



Linguagem de Programação Java

JDBC – Interação com o Banco de Dados

Profa. Joyce Miranda

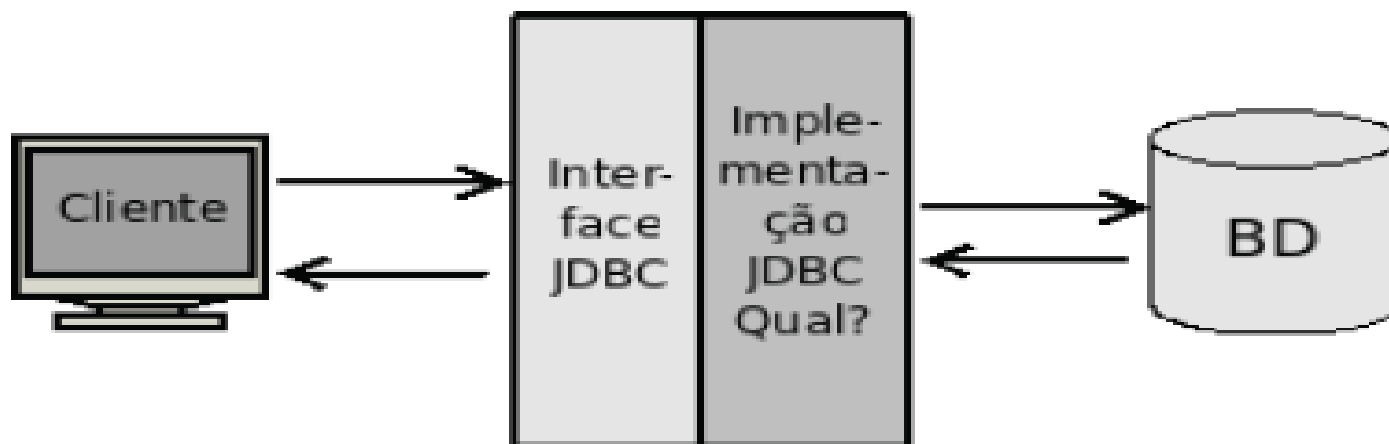
JDBC – Interação com o Banco de Dados

► JDBC

- Biblioteca JAVA padrão para persistência em banco de dados

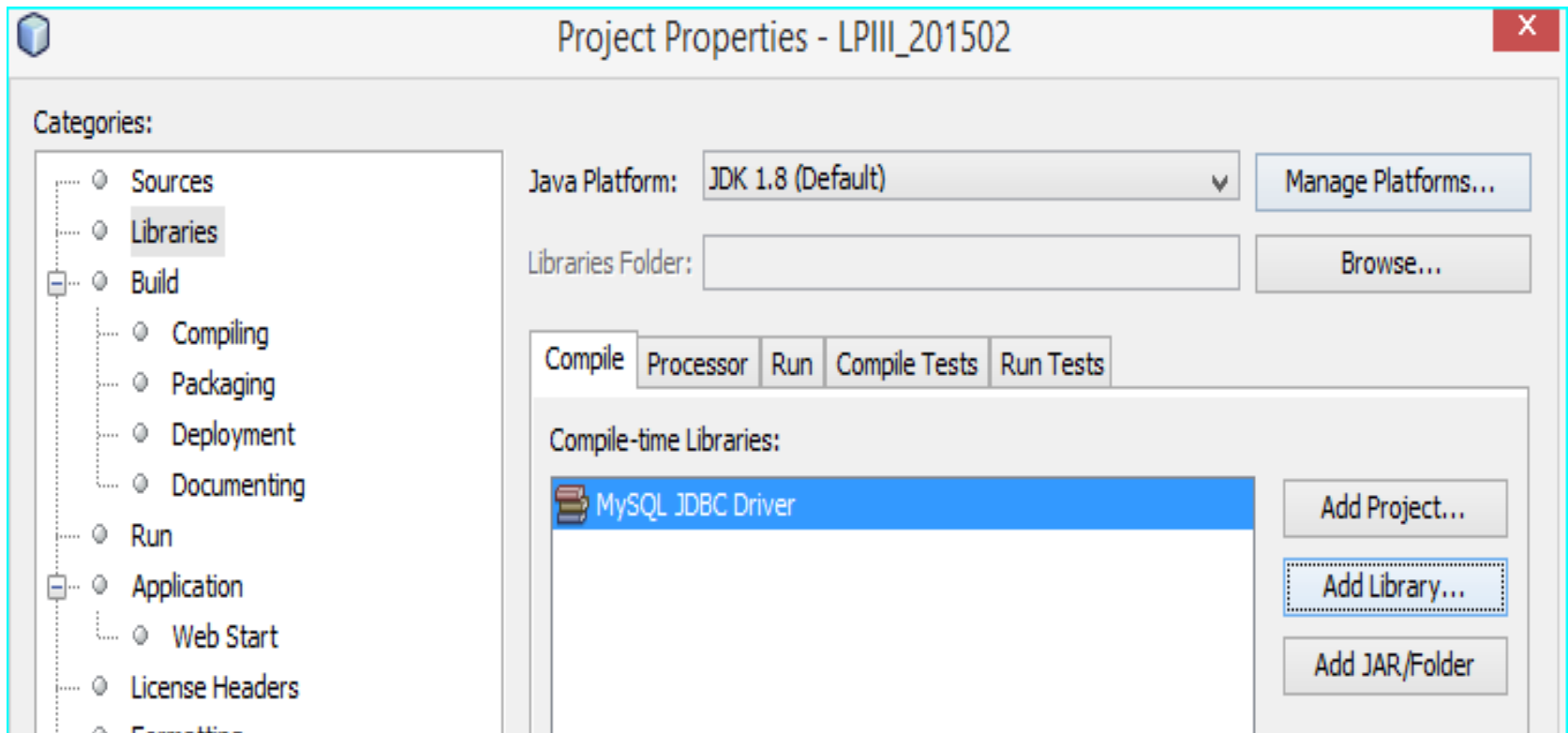
► *Driver*

- Implementa funcionalidades padrão que um banco de dados deve oferecer.



JDBC – Interação com o Banco de Dados

- ▶ O primeiro passo é adicionar o *Driver* de conexão do banco de dados ao projeto:



JDBC – Interação com o Banco de Dados

▶ **java.sql.***

▶ **DriverManager.getConnection(*parâmetros*)**

▶ Estabelece a conexão com o banco de dados.

▶ *Parâmetros*

□ URL padrão para acesso ao BD

▶ O padrão para o mysql é:

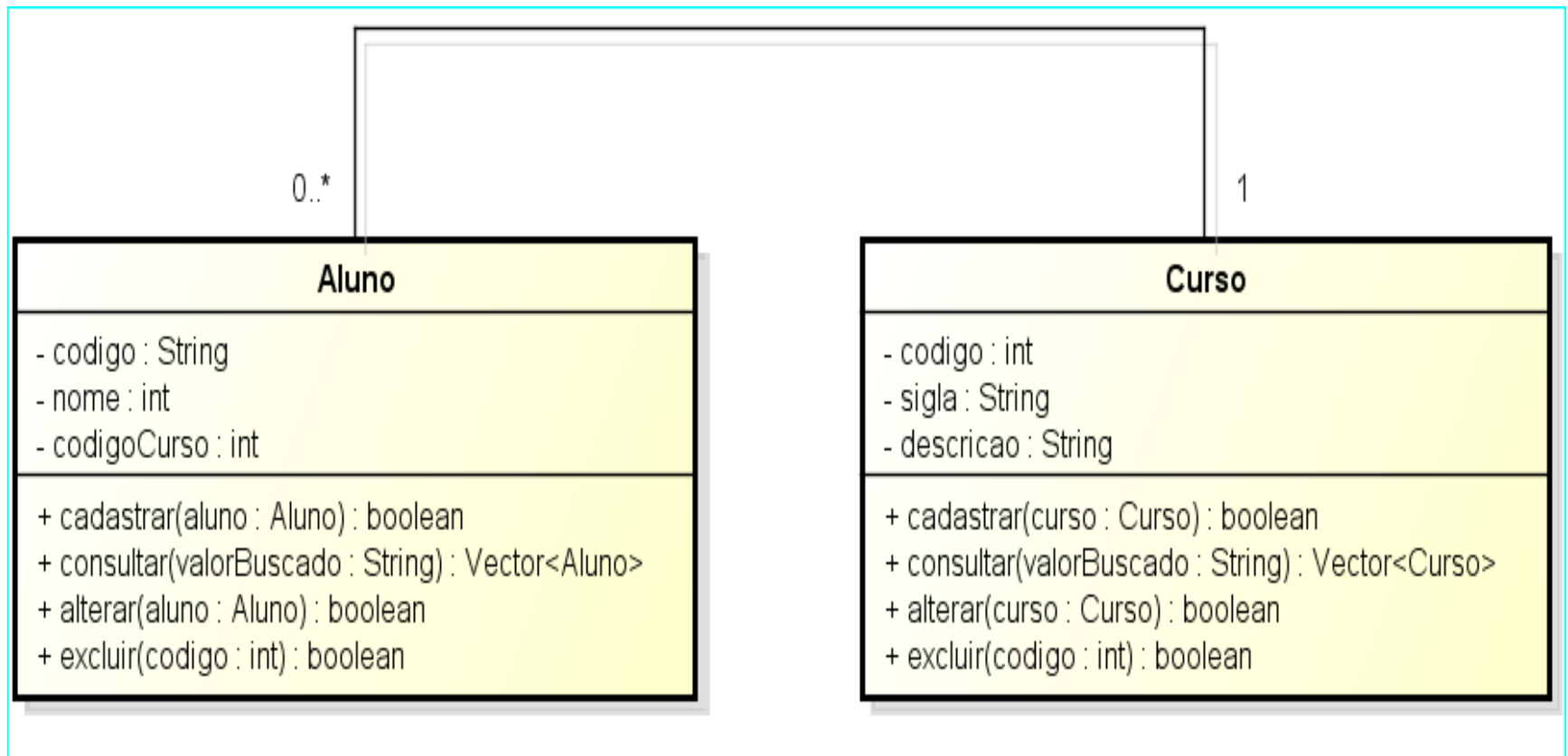
□ **jdbc:mysql://ip/banco**

□ **ip:** pelo endereço da máquina onde está o BD

□ **Banco:** nome do banco a ser utilizado.

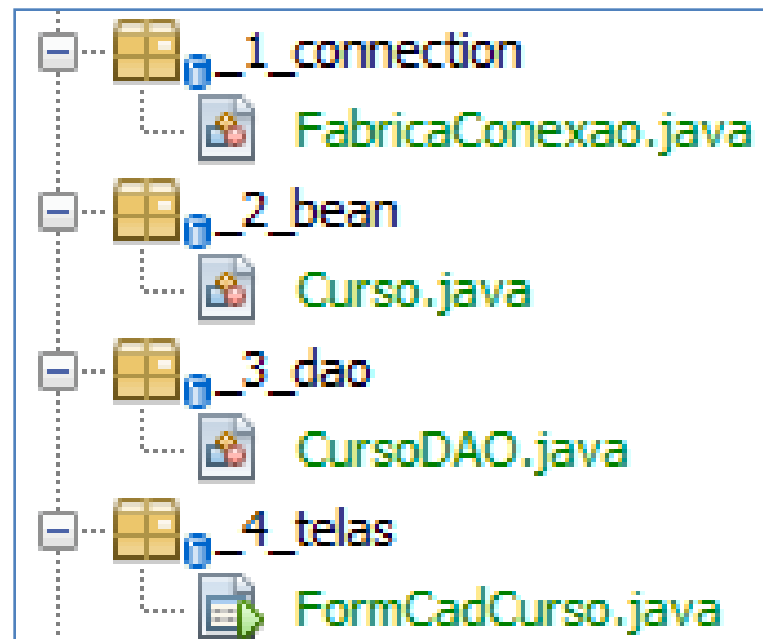
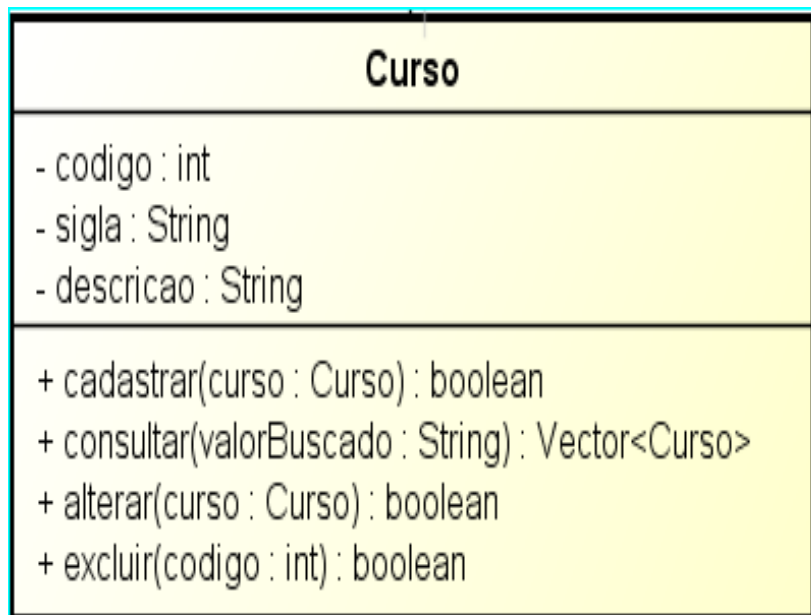
JDBC – Interação com o Banco de Dados

► Modelo



JDBC – Interação com o Banco de Dados

► Padrão DAO



► Código Disponível em

- https://github.com/joyceMiranda/classCodes/tree/master/LP_OO_GIT/src/conexaoBD

JDBC – Interação com o Banco de Dados

► connection.FabricaConexao

```
import java.sql.*;

public class FabricaConexao {

    public static Connection getConnection() {
        try{
            String host = "jdbc:mysql://localhost/yourDataBase";
            String user = "root";
            String password = "";
            return DriverManager.getConnection(
                host, user, password);

        }catch(SQLException e){
            throw new RuntimeException(e);
        }
    }
}
```

JDBC – Interação com o Banco de Dados

► bean.Curso

```
public class Curso {  
  
    private int codigo;  
    private String sigla;  
    private String descricao;  
  
    public Curso() { ...2 lines }  
    public Curso(int codigo, String sigla, String descricao)  
  
    public int getCodigo() { ...3 lines }  
    public void setCodigo(int codigo) { ...3 lines }  
  
    public String getSigla() { ...3 lines }  
    public void setSigla(String sigla) { ...3 lines }  
  
    public String getDescricao() { ...3 lines }  
    public void setDescricao(String descricao) { ...3 lines }  
}
```


JDBC – Interação com o Banco de Dados

► dao.CursorDAO

```
public class CursorDAO {

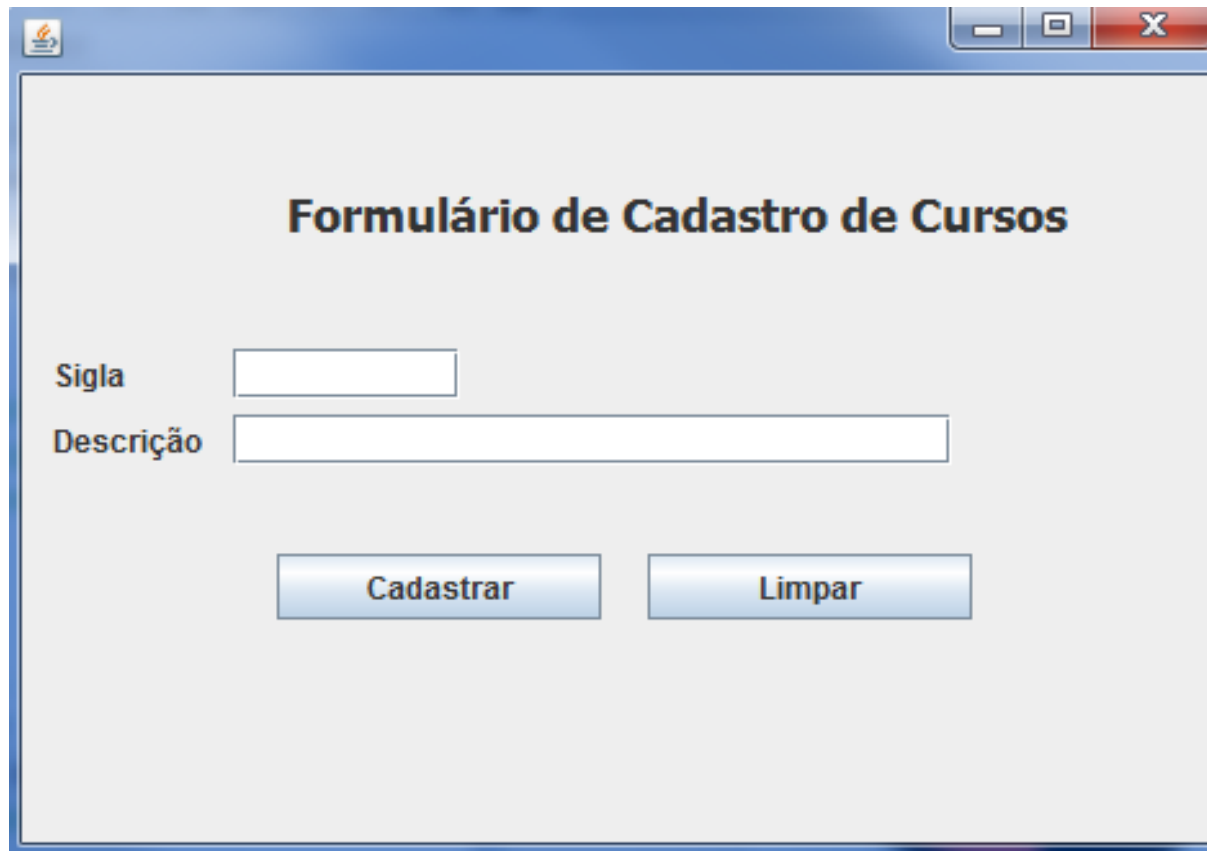
    private Connection connection;

    public CursorDAO() {
        this.connection = FabricaConexao.getConnection();
    }

    public boolean cadastrar(Curso curso) {
        String sql = "INSERT INTO CURSO VALUES (0, ?, ?)";
        try {
            PreparedStatement ps = connection.prepareStatement(sql);
            ps.setString(1, curso.getSigla());
            ps.setString(2, curso.getDescricao());
            ps.execute();
            connection.close();
            return true;
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}
```

JDBC – Interação com o Banco de Dados

- ▶ `telas.FormCadCurso`



The image shows a Java Swing window titled "Formulário de Cadastro de Cursos". The window has a standard Windows-style title bar with a minimize button, a maximize button, and a close button. The main content area is light gray and contains the following elements:

- A label "Sigla" followed by a single-line text input field.
- A label "Descrição" followed by a multi-line text input field.
- Two buttons at the bottom: "Cadastrar" and "Limpar", both with a blue gradient and a 3D effect.

JDBC – Interação com o Banco de Dados

► telas.FormCadCurso

```
private void btnCadastrarActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Curso curso = new Curso();  
    curso.setDescricao(txtDescricao.getText());  
    curso.setSigla(txtSigla.getText());  
  
    CursoDAO dao = new CursoDAO();  
    dao.cadastrar(curso);  
}
```

JDBC – Interação com o Banco de Dados

Formulário de Consulta de Cursos

Descrição

Sigla	Descrição

Curso
<div>- codigo : int</div> <div>- sigla : String</div> <div>- descricao : String</div>
<div>+ cadastrar(curso : Curso) : boolean</div> <div>+ consultar(valorBuscado : String) : Vector<Curso></div> <div>+ alterar(curso : Curso) : boolean</div> <div>+ excluir(codigo : int) : boolean</div>

```
public Vector<Curso> consultar(String valorBuscado){  
    String sql = "SELECT * FROM CURSO c WHERE c.descricao LIKE ? ";  
    try{  
        PreparedStatement ps = connection.prepareStatement(sql);  
        ps.setString(1, '%' + valorBuscado + '%');  
  
        ResultSet rs = ps.executeQuery();  
  
        Vector listaCursos = new Vector();  
  
        while(rs.next()){  
            int codigo = rs.getInt("idCurso"); /** nome do campo no BD **/  
            String sigla = rs.getString("sigla");  
            String descricao = rs.getString("descricao");  
            Curso curso = new Curso(codigo, sigla, descricao);  
            listaCursos.add(curso);  
        }  
        ps.close();  
        connection.close();  
        return listaCursos;  
    }catch (SQLException e) {  
        throw new RuntimeException(e);  
    }  
}
```

```
public Vector<Curso> consultar(String valorBuscado) {  
    String sql = "SELECT * FROM CURSO c WHERE c.descricao LIKE ? ";  
    try{  
        PreparedStatement ps = connection.prepareStatement(sql);  
        ps.setString(1, '%' + valorBuscado + '%');  
  
        ResultSet rs = ps.executeQuery();  
  
        Vector listaCursos = new Vector();  
  
        while(rs.next()){  
            int codigo = rs.getInt("idCurso"); /** nome do campo no BD **/  
            String sigla = rs.getString("sigla");  
            String descricao = rs.getString("descricao");  
            Curso curso = new Curso(codigo, sigla, descricao);  
            listaCursos.add(curso);  
        }  
        ps.close();  
        connection.close();  
        return listaCursos;  
    }catch (SQLException e) {  
        throw new RuntimeException(e);  
    }  
}
```

```
public Vector<Curso> consultar(String valorBuscado) {  
    String sql = "SELECT * FROM CURSO c WHERE c.descricao LIKE ? ";  
    try{  
        PreparedStatement ps = connection.prepareStatement(sql);  
        ps.setString(1, '%' + valorBuscado + '%');  
  
        ResultSet rs = ps.executeQuery();  
  
        Vector listaCursos = new Vector();  
  
        while(rs.next()){  
            int codigo = rs.getInt("idCurso"); /** nome do campo no BD **/  
            String sigla = rs.getString("sigla");  
            String descricao = rs.getString("descricao");  
            Curso curso = new Curso(codigo, sigla, descricao);  
            listaCursos.add(curso);  
        }  
        ps.close();  
        connection.close();  
        return listaCursos;  
  
    }catch (SQLException e) {  
        throw new RuntimeException(e);  
    }  
}
```

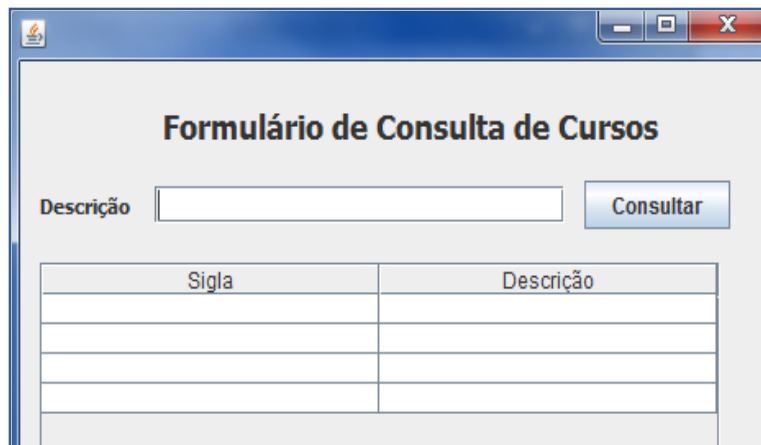
```
public Vector<Curso> consultar(String valorBuscado) {  
    String sql = "SELECT * FROM CURSO c WHERE c.descricao LIKE ? ";  
    try{  
        PreparedStatement ps = connection.prepareStatement(sql);  
        ps.setString(1, '%' + valorBuscado + '%');  
  
        ResultSet rs = ps.executeQuery();  
  
        Vector listaCursos = new Vector();  
  
        while(rs.next()){  
            int codigo = rs.getInt("idCurso"); /** nome do campo no BD **/  
            String sigla = rs.getString("sigla");  
            String descricao = rs.getString("descricao");  
            Curso curso = new Curso(codigo, sigla, descricao);  
            listaCursos.add(curso);  
        }  
  
        ps.close();  
        connection.close();  
        return listaCursos;  
  
    }catch (SQLException e) {  
        throw new RuntimeException(e);  
    }  
}
```



```
public Vector<Curso> consultar(String valorBuscado) {  
    String sql = "SELECT * FROM CURSO c WHERE c.descricao LIKE ? ";  
    try{  
        PreparedStatement ps = connection.prepareStatement(sql);  
        ps.setString(1, '%' + valorBuscado + '%');  
  
        ResultSet rs = ps.executeQuery();  
  
        Vector listaCursos = new Vector();  
  
        while(rs.next()){  
            int codigo = rs.getInt("idCurso"); /** nome do campo no BD **/  
            String sigla = rs.getString("sigla");  
            String descricao = rs.getString("descricao");  
            Curso curso = new Curso(codigo, sigla, descricao);  
            listaCursos.add(curso);  
        }  
        ps.close();  
        connection.close();  
        return listaCursos;  
    }catch (SQLException e) {  
        throw new RuntimeException(e);  
    }  
}
```

JDBC – Interação com o Banco de Dados

- ▶ Esquema preenchimento tabela da interface



Formulário de Consulta de Cursos

Descrição

Sigla	Descrição

1. Chamar método **consultar()**
2. Montar o **conjuntoLinhas**
3. Montar **conjuntoColunas**;
4. Adicionar **conjuntoLinhas** e **conjuntoColunas** à tabela;

```
//APLICANDO MODELO NA TABELA
DefaultTableModel modelo =
    new DefaultTableModel(conjuntoLinhas, conjuntoColunas);
tblCursos.setModel(modelo);
```

```

private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here
    CursoDAO dao = new CursoDAO();
    Vector<Curso> listaCursos = dao.consultar(txtValorBuscado.getText());
    Vector conjuntoLinhas = new Vector();

    for(Curso curso: listaCursos){
        Vector linha = new Vector();

        linha.add(curso.getCodigo());
        linha.add(curso.getSigla());
        linha.add(curso.getDescricao());

        conjuntoLinhas.add(linha);
    }

    Vector conjuntoColunas = new Vector();
    conjuntoColunas.add("Código");
    conjuntoColunas.add("Sigla");
    conjuntoColunas.add("Descrição");

    DefaultTableModel modeloTabela =
        new DefaultTableModel(conjuntoLinhas, conjuntoColunas);
    tblCursos.setModel(modeloTabela);
}

```

1. Chamar método consultar()

```
private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here  
    CursoDAO dao = new CursoDAO();  
    Vector<Curso> listaCursos = dao.consultar(txtValorBuscado.getText());  
    Vector conjuntoLinhas = new Vector();  
  
    for(Curso curso: listaCursos){  
        Vector linha = new Vector();  
  
        linha.add(curso.getCodigo());  
        linha.add(curso.getSigla());  
        linha.add(curso.getDescricao());  
  
        conjuntoLinhas.add(linha);  
    }  
}
```

2. Montar o conjuntoLinhas

```
Vector conjuntoColunas = new Vector();  
conjuntoColunas.add("Código");  
conjuntoColunas.add("Sigla");  
conjuntoColunas.add("Descrição");  
  
DefaultTableModel modeloTabela =  
    new DefaultTableModel(conjuntoLinhas, conjuntoColunas);  
tblCursos.setModel(modeloTabela);  
}
```

```
private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here  
    CursoDAO dao = new CursoDAO();  
    Vector<Curso> listaCursos = dao.consultar(txtValorBuscado.getText());  
    Vector conjuntoLinhas = new Vector();  
  
    for(Curso curso: listaCursos){  
        Vector linha = new Vector();  
  
        linha.add(curso.getCodigo());  
        linha.add(curso.getSigla());  
        linha.add(curso.getDescricao());  
  
        conjuntoLinhas.add(linha);  
    }  
}
```

```
Vector conjuntoColunas = new Vector();  
conjuntoColunas.add("Código");  
conjuntoColunas.add("Sigla");  
conjuntoColunas.add("Descrição");
```

3. Montar conjuntoColunas;

```
DefaultTableModel modeloTabela =  
    new DefaultTableModel(conjuntoLinhas, conjuntoColunas);  
tblCursos.setModel(modeloTabela);
```

```
}
```

```
private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here
    CursoDAO dao = new CursoDAO();
    Vector<Curso> listaCursos = dao.consultar(txtValorBuscado.getText());
    Vector conjuntoLinhas = new Vector();

    for(Curso curso: listaCursos){
        Vector linha = new Vector();

        linha.add(curso.getCodigo());
        linha.add(curso.getSigla());
        linha.add(curso.getDescricao());

        conjuntoLinhas.add(linha);
    }

    Vector conjuntoColunas = new Vector();
    conjuntoColunas.add("Código");
    conjuntoColunas.add("Sigla");
    conjuntoColunas.add("Descrição");
```

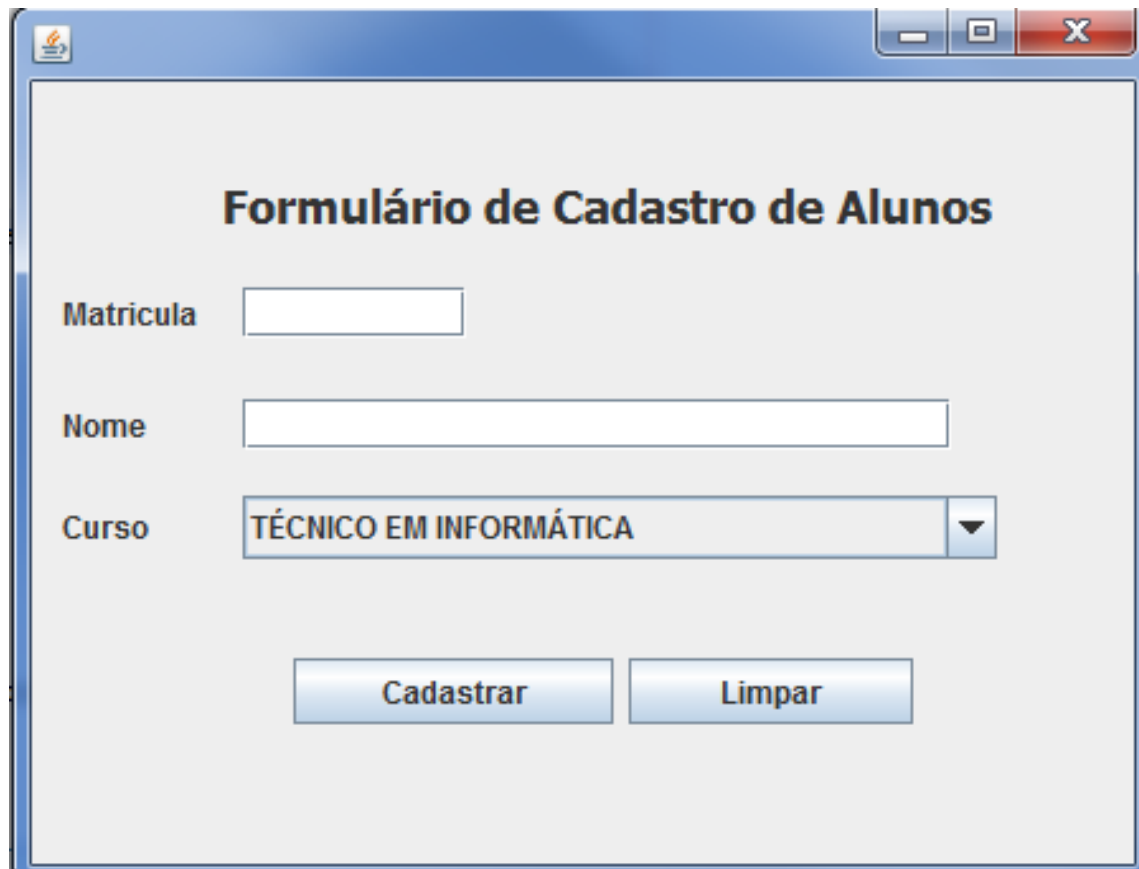
4. Adicionar conjuntoLinhas e conjuntoColunas à tabela;

```
DefaultTableModel modeloTabela =
    new DefaultTableModel(conjuntoLinhas, conjuntoColunas);
tblCursos.setModel(modeloTabela);
```

```
}
```

JDBC – Interação com o Banco de Dados

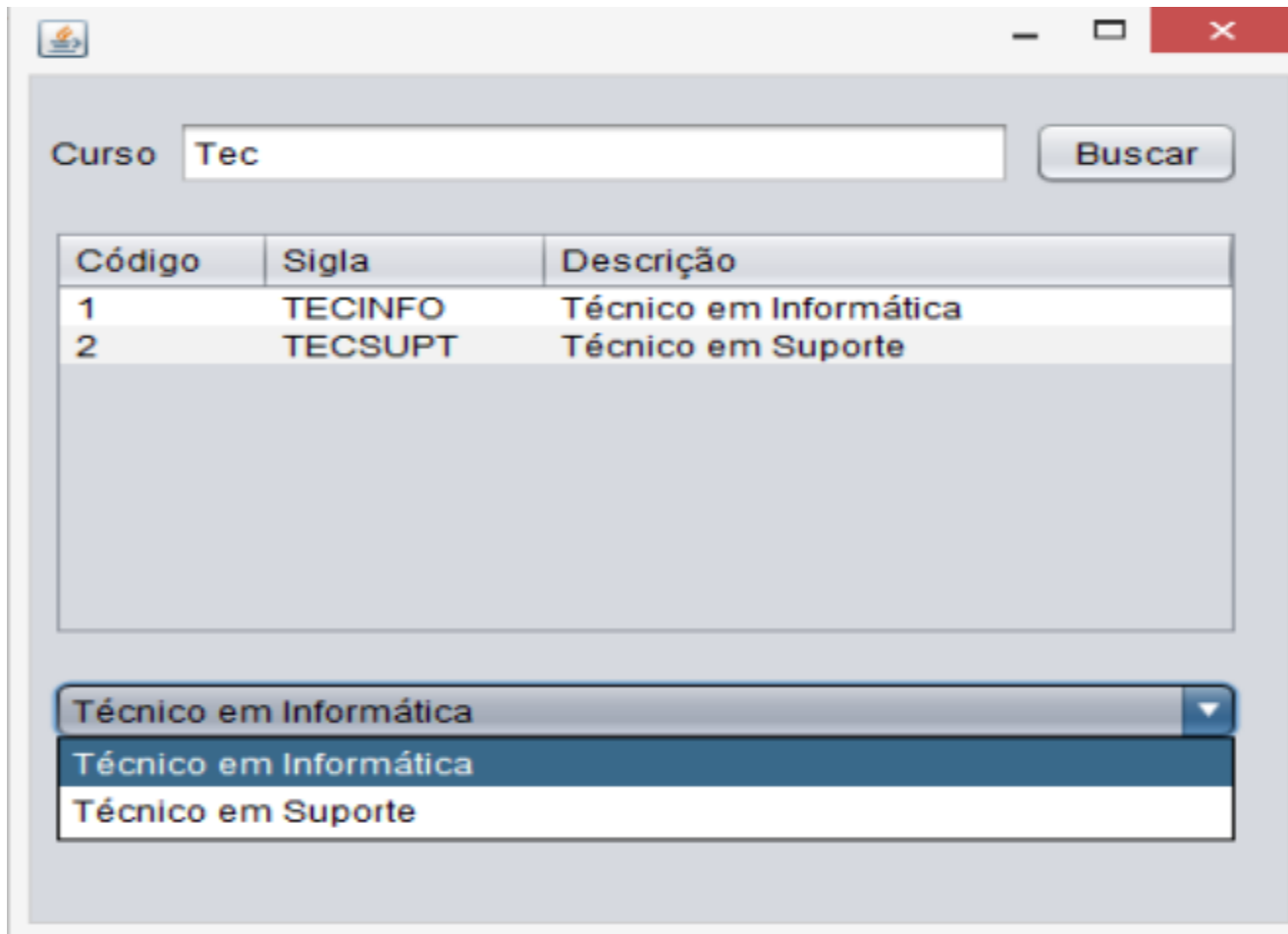
► Preenchendo ComboBox



The image shows a Java Swing window titled "Formulário de Cadastro de Alunos". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area is light gray and contains three labeled text fields: "Matricula" (a single-line text box), "Nome" (a single-line text box), and "Curso" (a ComboBox). The "Curso" ComboBox currently displays "TÉCNICO EM INFORMÁTICA" and has a downward arrow on its right side. Below the text fields are two buttons: "Cadastrar" and "Limpar".

JDBC – Interação com o Banco de Dados

► Preenchendo ComboBox



Curso

Código	Sigla	Descrição
1	TECINFO	Técnico em Informática
2	TECSUPT	Técnico em Suporte

Técnico em Informática
Técnico em Informática
Técnico em Suporte

JDBC – Interação com o Banco de Dados

► Preenchendo ComboBox

- Este código deve estar dentro do **construtor** da classe da **tela** (interface gráfica) logo abaixo do método *initComponents()*

- Ex: Construtor da classe FormBuscaCursos

```
Vector<Curso> listaCursos = null;  
  
public FormBuscaCursos() {  
  
    initComponents();  
  
    CursoDAO dao = new CursoDAO();  
    listaCursos = dao.consultar("");  
  
    DefaultComboBoxModel modeloCombo =  
        new DefaultComboBoxModel(listaCursos);  
    cboCursos.setModel(modeloCombo);  
}
```

* O método *toString()* deve ser incluído no *bean* Curso

```
public String toString() {  
    return this.descricao;  
}
```

JDBC – Interação com o Banco de Dados

- ▶ Preenchendo ComboBox
 - ▶ Recuperando informações
 - ▶ Exemplo

```
int indiceSelecioneado = cboCursos.getSelectedIndex();  
Curso curso = listaCursos.get(indiceSelecioneado);  
int codigo = curso.getCodigo();  
System.out.println("Código Curso selecionado: " + codigo);
```

JDBC – Interação com o Banco de Dados

► Visualizar/Alterar/Excluir Informações

Usuário
<ul style="list-style-type: none">- codigo : int- nome : String- login : String- senha : String
<ul style="list-style-type: none">+ cadastrar() : boolean+ consultar(nome : String) : Vector+ alterar() : boolean+ excluir() : boolean

powered by Astah

JDBC – Interação com o Banco de Dados

► Visualizar/Alterar/Excluir Informações

Curso

Código	Sigla	Descrição
5	TECINFO	TECNICO EM INFO

FormDetalhesCurso

Código

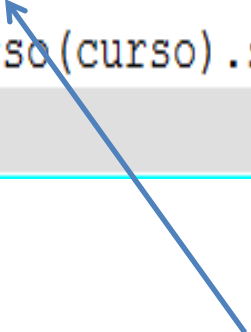
Sigla

Descricao

JDBC – Interação com o Banco de Dados

- ▶ Visualizar/Alterar/Excluir Informações
 - ▶ Evento Click do Mouse da Tabela

```
private void tblCursosMouseClicked(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
    int indice = tblCursos.getSelectedRow();  
    Curso curso = listaCursos.get(indice);  
    new FormDetalhesCurso(curso).setVisible(true);  
}
```



Lista retornada do método *consultar* deve ser declarada como variável global (como atributo da classe de interface gráfica)

JDBC – Interação com o Banco de Dados

- ▶ Visualizar/Alterar/Excluir Informações
 - ▶ Incluindo novo Construtor da classe **FormDetalhesCurso**

```
public FormDetalhesCurso() {  
    initComponents();  
}  
  
public FormDetalhesCurso(Curso curso) {  
    initComponents();  
    txtCodigo.setText(Integer.toString(curso.getCodigo()));  
    txtSigla.setText(curso.getSigla());  
    txtDescricao.setText(curso.getDescricao());  
}
```

JDBC – Interação com o Banco de Dados

```
public boolean alterar(Curso curso){
    String sql = "UPDATE CURSO SET sigla=?, descricao=? WHERE idCurso=?";
    try {
        PreparedStatement ps = connection.prepareStatement(sql);
        ps.setString(1, curso.getSigla());
        ps.setString(2, curso.getDescricao());
        ps.setInt(3, curso.getCodigo());
        ps.execute();
        connection.close();
        return true;
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

► Método “alterar”
USUARIODAO

JDBC – Interação com o Banco de Dados

► Adicionando evento ao botão 'Alterar'

```
private void btnAlterarActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Curso curso = new Curso(Integer.parseInt(txtCodigo.getText()),  
                             txtSigla.getText(),  
                             txtDescricao.getText());  
    CursoDAO dao = new CursoDAO();  
    boolean excluiu = dao.alterar(curso);  
    if(excluiu){  
        JOptionPane.showMessageDialog(null, "Alteração realizada com sucesso!!");  
    }else{  
        JOptionPane.showMessageDialog(null, "Alteração não realizada!!");  
    }  
}
```


JDBC – Interação com o Banco de Dados

```
public boolean excluir(int idCurso){  
    String sql = "DELETE FROM CURSO WHERE idCurso=?";  
    try {  
        PreparedStatement ps = connection.prepareStatement(sql);  
        ps.setInt(1, idCurso);  
        ps.execute();  
        connection.close();  
        return true;  
    } catch (SQLException e) {  
        throw new RuntimeException(e);  
    }  
}
```

► Método “excluir”
USUARIODAO

JDBC – Interação com o Banco de Dados

► Adicionando evento ao botão 'Excluir'

```
private void btnExcluirActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    CursoDAO dao = new CursoDAO();  
    boolean excluiu = dao.excluir(Integer.parseInt(txtCodigo.getText()));  
    if(excluiu){  
        JOptionPane.showMessageDialog(null, "Exclusão realizada com sucesso!!");  
    }else{  
        JOptionPane.showMessageDialog(null, "Exclusão não realizada!!");  
    }  
    this.dispose();  
}
```

JDBC - Extras

- ▶ Imagem armazenadas em disco

```
ImageIcon img = new ImageIcon(getClass().getResource("/imgs/logoIfam.gif"));  
jLabel3.setIcon(img);
```