



# SIN143 Laboratório de Programação

---

Prof. João Batista Ribeiro

[joao42ibatista@gmail.com](mailto:joao42ibatista@gmail.com)



---

Universidade Federal de Viçosa

---



# JDBC

---

- *Java Database Connectivity - JDBC*
- *Conjunto de classes para **integração** da linguagem Java com bancos de dados relacionais.*
- *API que possibilita a adição de comandos SQL em uma aplicação Java, suportando as funcionalidades básicas do SQL e possibilitando a integração do código Java com sistemas de banco de dados.*



# JDBC

---

- Acessa praticamente qualquer banco de dados relacional (ER).
- JDBC é uma API, ou seja, apenas especificação.
- Implementado através de um *driver* fornecido pelo fabricante.
- Pacote base: `java.sql`



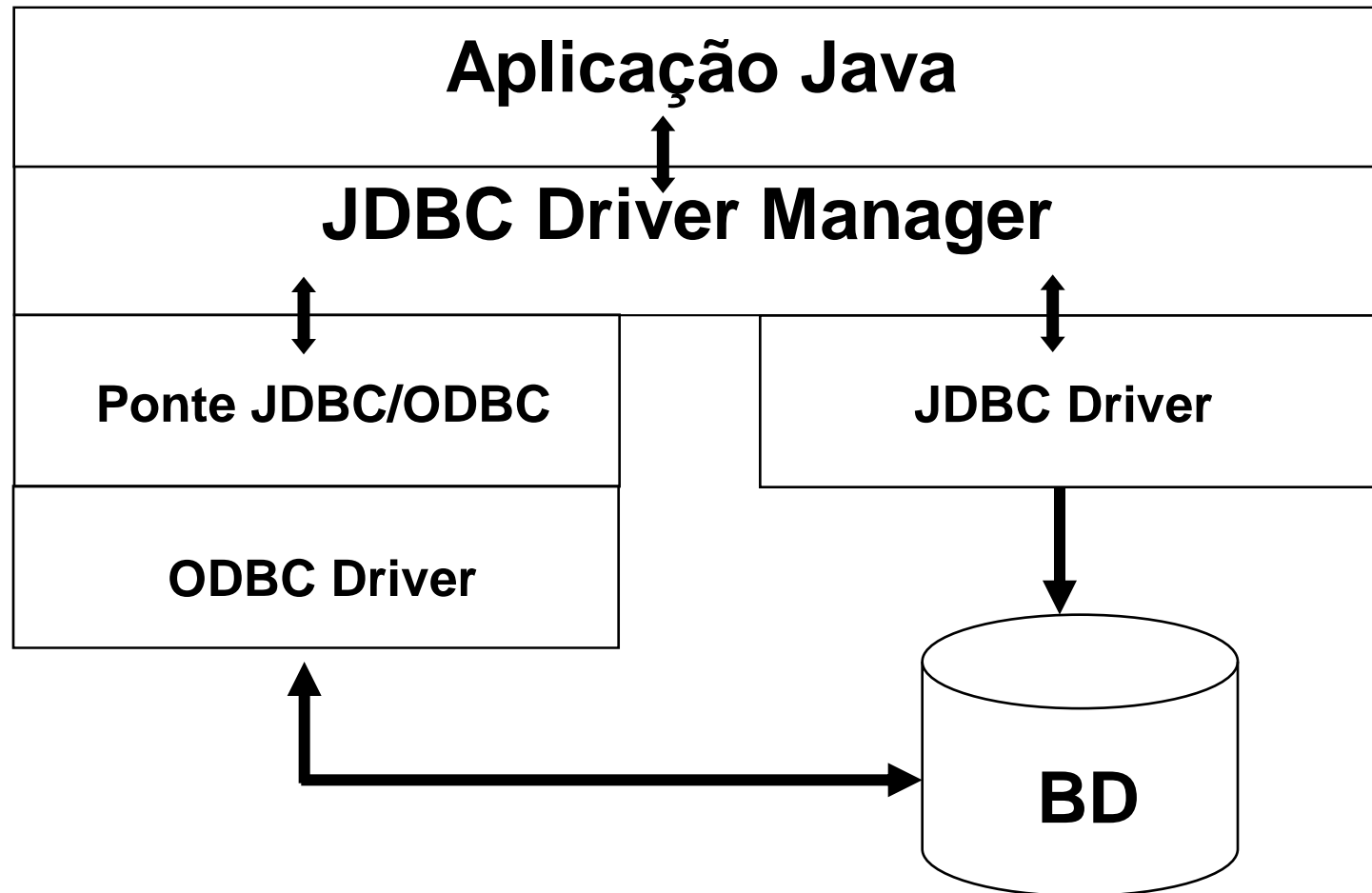
# JDBC API

---

- JDBC API é um conjunto de interfaces que são implementados pelos *drivers* dos fornecedores de banco de dados.
- O fato de utilizar interfaces garante a independência da aplicação em relação aos bancos de dados, garantindo a portabilidade.



# Estrutura JDBC x ODBC



# Driver JDBC

- Cada banco de dados tem sua implementação para a API JDBC. Chamamos essa implementação de *driver*
- O nome do *Driver* do MySQL é connectorJ
- Baixe o driver do site: <https://www.mysql.com/products/connector/>

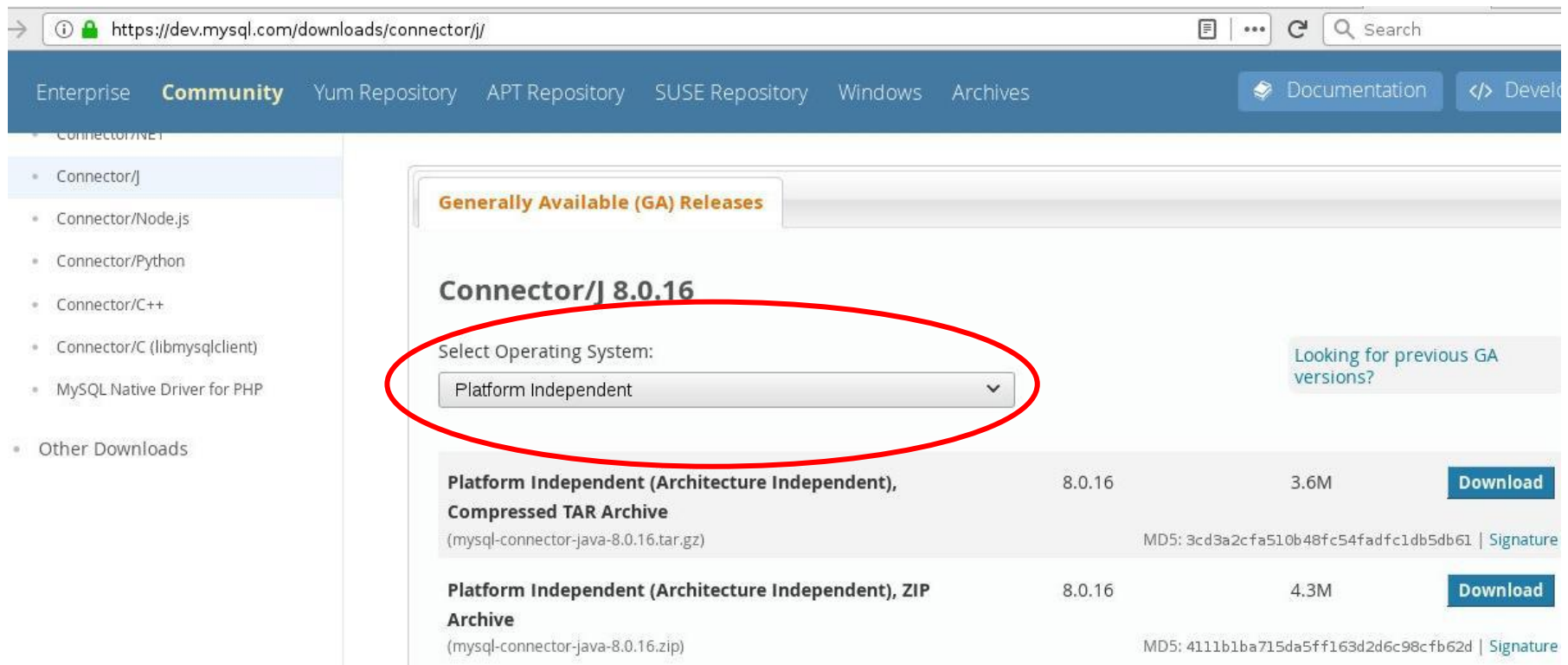


The screenshot shows the MySQL Connectors page. The left sidebar lists various MySQL products, with 'MySQL Enterprise Edition' expanded. The main content area is titled 'MySQL Connectors' and describes the standards-based drivers for JDBC, ODBC, and .Net. A table lists the available drivers, with the 'JDBC Driver for MySQL (Connector/J)' row highlighted by a red border.

Developed by MySQL	
ADO.NET Driver for MySQL (Connector/NET)	<a href="#">Download</a>
ODBC Driver for MySQL (Connector/ODBC)	<a href="#">Download</a>
<b>JDBC Driver for MySQL (Connector/J)</b>	<a href="#">Download</a>
Node.js Driver for MySQL (Connector/Node.js)	<a href="#">Download</a>

# Driver JDBC

- Extraia o arquivo compactado e procure pelo arquivo "mysql-connector-[versão\_atual].jar".



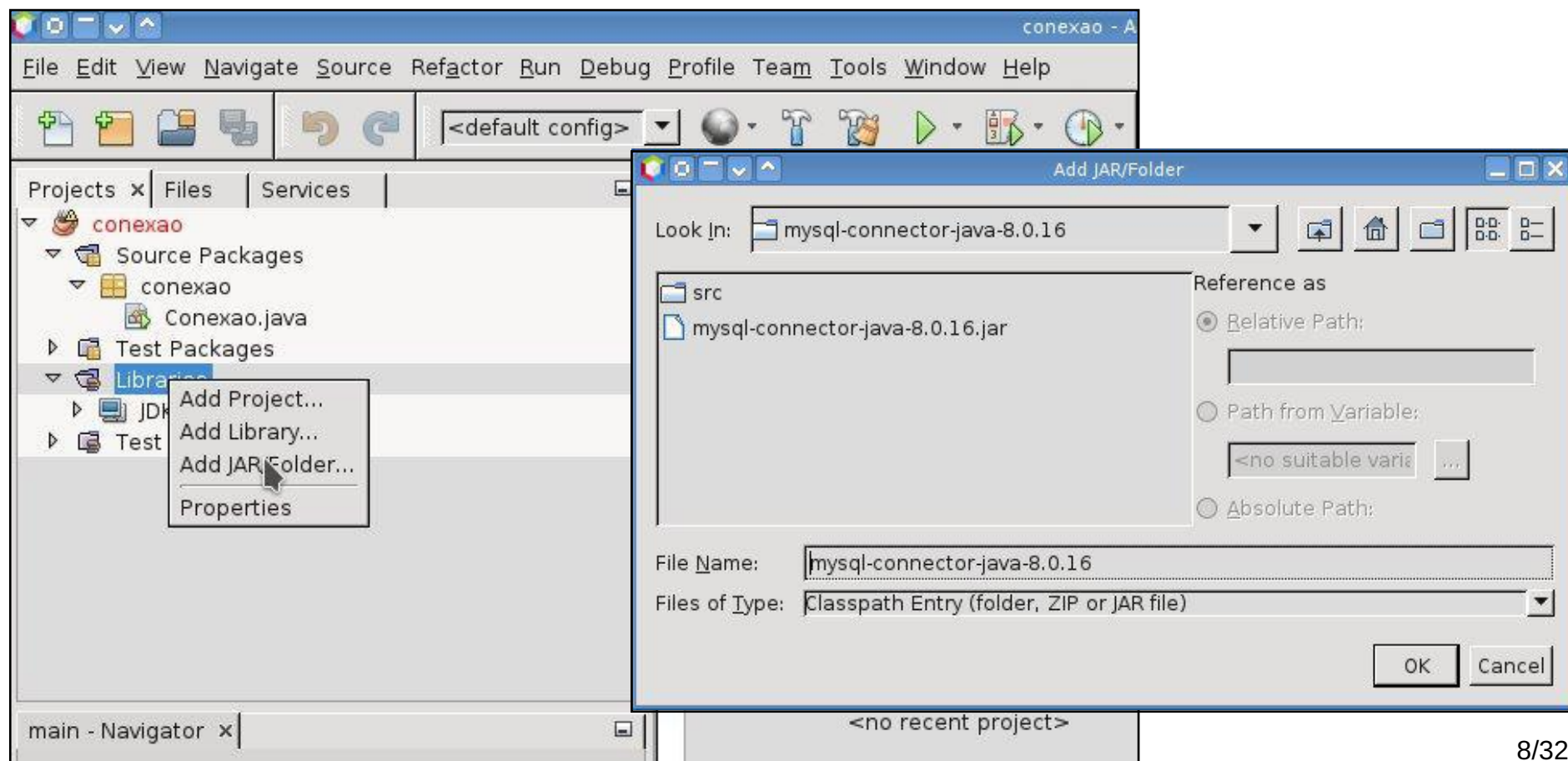
The screenshot shows the MySQL Connector/J download page. The browser address bar displays `https://dev.mysql.com/downloads/connector/j/`. The page has a blue header with navigation links: Enterprise, Community (selected), Yum Repository, APT Repository, SUSE Repository, Windows, and Archives. On the right of the header are links for Documentation and Developer. A left sidebar lists various connectors: Connector/NET, Connector/J (selected), Connector/Node.js, Connector/Python, Connector/C++, Connector/C (libmysqlclient), MySQL Native Driver for PHP, and Other Downloads. The main content area is titled 'Generally Available (GA) Releases' and features a section for 'Connector/J 8.0.16'. Below this, a 'Select Operating System:' dropdown menu is highlighted with a red circle, showing 'Platform Independent' as the selected option. To the right of this dropdown is a link: 'Looking for previous GA versions?'. Below the dropdown is a table of download options:

Download Option	Version	Size	Action
<b>Platform Independent (Architecture Independent), Compressed TAR Archive</b> (mysql-connector-java-8.0.16.tar.gz)	8.0.16	3.6M	<a href="#">Download</a>
<b>Platform Independent (Architecture Independent), ZIP Archive</b> (mysql-connector-java-8.0.16.zip)	8.0.16	4.3M	<a href="#">Download</a>

MD5 hashes and signature links are provided for each download option.

# Driver JDBC

- Adicione o arquivo .jar à sua aplicação.







# Elementos Básicos

---

- *Driver* : fornecido pelo fabricante do banco de dados, é específico para cada SGBD.
- *DriverManager* : responsável pela criação da conexão.
- *Connection* : representa uma conexão física com o SGBD.
- *Statement / PreparedStatement* : comando executado no SGBD (insert, delete, update).
- *ResultSet* : resultado da consulta a um banco de dados.



# Parâmetros da Conexão

---

- Para realização do acesso à um determinado SGBD é necessário informar o local onde este está localizado.
- Basicamente os parâmetros necessários são:
  - *Url* : protocolo e endereço do SGBD na rede.
  - *Driver* : nome do driver que você vai usar para acessar o banco
  - *User* : nome do usuário.
  - *Password* : senha do usuário.



# URL de conexão

---

- Usada para criar a conexão com a base de dados.
- Geralmente contém o nome do servidor, porta, protocolo, subprotocolo, database.
- É específico para cada banco de dados, sendo assim, você deve verificar a documentação.



# *Driver* de conexão

---

- A interface *Driver* é utilizada pelo mecanismo interno do JDBC (suas implementações).
- É obrigatório carregá-lo, pois só assim sua aplicação saberá como interpretar os comandos SQL.
- Exemplo:
  - `Class.forName("com.mysql.jdbc.Driver");`
  - `Class.forName("com.mysql.cj.jdbc.Driver");`



# Connection

---

RDBMS	Formato de URL de banco de dados
MySQL	<code>jdbc:mysql:// nomeDoHost:númeroDePorta/nomeDoBancoDeDados</code>
ORACLE	<code>jdbc:oracle:thin:@ nomeDoHost:númeroDePorta:nomeDoBancoDeDados</code>
DB2	<code>jdbc:db2: nomeDoHost:númeroDePorta/nomeDoBancoDeDados</code>
PostgreSQL	<code>jdbc:postgresql:// nomeDoHost:númeroDePorta/nomeDoBancoDeDados</code>
Java DB/Apache Derby	<code>jdbc:derby:nomeDoBancoDados</code> (incorporado) <code>jdbc:derby://nomeDoHost:númeroDaPorta/nomeDoBancoDeDados</code> (rede)
Microsoft SQL Server	<code>jdbc:sqlserver://nomeDoHost:númeroDaPorta;databaseName=nomeDoBancoDeDados</code>
Sybase	<code>jdbc:sybase:Tds: nomeDoHost:númeroDePorta/nomeDoBancoDeDados</code>



# *Connection*

---

- Representa uma conexão física com o SGBD.
- A sintaxe é fornecida pelo fabricante, mas todas são similares.
- Veja alguns exemplos:
  - jdbc:odbc:anuncios
  - jdbc:oracle:thin:@200.206.192.216:1521:exemplo
  - jdbc:mysql://alnitak.orion.org/clientes
  - jdbc:cloudscape:rmi://host:1098/MyDB;create=true
  
  - jdbc:mysql://localhost:3306/dbNome



# *DriverManager*

---

- O DriverManager manipula os Drivers informados anteriormente.
- Possui métodos para registrar um driver, remover ou listar todos.
- É utilizado para recuperar uma conexão baseando-se em parâmetros.
- Veja um exemplo:
  - ```
Connection con =  
DriverManager.getConnection("jdbc:mysql://nomeServidor/nomeDatabase",  
    "usuario", "senha");
```



# *Statement*

---

- Oferece meios para passar comandos ao SGBD.
  - `Statement stmt = con.createStatement();`
- Com o stmt podemos usar os métodos `execute()`, `executeQuery()`, `executeBatch()` e `executeUpdate()` para enviar instruções ao SGBD.
- Subinterfaces:
  - `PreparedStatement` (instruções SQL pré-compiladas) e `CallableStatement` (StoredProcedures).





# Executando um *Statement*

- Exemplo 1: Criando uma tabela pessoa.

```
stmt.execute("CREATE TABLE Pessoa"
+ " (id INT AUTO_INCREMENT"
+ " , nome varchar(50)"
+ " , endereco varchar(50)"
+ " , telefone varchar(20)"
+ " , cpf varchar(20)"
+ " , PRIMARY KEY(id))");
```

- Exemplo 2: Inserindo um registro na tabela criada.

```
stmt.executeUpdate("INSERT INTO"
+ " Pessoa (nome, endereco, telefone, cpf)"
+ " VALUES "
+ " ('Marco Antonio', 'CNB 14 LOTE 10', '33521134', '832869')");
```



# *ResultSet*

---

- O *ResultSet* encapsula o resultado de uma consulta a uma tabela.
- Métodos de navegação:
  - `next()`, `previous()`, `first()` e `last()`.
- Métodos para obter dados de uma coluna:
  - `getInt()`, `getString()`, `getDate()`, `getFloat()`, etc.



# Executando um *ResultSet*

---

```
String nomeDaPessoa;  
int idPessoa;  
String enderecoDaPessoa;
```

```
ResultSet rs = stmt.executeQuery("SELECT * FROM Pessoa");  
while (rs.next()) {  
    idPessoa = rs.getInt("id");  
    nomeDaPessoa = rs.getString("nome");  
    enderecoDaPessoa = rs.getString("endereco");  
    // Continua para todas as colunas  
  
    //Depois, faça algo com os valores obtidos, como mostrar numa tela  
    System.out.println("ID: " + idPessoa  
        + "Nome: " + nomeDaPessoa + "Endereco: " + enderecoDaPessoa);  
}
```



# *PreparedStatement*

---

- É um *Statement* pré-compilado que é mais eficiente quando temos o mesmo comando repetido várias vezes, mudando apenas os parâmetros.

Exemplo :

```
String sql = "INSERT INTO Pessoa (nome, endereco, telefone,"  
            + " cpf) values (?, ?, ?, ?)";  
PreparedStatement prepStmt = con.prepareStatement(sql);  
prepStmt.setString(1, "Marco Antonio");  
prepStmt.setString(2, "CNB 14 LOTE 10");  
prepStmt.setString(3, "33521134");  
prepStmt.setString(4, "832869");  
prepStmt.executeUpdate();
```



# Encerrando a conexão

---

- Lembre-se de fechar todos os recursos que você abriu para que outras pessoas possa utilizá-los.
- Exemplos:

```
rs.close ;
```

```
stmt.close ;
```

```
con.close ;
```



# Na prática – exemplo básico

---

- Criação da tabela de exemplo

```
CREATE TABLE tabela_01 (  
    tab01_indice int(10) unsigned NOT NULL AUTO_INCREMENT,  
    tab01_nome varchar(45) NOT NULL,  
    PRIMARY KEY (tab01_indice));
```



# Na prática – exemplo básico

---

- Definição da classe **Banco** e importação de pacotes para a classe:

```
//import java.sql.*;
```

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;
```

```
public class Banco {  
    ...  
}
```





# Na prática – exemplo básico

---

- Definição de atributos:

```
// login do Banco
private String login;
// senha do Banco
private String senha;
// IP do servidor do Banco
private String host;
// Nome do Banco de dados
private String dbname;
// URL de conexão do Banco
private String url;
// Objeto do tipo Connection para estabelecer a conexão
private Connection connection;
// Objeto Statement usado para enviar consultas para o Banco de dados
private Statement stmt;
// Objeto ResultSet utilizado nas consultas ao Banco de dados
private ResultSet rst;
// String utilizada para realizar as consultas ao Banco
private String query;
```





# Na prática – exemplo básico

---

- Definição do construtor padrão:

```
public Banco() {  
    login = "user2";  
    senha = "123456";  
    host = "localhost";  
    dbname = "exemploDB";  
    url = "jdbc:mysql://" + host + "/" + dbname;  
    conection = null;  
    stmt = null;  
}
```



# Na prática – exemplo básico

---

- Definição do método de conexão:

```
public boolean setConecta() {  
    try {  
        // Carrega o driver mysql  
        Class.forName("com.mysql.cj.jdbc.Driver");  
        // Estabelecendo a conexão  
        connection = DriverManager.getConnection(url, login, senha);  
        return true;  
    } catch (ClassNotFoundException | SQLException e) {  
        System.out.println("Error" + e);  
        return false;  
    }  
}
```



# Na prática – exemplo básico

---

- Definição do método para fechar a conexão:

```
public boolean setDesconecta() {  
    try {  
        connection.close();  
        return true;  
    } catch (SQLException e) {  
        System.out.println("Error: " + e);  
        return false;  
    }  
}
```



# Na prática – exemplo básico

---

- Exemplo:de:inserção:

```
public boolean insertExemplo() {  
    try {  
        // Instanciando o objeto Statement a partir do objeto  
        // Connection criado  
        stmt = conexao.createStatement();  
        // Query para inserção  
        query = "insert into tabela_01 (tab01_nome) values ('Jose')";  
        stmt.executeUpdate(query);  
        return true;  
    } catch(SQLException e) {  
        System.out.println("Error: " + e);  
        return false;  
    }  
}
```



# Na prática – exemplo básico

---

- Exemplo de exclusão:

```
public boolean deleteExemplo() {  
    try {  
        // Instanciando o objeto Statement a partir do  
        // objeto Connection criado  
        stmt = conection.createStatement();  
        // Query para deleção  
        query = "delete from tabela_01 where tab01_indice = 3";  
        stmt.executeUpdate(query);  
        return true;  
    } catch (SQLException e) {  
        System.out.println("Error: " + e);  
        return false;  
    }  
}
```



# Na prática – exemplo básico

---

- Exemplo de atualização:

```
public boolean updateExemplo() {  
    try {  
        // Instanciando o objeto Statement a partir do  
        // objeto Connection criado  
        stmt = conection.createStatement();  
        // Query para update  
        query = "update tabela_01 set tab01_nome='Jonas' where tab01_indice = 2";  
        stmt.executeUpdate(query);  
        return true;  
    } catch (SQLException e) {  
        System.out.println("Error: " + e);  
        return false;  
    }  
}
```



# Na prática – exemplo básico

- Exemplo de consulta:

```
public ResultSet selectExemplo() {  
    try {  
        // Instanciando o objeto Statement a partir do objeto  
        // Connection criado  
        stmt = conection.createStatement();  
        // Query para select do nome 'Jose'  
        //query = "select * from tabela_01 where tab01_nome = 'Jose'";  
        // Query para select de todos os nomes  
        query = "select * from tabela_01";  
        rst = stmt.executeQuery(query);  
        return rst;  
    } catch(SQLException e) {  
        System.out.println("Error: " + e);  
        return null;  
    }  
}
```



# Na prática – exemplo básico

- Classe de teste (principal):

```
import java.sql.ResultSet;
import java.sql.SQLException;

public class teste {
    public static void main(String[] args) {
        Banco bd = new Banco();
        System.out.println("Teste da conexão:");
        System.out.println(bd.setConecta());
        System.out.println("Inserir:");
        //System.out.println(bd.insertExemplo());
        System.out.println("Deletar:");
        //System.out.println(bd.deleteExemplo());
        System.out.println("Atualizar:");
        //System.out.println(bd.updateExemplo());
        System.out.println("Consulta:");
        ResultSet result = bd.selectExemplo();
        try {
            while (result.next()){
                System.out.println(result.getInt(1)+ " " + result.getString(2));
            }
        } catch (SQLException ex) {
            System.out.println("Error: " + ex);
        }
        System.out.println("Desconectar:");
        System.out.println(bd.setDesconecta());
    }
}
```