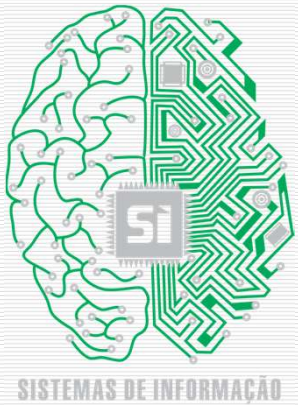
Prog. Orientada a Objetos II

Prof. Fernando Del Moro
fernando@esucri.com.br



1

Prog. Orientada a Objetos II

Conectando Java a um Banco de Dados

2

Prog. Orientada a Objetos II

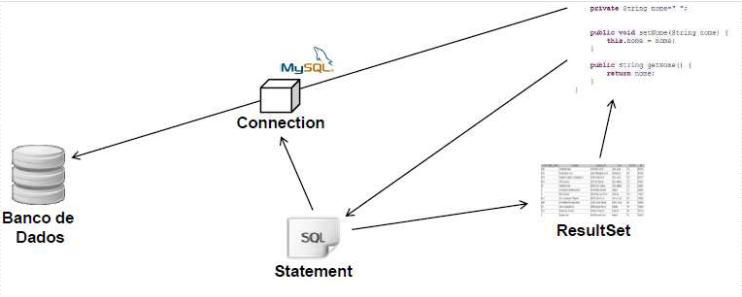
✓Para evitar que cada banco tenha a sua própria API e conjunto de classes e métodos, temos um único conjunto de *interfaces*

✓Esse conjunto de *interfaces* fica no pacote *java.sql*

3

Prog. Orientada a Objetos II

✓Para conectar um banco de dados, recuperar e manipular informações, precisamos de alguns objetos



4

Prog. Orientada a Objetos II

✓Para desenvolver uma aplicação baseada em **JDBC** é preciso entender algumas classes e *interfaces*.
✓A principal é a *Connection.*
✓Pertence ao pacote “java.sql”.
✓Esse é um fator a ser observado, pois é comum o desenvolvedor pensar em usar o pacote *com.mysql.jdbc* já que se está utilizando o *driver* JDBC do banco MySQL.

5

Prog. Orientada a Objetos II

Pacote java.sql
✓Esse pacote oferece ao Java o acesso e processamento de dados em uma fonte de dados. As classes e interfaces mais importantes são:

6

Prog. Orientada a Objetos II

Pacote java.sql

Classe	Interface
DriverManager	Driver
	Connection
	Statement
	PreparedStatement
	ResultSet

7

Prog. Orientada a Objetos II

✓São essas *classes/interfaces* que sabem se comunicar através do protocolo proprietário do banco de dados.

✓Todos os principais bancos de dados do mercado possuem *drivers* JDBC para que você possa utilizá-los com Java.

8

Prog. Orientada a Objetos II

Registrando e Carregando *Drivers*

✓Antes de utilizar uma conexão ao BD, é necessário registrar/carregar o *Driver*.

✓Vários drivers podem ser carregados através do método *Class.forName*.

✓Class.forName("oracle.jdbc.driver.OracleDriver");

✓Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

✓Class.forName("com.mysql.jdbc.Driver");

9

Prog. Orientada a Objetos II

Registrando e Carregando *Drivers*

✓Antes da versão 6 do Java, carregar o Driver era obrigatório e nada poderia ser feito antes dessa carga.

✓A partir da versão 6, ela se tornou opcional, utilizada para tratamento de erros.

10

Prog. Orientada a Objetos II

Estabelecendo uma Conexão

✓A abertura da conexão é feita pelo método *getConnection*, que recebe alguns parâmetros, normalmente:

- ✓url
- ✓Usuário
- ✓Senha

11

Prog. Orientada a Objetos II

Estabelecendo uma Conexão

✓Uma conexão é mantida através de um objeto chamado *connection*:

12

Prog. Orientada a Objetos II

Estabelecendo uma Conexão

✓Um exemplo de comando de conexão:

✓Connection conexao=
DriverManager.getConnection(url, usuario, senha);

Prog. Orientada a Objetos II

Envio de Comandos SQL

✓O envio de um comando SQL é feito por meio de um objeto específico.

✓Existem 2 classes para envio de comandos SQL:

- ✓*Statement*
- ✓*PreparedStatement*

Prog. Orientada a Objetos II

Envio de Comandos SQL

- ✓*Statement:* é utilizado para enviar comandos SQL únicos.
- ✓*PreparedStatement:* o comando SQL é pré-compilado e utilizado posteriormente, sendo mais eficiente nos casos onde o mesmo comando é utilizado várias vezes

15

Prog. Orientada a Objetos II

Envio de Comandos SQL

- ✓*Statement:* Um objeto desta classe é criado pelo envio da mensagem *createStatement* ao objeto de conexão

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

16

Prog. Orientada a Objetos II

Envio de Comandos SQL

✓*PreparedStatement:* criado com a instrução `prepareStatement(SQL)` é caracterizado por:

✓Manter instruções fixas e pré-compiladas

✓Execução mais rápida

✓Uso de parâmetros

17

Prog. Orientada a Objetos II

Envio de Comandos SQL

✓Os parâmetros são dados por “?”

```
PreparedStatement prep =
conexao.prepareStatement(
"SELECT * FROM TABELA WHERE ID=?");
```

18

Prog. Orientada a Objetos II

Envio de Comandos SQL

✓E passados por

```
prep.setTIPO(parametro,valor);
```

✓Onde:
✓TIPO – tipo do parâmetro
✓Parametro- número do parâmetro(iniciando de 1)
✓Valor- valor do parâmetro

19

Prog. Orientada a Objetos II

Execução de Comandos SQL

✓Ao criarmos um objeto de envio de comandos, devemos fazer a execução do comando.

✓A criação e a execução são realizados em comandos diferentes.

20

Prog. Orientada a Objetos II

Execução de Comandos SQL

- ✓Existem 3 maneiras distintas de executar uma instrução SQL:
  - ✓Execute
  - ✓ExecuteQuery
  - ✓ExecuteUpdate

21

Prog. Orientada a Objetos II

Execução de Comandos SQL

- ✓método *execute:* usado para executar comandos SQL sem retorno de resultado

22

Prog. Orientada a Objetos II

Execução de Comandos SQL

- ✓método *execute:*

```
stmt.execute("drop table teste");
```


Prog. Orientada a Objetos II

Execução de Comandos SQL

- ✓método *executeUpdate:* usado para executar comandos SQL que alteram a tabela ou o *banco(insert*, *delete* e *update*).
- ✓Tem como retorno um Inteiro, indicando o número de registros alterados

Prog. Orientada a Objetos II

Execução de Comandos SQL

- ✓método *executeUpdate:*

```
stmt.executeUpdate("INSERT INTO TABELA
(ID,NOME) VALUES (1,'NOME')");
```

25

Prog. Orientada a Objetos II

Execução de Comandos SQL

- ✓método *executeQuery:* utilizado para executar comandos SQL que retornam registros/valores das tabelas do banco de dados (Select).
- ✓As respostas devem ser associadas a um objeto do tipo *ResultSet*

26

Prog. Orientada a Objetos II

Execução de Comandos SQL

- ✓método *executeQuery:*

```
ResultSet rs = stmt.executeQuery("select *
from tabela");
```

27

Prog. Orientada a Objetos II

Execução de Comandos SQL

- ✓*ResultSet* oferece à aplicação a tabela resultante da execução de um *Select.*
- ✓mantém um cursor posicionado em uma linha da tabela.
- ✓Inicialmente este cursor está antes da primeira linha e a mensagem *next()* movimenta o cursor para frente.

28

Prog. Orientada a Objetos II

Execução de Comandos SQL

- ✓Permite recuperar os dados das colunas através do método `getTIPO(<coluna>)`
- ✓Onde:
  - ✓TIPO é o tipo da coluna
  - ✓<coluna> é o nome da coluna ou sua posição (a partir de 1).

29

Prog. Orientada a Objetos II

Execução de Comandos SQL

- ✓Permite recuperar os dados das colunas através do método `getXXX(<coluna>)`

```
ResultSet rs = stmt.executeQuery("select *
from tabela");
while (rs.next()) {
 int id = rs.getInt("ID");
 String nome = rs.getString(2);
}
```

Nome do Campo

Número do Campo

30

Prog. Orientada a Objetos II

Transações

✓Uma transação é um conjunto de *Statements* que são validados no BD com *commit* ou cancelados com *rollback*

✓Por default, todos os comandos no JDBC tem *commit* automático

```
conexao.setAutoCommit(false); // muda o default
```

31

Prog. Orientada a Objetos II

Transações

✓Exemplo:

```
conn.setAutoCommit(false);
Statement s = conexao.createStatement();
try {
     s.executeUpdate("SQL statement 1");
     s.executeUpdate("SQL statement 2");
     conexao.commit(); // valida os 2 updates
} catch (Exception e) {
     conexao.rollback(); //desfaz os updates
}
```

32