

# Linguagem Técnica de Programação Web

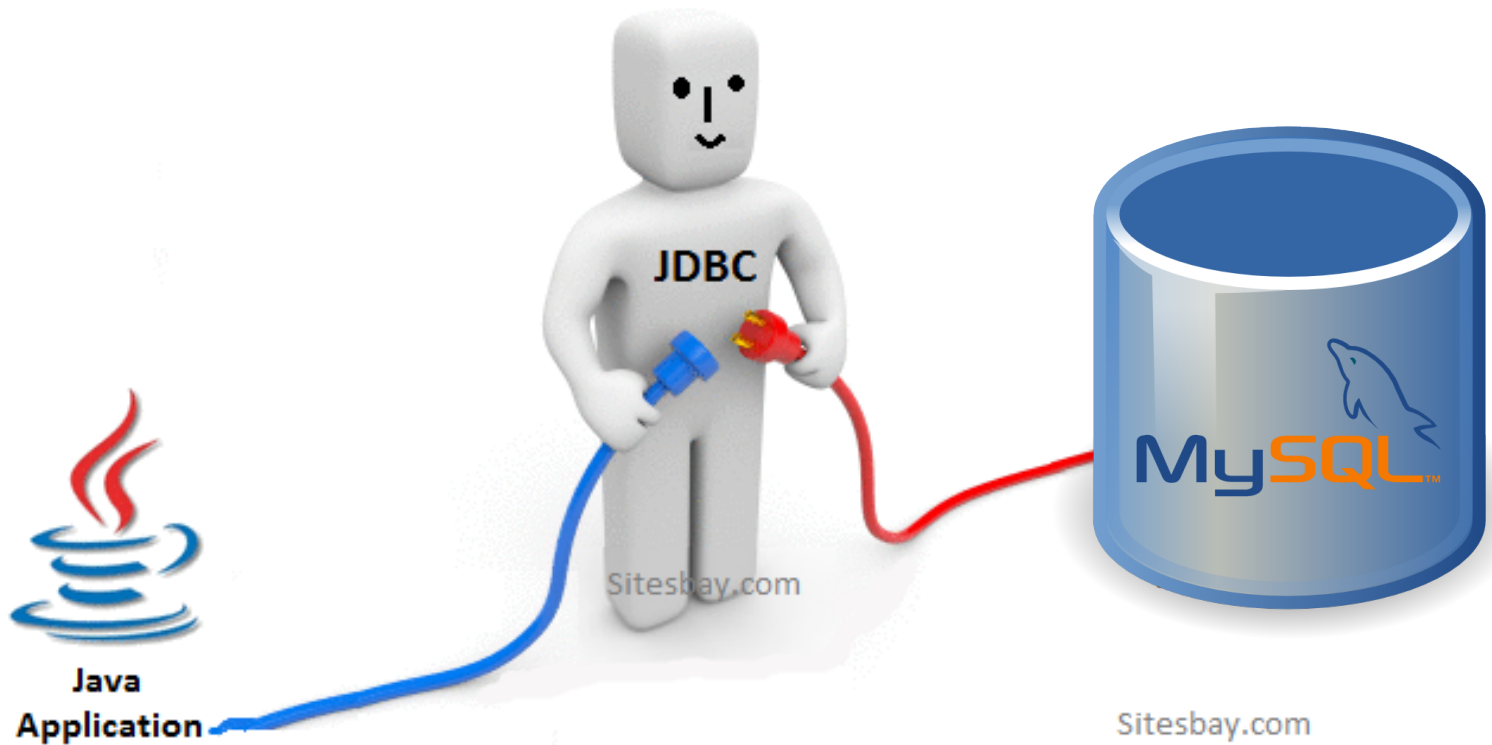
## Aula 4 – Introdução à utilização de banco de dados com JDBC

Prof. João Paulo Pimentel  
[joao.pimentel@projecao.br](mailto:joao.pimentel@projecao.br)

---

# JDBC e MySQL

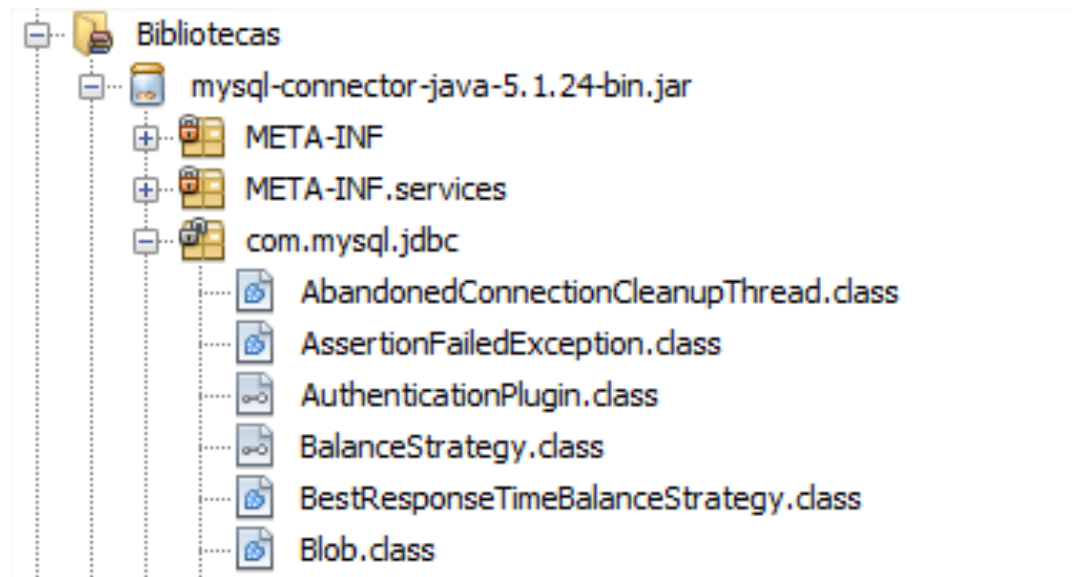
## JDBC - Java Database Connectivity



# JDBC e MySQL

## Definição de JDBC:

É um conjunto de classes e interfaces escritas em **Java** que fazem o envio de instruções SQL para qualquer banco de dados relacional.



# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans

### Ferramentas utilizadas:

- Netbeans IDE 7.2
  - Java 7
  - MySQL 5.5
  - Workbench 5.2
  - Driver JDBC 5.1
  - XAMPP v3.2.2
-

# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans

**1º Passo** - Vamos criar um banco de dados no MySQL utilizando a ferramenta Workbench, você pode utilizar a função assistente da ferramenta ou utilizar a seguinte query:

```
CREATE SCHEMA IF NOT EXISTS `Banco` DEFAULT  
CHARACTER SET utf8;
```

```
USE `Banco`;
```

---

# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans



```
CREATE TABLE Empregado (  
matricula SMALLINT NOT NULL,  
nome VARCHAR(50) NOT NULL,  
telefone VARCHAR(11),  
salario NUMERIC(15,2) NOT NULL,  
PRIMARY KEY (matricula)  
);
```

---

# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans

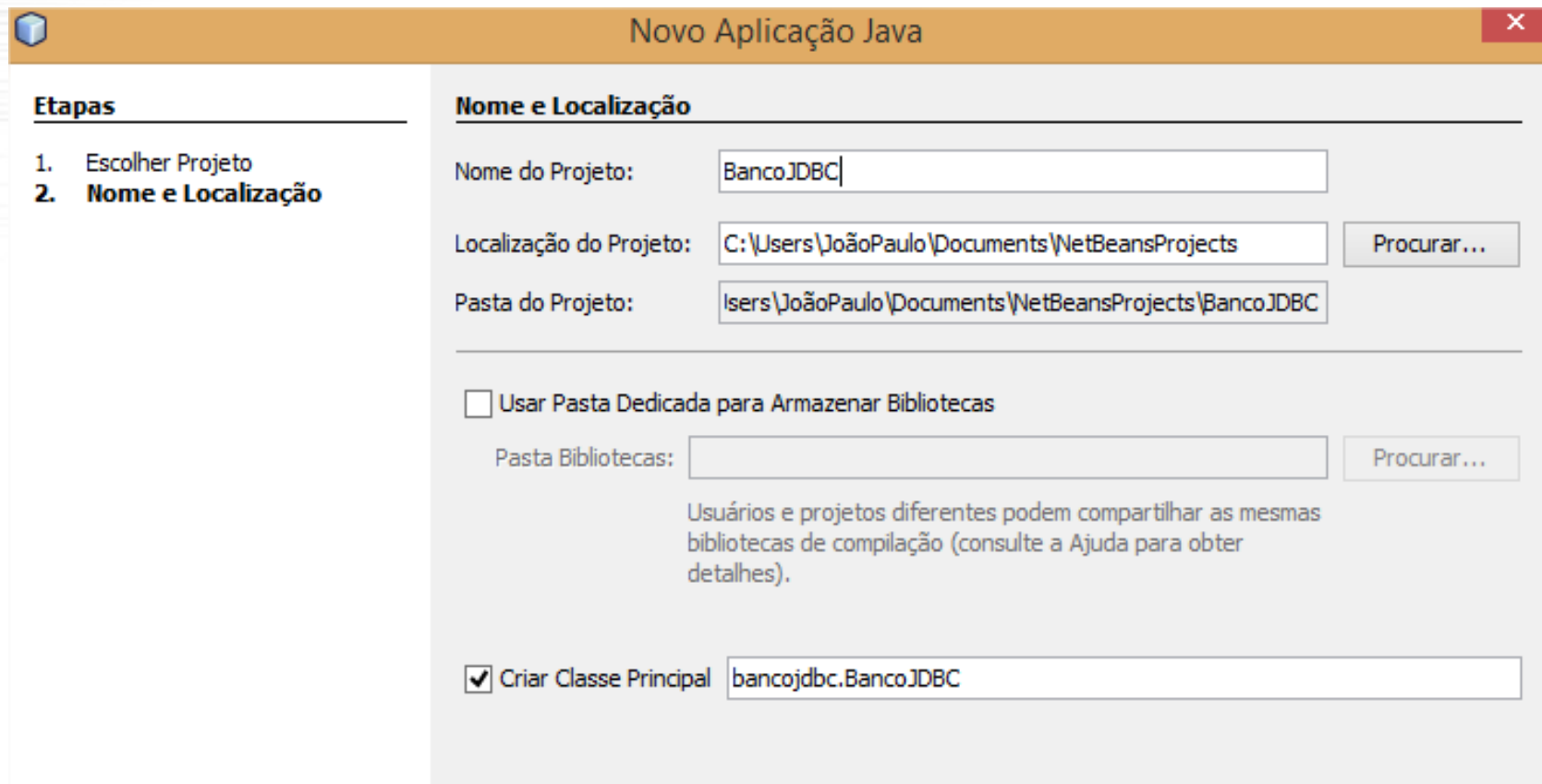
```
INSERT INTO Empregado VALUES  
(1, 'Joao', '61981192323', '15000.00');  
INSERT INTO Empregado VALUES  
(2, 'Jose', '61984192626', '35000.00');  
INSERT INTO Empregado VALUES  
(3, 'Paulo', '61985192886', '40000.00');  
  
SELECT * FROM Empregado;
```

Result Grid     Filter Rows: <input type="text"/>				
	matricula	nome	telefone	salario
	1	Joao	61981192323	15000.00
	2	Jose	61984192626	35000.00
	3	Paulo	61985192886	40000.00
	NULL	NULL	NULL	NULL

# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans

2º Passo: Abra o **Netbeans** e crie um novo projeto **java**:



**Novo Aplicação Java**

**Etapas**

1. Escolher Projeto
2. **Nome e Localização**

**Nome e Localização**

Nome do Projeto: BancoJDBC

Localização do Projeto: C:\Users\JoãoPaulo\Documents\NetBeansProjects Procurar...

Pasta do Projeto: Isern\JoãoPaulo\Documents\NetBeansProjects\BancoJDBC

☐ Usar Pasta Dedicada para Armazenar Bibliotecas

Pasta Bibliotecas: Procurar...

Usuários e projetos diferentes podem compartilhar as mesmas bibliotecas de compilação (consulte a Ajuda para obter detalhes).

☒ Criar Classe Principal bancojdbc.BancoJDBC



# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans

### 3º Passo:

→ Em propriedade do projeto a guia>Executar>Adicionar JAR/Pasta, adicione o driver connector JDBC do MySQL que você baixou do blog. Assim nossa aplicação poderá acessar o banco de dados.

---

# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans

### 3º Passo:

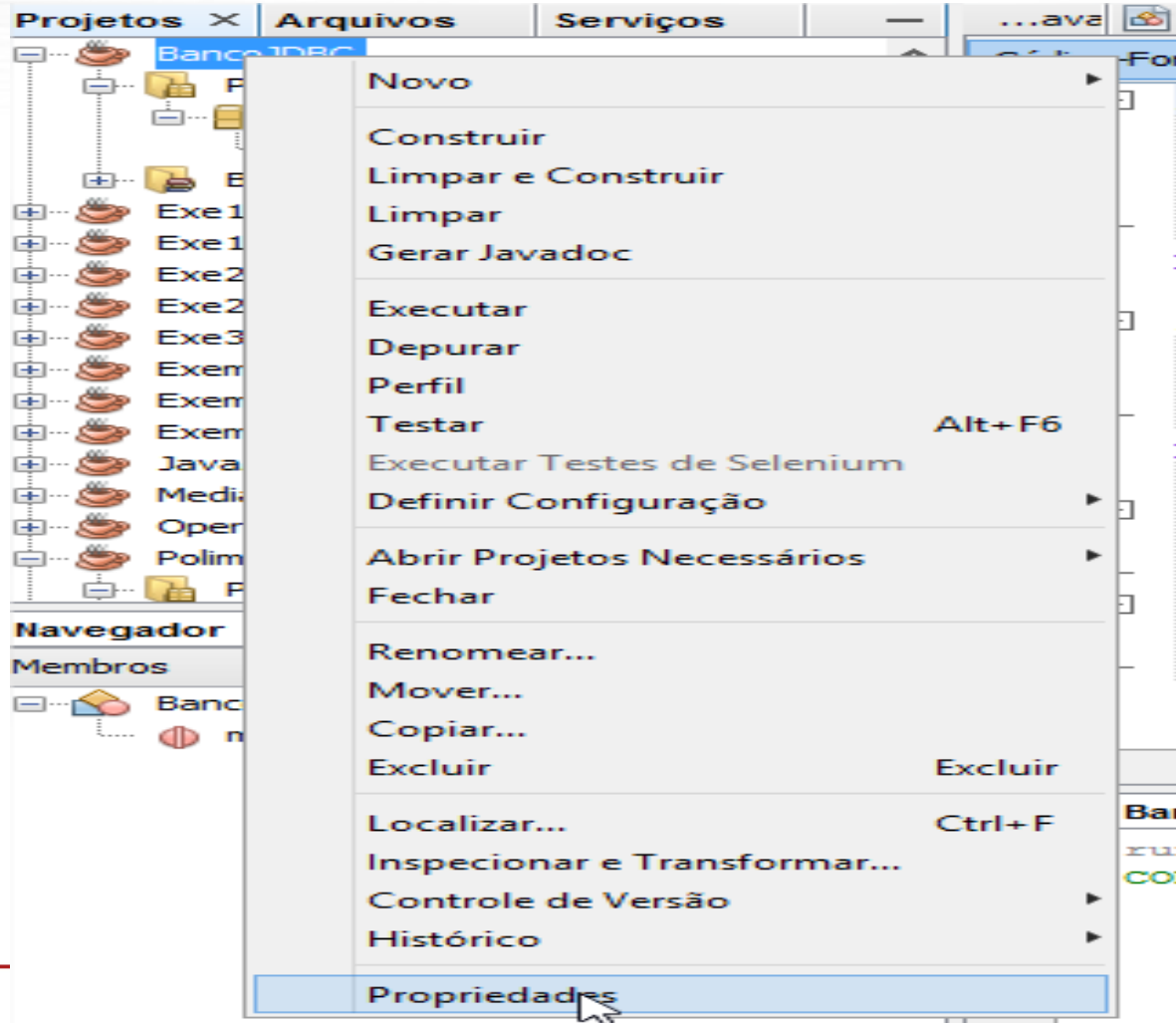
Em botão direito no Projeto → Propriedades → Bibliotecas → Executar → Adicionar JAR/Pasta → Selecciona o arquivo baixado do **Connector JDBC o MySQL** e clique em Abrir.

---

# JDBC e MySQL

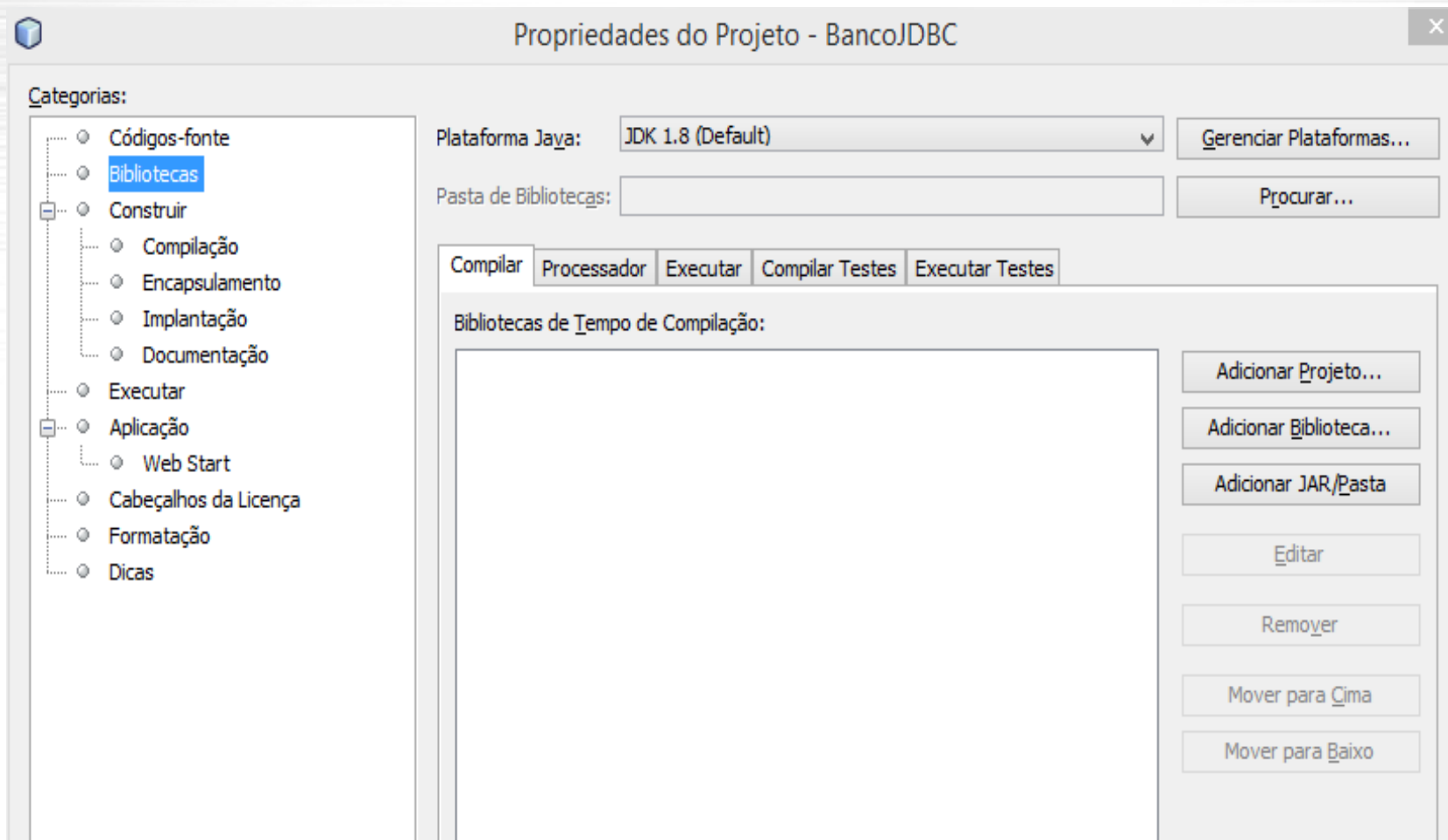
## JDBC - Configurando Banco de Dados no Netbeans

3º Passo:



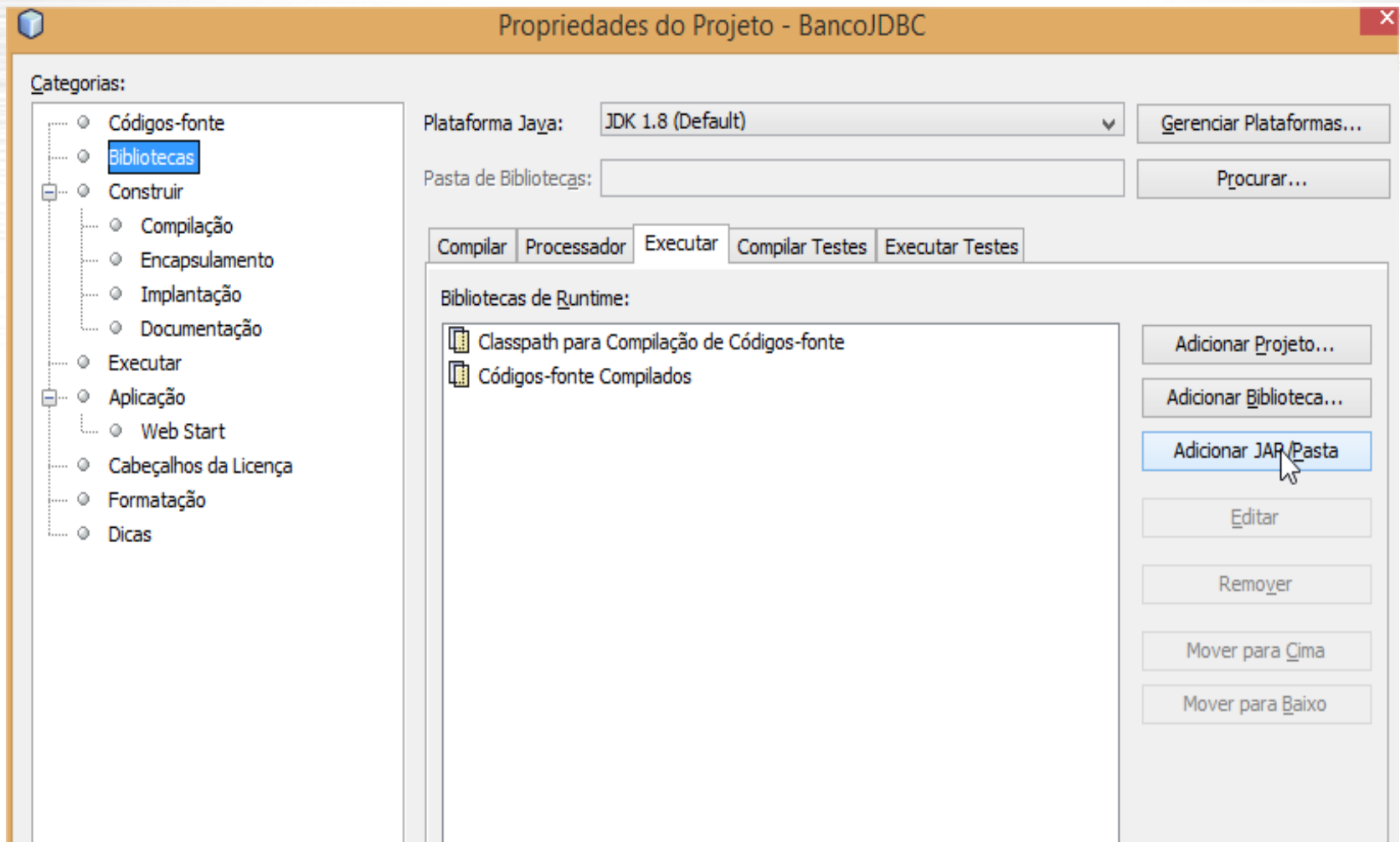
# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans



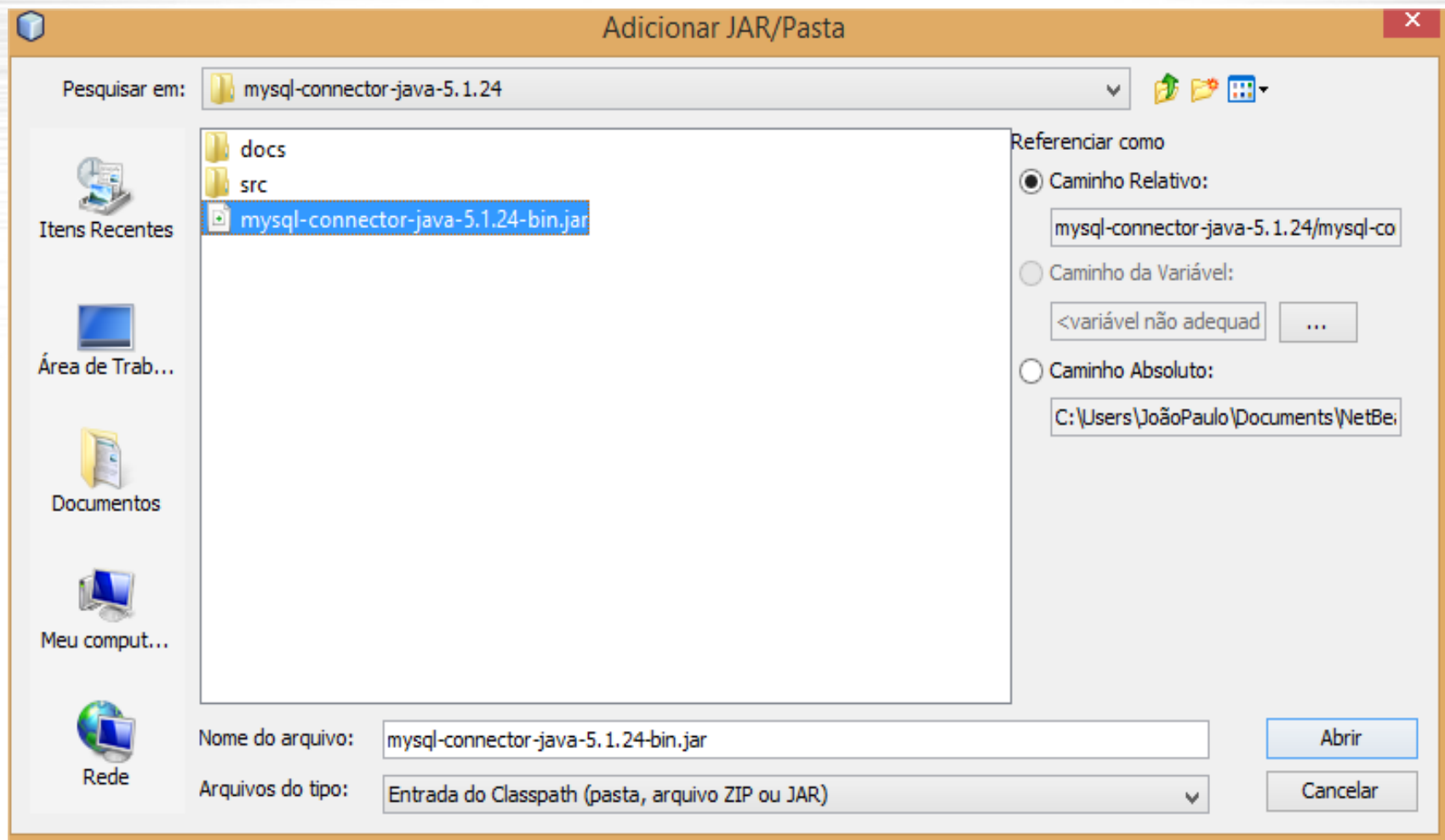
# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans



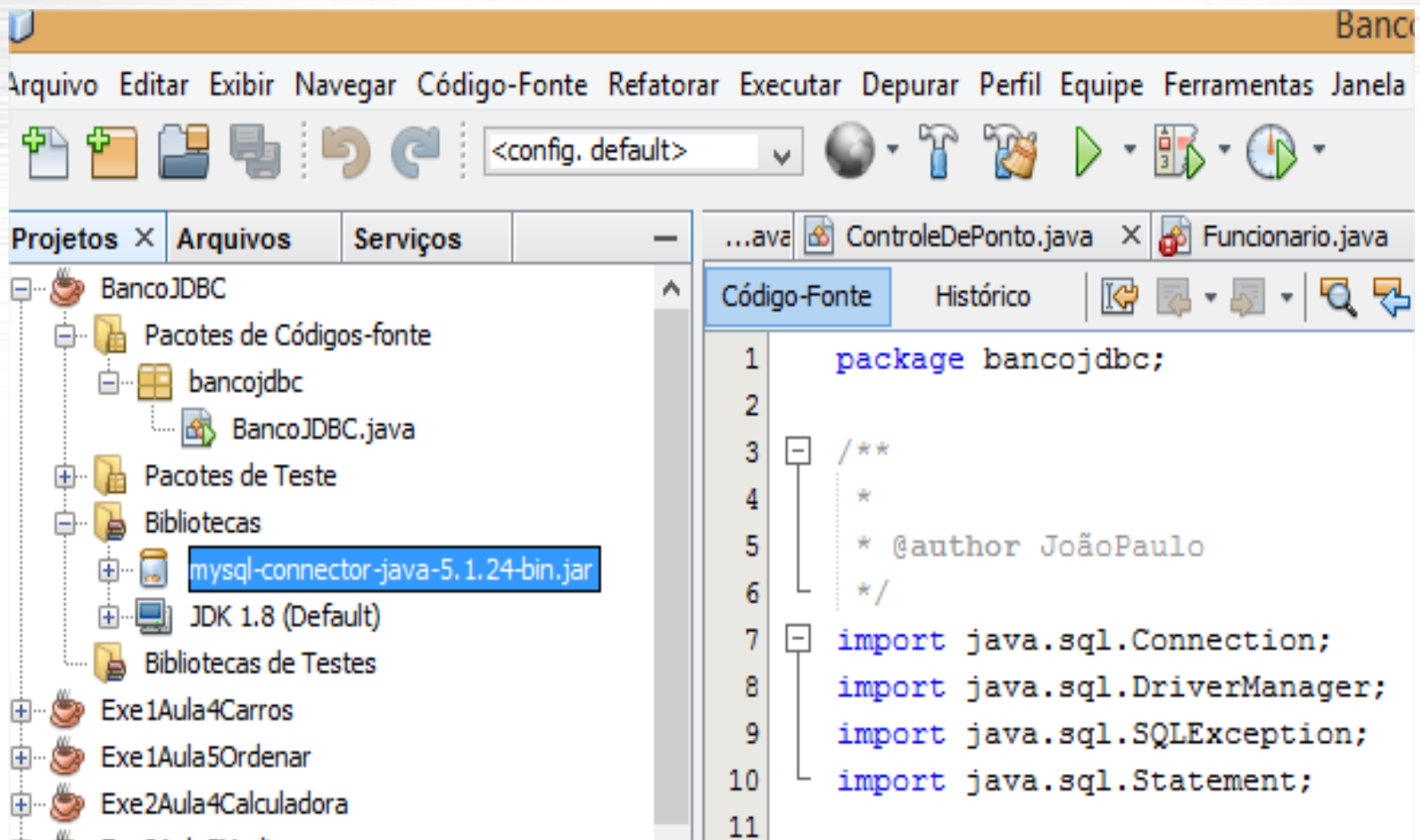
# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans



# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans



# JDBC e MySQL

## JDBC - Configurando Banco de Dados no Netbeans

### 4º Passo:

Agora vamos utilizar a seguinte classe para acesso:

```
package bancojdbc;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
import java.sql.Statement;
```

---



# JDBC e MySQL

## //continuação do código .java

```
public class BancoJDBC{  
    private Connection con;  
    private Statement stmt;  
  
    public BancoJDBC() {  
        try{  
            Class.forName("com.mysql.jdbc.Driver");  
            System.out.println("Driver encontrado!");  
        } catch (ClassNotFoundException e) {  
            System.out.println("Driver não encontrado!" +  
e);  
            System.out.println("Error: " + e.getMessage());  
        }  
    }  
}
```

---

# JDBC e MySQL

## //continuação do código .java

```
String url = "jdbc:mysql://localhost:3306/Banco";
    String user = "root";
    String password = "system";

try{

con=DriverManager.getConnection(url,user,password);
stmt = con.createStatement();

}catch(SQLException e){
    System.out.println("Error: "+ e.getMessage());
}

inserirRegistros();
}
```

---

# JDBC e MySQL

## //continuação do código .java

```
private void inserirRegistros() {  
    try{  
        stmt.executeUpdate("INSERT INTO Empregado  
VALUES (4, 'Evandro', '6291376654', '6780.00')");  
    }catch(SQLException e){  
        System.out.println("Error: "+ e.getMessage());  
    }  
  
}
```

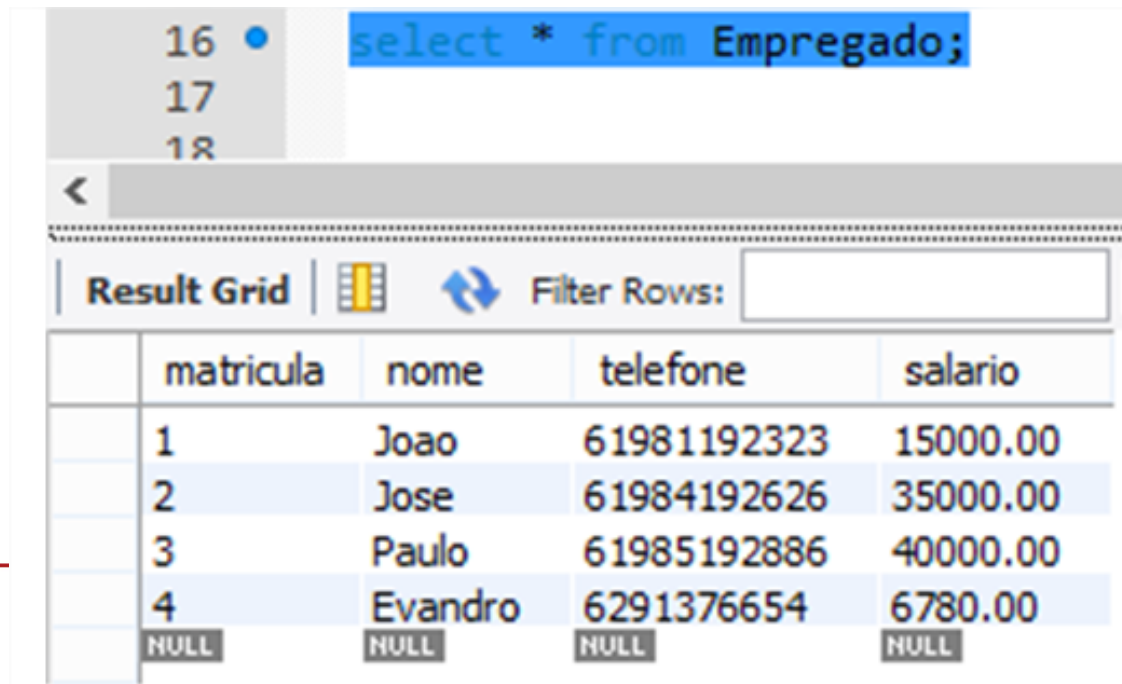
---

# JDBC e MySQL

## //continuação do código .java

```
public static void main(String[] args) {  
    BancoJDBC bancoJDBC = new BancoJDBC();  
}  
}
```

→ Nossa tabela Empregado após executar nosso método `inserirRegistros()`;



The screenshot shows a database IDE interface. At the top, a SQL query is entered in a text area: `select * from Empregado;`. Below the query area, there is a 'Result Grid' section. It contains a table with 5 columns: 'matricula', 'nome', 'telefone', and 'salario'. The table displays 4 rows of data. Below the data rows, there are four 'NULL' labels, one for each column, indicating that the results are empty. The interface also includes a 'Filter Rows' input field and some navigation icons.

	matricula	nome	telefone	salario
1	1	Joao	61981192323	15000.00
2	2	Jose	61984192626	35000.00
3	3	Paulo	61985192886	40000.00
4	4	Evandro	6291376654	6780.00
	NULL	NULL	NULL	NULL

# JDBC e MySQL

→ Como seria um método para listar os registros da tabela?

**//acrescentar a biblioteca:**

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

```
private void listarRegistros() {
```

```
try{
```

```
    ResultSet rs;
```

```
    rs = stmt.executeQuery("SELECT * FROM Empregado");
```

```
while ( rs.next() ) {
```

---

# JDBC e MySQL

## // continuação do método

```
int matricula = rs.getInt("matricula");
String nome = rs.getString("nome");
String telefone = rs.getString("telefone");
float salario = rs.getFloat("salario");



System.out.println(matricula + "\t" + nome + "\t" +
telefone + "\t" + salario);
}
    }catch(SQLException e){
        System.out.println("Erro: "+ e.getMessage());
    }
}
```

---

# Executando...

Código-Fonte   Histórico

```
50
51 }
52 private void listarRegistros() {
53
54     try{
55         ResultSet rs;
56         rs = stmt.executeQuery("SELECT * FROM Empregado");
57         while ( rs.next() ) {
58             int matricula = rs.getInt("matricula");
59             String nome = rs.getString("nome");
60             String telefone = rs.getString("telefone");
61             float salario = rs.getFloat("salario");
62             System.out.println(matricula + "\t" + nome + "\t" + telefone + "\t" + salario);
63         }
64     } catch (SQLException e){
65         System.out.println("Erro: " + e.getMessage());
66     }
67 }
68
69 public static void main(String[] args) {
```

Localizar:  Anterior Próximo Selecionar aA " " \*  

bancojdbc.BancoJDBC > main > bancoJDBC >

Saída - BancoJDBC (run) X

```
run:
Driver encontrado!
1      Joao      61981192323      15000.0
2      Jose      61984192626      35000.0
3      Paulo     61985192886      40000.0
4      Evandro   6291376654      6780.0
CONSTRUÍDO COM SUCESSO (tempo total: 7 segundos)
```

# JDBC e MySQL

## //exemplo para inserir registros por parâmetros

```
private void inserirRegistros(int mat, String n, String  
tel, String sal){  
    try{  
        stmt.executeUpdate("INSERT INTO Empregado  
VALUES (" + mat + ", '" + n + "', '" + tel + "', '" + sal + "')");  
    } catch (SQLException e) {  
        System.out.println("Erro: " +  
e.getMessage());  
    }  
}
```

---



# JDBC e MySQL

## //exemplo para listar os registros com parâmetros

```
private void listarRegistros() {
    try{
        ResultSet rs;
        rs = stmt.executeQuery("SELECT * FROM
Empregado");
        while ( rs.next() ) {
            int matricula = rs.getInt("matricula");
            String nome = rs.getString("nome");
            String telefone =
rs.getString("telefone");
            float salario = rs.getFloat("salario");
            System.out.println(matricula + "\t" + nome
+ "\t" + telefone + "\t" + salario);
        }
    }catch(SQLException e){
        System.out.println("Erro: "+
e.getMessage()); } }
```

---

# JDBC e MySQL

## //exemplo do alterar e apagar com parâmetros

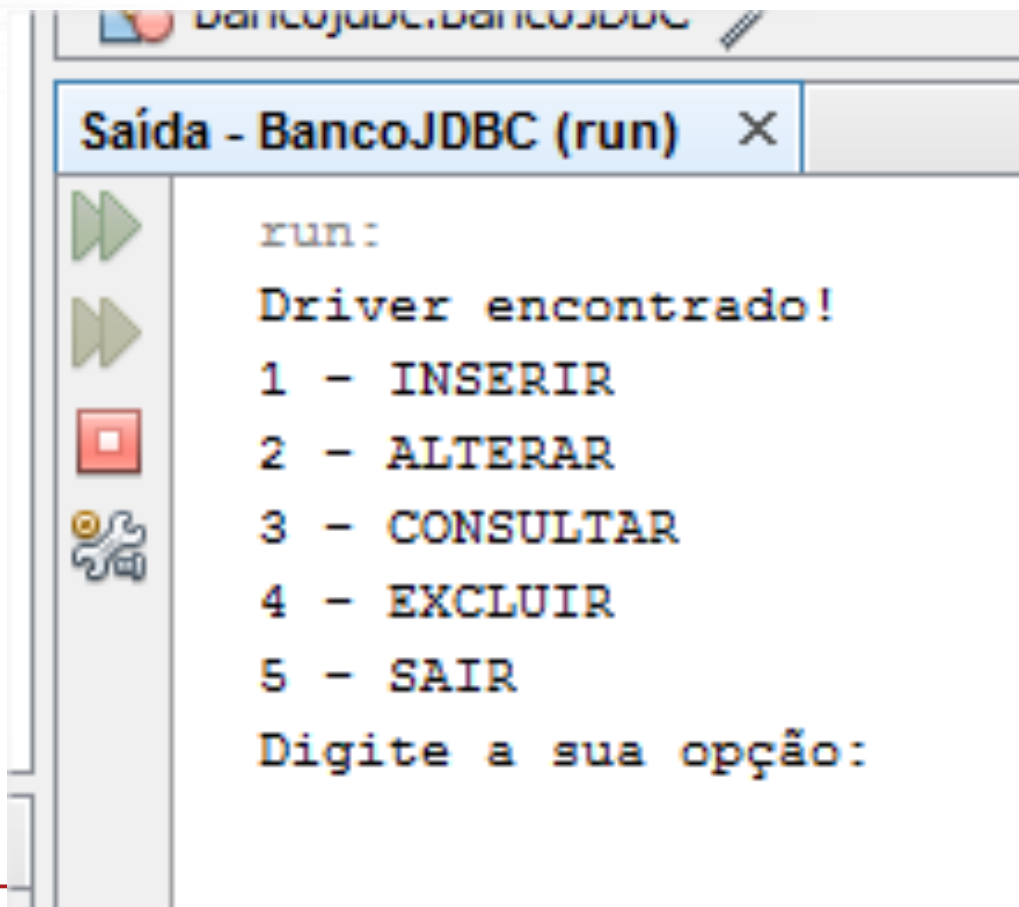
```
private void alterarRegistros(String sal, int mat){
    try{
        stmt.executeUpdate("UPDATE Empregado SET salario
= '"+sal+"' WHERE matricula="+mat+"");
    }catch(SQLException e){
        System.out.println("Erro: "+ e.getMessage());
    }
}

private void apagarRegistros(int mat){
    try{
        stmt.executeUpdate("DELETE FROM Empregado WHERE
matricula="+mat+"");
    }catch(SQLException e){
        System.out.println("Erro: "+ e.getMessage());
    }
}
```

---

# Exercício

**Desenvolver um menu para aplicação conforme apresentado nos próximos slides:**

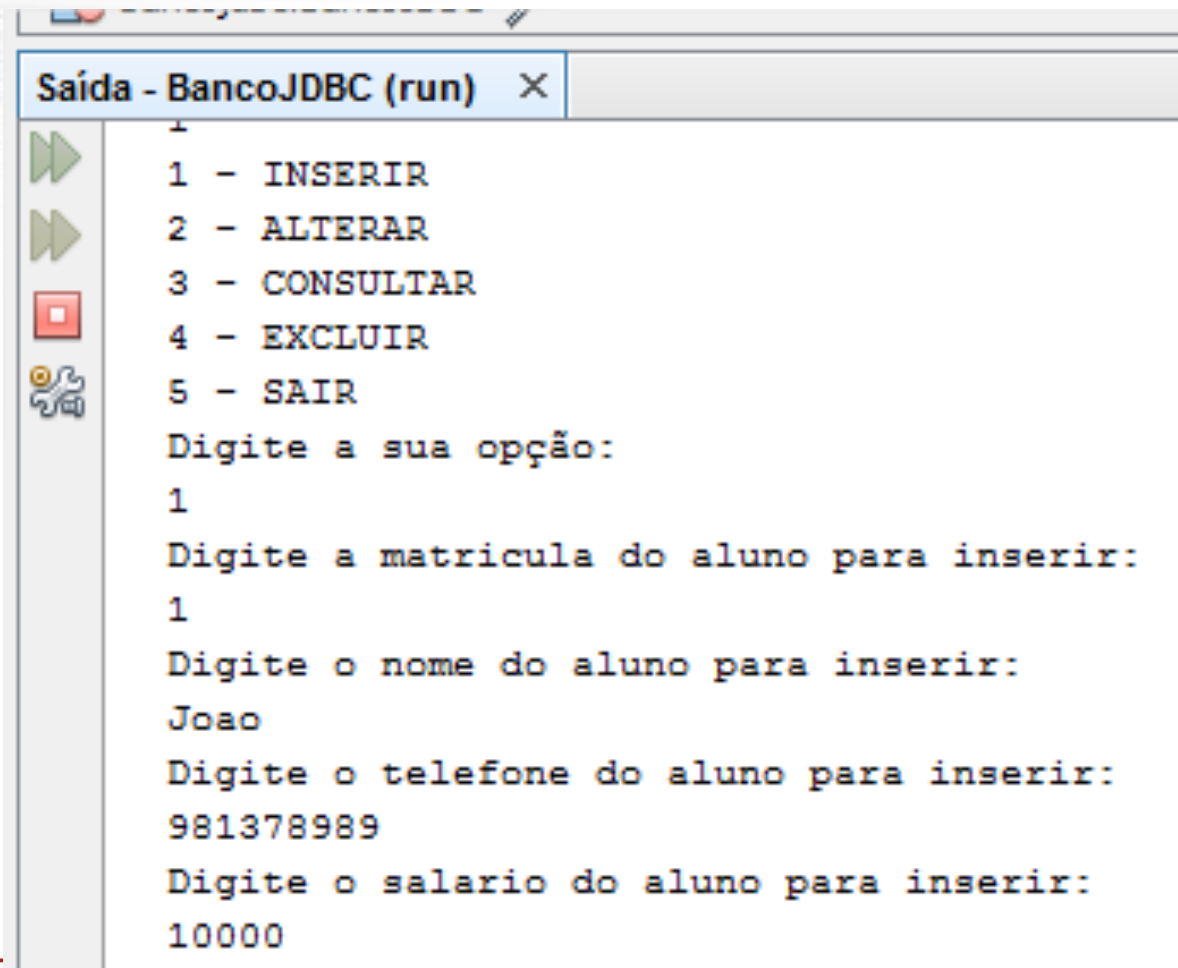


The screenshot shows a console window titled "Saída - BancoJDBC (run)". The output text is as follows:

```
run:
Driver encontrado!
1 - INSERIR
2 - ALTERAR
3 - CONSULTAR
4 - EXCLUIR
5 - SAIR
Digite a sua opção:
```

# Exercício


## //Opção 1 - Inserir



```
Saída - BancoJDBC (run) X
1
1 - INSERIR
2 - ALTERAR
3 - CONSULTAR
4 - EXCLUIR
5 - SAIR
Digite a sua opção:
1
Digite a matricula do aluno para inserir:
1
Digite o nome do aluno para inserir:
Joao
Digite o telefone do aluno para inserir:
981378989
Digite o salario do aluno para inserir:
10000
```

# Exercício

## //Opção 3 - Consultar



```
1 - INSERIR
2 - ALTERAR
3 - CONSULTAR
4 - EXCLUIR
5 - SAIR
Digite a sua opção:
3
1      Joao      981378989      10000.0
2      Jose      61984192626     35000.0
3      Paulo     61985192886     40000.0
4      Evandro   6291376654      6780.0
5      Luciano   9898987766      23000.0
1 - INSERIR
```

# Exercício

## //Opção 2 – Alterar salário



1 - INSERIR

2 - ALTERAR

3 - CONSULTAR

4 - EXCLUIR

5 - SAIR

Digite a sua opção:

2

Digite a matricula do aluno para alterar o salário:

1

Digite o valor do novo salário:

20000

# Exercício

## //Opção 3 – Consultar para conferir a alteração

1 - INSERIR

2 - ALTERAR

3 - CONSULTAR

4 - EXCLUIR

5 - SAIR

Digite a sua opção:

3

1	Joao	981378989	20000.0
---	------	-----------	---------

2	Jose	61984192626	35000.0
---	------	-------------	---------

3	Paulo	61985192886	40000.0
---	-------	-------------	---------

4	Evandro	6291376654	6780.0
---	---------	------------	--------

5	Luciano	9898987766	23000.0
---	---------	------------	---------

# Exercício

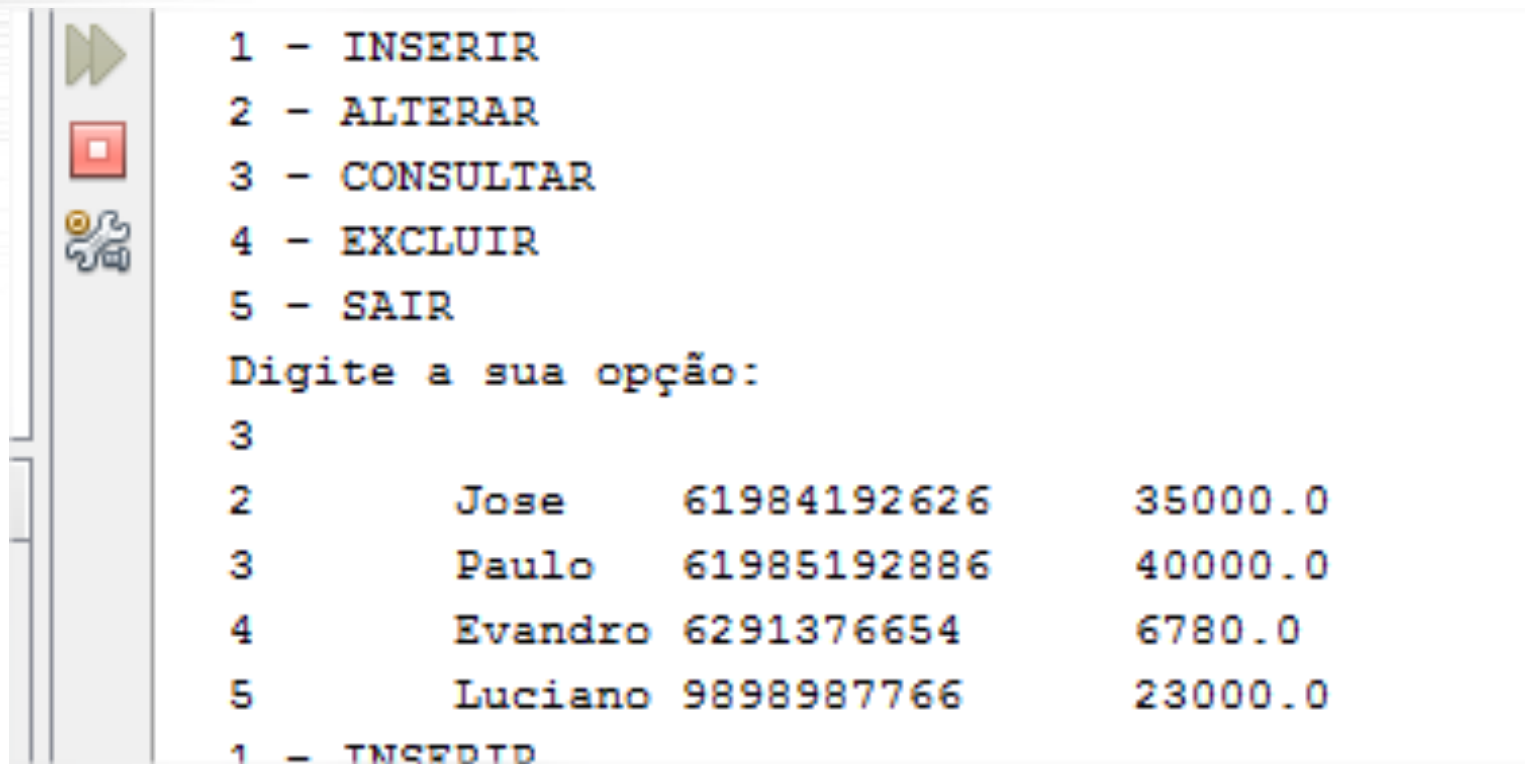
## //Opção 4 – Excluir

```
Saida - BancoJDBC (run) X
Digite a sua opção:
3
1      Joao      981378989      20000.0
2      Jose      61984192626     35000.0
3      Paulo     61985192886     40000.0
4      Evandro   6291376654      6780.0
5      Luciano   9898987766      23000.0
1 - INSERIR
2 - ALTERAR
3 - CONSULTAR
4 - EXCLUIR
5 - SAIR
Digite a sua opção:
4
```



# Exercício

**//Consultar para verificar a exclusão**



1 - INSERIR  
2 - ALTERAR  
3 - CONSULTAR  
4 - EXCLUIR  
5 - SAIR

Digite a sua opção:  
3

2	Jose	61984192626	35000.0
3	Paulo	61985192886	40000.0
4	Evandro	6291376654	6780.0
5	Luciano	9898987766	23000.0
1	INSEDTD		



Até a próxima aula...

---