == JOptionPane.OK_OPTION) {  
155              TMCidade tmc = (TMCidade) jTable1.getModel();  
156              dao.delete(tmc.getElemento(selecionado));  
157              jTable1.setModel(new TMCidade());  
158              jTable1.updateUI();  
159          }  
160      }  
161  }
```

Projeto "ExemploCadastro"

Classe Principal “Main”

```
1 package exemplodaogenerico;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         new CadastroCidades().setVisible(true);
8
9     }
10 }
```

FIM
PROJETO "EXEMPLOCADASTRO"

Mais Exemplos

```
public class Livro {  
    private Long id;  
  
    private String nome;  
  
    private String descricao;  
  
    // getters e setters  
}
```

Mais Exemplos

```
@Entity  
public class Livro {  
  
    private Long id;  
  
    private String nome;  
  
    private String descricao;  
  
    // getters e setters  
}
```

Mais Exemplos

```
@Entity  
public class Livro {  
  
    @Id @GeneratedValue  
    private Long id;  
  
    private String nome;  
  
    private String descricao;  
  
    // getters e setters  
}
```

Mais Exemplos

```
@Entity  
public class Livro {  
  
    @Id @GeneratedValue  
    private Long id;  
  
    @Column(length=100)  
    private String nome;  
  
    private String descricao;  
  
    // getters e setters  
}
```

Mais Exemplos

```
Livro l = new Livro();
l.setNome("Alice no País das Maravilhas");
l.setDescricao("Um excelente livro");
```

Mais Exemplos

```
Livro l = new Livro();
l.setNome("Alice no País das Maravilhas");
l.setDescricao("Um excelente livro");

Session session = //....
```

Mais Exemplos

```
Livro l = new Livro();
l.setNome("Alice no País das Maravilhas");
l.setDescricao("Um excelente livro");

Session session = //.....

session.save(l);
```

Mais Exemplos

```
Livro l = new Livro();
l.setNome("Alice no País das Maravilhas");
l.setDescricao("Um excelente livro");

Session session = //.....
session.beginTransaction();
session.save(l);
session.getTransaction().commit();
```

Mais Exemplos

```
@Entity  
public class Autor {  
  
    @Id @GeneratedValue  
    private Long id;  
  
    private String nome;  
  
    private String sobrenome;  
  
    // getters e setters  
}
```

Mais Exemplos

```
@Entity  
public class Livro {  
  
    @Id @GeneratedValue  
    private Long id;  
  
    @Column(length=100)  
    private String nome;  
  
    @Lob  
    private String descricao;  
  
    // getters e setters  
}
```

Mais Exemplos

```
@Entity  
public class Livro {  
  
    @Id @GeneratedValue  
    private Long id;  
  
    @Column(length=100)  
    private String nome;  
  
    @Lob  
    private String descricao;  
  
    @ManyToOne  
    private Autor autor;  
  
    // getters e setters  
}
```

Mais Exemplos

```
Livro l = new Livro();
l.setNome("Alice no País das Maravilhas");
l.setDescricao("Um excelente livro");

Autor a = new Autor();
a.setNome("Lewis");
a.setSobrenome("Carroll");

l.setAutor(a);

Session session = //....
session.beginTransaction();
session.save(a);
session.save(l);
session.getTransaction().commit();
```

Mais Exemplos

```
@Entity  
public class Livro {  
  
    @Id @GeneratedValue  
    private Long id;  
  
    @Column(length=100)  
    private String nome;  
  
    @Lob  
    private String descricao;  
  
    @ManyToMany  
    private List<Autor> autores;  
  
    // getters e setters  
}
```

Dúvidas



Próxima Aula...

- ▶ Continuação do Desenvolvimento do Projeto do Trabalho da Disciplina...

Bibliografia

- ▶ LOPES, SÉRGIO. Persistência no Java – Hibernate e JPA (Slides). Caelum.
- ▶ CASCARROLHO, RODRIGO. JPA - Java Persistence API (Slides). 2009
- ▶ REIS, MARCO. Java Persistence API JPA (Slides). 2011.