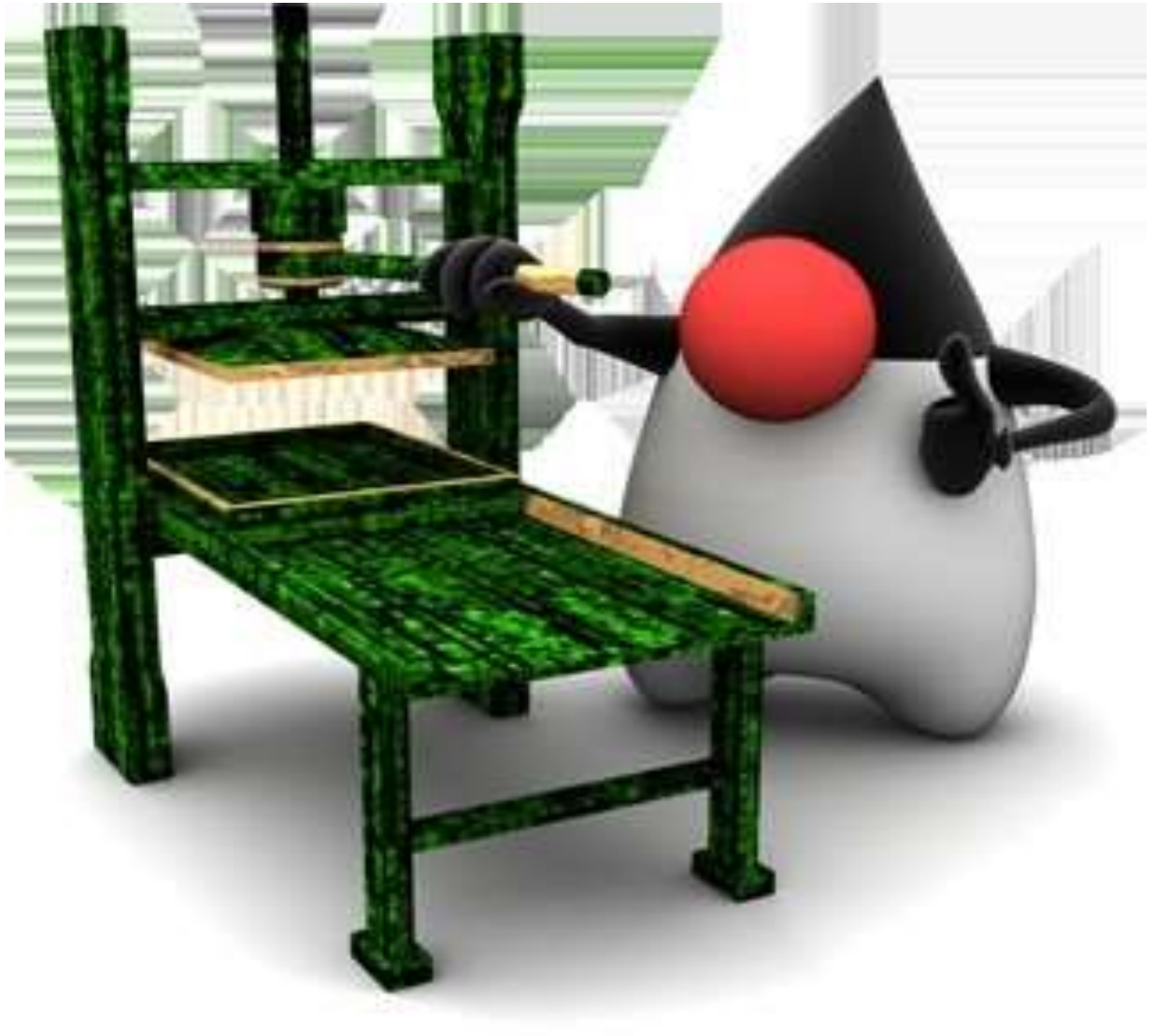
ALPOO – Aplicações de Linguagem de Programação Orientada a Objetos

Prof. Ms. Gustavo Molina

msc.gustavo.unip@gmail.com

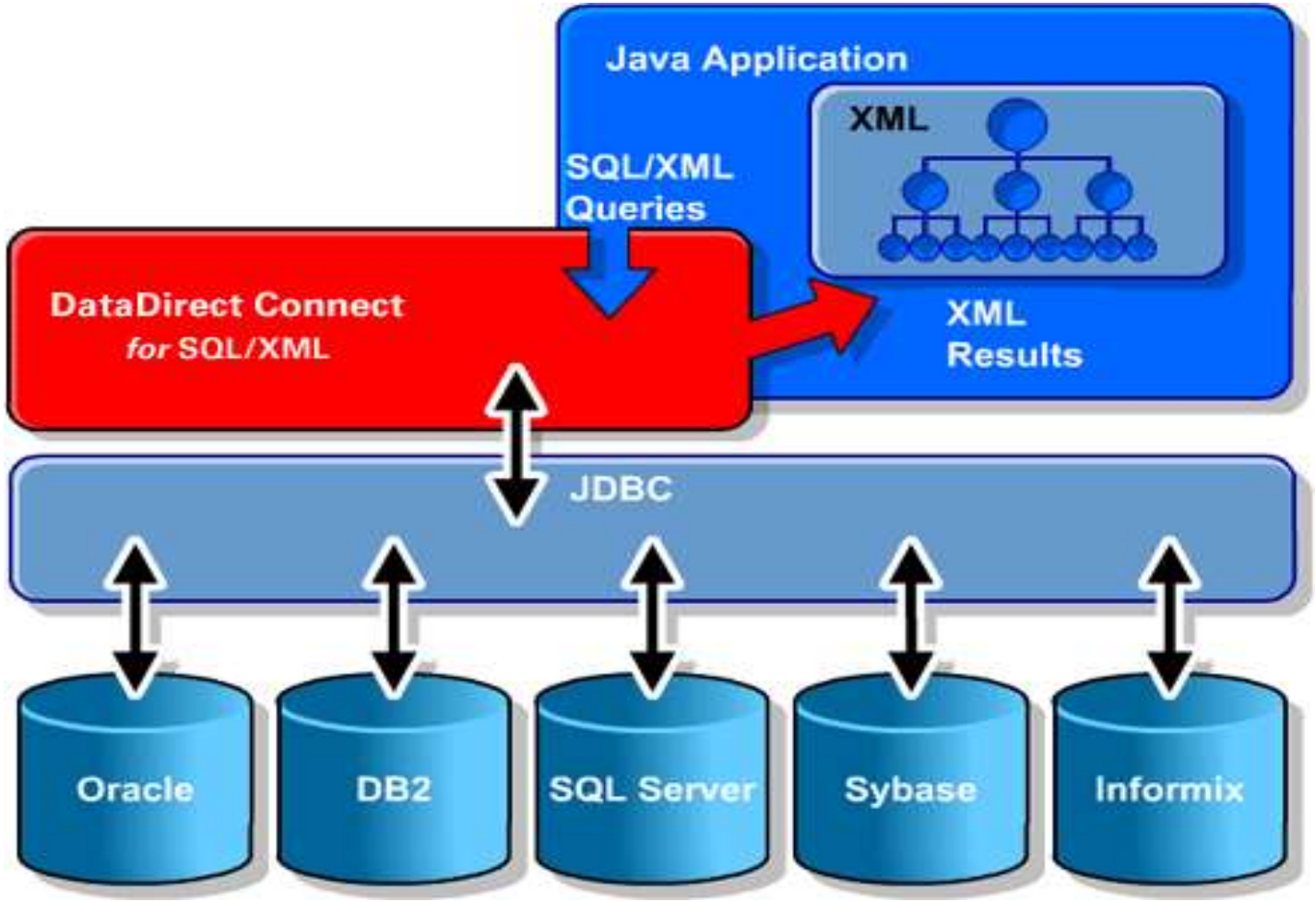
Aula 05 – JDBC



Introdução ao JDBC

- ✓ Os programas Java comunicam-se com BD e manipulam seus dados utilizando a API JDBC (Java DataBase Connectivity).
- ✓ Um driver JDBC permite aos aplicativos Java conectar-se a um BD em um SGBD particular e permite aos programadores manipular esse banco de dados utilizando a API JDBC.
- ✓ O JDBC é quase sempre utilizado com um BD relacional, mas pode ser utilizado com qualquer origem de dados baseada em tabela.

Introdução ao JDBC



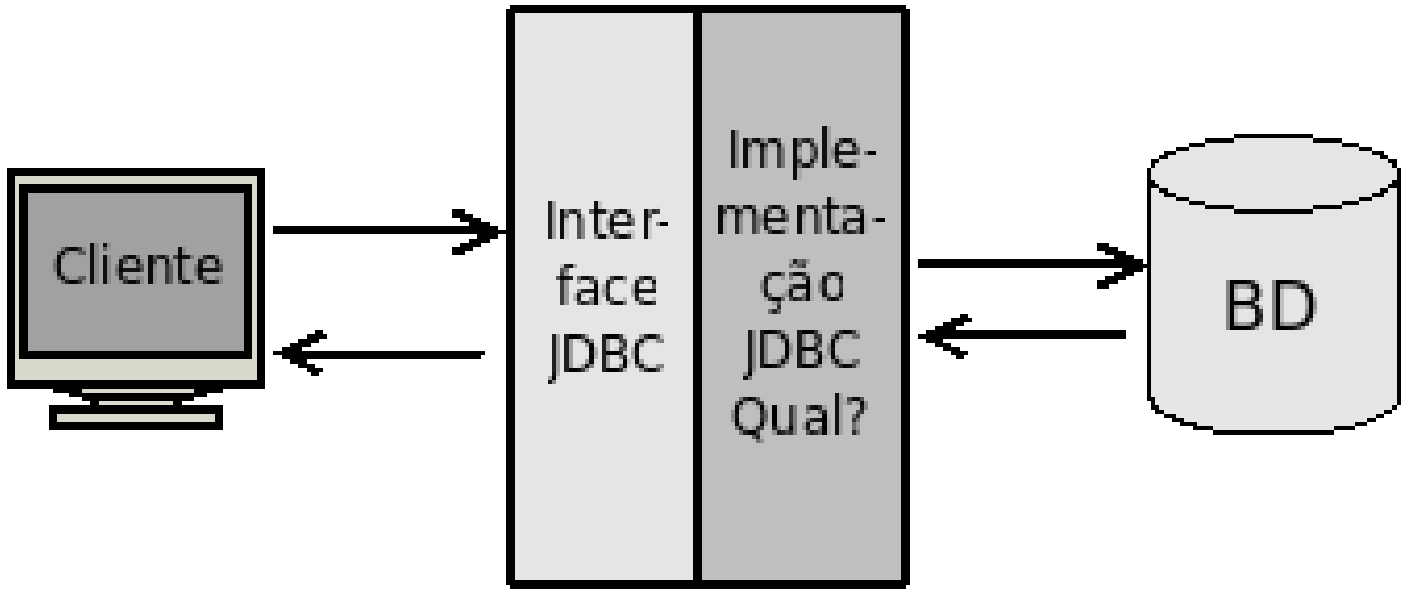
Introdução ao JDBC

- ✓ Java Database Connectivity (JDBC) é uma API definida nos pacotes java.sql e javax.sql utilizada para estabelecer uma conexão com um BD a um programa Java.
- ✓ A API JDBC é um conjunto de classes e interfaces que provê um padrão para tornar possível, aos desenvolvedores de aplicações e ferramentas, a construção de software que acesse BD.
- ✓ Provê acesso universal a dados para a linguagem de programação Java.
- ✓ Permite o acesso a qualquer tipo de fonte de dados (BDR, planilhas e arquivos de dados).

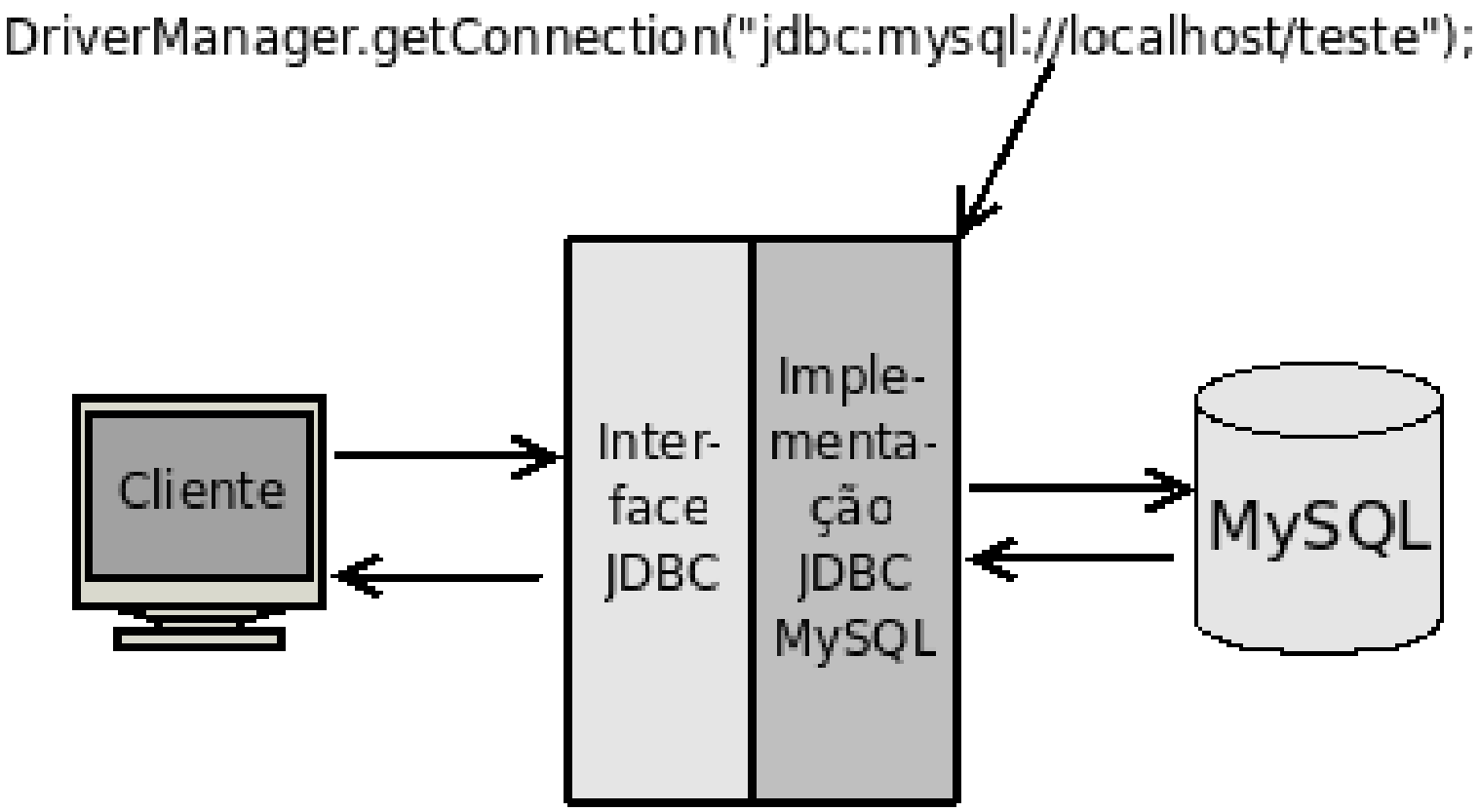
Introdução ao JDBC

- ✓ A arquitetura JDBC é conceitualmente semelhante ao Open Database Connectivity (ODBC), amplamente utilizada em ambiente Windows.
- ✓ No entanto, JDBC é mais flexível, e seu emprego independe do sistema operacional em uso, um princípio fundamental do Java.
- ✓ Através do JDBC, torna-se possível o acesso, genérico e uniforme, a qualquer SGBDR, sendo que os dados são efetivamente acessados por meio de linguagem SQL.
- ✓ Cabe ao desenvolvedor escrever uma interface simples com os elementos da API JDBC para conectar-se ao BD, escolhendo a forma de interação.
- ✓ Com o auxílio do SQL deverão ser especificadas as operações de consulta, inclusão, remoção ou alteração de dados.

Acessando o Banco de Dados



Acessando o Banco de Dados



Pacote Java.SQL

- ✓ Fornece a API para acesso e processamento de dados;
- ✓ Principais classes e interfaces são:

DriverManager

Responsável por criar uma conexão com o BD

Connection

Classe responsável por manter uma conexão aberta com o BD

Statement

Gerencia e executa instruções SQL;

PreparedStatement

Gerencia e executa instruções SQL, permitindo também a passagem de parâmetros em uma instrução

ResultSet

Responsável por receber os dados obtidos em uma pesquisa ao banco.

Conexão com o Bando de Dados

Para abrir uma conexão com um banco de dados é preciso realizar duas tarefas distintas:

1. Verificar a existência do driver de conexão;
2. Solicitar a abertura da conexão.

Conexão com o Bando de Dados

1. Verificando a existência do driver de conexão:

Exemplo com MySQL

```
try {
    Class.forName("com.mysql.jdbc.Driver");
}
catch ( ClassNotFoundException e)  {
    System.out.println(“Driver não Encontrado!”);
}
```

Exemplo com SQL Server

```
try {
    Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
}
catch ( ClassNotFoundException e)  {
    System.out.println(“Driver não Encontrado!”);
}
```

Conexão com o Bando de Dados

2. Solicitando a abertura da conexão

- A utilização de um banco de dados requer uma conexão estabelecida por meio de um driver adequado.
- A classe ***DriverManager*** é responsável por administrar e selecionar tal driver conforme o banco de dados específico, possibilitando efetuar a conexão.
- A solicitação de uma conexão com um banco de dados é realizada pelo método ***getConnection(String)*** da classe DriverManager.

Conexão com o Bando de Dados

2. Solicitando a abertura da conexão

Connection

- Representa a conexão com o banco de dados
- Proporciona informações sobre as tabelas do BD por meio de transações;
- Os métodos desta interface frequentemente utilizados são:

Commit ()

executa todas as alterações feitas com o banco de dados pela atual transação.

Rollback ()

desfaz qualquer alteração feita com o banco de dados pela atual transação.

Close ()

libera o recurso que estava sendo utilizado pelo objeto.

Conexão com o Bando de Dados

2. Solicitando a abertura da conexão

Um pedido de conexão:

```
try {
        Class.forName("Driver");
         String bdUrl = “URL";
         String bdUsuario = “nome usuário";
         String bdSenha = “senha";
         Connection c =
DriverManager.getConnection(bdUrl,bdUsuario,bdSenha);
}
catch(ClassNotFoundException ex){ }
catch(SQLException ex) { }
```

Conexão com o Bando de Dados

2. Solicitando a abertura da conexão

Um pedido de conexão para um servidor SQL Server:

```
try {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
         String bdUrl =
"jdbc:sqlserver://localhost:1433;databaseName=Pessoa_Veiculo";
         String bdUsuario = "ale";
         String bdSenha = "ale";
         Connection c =
DriverManager.getConnection(bdUrl,bdUsuario,bdSenha);
}
catch(ClassNotFoundException ex){ }
catch(SQLException ex) { }
```

Conexão com o Bando de Dados

- Fechamento da conexão

Depois de ser utilizada, a conexão deve ser finalizada.

Para terminar uma conexão representada por uma instância de classe ***Connection***, basta invocar seu método ***close()***

```
try {
 c.close();
}
catch (SQLException sqle) { }
```

Atualizar ou Visualizar Informações

- Para enviar uma instrução SQL ao banco de dados (BD), a aplicação Java precisa criar uma instância da interface ***Statement***.
- A função desta interface é enviar instruções SQL ao banco e captar o retorno produzido.
- Para instanciar a interface ***Statement***, é preciso ter uma conexão aberta com o BD.
- Uma instância da interface ***Connection*** é criada utilizando-se o método estático ***getConnection()*** da classe ***DriverManager.***
- Uma instancia da interface ***Statement*** e criada utilizando-se o metodo estático createStatement() da interface ***Connection***.
- Toda instancia da interface ***Statement*** é vinculada, no momento de sua criação, a uma conexão ativa e, portanto, as instruções SQL a serem enviadas por ela já tem um BD especifico como destino.

Atualizar ou Visualizar Informações

- Uma instância da interface ***Statement*** é criada utilizando-se o método estático ***createStatement()*** da interface ***Connection***.
- Toda instância da interface ***Statement*** é vinculada, no momento de sua criação, a uma conexão ativa e, portanto, as instruções SQL a serem enviadas por ela já têm um BD específico como destino.

Atualizar ou Visualizar Informações

- Para executar uma instrução SQL no banco de dados deve ser utilizado os métodos:

  executeQuery

  para executar somente a instrução SELECT.

  executeUpdate

  para executar as instruções DELETE, INSERT e UPDATE.

Atualizar ou Visualizar Informações

- A interface ***ResultSet*** representa uma estrutura de dados bidimensional (matriz) resultante de uma consulta a um banco.
- Uma instância dessa interface mantém um cursor apontando para uma linha dessa estrutura de dados, que inicialmente está posicionada antes do primeiro registro.
- O método ***next()*** move o cursor para a próxima linha retornando um valor booleano (false) quando é invocado e já se encontra na última linha.
- Em conjunto com o laço ***while*** possibilita a varredura dos dados contidos nessa estrutura.
- O método ***get<tipo>(<campo>)*** permite recuperar o valor do campo para o registro atual.

Atualizar ou Visualizar Informações

Exemplo

```
try{
        stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery ("Select * From Veiculo");
        while (rs.next()){
                System.out.print(rs.getString(2) ;
        }
    }
}
```

Dúvidas?