Sistemas de Bancos de Dados

Aula 12 – Conectando sem as classes do tutorial JDBC

Murielly Oliveira Nascimento – 11921BSI222

1) Inicialmente, a quantidade de bibliotecas a ser importada será menor quando comparada com as bibliotecas da classe MyQueries:

```
1 import java.util.Properties; //Objeto genérico que armazena propriedades com usuário e senha
2 import java.sql.DriverManager; //Objeto que criará a conexão do sistema de banco de dados
3 import java.sql.Connection;  //Objeto que armazenará o objeto de conexão ao banco de dados
4 import java.sql.Statement; //Objeto para disparar um comando para o SGBD
5 import java.sql.ResultSet;  //Objeto que armazenará as tuplas resultantes de um comando SQL
6 import java.sql.SQLException; //Objeto para capturar eventos de erro no acesso ao banco de dados
```

2) Para implementação do código você precisa de, pelo menos, uma classe e quatro métodos:

```
1 import java.util.Properties; //Objeto genérico que armazena propriedades com usuário e senha
2 import java.sql.DriverManager; //Objeto que criará a conexão do sistema de banco de dados
3 import java.sql.Connection;  //Objeto que armazenará o objeto de conexão ao banco de dados
4 import java.sql.Statement; //Objeto para disparar um comando para o SGBD
5 import java.sql.ResultSet;  //Objeto que armazenará as tuplas resultantes de um comando SQL
6 import java.sql.SQLException; //Objeto para capturar eventos de erro no acesso ao banco de dados
7
8 public class StandAloneJDBCCode {
9         public static Connection getConnection(){...}
10        public static void myquery(Connection con) throws SQLException {...}
11        public static void closeConnection(Connection con) {...}
12        public static void main(String[] args) {...}
13
14 }
```

3) O código abaixo mostra um esboço do método criador do objeto armazenador da conexão ao banco de dados. Perceba que este método retorna um objeto de conexão do tipo Connection, cuja biblioteca da classe foi incorporada no início do código.

```
public static Connection getConnection(){
        Connection con = null;
        String currentUrlString = null;
        Properties connectionProps = new Properties();

        connectionProps.put("user", "postgres");
        connectionProps.put("password", "postgres");

        currentUrlString = "jdbc:postgresql://localhost:5432/IB2";
        //atenção para os símbolos de barra "/" na linha acima, não confunda com a letra l
        try {
                con = DriverManager.getConnection(currentUrlString, connectionProps);
        } catch (SQLException e) {
                e.printStackTrace();
        }
        return con;
}
```

4) Este código mostra o método que é o objetivo da existência deste programa, uma consulta ao banco de dados. Percebam que o método apenas recebe um objeto de conexão ao banco de dados, não há uma implementação de código para determinar os parâmetros de conexão (função do código do item anterior), mas apenas a preocupação de executar uma consulta nesta conexão e tratar os dados recebidos:

A consulta retorna o nome de todos os clientes.

```
public static void myquery(Connection con) throws SQLException {
            Statement stmt = null;
            String query = "select * from cliente";

            try {
                stmt = con.createStatement();
                ResultSet rs = stmt.executeQuery(query);
                System.out.println("Table list: ");
                while ( rs.next() ) {
                        String name   = rs.getString(1);
                        System.out.println(name + ", " +  " " );
                }
        } catch (SQLException e) {
                e.printStackTrace();
        }
        finally {
                if (stmt != null) {
                stmt.close();
                }
        }
}
```

5) O código abaixo tem o objetivo de finalizar a conexão criado com o banco de dados. Tal hábito é saudável visto que um sistema de banco de dados pode criar diversas conexões simultâneas a um banco de dados com potencial de prejudicar o desempenho de um servidor de banco de dados, bem como limitar a quantidade de conexões para outros usuários/sistemas:

```
public static void closeConnection(Connection con) {
        try {
                if (con != null) {
                        con.close();
                        con = null;
                }
                System.out.println("Released all database resources.");
        } catch (SQLException e) {
                e.printStackTrace();
        }
}
```

6) Por último, o código abaixo é o método executável deste programa.:

```java
public static void main(String[] args) {
        if (args.length == 0) {
                System.err.println("No arguments.");
        }
        Connection myConnection = null;
        try {
                myConnection = StandAloneJDBCCode.getConnection();
                StandAloneJDBCCode.myquery(myConnection);
        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                StandAloneJDBCCode.closeConnection(myConnection);
        }
}
```

```
mury@ubuntu:~/JDBCTutorial$ javac -cp "/home/mury/JDBCTutorial/BD2-14-postgresql-42.2.4.jar" StandAloneJDBCCode.java
mury@ubuntu:~/JDBCTutorial$ java -cp "/home/mury/JDBCTutorial/BD2-14-postgresql-42.2.4.jar:./" StandAloneJDBCCode
No arguments.
Table list:
José Moreira da Silva,
Pedro Alvares Sousa,
Maria Lúcia Alves,
Marta Avelar Santos,
João Boiadeiro,
Everardo Monfort Leitão,
Marco Aurélio Santos,
Maria das Dores,
Cláudia Santos Mota,
Carolina Soares Souza,
Marcos Andrade,
Maria do Socorro,
Gomes Dias Santos,
Carla Soares Sousa,
Manuel Oliveira,
Joaquim Carlos Reis,
Marcos Cláudio,
Jefferson Oliveira,
Andrade de Freitas,
Geraldo Oliveira,
André Cabral da Silva,
Marcos Ferreira Dinardi,
Alexandre Márcio de Souza,
Eurides Alves da Silva,
```