UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

Programação Orientada a Objectos II

JDBC

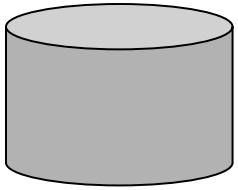
Docente: Ruben Manhiça

Maputo, 5 de Setembro de 2015

Conteúdo da Aula

1. JDBC

JDBC



JDBC (Java Database Connectivity) é uma API padronizada para acesso a dados para a linguagem Java.

- A API JDBC é baseada em interfaces, o que permite que os detalhes de implementação de cada base de dados sejam encapsulados no driver de conexão.
- Garante a manutenção da portabilidade de uma aplicação dentre vários SGBD.
- Basicamente estabelece conexões com o banco de dados, envia instruções SQL e processa os resultados recebidos.

JDBC

- API para executar comandos SQL:
  - classes e interfaces escritas em Java
  - permite envio de comandos SQL para **QUALQUER** SGBD Relacional
- com Java e API JDBC,
  - alunos acessam BD em Intranet a partir de PCs, Macs ou workstations UNIX

Tipos de Drivers JDBC

1: **JDBC-ODBC bridge mais ODBC (Open Database Connectivity) driver**

- JDBC acede a base de dados via ODBC driver. O ODBC atinge a independência de bancos de dados usando drivers para operarem como uma camada de tradução entre a aplicação e o SGBD

2: **Driver implementado com API nativa parcialmente escrita em Java**

- Chamadas JDBC convertidas em código específico da base de dados
- Neste caso as chamadas JDBC são convertidas diretamente em chamadas para a API dos banco de dados

Tipos de Drivers JDBC

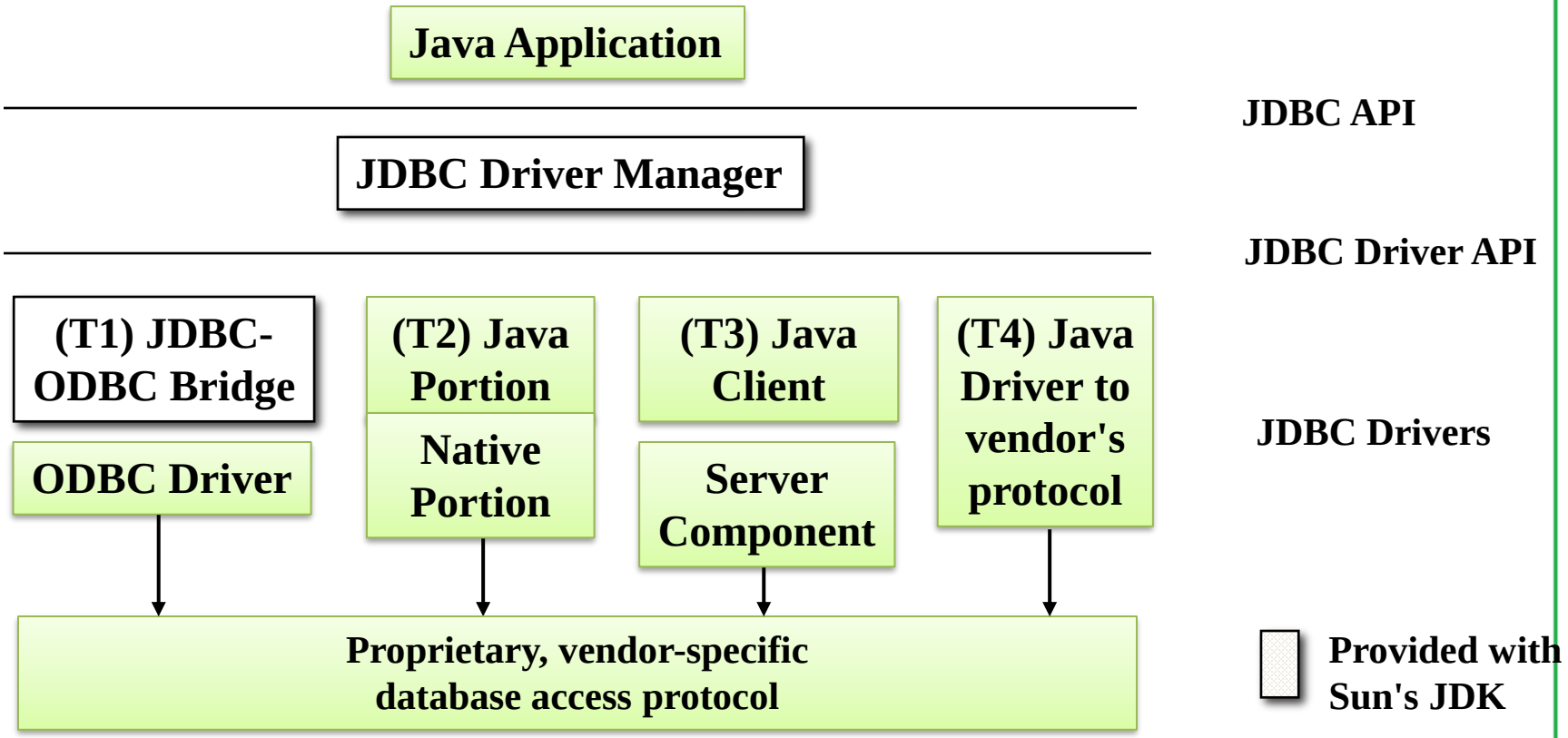
3: **Driver Java puro usando protocolo JDBC-Net:**

- JDBC usa protocolo de rede indenpendente to BD
- Este driver traduz chamadas JDBC em chamadas para um protocolo de Rede/DBMS independente que em seguida é traduzido para o DBMS por um servidor. Este Middleware permite que cliente java “puros” se conectem com diferentes BD

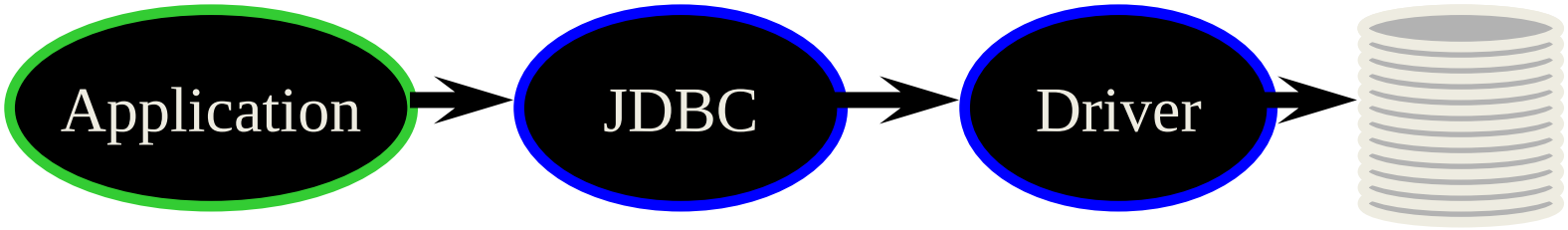
4: **Driver Java puro usando protocolo nativo:**

- JDBC usa protocolo de rede usado pelo BD
- Neste caso as chamadas JDBC são convertidas diretamente para o protocolo utilizado pelo DBMS, permitindo uma chamada direta do cliente para o servidor. A maioria destes drivers é proprietário.

JDBC Architecture



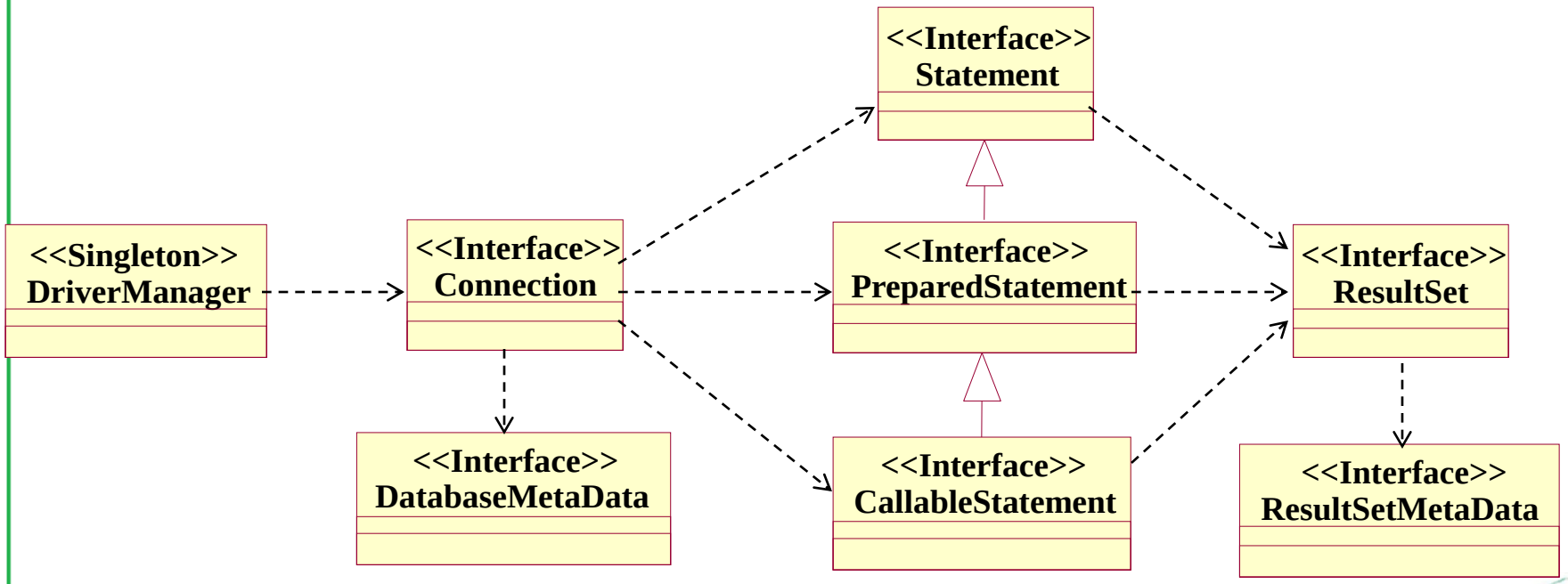
JDBC Architecture (cont.)



- Cógigo Java chama biblioteca JDBC
- JDBC carrega um *driver*
- Driver conversa com o SGBD
- Podemos ter mais de um driver -> mais do que um SGBD
- Ideal: podemos mudar o banco de dados sem termos que mudar o código da aplicação

Classes JDBC

- java.sql.* prove classes para serem usadas pelas aplicações



Pacotes importantes

```
import java.sql.*;  //JDBC packages
import oracle.jdbc.driver.*; // Driver Oracle
```

Sete passos básicos

1. Carregar Driver
2. Definir URL de conexão
3. Estabelecer conexão
4. Criar objeto do tipo statement
5. Executar uma consulta
6. Processar resultado
7. Fechar Conexão

Conexão na Base de Dados

Registra Driver

Cria Conexão

```
Class.forName("org.postgresql.Driver");

Connection con = DriverManager.getConnection(
"jdbc:postgresql://localhost/teste”,
"admin", "admin");

Statement stm = con.createStatement();
ResultSet rs = stm.executeQuery("SELECT * FROM TESTE");

//Trabalha com o ResultSet

rs.close();
stm.close();
con.close();
```

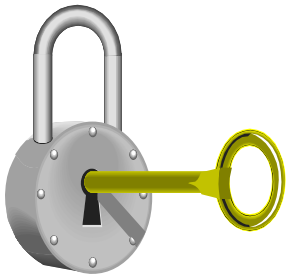
Cria o comando
e executa no BD

Fechas as Conexões

Passando Parâmetros

Os parâmetros podem ser concatenados para formar o comando SQL enviado para a base de dados. Isto não é uma boa prática pois pode criar uma brecha de segurança....

```
Statement stm = con.createStatement();
String query = "SELECT * FROM USUARIO" +
               "WHERE LOGIN = '" + login + "' " +
               "AND SENHA = '" + senha + "'";
ResultSet rs = stm.executeQuery(query);
```



O ataque SQL Injection, envia entradas a aplicação de forma a modificar o comando SQL formado por concatenação de String que é executado no banco de dados.

PreparedStatement

Utilizando o objeto PreparedStatement, a String que representa a query possui "?" no lugar dos parâmetros, e existem métodos que colocam as variáveis na query com o devido tratamento.

```
String query = "SELECT * FROM USUARIO" +
               "WHERE LOGIN = ? " +
               "AND SENHA = ? ";
PreparedStatement stm = con.prepareStatement(query);
stm.setString(1,login);
stm.setString(2,senha);
ResultSet rs = stm.executeQuery();
```

O PreparedStatement é mais eficiente pois o driver/BD processa o texto SQL somente uma vez, fazendo um cache de sua representação compilada, o que aumenta o desempenho para um número grande de execuções.

Trabalhando com o ResultSet

A classe ResultSet representa o resultado retornado por uma query ao banco de dados. Ele funciona como se fosse um ponteiro que aponta para a linha corrente.

O método next() passa o cursor para próxima linha e retorna false se estiver no final.

Recupera o resultado da query como um ResultSet

```
ResultSet rs = stm.executeQuery();

while(rs.next()){
   rs.getString(1);
   rs.getInt("nome_coluna");
}
```

Existem vários métodos que recuperam o valor das colunas pela ordem ou pelo nome.

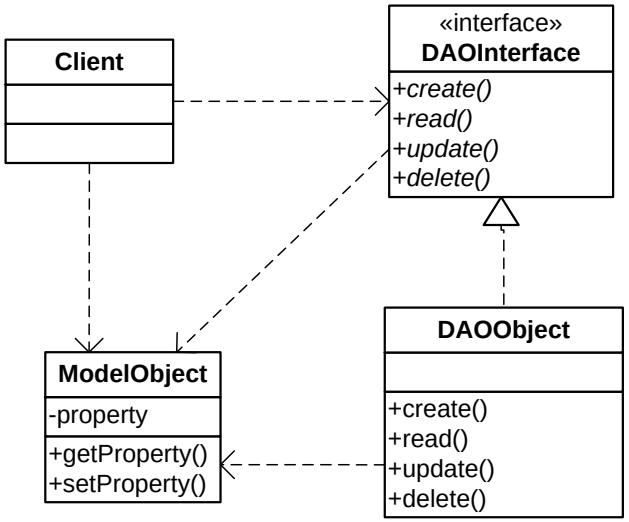
Excepções de Base de Dados

Muitos erros podem acontecer enquanto se acessa a base de dados, e até mesmo quando se tenta fechar a conexão com ele. Segue como é uma estrutura típica de um código que faz acesso a dados.

```
// Declarar variáveis
try {
    // Criar um objeto Connection
    // Criar os demais objetos
    // Executa e faz o que tem que fazer
} catch(SQLException e){
    // Tratar erros do banco de dados
} catch(Exception ex) {
    // Tratar demais exceções
} finally {
    try {
        // Fechar possíveis ResultSets, Statements e Connections
    } catch(Exception ef) {
        // Tratar aqui erros do fechamento das Conexões
    }
}
```

Data Access Objects (DAO)

DAO é um padrão de projeto utilizado para encapsular e abstrair todo acesso a base de dados.



- Uma interface abstrai os métodos do DAO de sua implementação.
- O objeto implementa o acesso a dados da forma desejada (JDBC, Hibernate, iBatis e etc...)
- O objeto de negócios é utilizado nas trocas de mensagens entre o DAO e a classe Cliente.
- O uso da interface permite que depois seja adicionado cache, verificação de permissões e etc... antes do DAO.

Exercício – Acedendo a Dados

Criar um DAO para uma determinada classe que execute inserção, atualização, exclusão e a recuperação individual e de listagens.

- Utilizar JDBC para o acesso a dados
- Criar a tabela e a classe de negócios
- Fazer testes para verificar o funcionamento do DAO
- Criar uma interface e um método fábrica

FIM!!!

Duvidas e Questões?