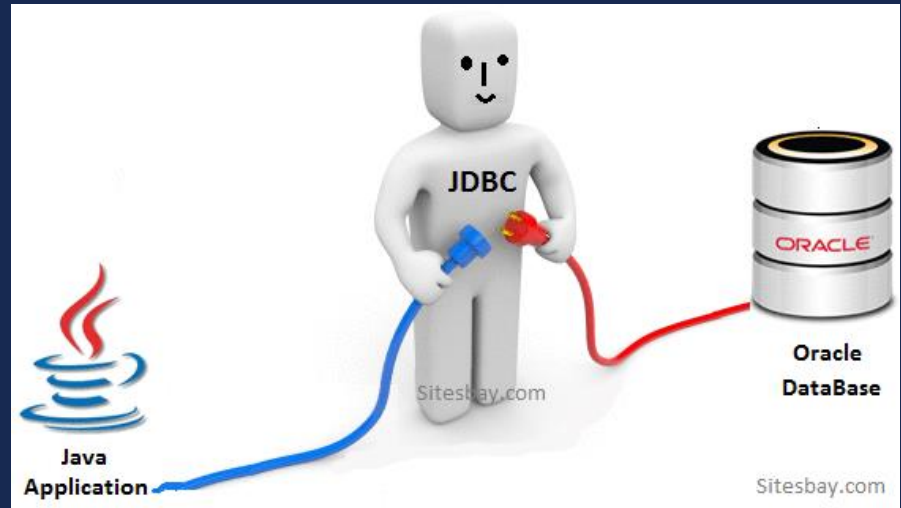


Programação Orientada a Objetos

Unidade 6 – Conectividade com Banco de Dados



Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP
aparecidovfreitas@gmail.com

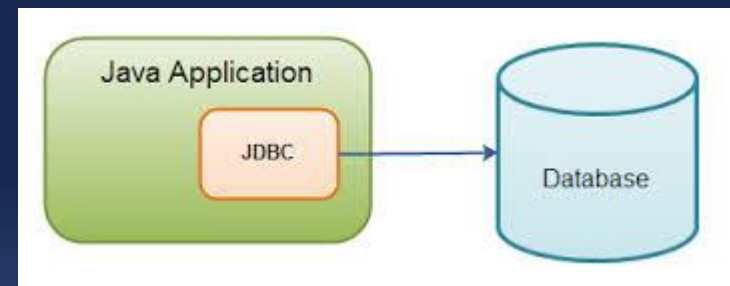
Introdução

- ✓ A **SQL Embutida** às vezes é chamada de **Técnica de Programação de Banco de Dados Estática**, pois o texto da consulta é escrito no código fonte do programa e **não** pode ser alterado sem uma nova compilação ou reprocessamento do código fonte;



Programação SQL – CLI

- ✓ Nesta técnica de programação **SQL**, disponibiliza-se ao programa **SQL** uma biblioteca de funções (também conhecida por interface de programação de aplicação (**API**), para acesso ao sistema gerenciador de banco de dados;
- ✓ Nesta técnica, a partir do programa **SQL** efetuam-se chamadas de função de biblioteca, e por essa razão a técnica é chamada **SQL/CLI – CALL LEVEL INTERFACE**;
- ✓ Um exemplo deste tipo de programação SQL é **JDBC – Java DataBase Connectivity**, uma biblioteca de funções (driver) para acessar banco de dados com Java. Outro exemplo é **ODBC – Open DataBase Connectivity**.



Quais as vantagens de se usar CLI ao invés de SQL embutido ?



Programação SQL CLI – Vantagens

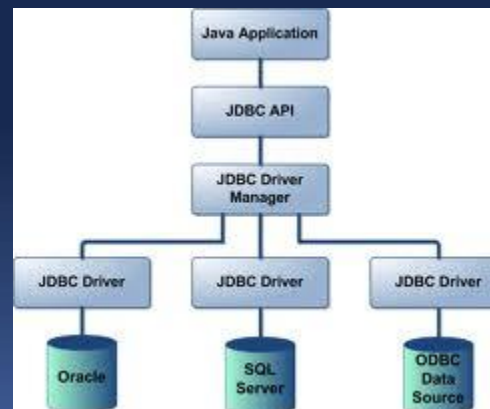
- ✓ A principal vantagem do uso de uma interface de chamada de função (**CLI**) é que ela facilita o acesso à múltiplos bancos de dados no mesmo programa de aplicação, mesmo que eles estejam armazenados em diferentes sistemas de gerenciamento de banco de dados;
- ✓ Ao se empregar **SQL CLI**, **não** há necessidade de se empregar **pré-processadores** para o processamento do código SQL. No entanto, a sintaxe e outras verificações dos comandos **SQL** precisam ser feitas **em tempo de execução**.



JDBC



- ✓ **API** for Java que define a forma pela qual um programa acessa um banco de dados;
- ✓ Primeira versão do **JDBC – Java Database Connectivity** liberada pela Sun em 1996;
- ✓ Esta liberação permitiu que programadores Java pudessem fazer conexão a um banco de dados, atualização e consultas através da linguagem **SQL**;
- ✓ Características: Portabilidade, API independente do SGBD subjacente, Estrutura em camadas.



Conectividade JDBC

- ✓ Programas desenvolvidos com **Java** e **JDBC** são **independentes** de plataforma e de fornecedores de **SGBD**.
- ✓ Pode-se mover dados de um **SGBD** para outro (por exemplo, **SQL Server** para **DB/2**).



Padrão JDBC de acesso a Bases de Dados

- ✓ API de acesso para executar comandos SQL;
- ✓ Implementado no **pacote padrão java.sql**;
- ✓ Envio para qualquer tipo de Banco de Dados relacional;
- ✓ Interface baseada no **X/OPEN SQL CLI**;
- ✓ Independente de **API/Linguagem** proprietária dos fabricantes de SGBD (IBM DB/2, Microsoft, Oracle, Informix, ...)
- ✓ Uso de drivers específicos de fabricantes do SGBD.



A arquitetura JDBC

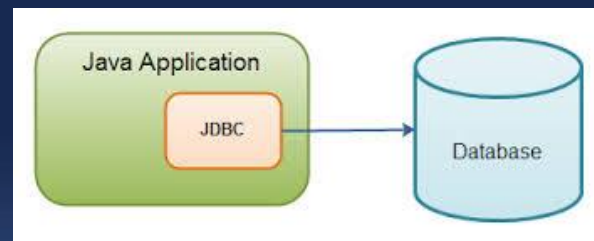
- ✓ **JDBC** é composto por um conjunto de interfaces, cada qual implementada diferentemente pelos fornecedores;
- ✓ O conjunto de classes que implementam as interfaces **JDBC** para um particular banco de dados é chamada **JDBC driver**;
- ✓ Os detalhes de como esta implementação foi feita é irrelevante (**encapsulamento**).



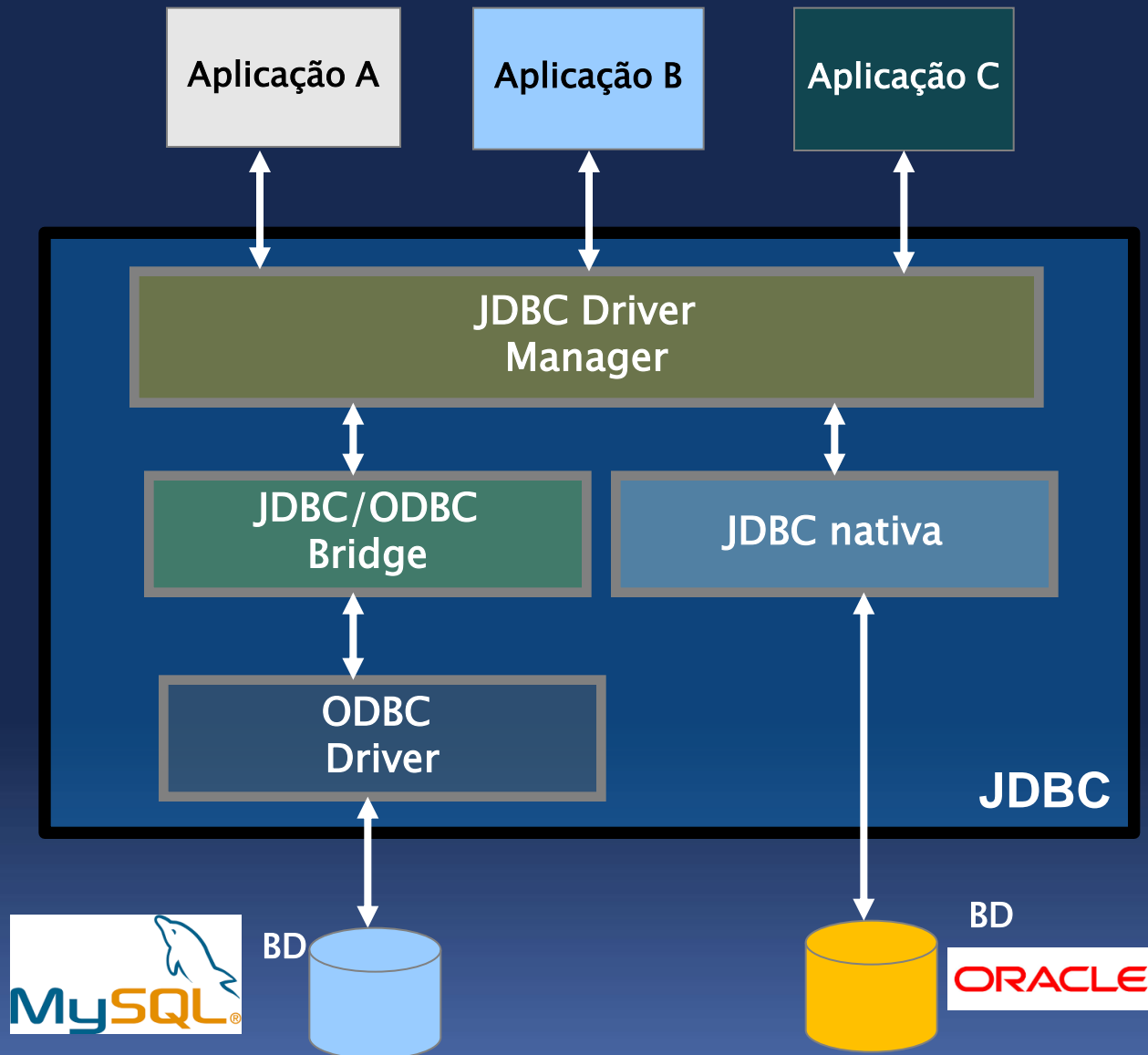
Arquitetura JDBC



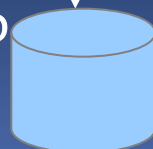
- ❖ Aplicações Java “**conversam**” com o gerenciador de drivers JDBC (Driver Manager);
- ❖ Este, por sua vez, se comunica com algum **driver carregado em tempo de execução**;
- ❖ O programa de aplicação interage **apenas** com a API do gerenciador de drivers;
- ❖ O gerenciador de drivers interage com o driver específico do SGBD, que por sua vez interage com o SGBD;



Arquitetura JDBC



BD



BD



ORACLE®



Pacote java.sql

- Na maioria definida por meio de **interfaces**;
- A implementação destas interfaces é feita pelo fornecedor do driver do banco de dados(por exemplo: **DB2**, **Oracle**, etc) ;
- Com isso, **JDBC** padronizou o modo de se conectar ao banco de dados, liberando o driver para ser implementado pelos fornecedores, que na verdade é que são os especialistas em SGBD.





Implementações JDBC

- O JDBC pode ser visto como um conjunto de **interfaces** cuja implementação deve ser fornecida por fabricantes de **SGBD**;
- Cada fabricante deve fornecer implementações de:
 - **java.sql.Connection**
 - **java.sql.Statement**
 - **java.sql.PreparedStatement**
 - **java.sql.CallableStatement**
 - **java.sql.ResultSet**
 - **java.sql.Driver**
- O objetivo é que fique transparente para o programador qual a implementação **JDBC** está sendo utilizada.



Instalação JDBC

- O pacote **JDBC** faz parte da **JDK**, incluso com as distribuições Java;
- As classes e interfaces que compõem a **API JDBC** estão nos packages **java.sql** e **javax.sql**;
- Entretanto, deve-se obter um **driver** para o sistema de gerência de banco de dados a ser utilizado.



- Lista de drivers **JDBC** disponíveis:

<http://www.oracle.com/technetwork/java/index-136695.html>

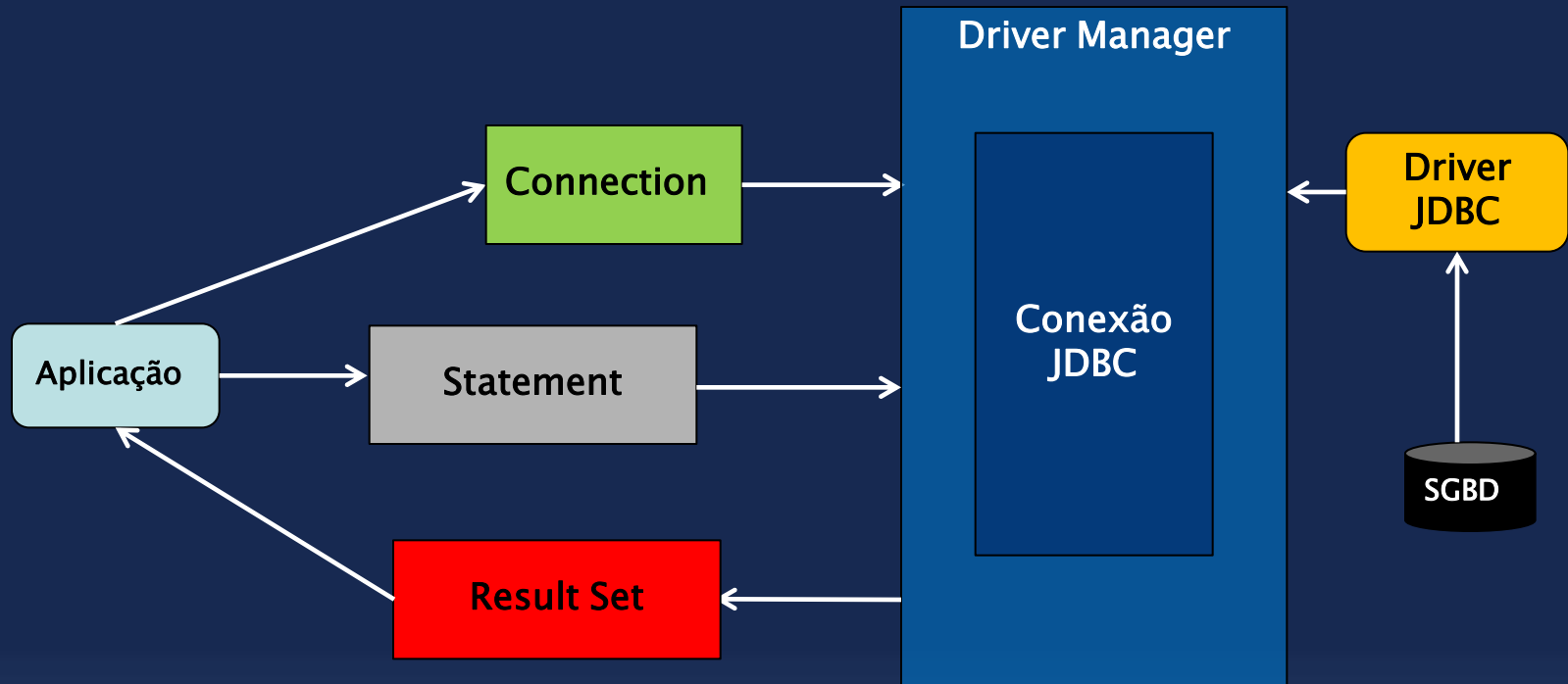


Classes Principais – JDBC

- ✓ **java.sql.DriverManager**
 - Provê serviços básicos para gerenciar diversos drivers **JDBC**;
- ✓ **java.sql.Connection**
 - Representa uma conexão estabelecida com o **BD**;
- ✓ **java.sql.Statement**
 - Representa sentenças onde são inseridos os comandos **SQL**;
 - Permite realizar todo o tratamento das consultas (select) e dos comandos de atualizações (insert, delete, update)
- ✓ **java.sql.ResultSet**
 - Representa o conjunto de registros resultante de uma consulta;
 - Permite manipular os resultados;
 - Permite realizar coerção (cast) entre tipos Java e **SQL**;



Interação Programa JDBC



Processamento de aplicação JDBC

- ✓ Definição de qual **driver** será utilizado na aplicação;
- ✓ **Carga** do driver;
- ✓ Criação do objeto **Connection** que será responsável pelas atividades de conexão banco de dados;
- ✓ Criação dos objetos **Statement** e **ResultSet** para envio de queries;
- ✓ **Execução** das queries;
- ✓ Processamento dos resultados;
- ✓ Fechamento (**Close**) da conexão.



Como efetuar conexão com o servidor de banco de dados ?





Classe DriverManager

- ✓ Responsável por abrir uma conexão, especificada através de uma URL, com uma base de dados, utilizando um determinado driver;
- ✓ Possui registro de todos os drivers já carregados;



Definição do Driver

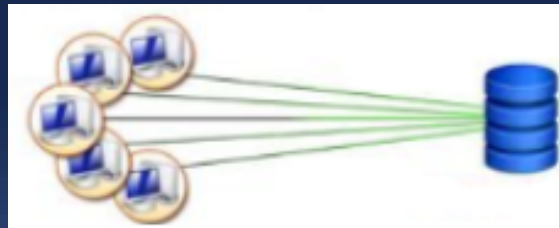


- ✓ Numa aplicação **Java**, podemos ter vários drivers trabalhando ao mesmo tempo;
- ✓ A definição do **driver** é feita por meio de um **String de conexão**;
- ✓ O driver é um arquivo **.jar** e devemos tê-lo em um **classpath**, do contrário a aplicação não o encontrará.
- ✓ Deve-se anexar o driver no **classpath**, no instante da execução da aplicação **Java**.

```
java -classpath diretorio/meudriver.jar Minhaclasse
```

Conexão ao Banco de Dados

- Para a conexão com o banco deve-se instanciar um objeto do tipo **Connection** através da chamada do método **getConnection()** da classe **DriverManager**.
- O método **getConnection()** recebe como parâmetros informações relativas ao **data-source (URL)**, **usuário** e **senha** para autenticação.





Exemplo getConnection() com Driver nativo MySQL

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

public class TestConnection {

    public static Connection createConnection() throws SQLException {

        String url = "jdbc:mysql://localhost:3306/loja";
        String user = "root";
        String password = "";

        Connection conexao = null;

        conexao = DriverManager.getConnection( url, user, password );

        return conexao;
    }
}
```





Exemplo – Driver **MariaDB** – nativo

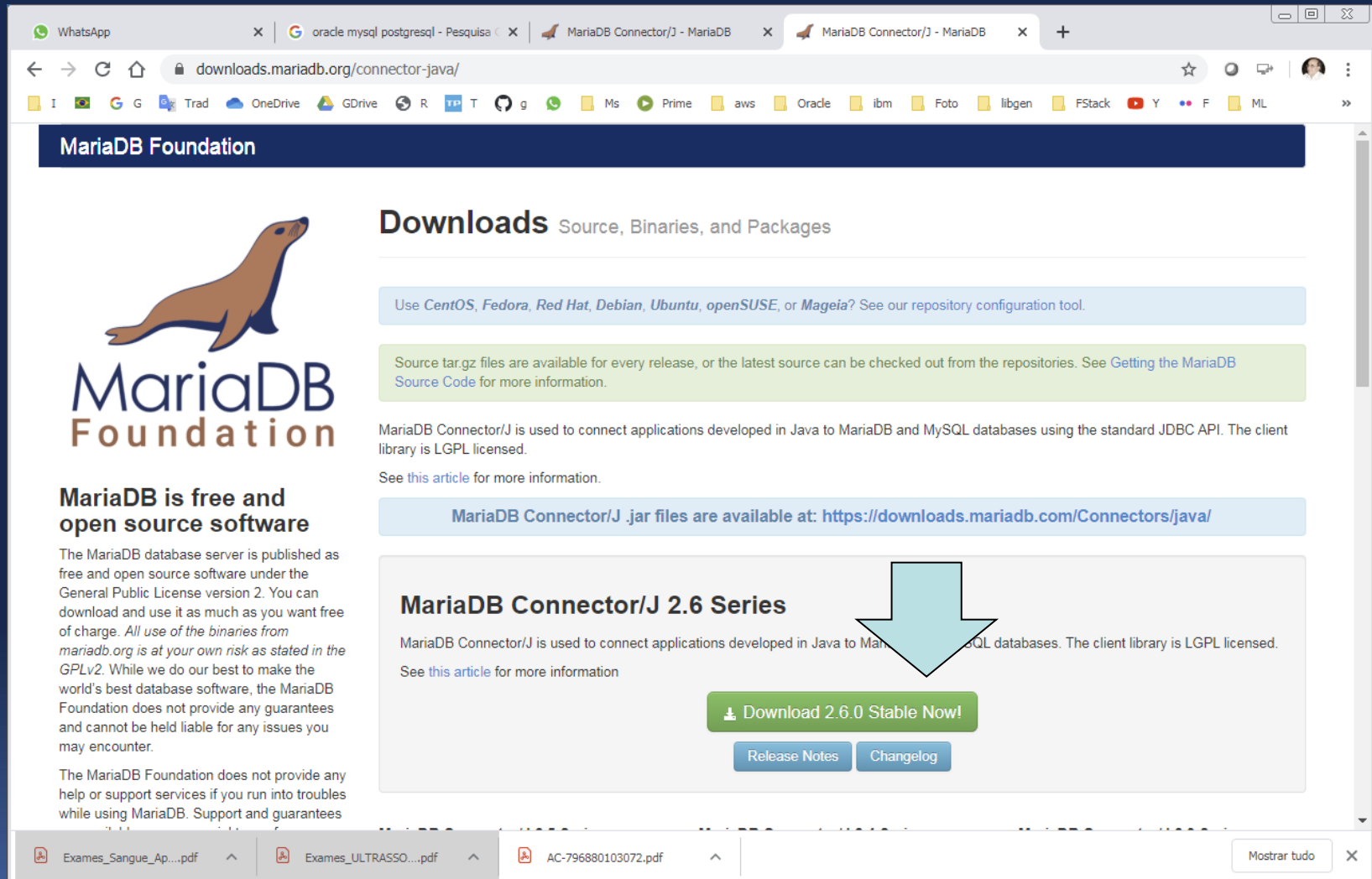
- ✓ Baixar o driver a partir do endereço:

<https://downloads.mariadb.org/connector-java/>

- ✓ Salvar em algum diretório do Servidor;
- ✓ Configurar o Path do **Eclipse** para que o projeto visualize o driver.



Exemplo – Driver **MariaDB** – nativo



The screenshot shows the MariaDB Foundation website's Downloads page. The browser's address bar displays 'downloads.mariadb.org/connector-java/'. The page features the MariaDB Foundation logo, which includes a seal illustration and the text 'MariaDB Foundation'. A section titled 'Downloads' with the subtitle 'Source, Binaries, and Packages' contains several informational boxes. A blue box suggests using various Linux distributions like CentOS, Fedora, Red Hat, Debian, Ubuntu, openSUSE, or Mageia, with a link to the repository configuration tool. A green box states that source tar.gz files are available for every release, with a link to 'Getting the MariaDB Source Code'. A paragraph explains that MariaDB Connector/J is used to connect Java applications to MariaDB and MySQL databases using the standard JDBC API, and is LGPL licensed, with a link to a related article. A blue box provides the URL for downloading MariaDB Connector/J .jar files: 'https://downloads.mariadb.com/Connectors/java/'. A large section for 'MariaDB Connector/J 2.6 Series' features a large blue downward-pointing arrow, a paragraph about its use, a link to an article, and a prominent green button labeled 'Download 2.6.0 Stable Now!'. Below this button are two smaller buttons: 'Release Notes' and 'Changelog'. The browser's taskbar at the bottom shows several open PDF files: 'Exames_Sangue_Ap...pdf', 'Exames_ULTRASSO...pdf', and 'AC-796880103072.pdf', along with a 'Mostrar tudo' button.

MariaDB Foundation

Downloads

Source, Binaries, and Packages

Use CentOS, Fedora, Red Hat, Debian, Ubuntu, openSUSE, or Mageia? See our repository configuration tool.

Source tar.gz files are available for every release, or the latest source can be checked out from the repositories. See [Getting the MariaDB Source Code](#) for more information.

MariaDB Connector/J is used to connect applications developed in Java to MariaDB and MySQL databases using the standard JDBC API. The client library is LGPL licensed.

See [this article](#) for more information.

MariaDB Connector/J .jar files are available at: <https://downloads.mariadb.com/Connectors/java/>

MariaDB Connector/J 2.6 Series

MariaDB Connector/J is used to connect applications developed in Java to MariaDB and MySQL databases. The client library is LGPL licensed.

See [this article](#) for more information

[Download 2.6.0 Stable Now!](#)

[Release Notes](#) [Changelog](#)

Exemplo – Driver **MariaDB** – nativo

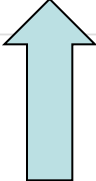
[View all releases](#)

MariaDB Connector/J 2.6.0 Stable 2020-03-20

Release Notes
Changelog

Affordable, enterprise class product support, professional services, and training for your MariaDB database is available from the MariaDB Foundation's release sponsor, MariaDB Corporation. To learn more about them and their services for MariaDB, visit their [website](#), or email MariaDB Corporation at sales@mariadb.com.

File Name	Package Type	OS / CPU	Size	Meta
mariadb-java-client-2.6.0-sources.jar	java source jar	Source	626.3 kB	Checksum Instructions
MariaDB Connector/J .jar files	jar	Universal		Checksum Instructions



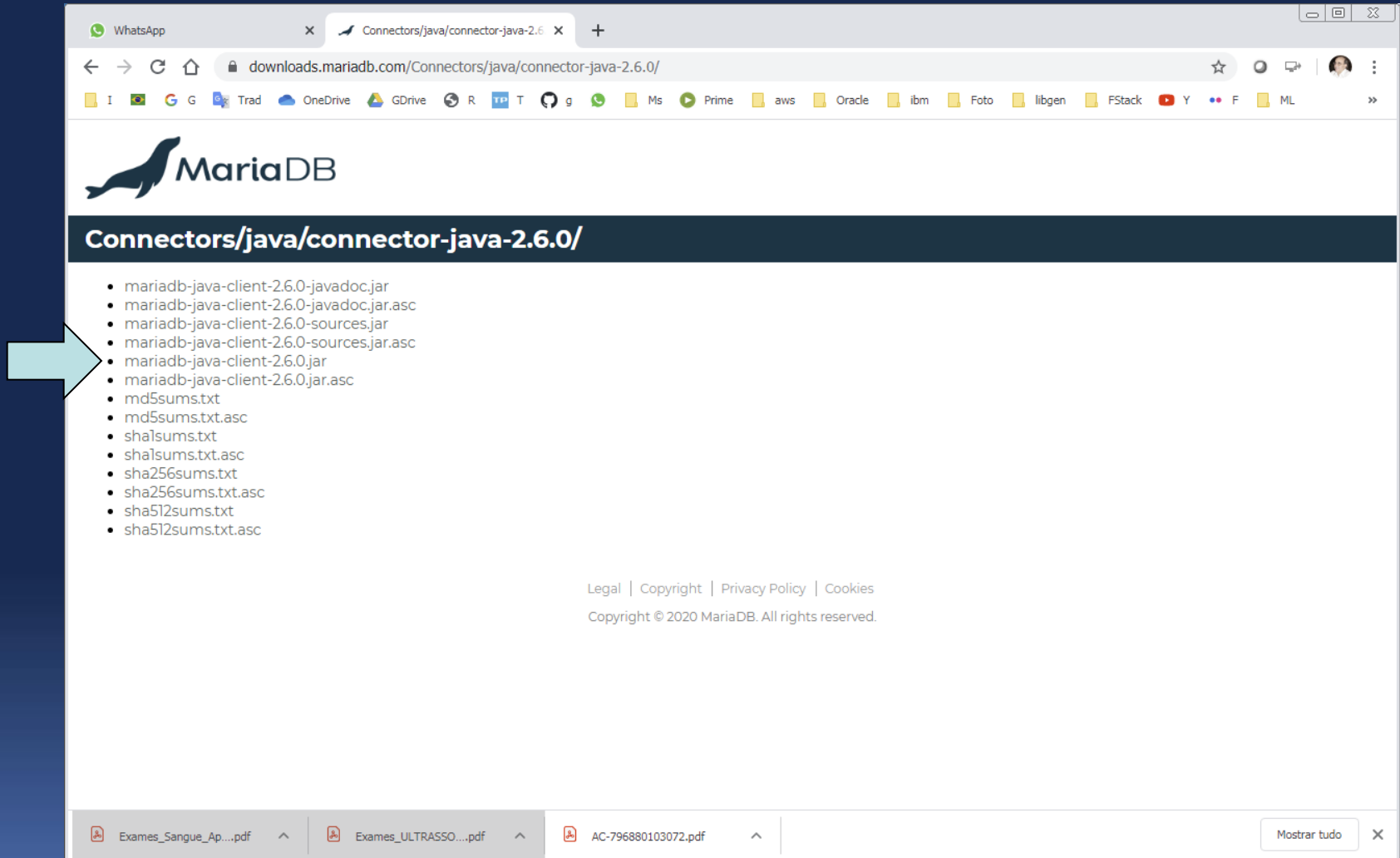
Mirror

- ☐ Liquid Telecom - Nairobi
- ☐ Marwan - Morocco
- ☐ 清华大学 TUNA 协会 (Tsinghua University TUNA Association)
- ☐ 網匯在線有限公司 - Nethub Online Limited - Hong Kong
- ☐ PT. Biznet Gio Nusantara
- ☒ Limestone Networks

[Show All Mirrors](#)

Exemplo – Driver **MariaDB** – nativo

✓ Baixar o driver em alguma pasta de seu sistema.



The screenshot shows a web browser window with the URL `downloads.mariadb.com/Connectors/java/connector-java-2.6.0/`. The page features the MariaDB logo and a list of files for download. A blue arrow points to the list of files.

Connectors/java/connector-java-2.6.0/

- mariadb-java-client-2.6.0-javadoc.jar
- mariadb-java-client-2.6.0-javadoc.jar.asc
- mariadb-java-client-2.6.0-sources.jar
- mariadb-java-client-2.6.0-sources.jar.asc
- mariadb-java-client-2.6.0.jar
- mariadb-java-client-2.6.0.jar.asc
- md5sums.txt
- md5sums.txt.asc
- sha1sums.txt
- sha1sums.txt.asc
- sha256sums.txt
- sha256sums.txt.asc
- sha512sums.txt
- sha512sums.txt.asc

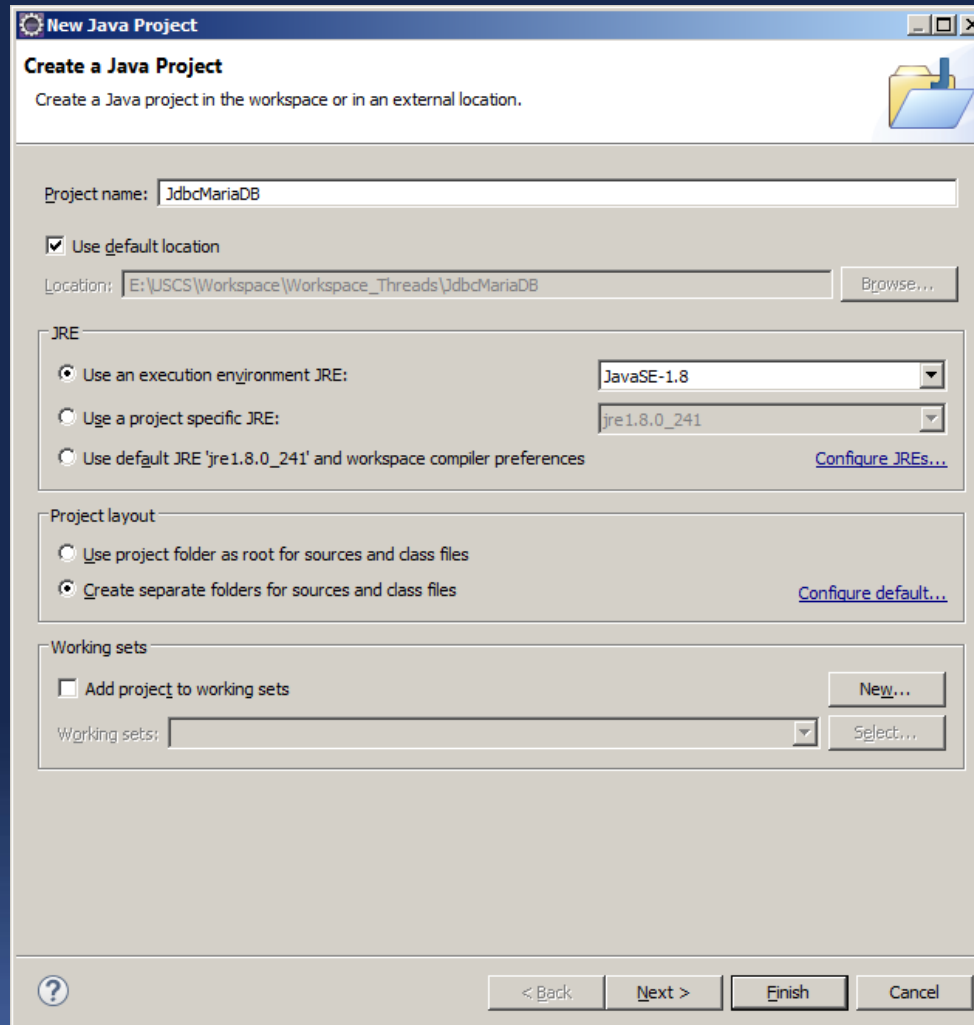
Legal | Copyright | Privacy Policy | Cookies
Copyright © 2020 MariaDB. All rights reserved.

Exames_Sangue_Ap...pdf Exames_ULTRASSO...pdf AC-796880103072.pdf

Mostrar tudo

Exemplo – Driver **MariaDB** – nativo

- ✓ Criar um Projeto Java no Eclipse



The screenshot shows the 'New Java Project' dialog box in the Eclipse IDE. The dialog is titled 'New Java Project' and has a subtitle 'Create a Java Project'. Below the subtitle, it says 'Create a Java project in the workspace or in an external location.' The 'Project name' field is filled with 'JdbcMariaDB'. The 'Use default location' checkbox is checked. The 'Location' field shows the path 'E:\USCS\Workspace\Workspace_Threads\JdbcMariaDB'. The 'JRE' section has three radio buttons: 'Use an execution environment JRE:' (selected), 'Use a project specific JRE:', and 'Use default JRE 'jre1.8.0_241' and workspace compiler preferences'. The 'Use an execution environment JRE:' option has a dropdown menu showing 'JavaSE-1.8'. The 'Use a project specific JRE:' option has a dropdown menu showing 'jre1.8.0_241'. The 'Project layout' section has two radio buttons: 'Use project folder as root for sources and class files' and 'Create separate folders for sources and class files' (selected). The 'Working sets' section has a checkbox 'Add project to working sets' which is unchecked. The 'Working sets' field is empty. At the bottom, there are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: JdbcMariaDB

☒ Use default location

Location: E:\USCS\Workspace\Workspace_Threads\JdbcMariaDB [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-1.8

☐ Use a project specific JRE: jre1.8.0_241

☐ Use default JRE 'jre1.8.0_241' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

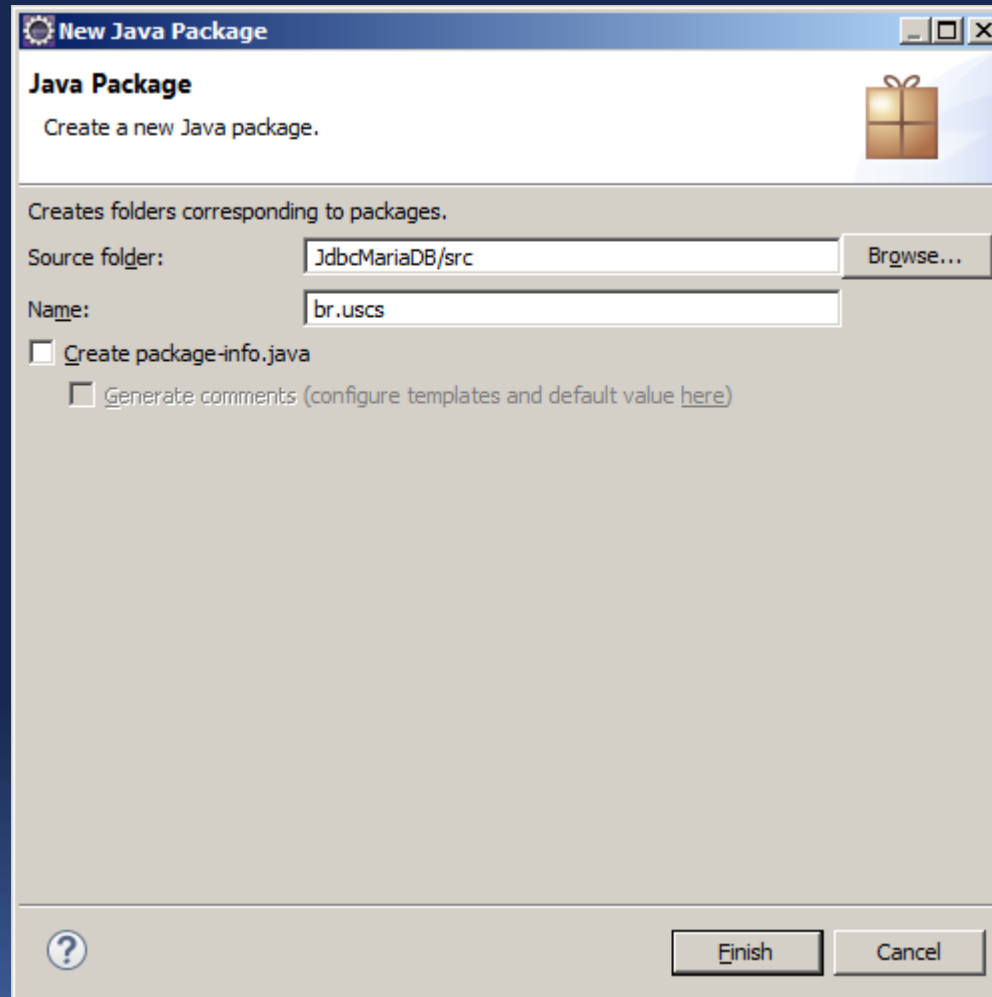
☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

[? < Back Next > Finish Cancel](#)

Exemplo – Driver **MariaDB** – nativo

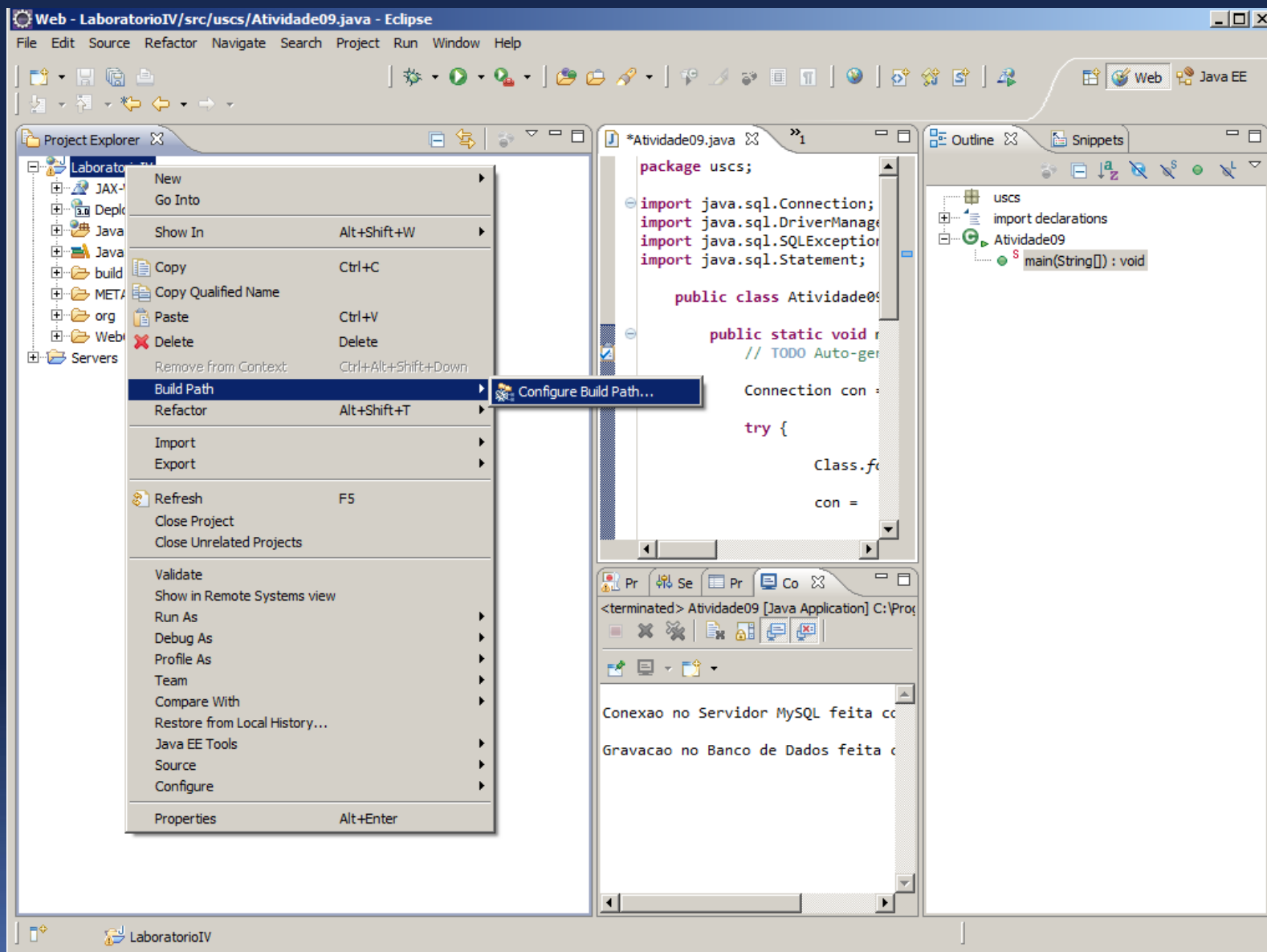
✓ Criar um package



Configuração do Path – Eclipse



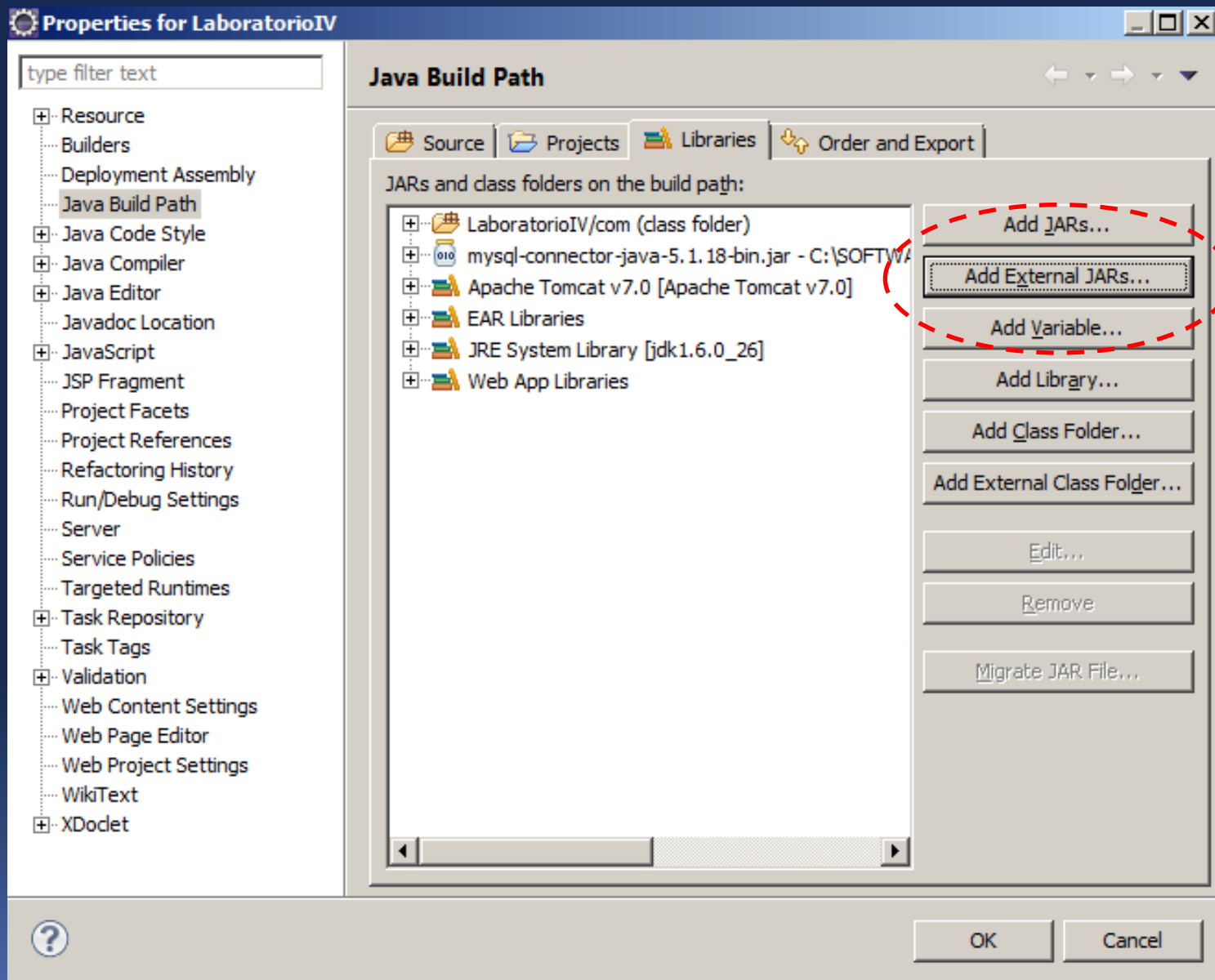
✓ Botão direito no Projeto > Build Path > Configure Build Path ...



Configuração do Path – Eclipse



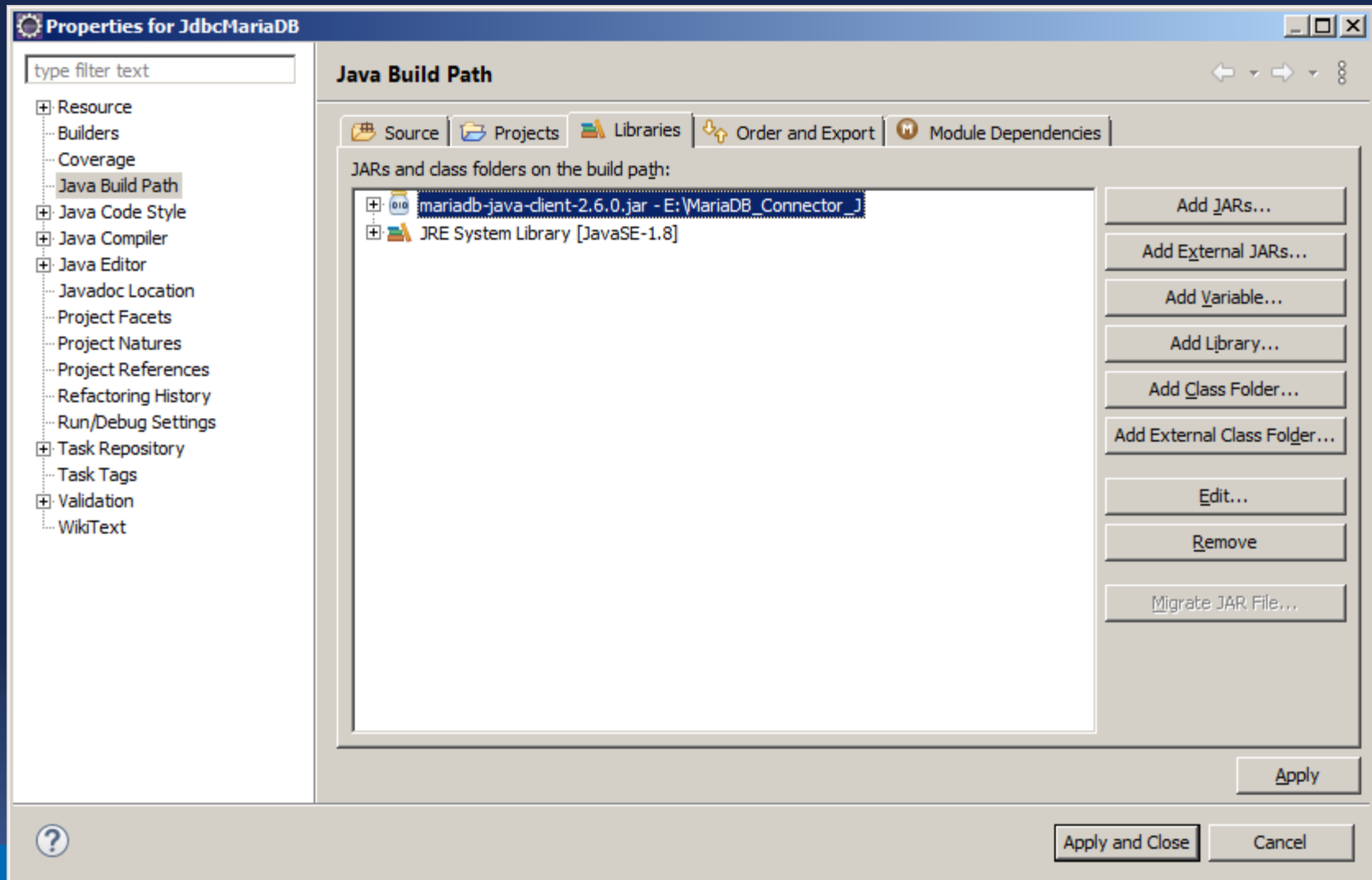
✓ Clicar em Add External JARS... e selecionar o driver JDBC salvo



Configuração do Path – Eclipse



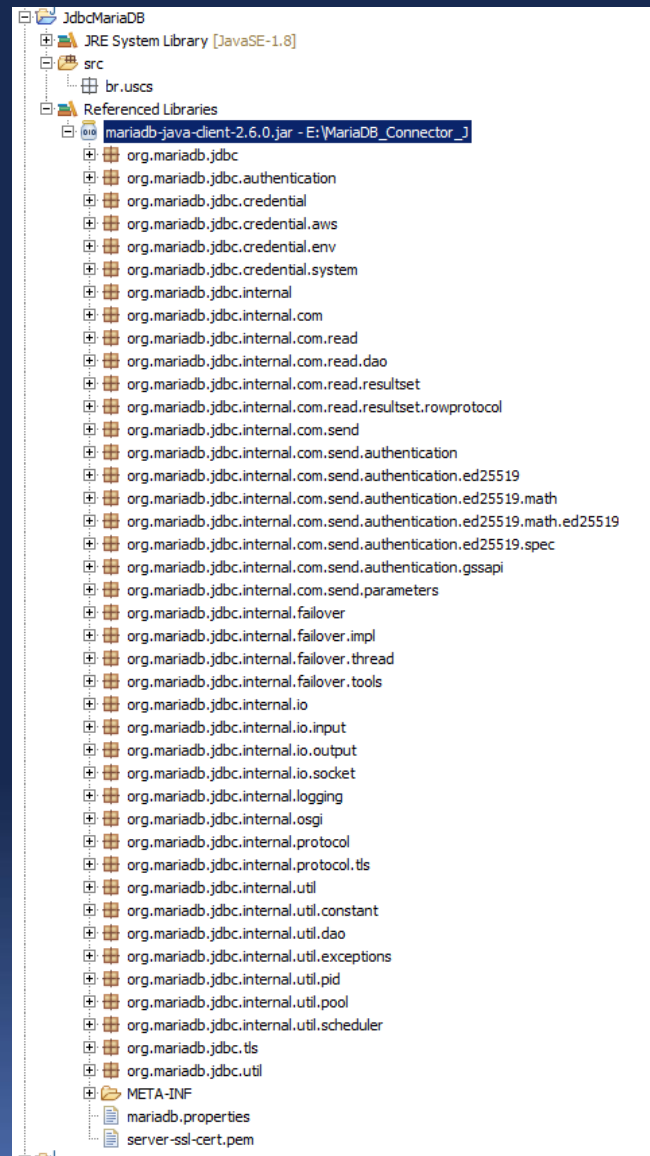
✓ Clicar Apply and Close



Configuração do Path – Eclipse



- ✓ Checar se o driver JDBC está anexado ao Projeto



Configuração do JDBC – MariaDB

- ✓ Verificar na documentação do MariaDB qual o nome do String de conexão JDBC do SGBD MariaDB.

<https://mariadb.com/kb/en/about-mariadb-connector-j/>

Connection Strings

The format of the JDBC connection string is:

```
jdbc:(mysql|mariadb):[replication:|loadbalance:|sequential:|aurora:][//<hostDescription>[,<hostDescription>]
```

HostDescription:

```
<host>[:<portnumber>] or address=(host=<host>)[(port=<portnumber>)][(type=(master|slave))]
```

Some notes about this:

- The host must be a DNS name or IP address.
- If the host is an IPv6 address, then it must be inside square brackets.
- The default port is 3306.
- The default type is master.
- If the failover and load-balancing mode is set to replication, then the connector assumes that the first host is master, and the others are slaves by default, if their types are not explicitly mentioned.

Examples:

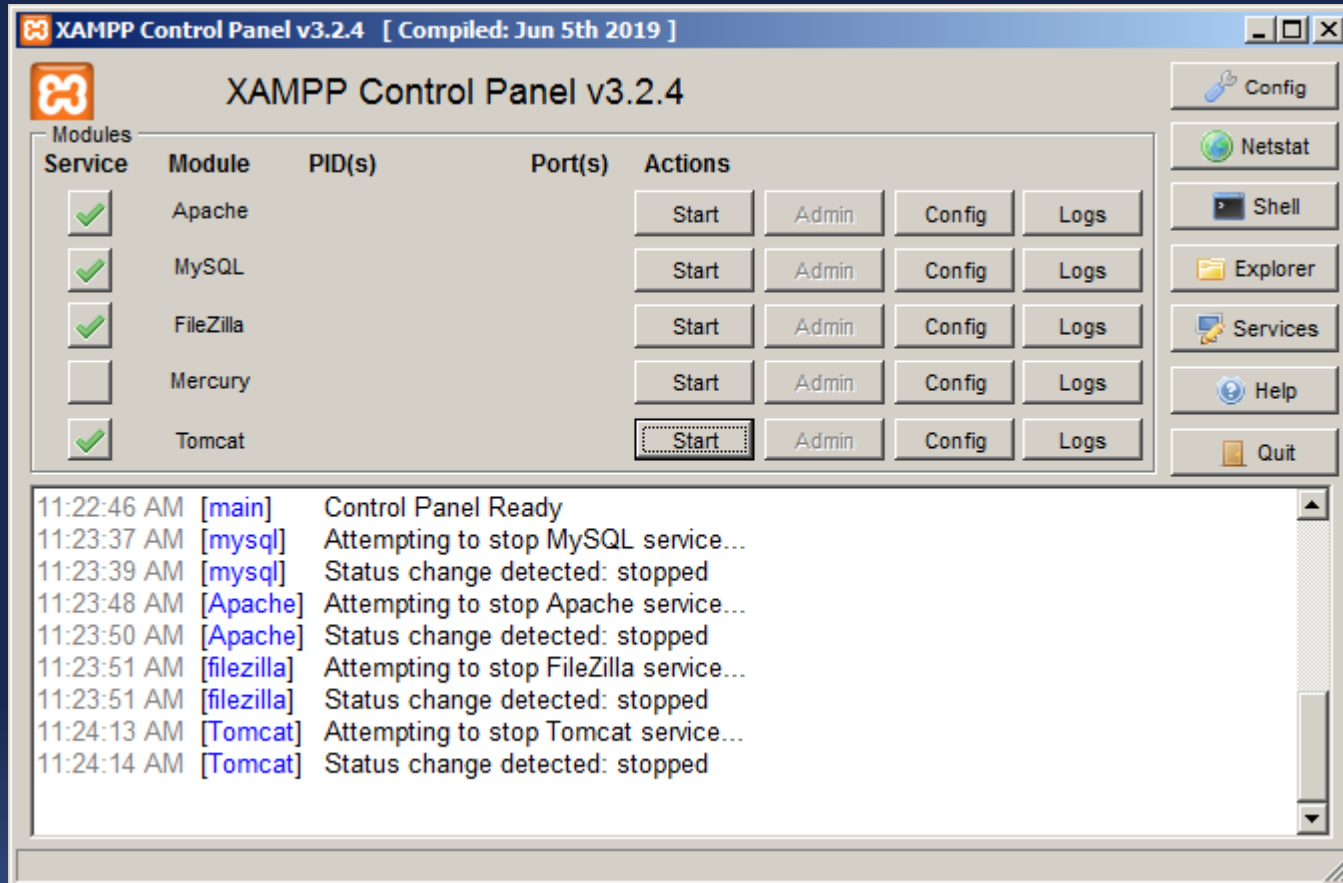
- localhost:3306
- [2001:0660:7401:0200:0000:0000:0edf:bdd7]:3306
- somehost.com:3306
- address=(host=localhost)(port=3306)(type=master)

Exemplo: **"jdbc:mariadb://localhost:3306/db?"+"user=root&password="+"**);



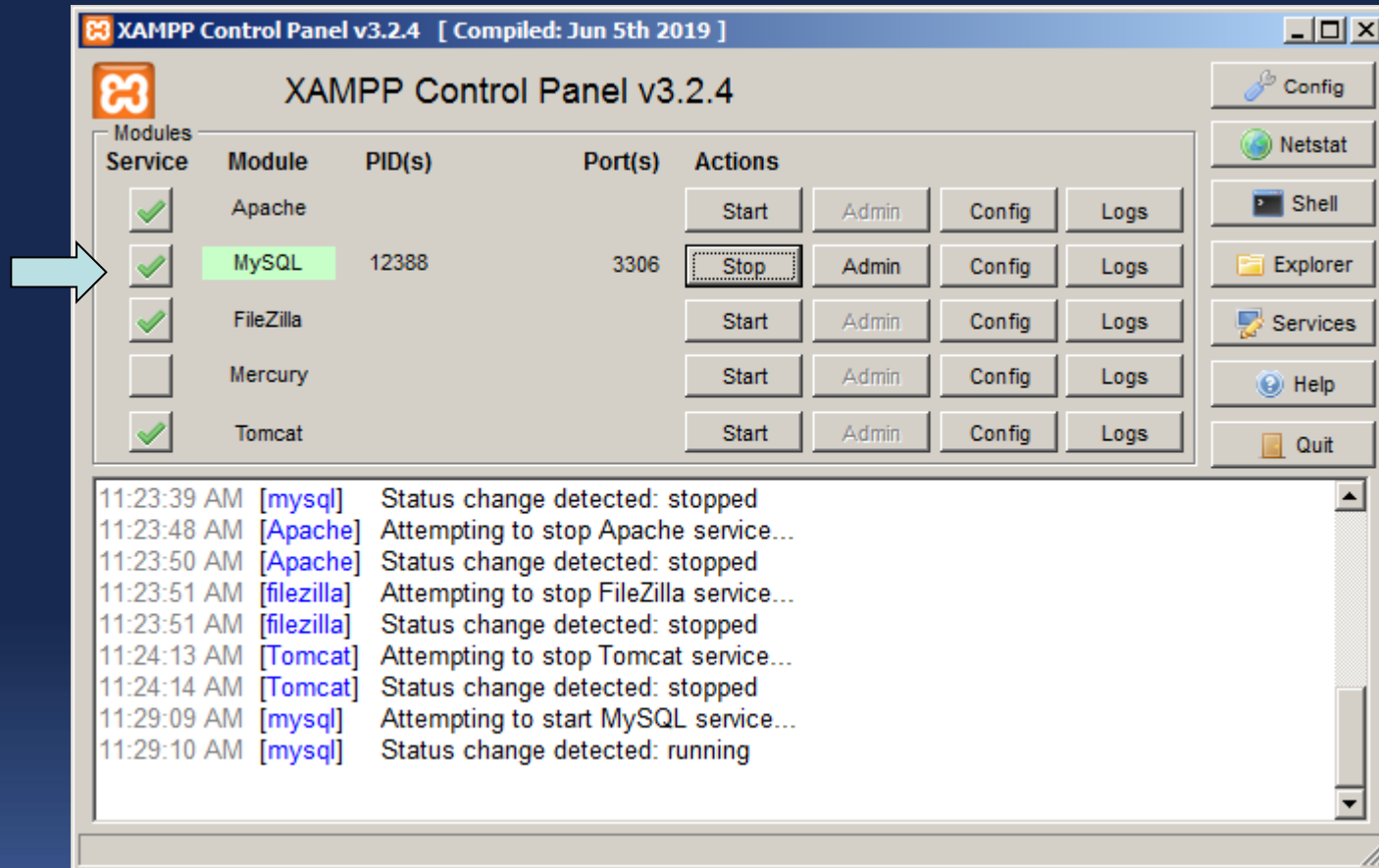
Programa Java para conectar MariaDB

- ✓ Criação do Banco de Dados
- ✓ Ativar XAMPP



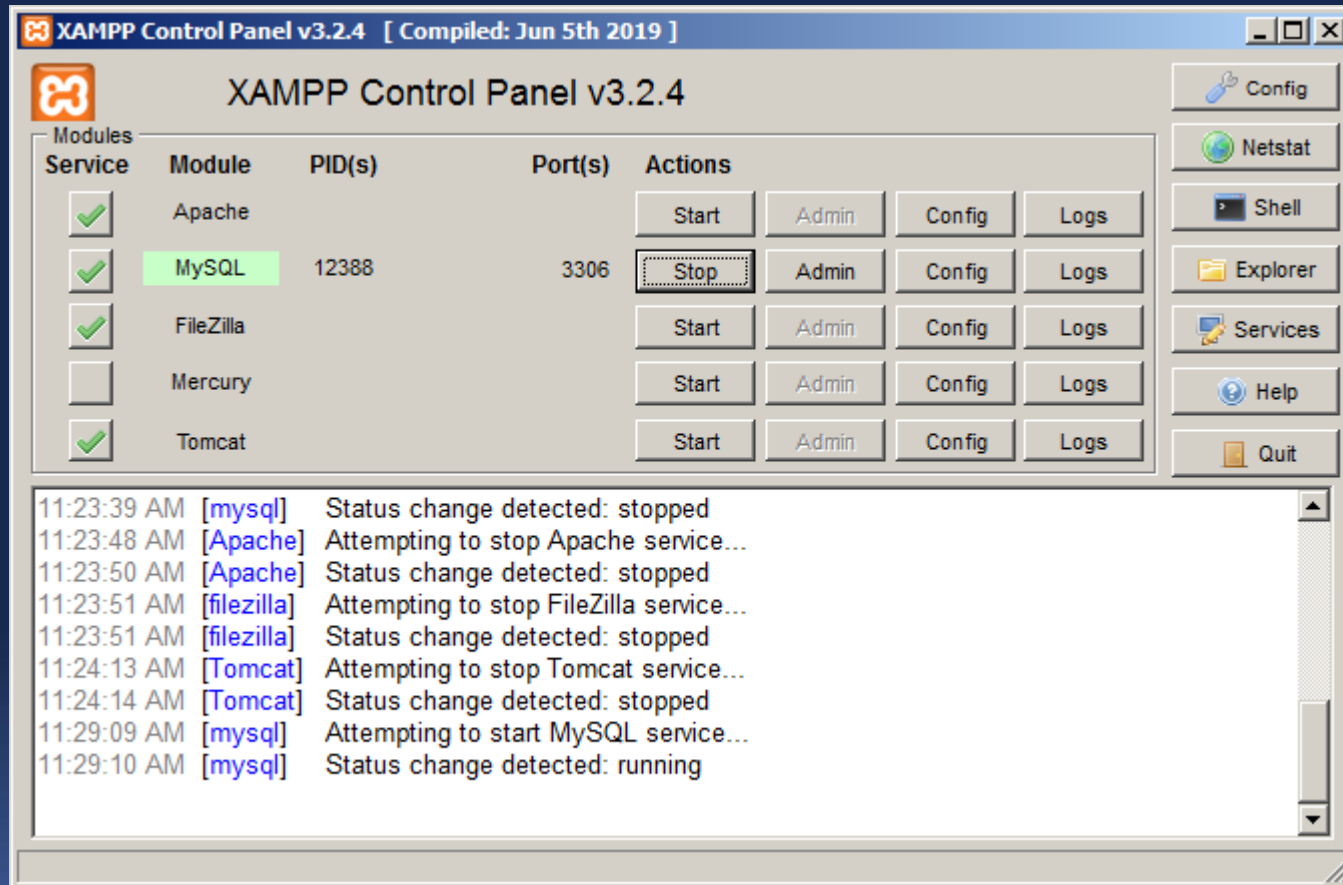
Programa Java para conectar MariaDB

- ✓ Criação do Banco de Dados
- ✓ Ativar serviço MySQL (MariaDB)



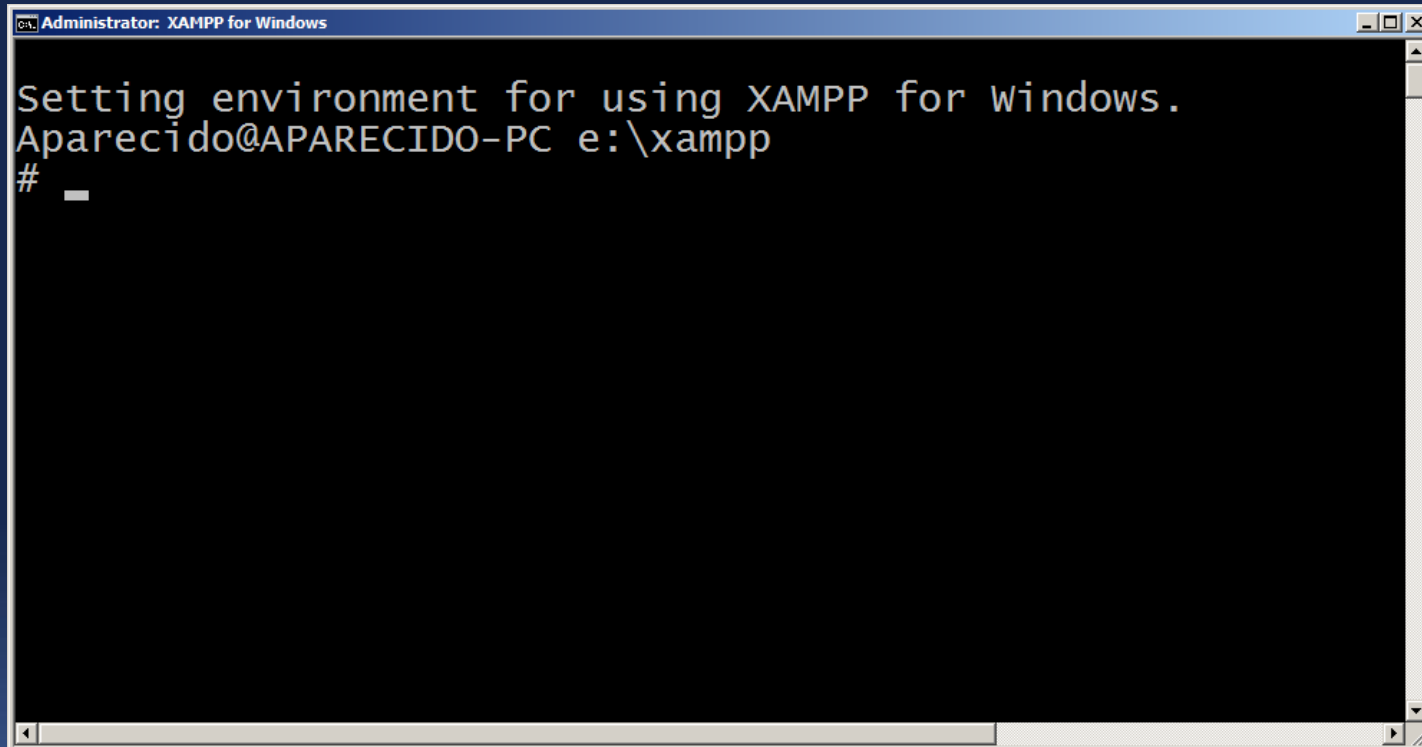
Programa Java para conectar MariaDB

- ✓ Criação do Banco de Dados
- ✓ Conectar MariaDB com user e password , através do Shell



Programa Java para conectar MariaDB

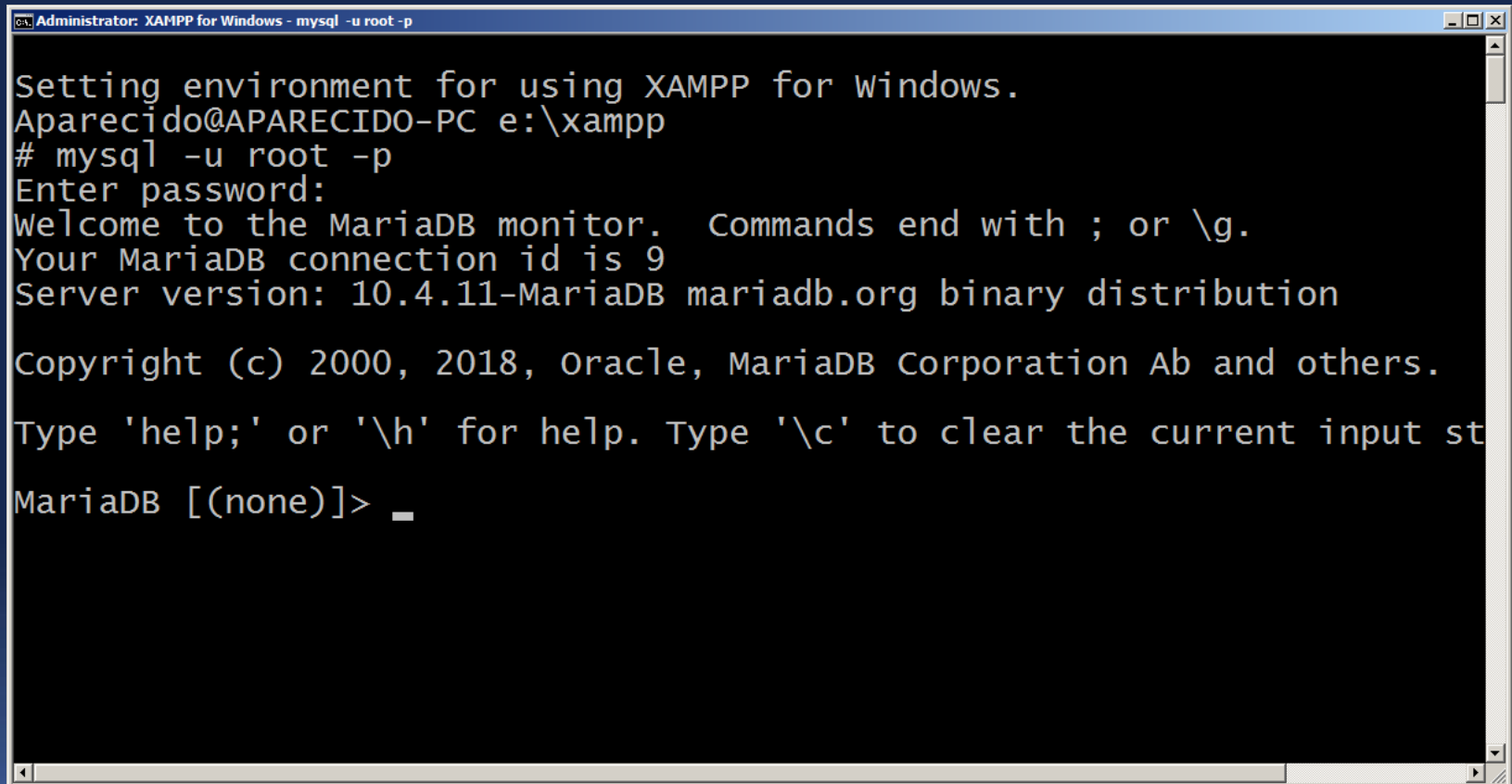
✓ Shell – XAMPP



```
Administrator: XAMPP for Windows
Setting environment for using XAMPP for windows.
Aparecido@APARECIDO-PC e:\xampp
# _
```

Programa Java para conectar MariaDB

- ✓ Fazer conexão client com o MariaDB
- ✓ Comando: `mysql -u root -p`



```
Administrator: XAMPP for Windows - mysql -u root -p
Setting environment for using XAMPP for windows.
Aparecido@APARECIDO-PC e:\xampp
# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.11-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input st
MariaDB [(none)]> _
```

Programa Java para conectar MariaDB

- ✓ Criando um banco de Dados: TesteConexao

```
Administrator: XAMPP for Windows - mysql -u root -p

Setting environment for using XAMPP for windows.
Aparecido@APARECIDO-PC e:\xampp
# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.11-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement

MariaDB [(none)]> create database TesteConexao;
```



Programa Java para conectar MariaDB

✓ Exibindo os bancos de Dados

```
Administrator: XAMPP for Windows - mysql -u root -p

Setting environment for using XAMPP for windows.
Aparecido@APARECIDO-PC e:\xampp
# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.11-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database TesteConexao;
Query OK, 1 row affected (0.003 sec)

MariaDB [(none)]> show databases;
```



Programa Java para conectar MariaDB

✓ Exibindo os bancos de Dados

```
Administrator: XAMPP for Windows - mysql -u root -p
Type 'help;' or '\h' for help. Type '\c' to clear the current input st

MariaDB [(none)]> create database TesteConexao;
Query OK, 1 row affected (0.003 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| db       |
| information_schema |
| mysql    |
| performance_schema |
| phpmyadmin |
| scpe     |
| test     |
| testeconexao |
+-----+
8 rows in set (0.019 sec)

MariaDB [(none)]>
```

Programa Java para conectar MariaDB

✓ Abertura do Banco de Dados

```
Administrator: XAMPP for Windows - mysql -u root -p
MariaDB [(none)]> create database TesteConexao;
Query OK, 1 row affected (0.003 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| db        |
| information_schema |
| mysql     |
| performance_schema |
| phpmyadmin |
| scke      |
| test      |
| testeconexao |
+-----+
8 rows in set (0.019 sec)

MariaDB [(none)]> use testeconexao
Database changed
MariaDB [testeconexao]>
```



- ✓ Criação da tabela: TabProduto

```
Administrator: XAMPP for Windows - mysql -u root -p
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] >
MariaDB [testeconexao] > create table tabProduto (codigoProduto int, descProduto
varchar(50) , primary key(codigoProduto));
Query OK, 0 rows affected (0.048 sec)

MariaDB [testeconexao] > _
```



Programa Java para conectar MariaDB

✓ Verificação da Estrutura da Tabela

```

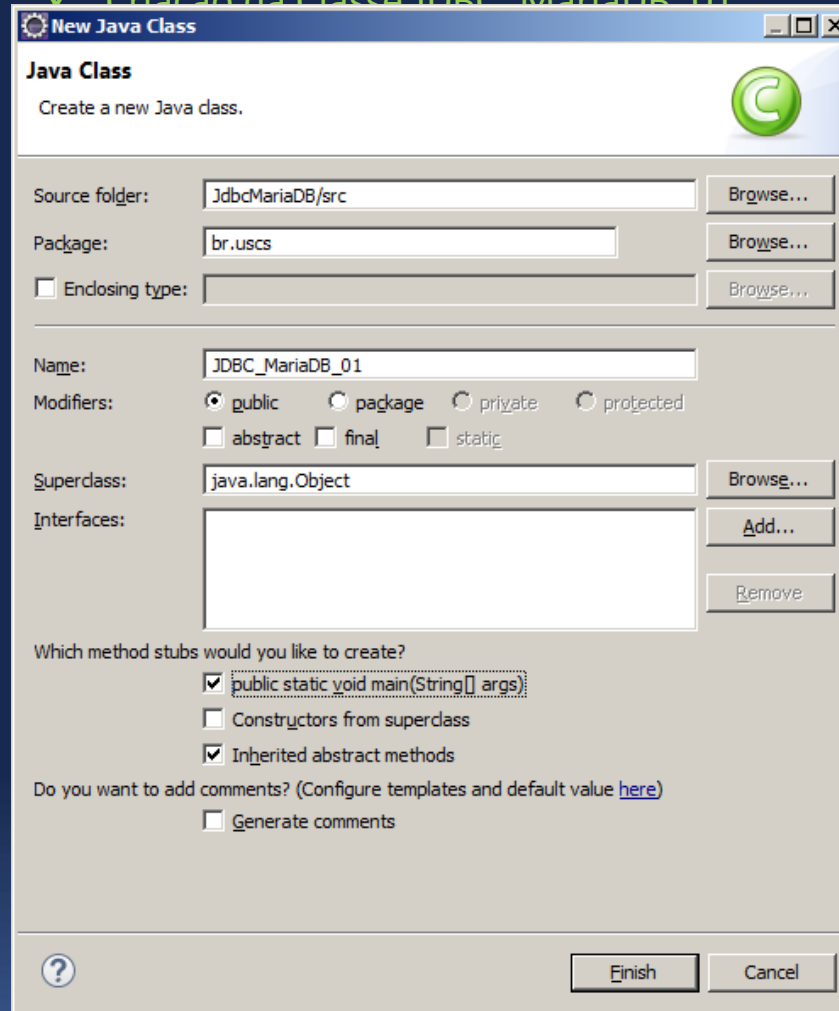
Administrator: XAMPP for Windows - mysql -u root -p
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]>
MariaDB [testeconexao]> describe tabproduto;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| codigoProduto  | int(11)       | NO   | PRI | NULL    |       |
| descProduto    | varchar(50)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.020 sec)

MariaDB [testeconexao]>

```

Programa Java para conectar MariaDB

✓ Criação da Classe JDBC_MariaDB_01



New Java Class

Create a new Java class.

Source folder: JdbcMariaDB/src Browse...

Package: br.uscs Browse...

☐ Enclosing type: Browse...

Name: JDBC_MariaDB_01

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add...
Remove

Which method stubs would you like to create?

☒ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

? Finish Cancel

✓ código Java para conexão ao MariaDB

```
package br.uscs;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class JDBC_MariaDB_01 {


    public static void main(String[] args) {

        Connection conexao;
        String url = "jdbc:mariadb://localhost:3306/testeconexao";
        String user = "root";
        String password = "";

        try {
            conexao = DriverManager.getConnection(url, user, password);
            System.out.println("\nMariaDB: Conexao ao Banco de Dados testeconexao OK...");
        }
        catch (SQLException ex) {

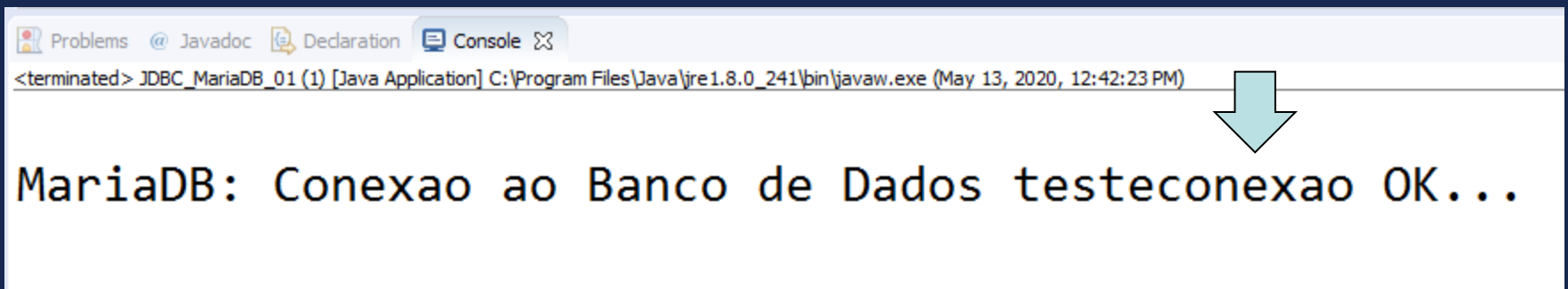
            System.out.println ("**** ERRO DE ACESSO AO BANCO DE DADOS...\n");
            System.out.println ("****SQLException: " + ex);

        }
        catch (Exception ex) {
            System.out.println("*****Exception: " + ex);
        }
    }
}
```



Programa Java para conectar MariaDB

✓ Testando a conexão com o MariaDB



The screenshot shows an IDE console window with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of a Java application. The output text is "MariaDB: Conexao ao Banco de Dados testeconexao OK...". A large blue arrow points from the console output area down to the text "MariaDB: Conexao ao Banco de Dados testeconexao OK...".

```
<terminated> JDBC_MariaDB_01 (1) [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (May 13, 2020, 12:42:23 PM)
```

MariaDB: Conexao ao Banco de Dados testeconexao OK...

Uma vez conectado ao BD, como enviar comandos para o SGBD?





Comandos SQL

- Comandos **SQL** podem ser diretamente enviados ao **SGBD** por meio de um objeto instanciado por uma classe que implemente a interface **Statement**;
- Comandos de definição de dados (**DDL**) e de consultas são aceitos;
- Há dois tipos básicos de comandos **SQL**:
 - **Statement**: Envia texto **SQL** ao **SQGD**;
 - **PreparedStatement**: Pré-compila o texto **SQL**, com posterior envio ao **SGBD**;



Statements

- ✓ Um objeto **Statement** é uma espécie de canal que envia comandos **SQL** através de uma conexão;
- ✓ O mesmo **Statement** pode enviar vários comandos;
- ✓ Para se criar um **Statement**, é preciso ter criado anteriormente um objeto **Connection**;
- ✓ A partir de uma conexão, pode-se criar diversos objetos **Statement**.



Criação do objeto Statement

- O objeto **Statement** será responsável pelo envio dos comandos **SQL** ao **DBMS**.
- Este objeto é criado pelo método **createStatement()** executado pelo objeto **Connection**.

```
Connection dbCon = DriverManager.getConnection (
    "jdbc:odbc:curso",
    "admin",
    "secret" ) ;
```

```
Statement stmt = dbCon.createStatement() ;
```



Como se executa os comandos SQL ?





Executando Statements

- Há dois métodos da classe Statement para envio de comandos ao **SGBD**.
- Modificações: **executeUpdate()**
 - ✓ Para comandos SQL **"INSERT"**, **"UPDATE"**, **"DELETE"**, ou outros que alterem a base de dados e não retornem dados;
 - ✓ Forma geral: **executeUpdate(<comando>);**
 - ✓ Exemplo: **stmt.executeUpdate("DELETE FROM Cliente");**





Executando Statements

■ Consultas: **executeQuery()**

- ✓ Para comandos SQL “**SELECT**” ou outros retornem tuplas;
- ✓ Forma geral: **executeQuery(<comando>);**
- ✓ Esse método retorna um objeto da Classe **ResultSet**;
- ✓ Exemplo: **stmt.executeQuery(“**SELECT * FROM Cliente**”);**





Exemplo – Statement

```
Connection conexao = DriverManager.getConnection( "jdbc:postgresql:usuarios");
```

```
Statement stat = conexao.createStatement();
```

```
ResultSet nomes = stat.executeQuery("SELECT nomes FROM pessoas");
```





Exemplo – executeQuery()

- O método **executeQuery()** executa comandos **SQL** do tipo **SELECT**;
- Retorna um objeto do tipo **ResultSet**.

```
Connection    dbCon = DriverManager.getConnection (
                                "jdbc:odbc:curso",
                                "admin",
                                "secret" ) ;

Statement     stmt = dbCon.createStatement () ;

ResultSet     rs = stmt.executeQuery (
    "SELECT nome_curso FROM curso" );
```



Exemplo – executeUpdate()

- O método `executeQuery()` é usado para submeter statements SQL do tipo DML/DDL;
- DML é usado para manipular dados existentes em objetos (por meio de UPDATE, INSERT, DELETE statements).
- DDL é usado para manipular objetos database (CREATE, ALTER, DROP).

```
Statement      stmt = dbCon.createStatement();  
  
stmt.executeUpdate("INSERT INTO tabcurso  
VALUES(1, 'Psicologia')");
```

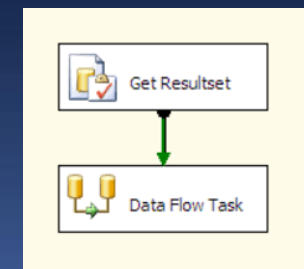
O Objeto ResultSet

- Mantém o posicionamento do cursor em sua corrente linha de dados;
- Provê métodos para recuperar valores de colunas.

```
ResultSet      rs = stmt.executeQuery(  
    "SELECT nome_curso FROM curso");  
  
while (rs.next() ) {  
  
    String nome_curso = rs.getString("nome_curso");  
    double valor = rs.getDouble("preco");  
  
}
```

Funções de acesso ao ResultSet

- Métodos de acesso aos dados têm duas formas: Uma forma tem um argumento numérico e outra com argumento **String**.
- Quando se fornece um argumento numérico, está se referindo à coluna que corresponde àquele valor.
- Quando se fornece um argumento String se refere à coluna cujo nome corresponde ao String fornecido.



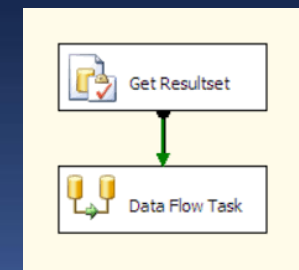
Manipulação de objetos ResultSet

✓ Métodos **getXXX**

- Recuperam um dado de acordo com o tipo;
- Formas: rs.**getXXX**(<nome do campo>) ou rs.**getXXX**(<posição do campo >);
- Exemplo:rs. **getString**("nm_cliente") ou rs.**getString**(2);

✓ Método **next()** , **previous()**

- Retorna para o próximo registro no conjunto ou para o anterior;
- Retornam valor lógico;
- Valor de retorno **true** indica que há outros registros para serem processados.



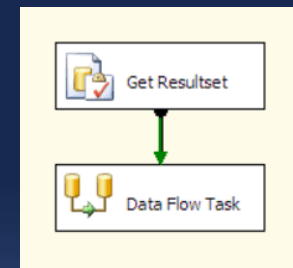
Manipulação de objetos ResultSet

✓ Métodos `first()` , `last()`

- Posicionam o cursor no início ou no final do conjunto de dados;

✓ Método `next()` , `previous()`

- Testam a posição do curso;
- Retornam valor lógico.





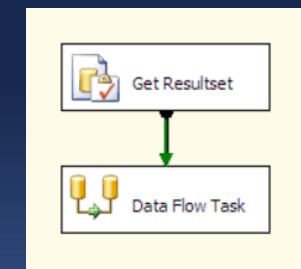
Acessores para tipos Java

- **rs.getString(1)** retorna o valor da primeira coluna na linha corrente.
- **rs.getDouble("Price")** retorna o valor da coluna com nome "Price".



Encerramento da conexão

- ✓ Explicitamente fecham a conexão, por meio da função **close()** aplicada aos objetos **Connection**, **Statement** e **ResultSet**.
- ✓ Este procedimento irá liberar os recursos que não são mais necessários à aplicação.





PreparedStatement

- ✓ Os métodos **executeQuery** e **executeUpdate** da classe **Statement** não recebem parâmetros;
- ✓ **PreparedStatement** é uma subinterface de **Statement** cujos objetos permitem a passagem de parâmetros;
- ✓ Em um comando **SQL** de um objeto **PreparedStatement**:
 - Parâmetros são simbolizados por pontos de interrogação;
 - Configuração dos valores dos parâmetros: métodos **setXXX**

PreparedStatement **pst** =

```
con.prepareStatement("INSERT INTO Clientes (codigo, nome) VALUES (?,?)");  
pst.setInt(1,10);  
pst.setString(2,"Eduardo");
```





PreparedStatement – Exemplo

```
PreparedStatement stat = conn.prepareStatement("SELECT * FROM ?");  
// percorre os funcionários  
stat.setString(1, "Funcionarios");  
ResultSet funcionarios = stat.executeQuery();  
  
.....  
  
// percorre os produtos  
stat.setString(1, "Produtos");  
ResultSet produtos = stat.executeQuery();  
  
.....
```



Atividade – JDBC

Escrever um **programa** desktop que faça uma conexão a um banco de dados e insira um registro. **Acessar o Servidor de Banco de Dados MySQL. Utilizar o Driver JDBC nativo.**

Obs. a) Nome do database: CURSO
b) Nome da tabela: TABCURSO



Código do Curso	Nome do Curso
CODCURSO int(2)	NOMECURSO char(50)

Driver JDBC MySQL nativo

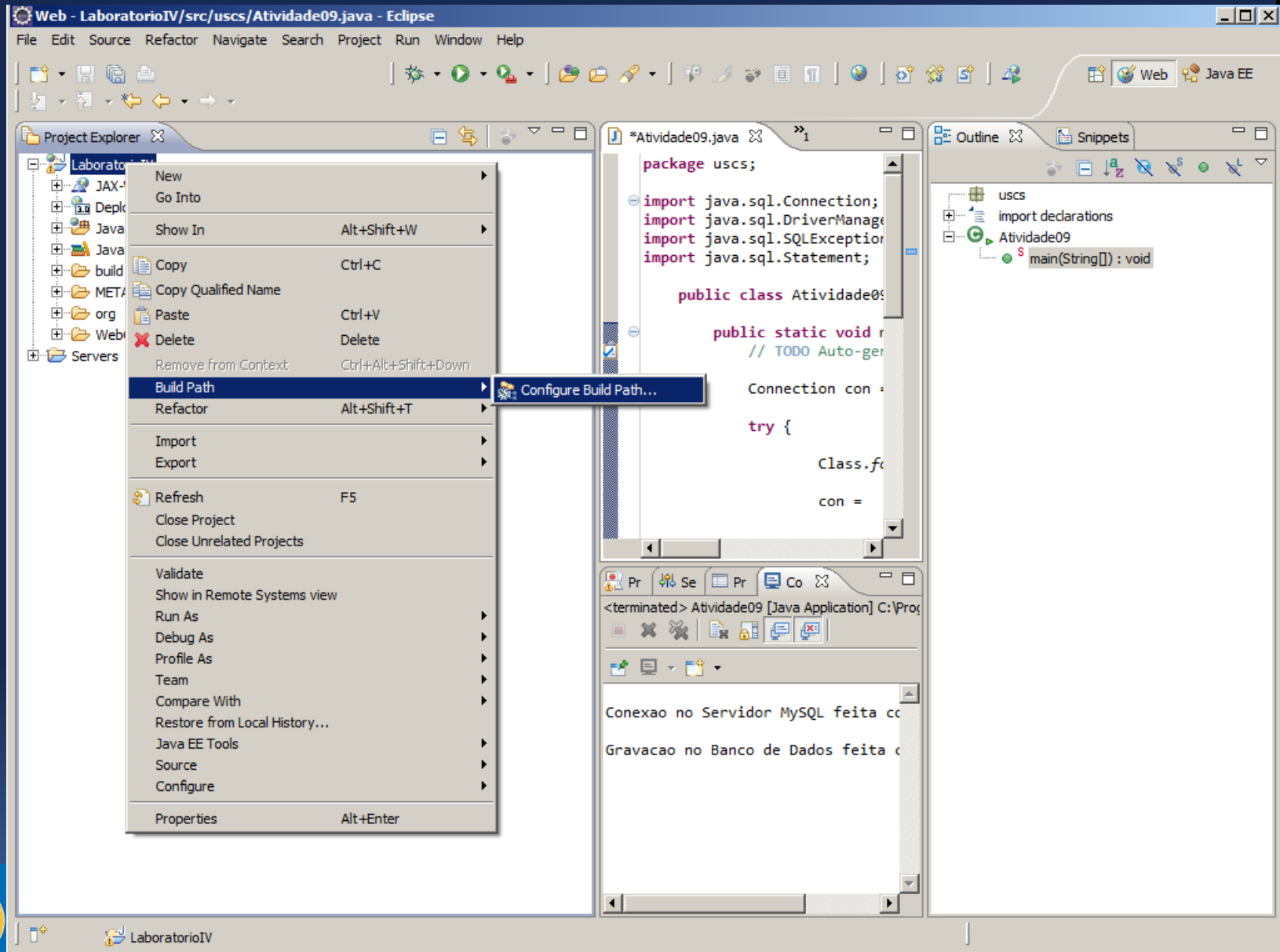
- ✓ Baixar o driver a partir do endereço:

<https://dev.mysql.com/downloads/connector/j/>

- ✓ Salvar em algum diretório do Servidor;
- ✓ Configurar o Path do Eclipse para que o projeto visualize o driver.



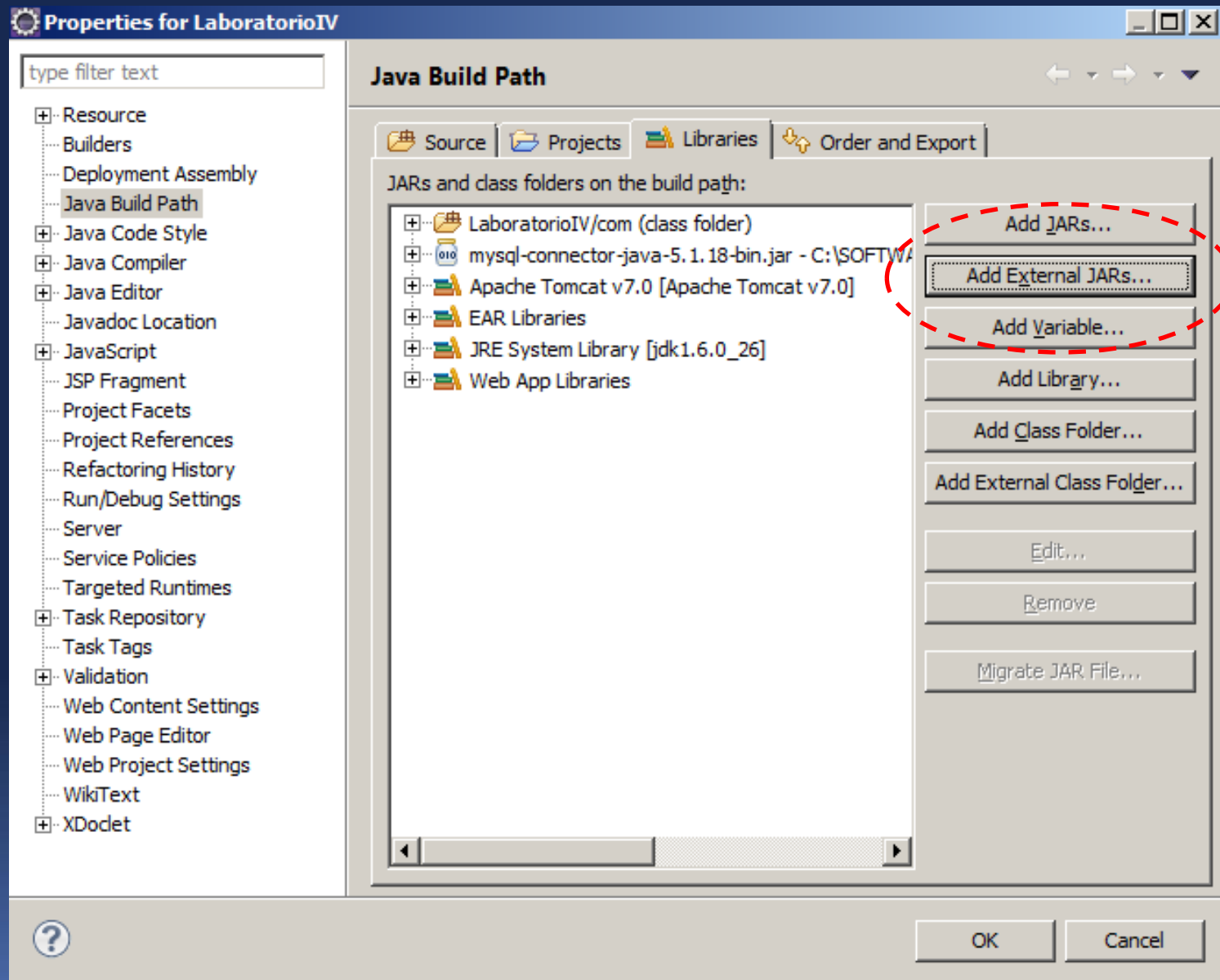
Configuração do Path – Eclipse



The screenshot shows the Eclipse IDE interface with the following components:

- Project Explorer:** Displays the project structure. A context menu is open over the 'LaboratorioIV' project, with 'Build Path' selected. A sub-menu is visible showing 'Configure Build Path...' as the option to click.
- Editor:** Shows the source code of 'Atividade09.java'. The code includes imports for JDBC classes and a public class 'Atividade09' with a 'main' method.
- Outline:** Shows the class hierarchy, including 'Atividade09' and its 'main' method.
- Console:** Displays the output of the application, showing messages like 'Conexao no Servidor MySQL feita com sucesso' and 'Gravacao no Banco de Dados feita com sucesso'.

Configuração do Path – Eclipse



Parâmetros de Conexão

```
Connection con = null;
```

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
```

```
con = DriverManager.getConnection("jdbc:mysql://localhost/curso?" +  
    "user=root&password="+ "" );
```





```
package uscs;
```

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

```
public class Atividade_MySQL {
```

```
    public static void main(String[] args) {  
        // TODO Auto-generated method stub
```

```
        Connection con = null;
```

```
        try {
```

```
            Class.forName(" com.mysql.jdbc.Driver ").newInstance();
```

```
            con = DriverManager.getConnection( " jdbc:mysql://localhost/curso?" +  
                " user=root&password="+ "" );
```

```
            System.out.println("\nConexao no Servidor MySQL feita com sucesso...");
```





```
Statement stmt = con.createStatement();
String command = "INSERT INTO tabcurso VALUES(2,'Matematica')";

stmt.executeUpdate(command);
System.out.println("\nGravacao no Banco de Dados feita com sucesso...");

}

catch (SQLException ex) {

    System.out.println ("**** ERRO DE ACESSO AO BANCO DE DADOS...\n");
    System.out.println ("****SQLException: " + ex);

}

catch (Exception ex) {

    System.out.println("*****Exception: " + ex);

}

}

}
```

