FIAP GRADUAÇÃO

FIAP

17. JDBC

AGENDA

- JDBC
- CLASSES PARA CONEXÃO
- SQL INJECTION
- DESCANSO

JDBC

Java DataBase Connectivity

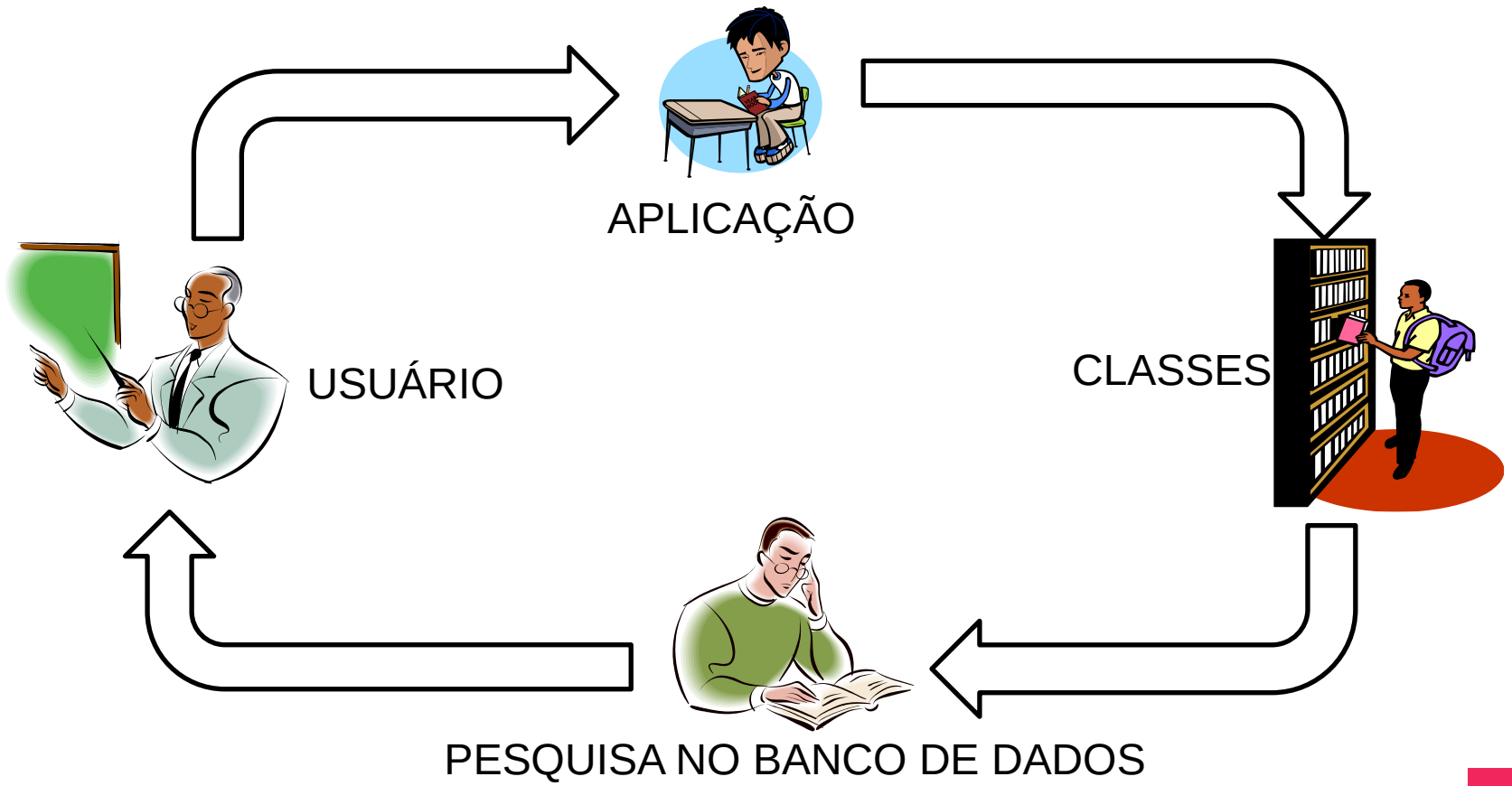
- Criada em 1996 pela Sun. Trata-se de um pacote (JAR) embutido no JDK a partir da versão 1.1.

- JDBC é um sinônimo para Database Access from Java. Trata-se de uma API para acesso a banco de dados.

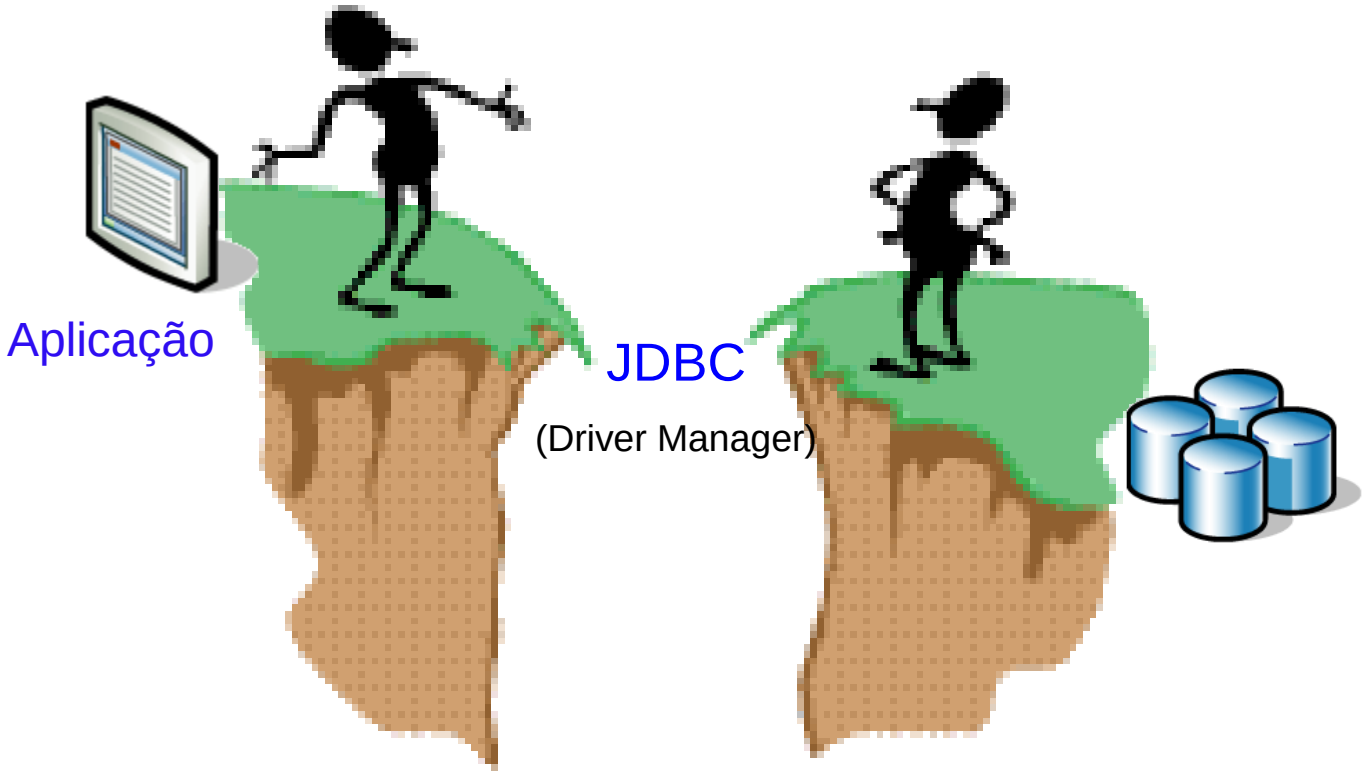
- Nas primeiras versões do JAVA não existia JDBC, era necessário criar uma conexão via SOCKET ou através de bibliotecas escritas em linguagem C.

JDBC

Classes para conexão



JDBC



O que é uma conexão JDBC?

Uma conexão representa uma ligação com determinado banco de dados.

Realiza-se a conexão através de duas etapas. Inicialmente apontamos para o driver desejado através da chamada ao DriverManager e em seguida obtemos a conexão atráves da inicialização da classe Connection.

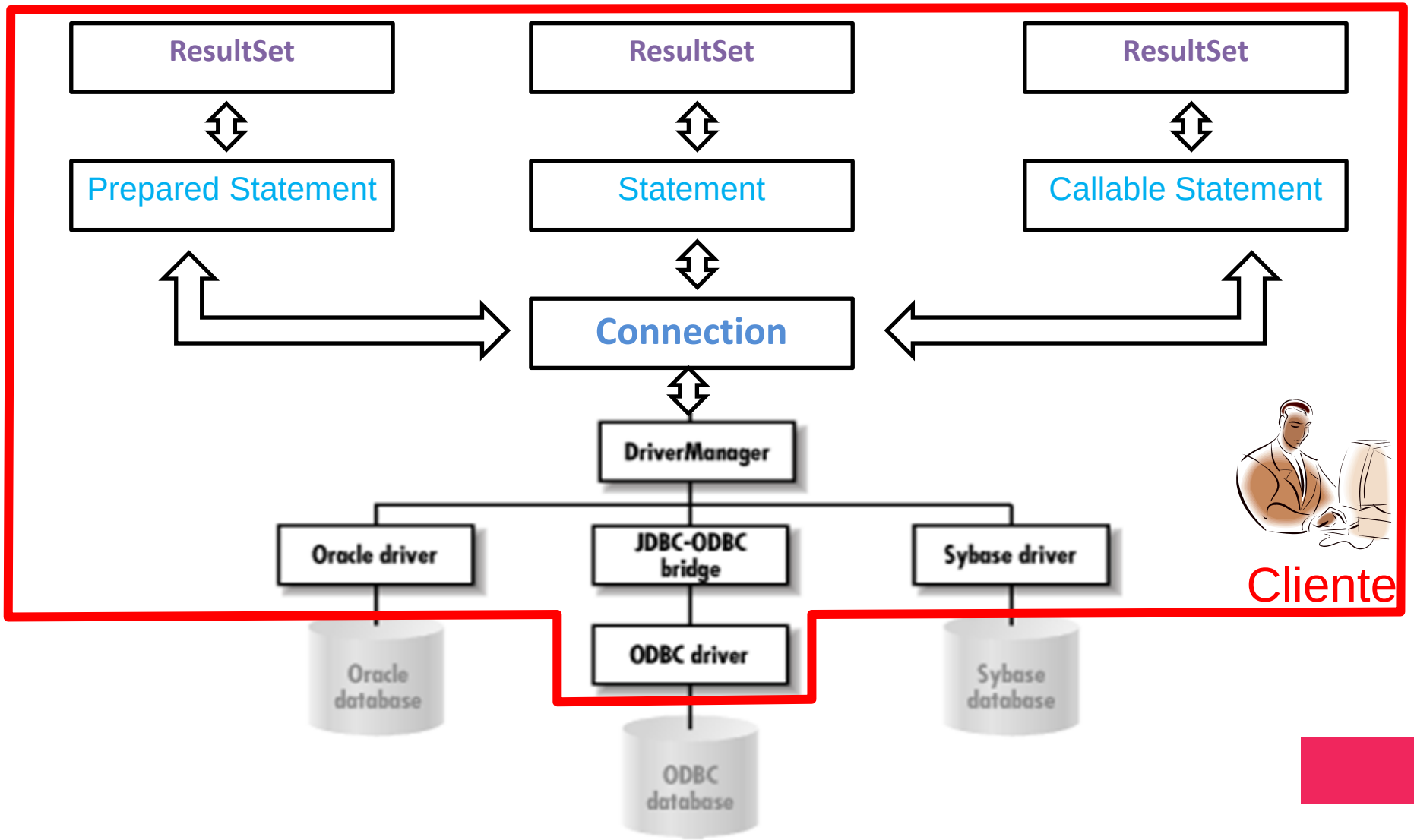
Como fazer o acesso ao BD usando JDBC ?

1. Abrir uma conexão com o BD;

2. Enviar instruções SQL ao BD;

3. Processar o resultado.

JDBC – Classes para Conexão



JDBC - CONEXÃO

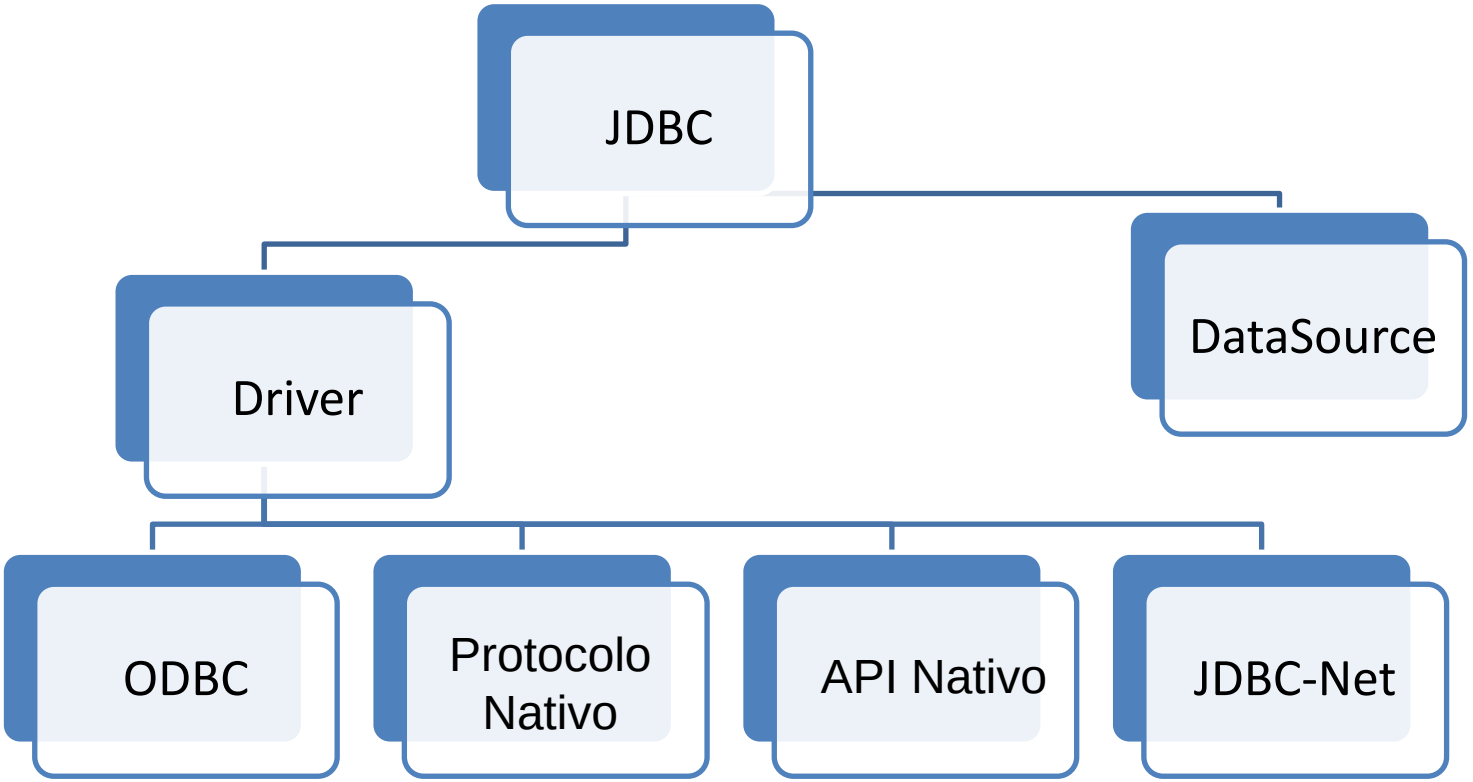
Existem 2 tipos de conexão.

1. Via Driver

2. Via Fonte de Dados (DataSource)

   Este tipo de conexão faz uso do Driver.

JDBC – CONEXÃO VIA DRIVER



Tipo	100% Java	Protocolo de rede	Facilidade de uso
JDBC-ODBC	Não	Direto	Simples. Ideal para aprendizado.
Native API	Não	Direto	Depende que o BD forneça acesso.
JDBC-NET	Sim	Requer conector	Depende de rede.
Protocolo Nativo	Sim	Direto	Solução mais prática.

JDBC – CONEXÃO VIA DATA SOURCE

1. Arquivo de configuração de datasource: mysql-ds.xml
2. Console administrativo.

```
<datasources>
  <local-tx-datasource>
    <jndi-name>jdbc/estatistica</jndi-name>
    <connection-url>jdbc:mysql://localhost:3306/estatistica</connection-url>
    <driver-class>com.mysql.jdbc.Driver</driver-class>
    <user-name>root</user-name>
    <password></password>
</local-tx-datasource>
</datasources>
```

- name=DefaultDS,service=LocalTxCM
- name=DefaultDS,service=ManagedConnectionFactory
- name=DefaultDS,service=ManagedConnectionPool
- name=JBoss JDBC XATransaction ResourceAdapter,service=RARDeployment
- name=JBoss LocalTransaction JDBC Wrapper,service=RARDeployment
- name=JMS Adapter,service=RARDeployment
- name=JmsXA,service=ManagedConnectionFactory
- name=JmsXA,service=ManagedConnectionPool
- name=JmsXA,service=TxCM
- service=CachedConnectionManager
- service=ConnectionFactoryDeployer
- service=RARDeployer

jboss.jdbc

- service=SQLExceptionProcessor

jboss.jmx

- alias=jmx/rmi/RMIAdaptor
- name=Invoker,protocol=jrmp,service=proxyFactory,type=adaptor
- name=Invoker,type=adaptor

JDBC – DATA SOURCE X DRIVERMANAGER

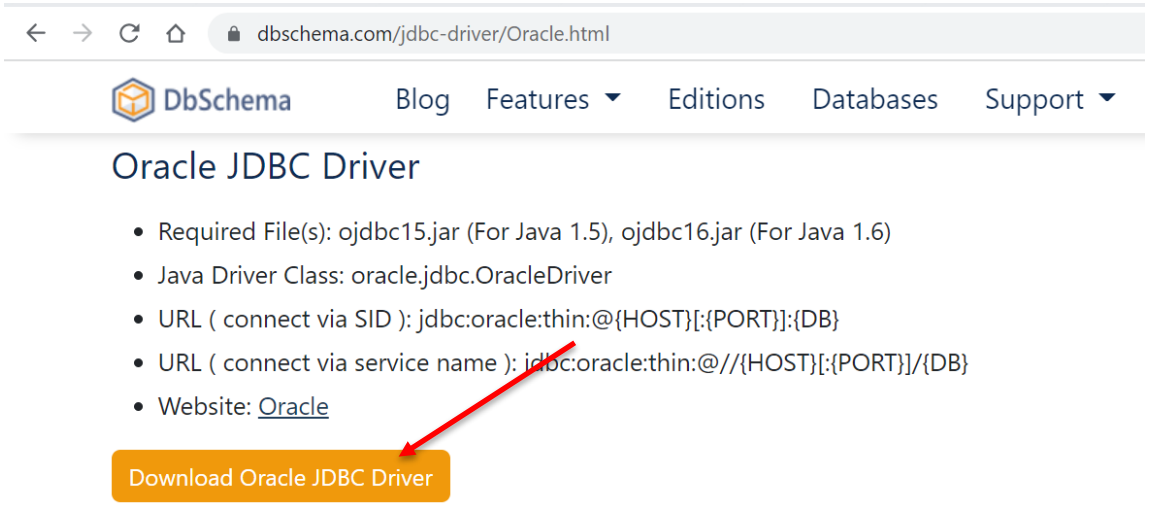
- Data Sources permitem uso mais adequado de conexões (pooling).

- DriverManager é configurado hard-coded enquanto data source roda no servidor.

JDBC – PROTOCOLO NATIVO

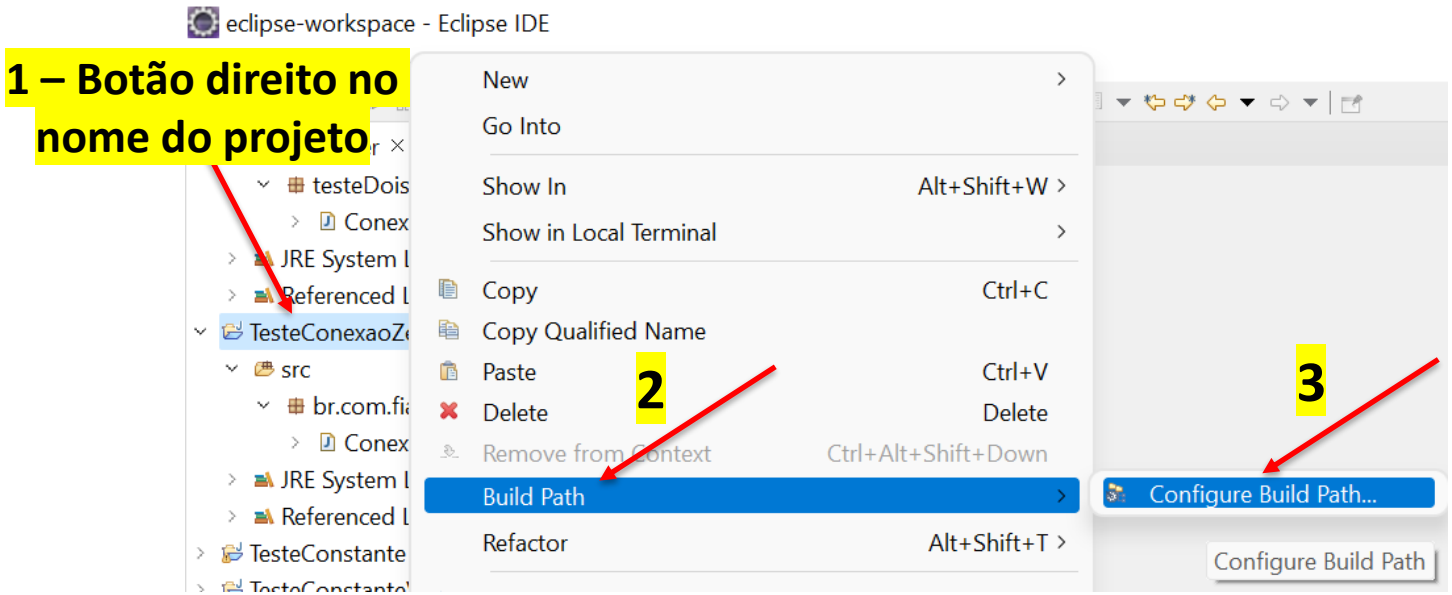
Como instalar no Eclipse o driver JDBC – "jar" dentro do projeto

- Baixe o driver JDBC da Oracle:
  - https://dbschema.com/jdbc-driver/Oracle.html



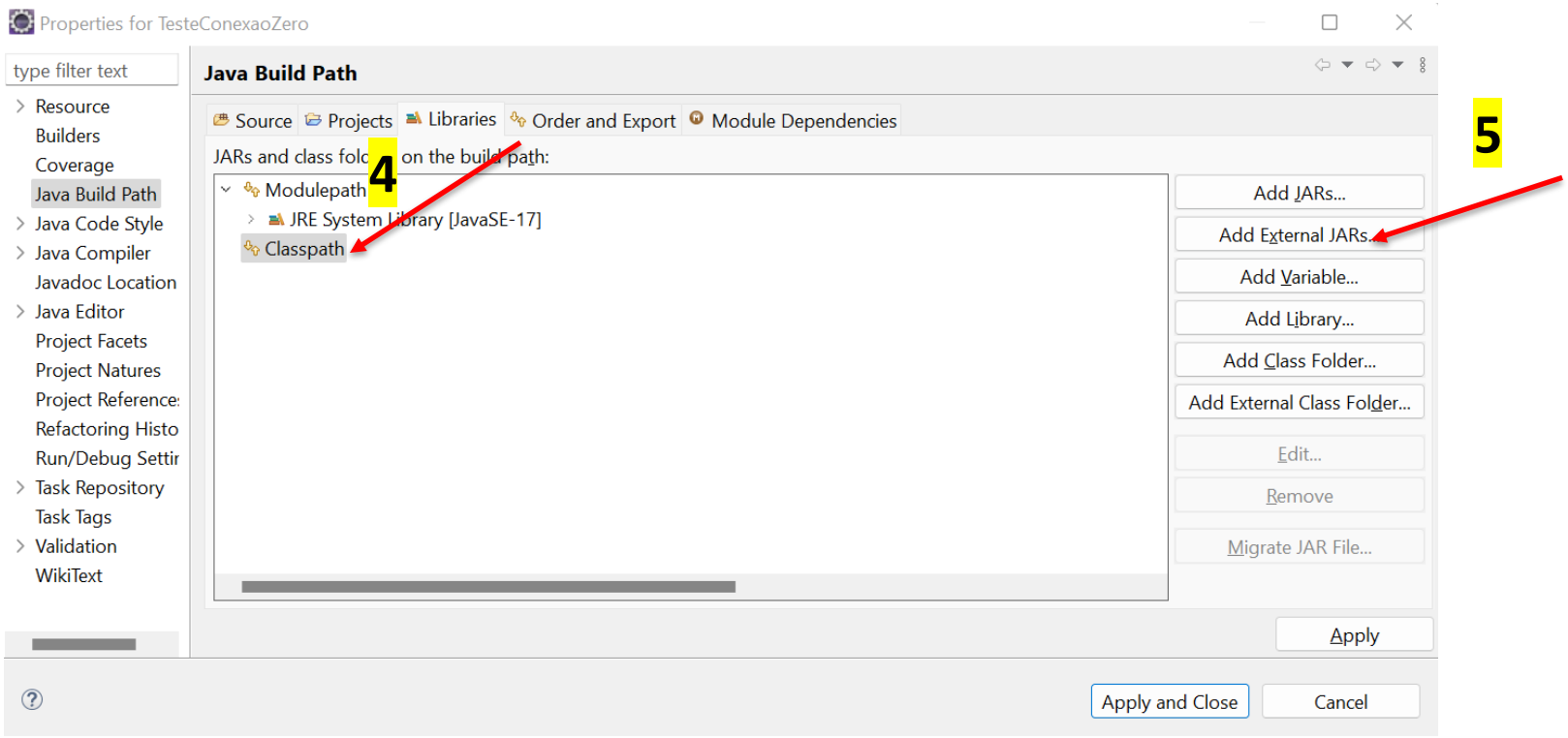
JDBC – PROTOCOLO NATIVO

Como instalar no Eclipse o driver JDBC – "jar" dentro do projeto



JDBC – PROTOCOLO NATIVO

Como instalar no Eclipse o driver JDBC – "jar" dentro do projeto

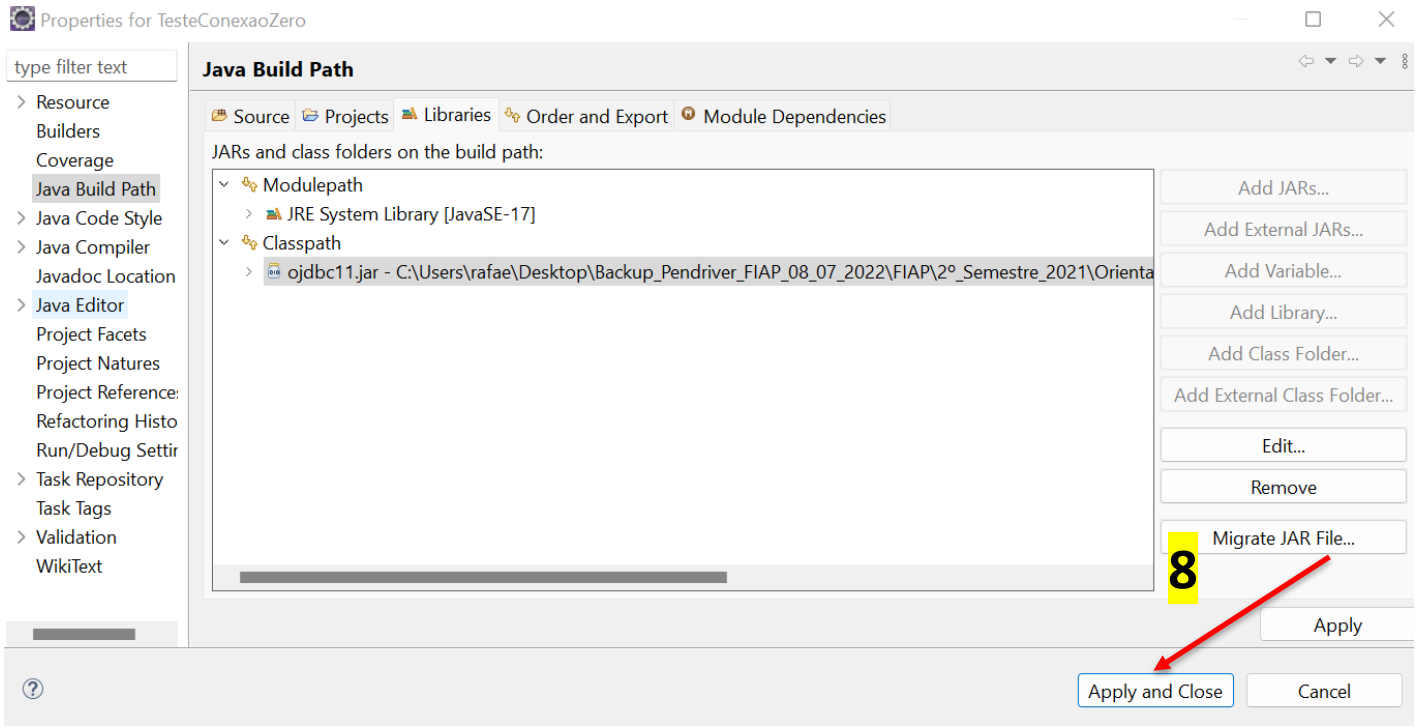


JDBC – PROTOCOLO NATIVO

Como instalar no Eclipse o driver JDBC – "jar" dentro do projeto

JDBC – PROTOCOLO NATIVO

Como instalar no Eclipse o driver JDBC – "jar" dentro do projeto



JDBC - CONEXÃO

Como é feita a conexão ?

Após o registro do driver deve-se inicializar a variável que representa a conexão propriamente dita.

Isso é feito através de uma chamada ao método

DriverManager.getConnection("URL")

jdbc:odbc:votacao

jdbc:oracle:thin:@192.168.60.15:1521:ORCL

jdbc:mysql://localhost:3306/estatistica

JDBC - CONEXÃO

Exemplo de conexão:

TesteConexao.java

```java
package br.com.fiap.testes;

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

import br.com.fiap.excecoes.Excecao;

public class TesteConexao {

    public static void main(String[] args) throws Excecao, SQLException {
        Connection conexao = null;
        String url = "jdbc:oracle:thin:@oracle.fiap.com.br:1521:ORCL";
        conexao = DriverManager.getConnection(url, "PF1153", "Inserir a senha aqui");
        System.out.println("Abriu a conexão.");
        conexao.close();
    }
}
```

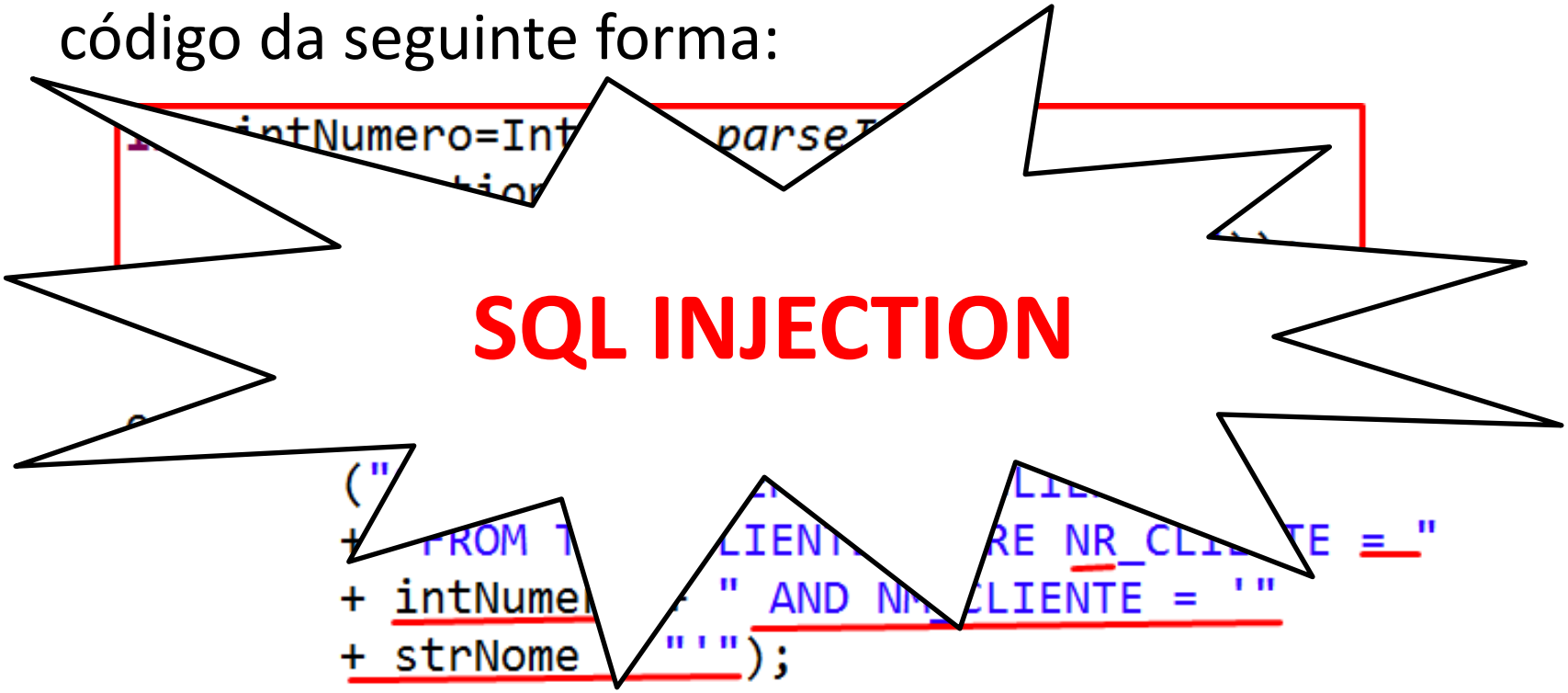
```java
package br.com.fiap.conexao;

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

public class ConexaoFactory {

        public static void main(String[] args) throws SQLException {
                Connection conexao = null;
                String url = "jdbc:oracle:thin:@oracle.fiap.com.br:1521:ORCL";
                conexao = DriverManager.getConnection(url, "pf1153", “Inserir a sua senha do BD aqui");
                System.out.println("Abriu a conexão");
                conexao.close();
        }
}
```

JDBC - CONEXÃO

Mais exemplos e integrados aos projetos serão ensinados em sala de aula

JDBC - CONEXÃO

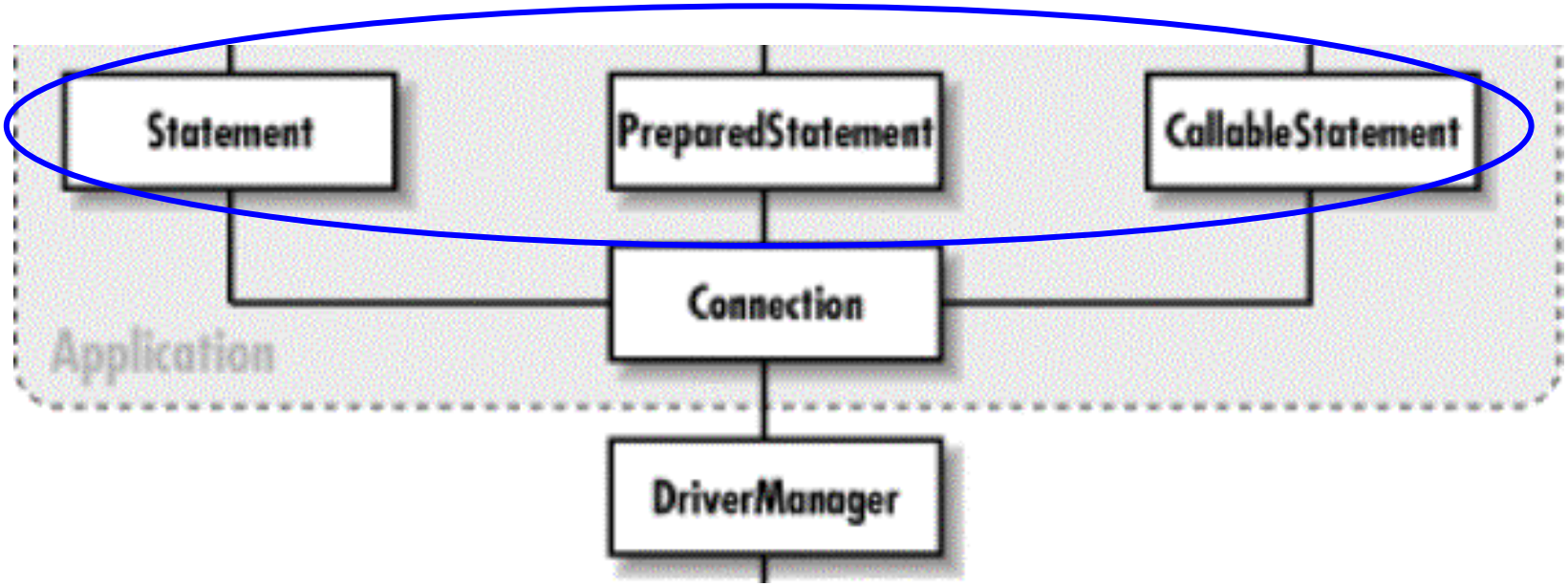
Vamos simular a situação onde o NR_CLIENTE faria o papel da senha e o NM_CLIENTE o login, altere o código da seguinte forma:



Pesquise: strNumero => 123 e strNome => ZE' or '1'='1

JDBC – CLASSES PARA ESTRUTURA

Existem 3 classes (sendo duas derivadas) que permitem criar os objetos utilizados para passagem de comandos SQL ao banco de dados.



JDBC – CLASSES PARA ESTRUTURA

- Statement

  Usado para passagem de declarações SQL simples.

  (SELECT * FROM votacao)

- Prepared Statement

  Usado quando a declaração SQL tiver um ou mais parâmetros que podem ser enviados em conjunto.

  (SELECT * FROM votacao WHERE total_votos > ?)

- Callable Statement

  Usado para execução de funções do tipo Stored Procedures declaradas em alguns tipos de bancos.

  CallableStatement cstmt = conn.prepareCall("{ call proc2 }");

JDBC – CLASSES PARA RESULTADO

- ResultSet

  Uma declaração do tipo SELECT devolve como resultado uma lista que é interceptada pelo JDBC através de um objeto do tipo ResultSet.

```
Statement stmt = conn.createStatement();
String sql = "SELECT CODCARGO,DESCCARGO FROM CARGO";
ResultSet rs = stmt.executeQuery(sql);
```

- ResultSetMetadata e DatabaseMetaData

  O JDBC permite obter informações do BD e de uma tabela através destas interfaces.

```
DatabaseMetaData meta = conn.getMetaData();
String types[] = {"TABLES"};
ResultSet rs = meta.getTables(null,null,"%",types);
while(rs.next()){
      System.out.println(“TABELA ” + rs.getString(3));
}
```

DÚVIDAS...