**Kiểm tra email**

package Test;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class EmailValidator{

private static final String EMAIL\_PATTERN = "^[\_A-Za-z0-9-\\+]+(\\.[\_A-Za-z0-9-]+)\*@"

            + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)\*(\\.[A-Za-z]{2,})$";// cấu trúc 1 email thông thường

    public EmailValidator() {

        pattern = Pattern.compile(EMAIL\_PATTERN);

    }

    // Class kiểm định dạng email

    public boolean validate(final String hex) {

        matcher = pattern.matcher(hex);

        return matcher.matches();

    }

    // main và test

    public static void main(String[] args) {

        String email = "nguyendangkhiemit@gmail.com";

        EmailValidator validator = new EmailValidator();

        if (validator.validate(email)) {

            System.out.println("Email hợp lệ");

        } else {

            System.out.println("Email không hợp lệ");

        }

    }

}

**Đọc ghi file**

**public** **void** Exercise1() **throws** FileNotFoundException{

LinkedList<String> list = **new** LinkedList<String>();

Scanner kb = **new** Scanner(**new** File("List.inp"));

**while** (kb.hasNext()){

String x=kb.next();

**if** (!list.contains(x)) list.add(x);

}

PrintStream ps = **new** PrintStream(**new** File("List.out"));

**for** (**int** i=0;i<list.size();i++)

ps.print(list.get(i));

}

**Kiểm tra ngày tháng**

**public** **class** DateUtils {

**private** **static** Pattern *dateRegexPattern* = Pattern.*compile*("(0?[1-9]|[12][0-9]|3[01])/(0?[1-9]|1[012])/((19|20)\\d\\d)");

**public** **static** **boolean** isValidDate(String dateString) {

Matcher dateMatcher = *dateRegexPattern*.matcher(dateString);

**if** (dateMatcher.matches()) {

dateMatcher.reset();

**if** (dateMatcher.find()) {

String day = dateMatcher.group(1);

String month = dateMatcher.group(2);

**int** year = Integer.*parseInt*(dateMatcher.group(3));

**if** ("31".equals(day) &&

("4".equals(month) || "6".equals(month) || "9".equals(month) ||

"11".equals(month) || "04".equals(month) || "06".equals(month) ||

"09".equals(month))) {

**return** **false**; // 1, 3, 5, 7, 8, 10, 12 has 31 days

} **else** **if** ("2".equals(month) || "02".equals(month)) {

//leap year

**if** (year % 4 == 0) {

**return** !"30".equals(day) && !"31".equals(day);

} **else** {

**return** !"29".equals(day) && !"30".equals(day) && !"31".equals(day);

}

} **else** {

**return** **true**;

}

} **else** {

**return** **false**;

}

} **else** {

**return** **false**;

}

}

**public** **static** **void** main(String[] args) {

System.***out***.println("Enter date: ");

Scanner kb = **new** Scanner(System.***in***);

String s= kb.next();

System.***out***.println(DateUtils.*isValidDate*(s) ? "PASSED: " + s : "FAIL: " + s);

}

}

**Kết nối database**

4- Download một số các driver quan trọng

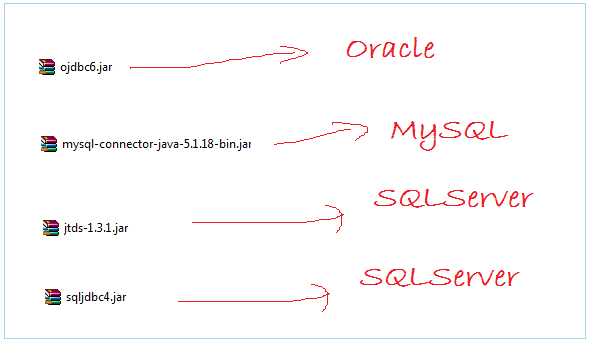
Trong trường hợp nếu bạn không muốn sử dụng **JDBC-ODBC**, bạn có thể sử dụng cách trực tiếp kết nối vào Database, trong trường hợp đó cần phải download Driver ứng với mỗi loại DB này. Tại đây tôi hướng dẫn download một loại Driver cho các Database thông dụng:

* **Oracle**
* **MySQL**
* **SQLServer**
* ....

Bạn có thể xem hướng dẫn tại:

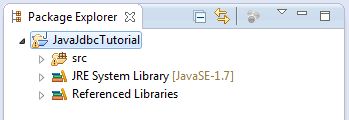
* <http://o7planning.org/vi/10227/thu-vien-dieu-khien-cac-loai-co-so-du-lieu-khac-nhau-trong-java>

Kết quả chúng ta có một vài file:



5- Tạo project để bắt đầu ví dụ với JDBC

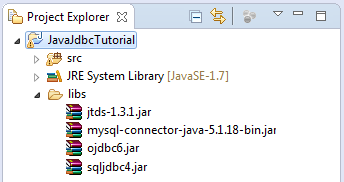
Tạo mới project **JavaJdbcTutorial**:



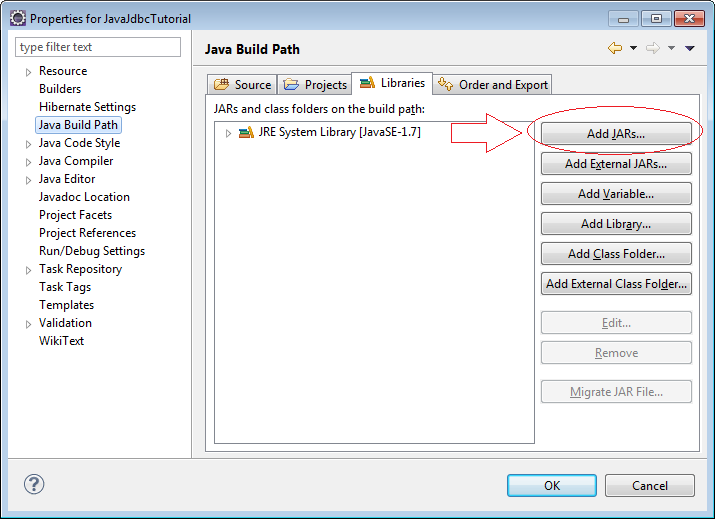
Tạo thư mục **libs** trên project và copy các thư viện kết nối trực tiếp các loại database ***Oracle, MySQL, SQLServer*** mà bạn vừa download được ở trên vào. Bạn có thể copy hết hoặc một trong các thư viện đó, theo loại DB mà bạn sử dụng.

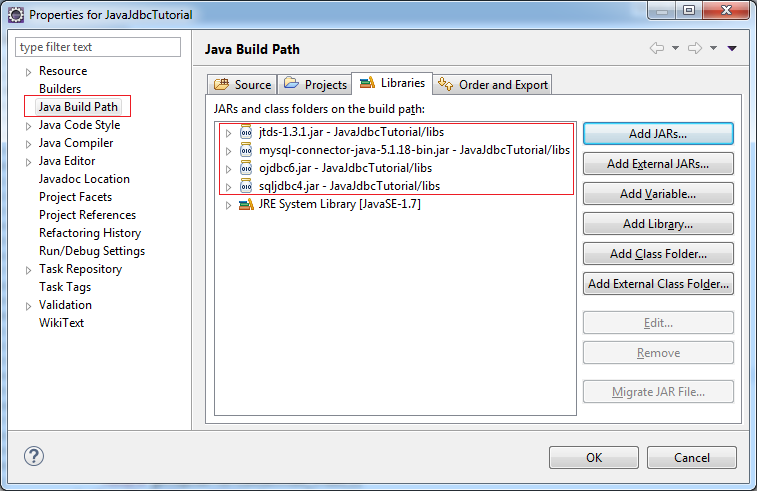
*Chú ý: Bạn chỉ cần download một Driver ứng với loại Database mà bạn quen thuộc. Cơ sở dữ liệu dùng làm ví dụ trong tài liệu này bạn có thể lấy tại:*

* [*http://o7planning.org/vi/10117/co-so-du-lieu-mau*](http://o7planning.org/vi/10117/co-so-du-lieu-mau)



Nhấn phải vào Project chọn Properties:





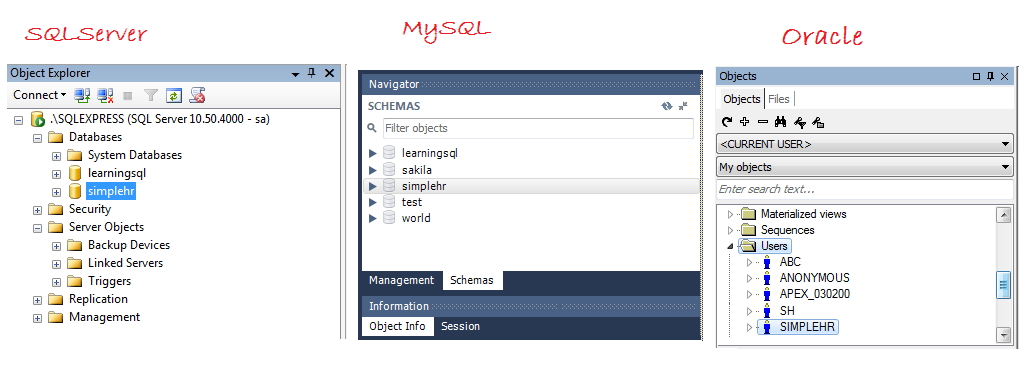
Giờ thì bạn có thể sẵn sàng làm việc với một trong các Database (Oracle, MySQL, SQLServer)

6- Connection

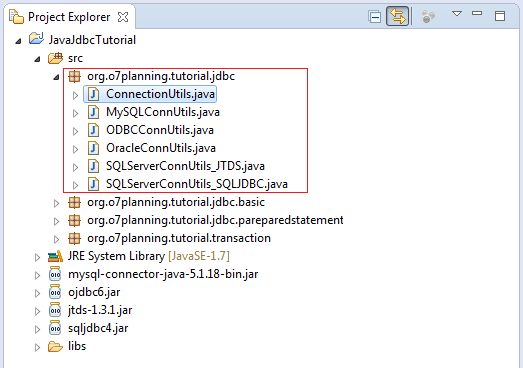
Trong tài liệu hướng dẫn này tôi sẽ hướng dẫn các kết nối vào cả 3 loại database:

* **MySQL**
* **SQLServer**
* **Oracle**

Trong khi thực hành, bạn chỉ cần làm việc với một loại **DB**nào mà bạn quen thuộc.



Chúng ta tạo class **ConnectionUtils**để lấy ra đối tượng ***Connection***kết nối với Database.



**ConnectionUtils.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | package org.o7planning.tutorial.jdbc;    import java.sql.Connection;  import java.sql.SQLException;    public class ConnectionUtils {      public static Connection getMyConnection() throws SQLException,            ClassNotFoundException {        // Sử dụng Oracle.        // Bạn có thể thay thế bởi Database nào đó.        return OracleConnUtils.getOracleConnection();    }      //    // Test Connection ...    //    public static void main(String[] args) throws SQLException,            ClassNotFoundException {          System.out.println("Get connection ... ");          // Lấy ra đối tượng Connection kết nối vào database.        Connection conn = ConnectionUtils.getMyConnection();          System.out.println("Get connection " + conn);          System.out.println("Done!");    }    } |

**OracleConnUtils.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | package org.o7planning.tutorial.jdbc;    import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;    public class OracleConnUtils {     // Kết nối vào ORACLE.   public static Connection getOracleConnection() throws SQLException,           ClassNotFoundException {       String hostName = "localhost";       String sid = "db11g";       String userName = "simplehr";       String password = "simplehr";         return getOracleConnection(hostName, sid, userName, password);   }     public static Connection getOracleConnection(String hostName, String sid,           String userName, String password) throws ClassNotFoundException,           SQLException {         // Khai báo class Driver cho DB Oracle       // Việc này cần thiết với Java 5       // Java6 tự động tìm kiếm Driver thích hợp.       // Nếu bạn dùng Java6, thì ko cần dòng này cũng được.       Class.forName("oracle.jdbc.driver.OracleDriver");         // Cấu trúc URL Connection dành cho Oracle       // Ví dụ: jdbc:oracle:thin:@localhost:1521:db11g       String connectionURL = "jdbc:oracle:thin:@" + hostName + ":1521:" + sid;         Connection conn = DriverManager.getConnection(connectionURL, userName,               password);       return conn;   }  } |

**MySQLConnUtils.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | package org.o7planning.tutorial.jdbc;    import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;    public class MySQLConnUtils {     // Kết nối vào MySQL.   public static Connection getMySQLConnection() throws SQLException,           ClassNotFoundException {       String hostName = "localhost";         String dbName = "simplehr";       String userName = "root";       String password = "1234";         return getMySQLConnection(hostName, dbName, userName, password);   }     public static Connection getMySQLConnection(String hostName, String dbName,           String userName, String password) throws SQLException,           ClassNotFoundException {       // Khai báo class Driver cho DB MySQL       // Việc này cần thiết với Java 5       // Java6 tự động tìm kiếm Driver thích hợp.       // Nếu bạn dùng Java6, thì ko cần dòng này cũng được.       Class.forName("com.mysql.jdbc.Driver");         // Cấu trúc URL Connection dành cho Oracle       // Ví dụ: jdbc:<mysql://localhost:3306/simplehr>       String connectionURL = "jdbc:mysql://" + hostName + ":3306/" + dbName;         Connection conn = DriverManager.getConnection(connectionURL, userName,               password);       return conn;   }  } |

**SQLServerConnUtils\_JTDS.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | package org.o7planning.tutorial.jdbc;    import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;    public class SQLServerConnUtils\_JTDS {       // Kết nối vào SQLServer.   // (Sử dụng thư viện điều khiển JTDS)   public static Connection getSQLServerConnection()           throws SQLException, ClassNotFoundException {       String hostName = "localhost";       String sqlInstanceName = "SQLEXPRESS";       String database = "simplehr";       String userName = "sa";       String password = "1234";         return getSQLServerConnection(hostName, sqlInstanceName, database,               userName, password);   }       // Trường hợp sử dụng SQLServer.   // Và thư viện JTDS.   public static Connection getSQLServerConnection(String hostName,           String sqlInstanceName, String database, String userName,           String password) throws ClassNotFoundException, SQLException {       // Khai báo class Driver cho DB SQLServer       // Việc này cần thiết với Java 5       // Java6 tự động tìm kiếm Driver thích hợp.       // Nếu bạn dùng Java6, thì ko cần dòng này cũng được.       Class.forName("net.sourceforge.jtds.jdbc.Driver");         // Cấu trúc URL Connection dành cho SQLServer       // Ví dụ:       // jdbc:jtds:<sqlserver://localhost:1433/simplehr;instance=SQLEXPRESS>       String connectionURL = "jdbc:jtds:sqlserver://" + hostName + ":1433/"               + database + ";instance=" + sqlInstanceName;         Connection conn = DriverManager.getConnection(connectionURL, userName,               password);       return conn;   }    } |

**SQLServerConnUtils\_SQLJDBC.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | package org.o7planning.tutorial.jdbc;    import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;    public class SQLServerConnUtils\_SQLJDBC {     // Kết nối vào SQLServer.   // (Sử dụng thư viện điều khiển SQLJDBC)   public static Connection getSQLServerConnection()           throws SQLException, ClassNotFoundException {       String hostName = "localhost";       String sqlInstanceName = "SQLEXPRESS";       String database = "simplehr";       String userName = "sa";       String password = "1234";         return getSQLServerConnection(hostName, sqlInstanceName,               database, userName, password);   }     // Trường hợp sử dụng SQLServer.   // Và thư viện SQLJDBC.   public static Connection getSQLServerConnection(String hostName,           String sqlInstanceName, String database, String userName,           String password) throws ClassNotFoundException, SQLException {       // Khai báo class Driver cho DB SQLServer       // Việc này cần thiết với Java 5       // Java6 tự động tìm kiếm Driver thích hợp.       // Nếu bạn dùng Java6, thì ko cần dòng này cũng được.       Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");         // Cấu trúc URL Connection dành cho SQLServer       // Ví dụ:       // jdbc:<sqlserver://ServerIp:1433/SQLEXPRESS;databaseName=simplehr>       String connectionURL = "jdbc:sqlserver://" + hostName + ":1433"               + ";instance=" + sqlInstanceName + ";databaseName=" + database;         Connection conn = DriverManager.getConnection(connectionURL, userName,               password);       return conn;   }    } |

**ODBCConnUtils.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | package org.o7planning.tutorial.jdbc;    import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;    public class ODBCConnUtils {      // Lấy ra kết nối vào ODBC Data Source có tên "simplehr-ds".    public static Connection getJdbcOdbcConnection() throws SQLException,            ClassNotFoundException {        String odbcDataSourceName = "simplehr-ds";        String userName = "simplehr";        String password = "simplehr";        return getJdbcOdbcConnection(odbcDataSourceName, userName, password);    }      public static Connection getJdbcOdbcConnection(String odbcDataSourceName,            String userName, String password) throws SQLException,            ClassNotFoundException {          // Khai báo class Driver (Cầu nối Jdbc-Odbc)        // Việc này cần thiết với Java 5        // Java6 tự động tìm kiếm Driver thích hợp.        // Nếu bạn dùng Java6, thì ko cần dòng này cũng được.        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");          // Cấu trúc URL Connection dành cho JDBC-ODBC        String connectionURL = "jdbc:odbc:" + odbcDataSourceName;          Connection conn = DriverManager.getConnection(connectionURL, userName,                password);        return conn;    }  } |

Bạn có thể thay đổi Class **ConnectionUtils**để sử dụng kết nối tới một Database nào đó quen thuộc. Và chạy class này để test kết nối.

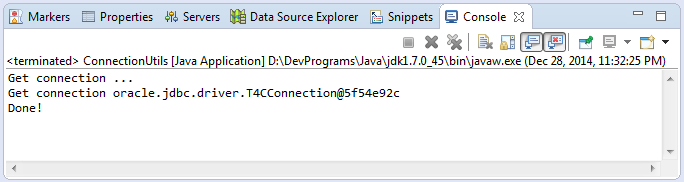
*Chú ý: Nếu bạn sử dụng****MySQL****hoặc****SQL Server****mặc định 2 Database này chặn không cho phép kết nối vào nó từ một IP khác. Bạn cần cấu hình để cho phép điều này. Bạn có thể xem hướng dẫn trong tài liệu cài đặt và cấu hình****MySQL****,****SQL Server****trên****o7planning****.*

*Cài đặt và cấu hình****MySQL Community****:*

* [*http://o7planning.org/vi/10221/huong-dan-cai-dat-va-cau-hinh-mysql-community*](http://o7planning.org/vi/10221/huong-dan-cai-dat-va-cau-hinh-mysql-community)

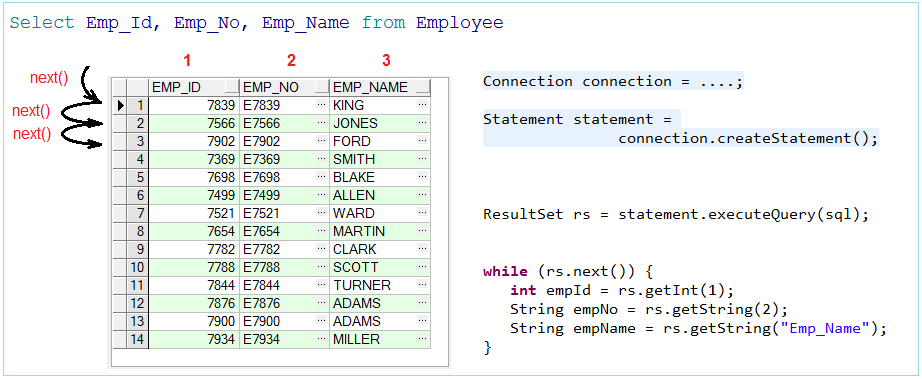
*Cài đặt và cấu hình SQL Server:*

* [*http://o7planning.org/vi/10299/huong-dan-cai-dat-va-cau-hinh-sql-server-express-2014*](http://o7planning.org/vi/10299/huong-dan-cai-dat-va-cau-hinh-sql-server-express-2014)



7- Sử dụng JDBC API truy vấn dữ liệu

Đây là hình ảnh dữ liệu trong bảng **Employee**. Chúng ta sẽ xem cách **Java**lấy ra dữ liệu thế nào thông qua một ví dụ:



***ResultSet***là một đối tượng Java, nó được trả về khi bạn truy vấn (query) dữ liệu. Sử dụng ***ResultSet.next()*** để di chuyển con trỏ tới các bản ghi tiếp theo (Di chuyển dòng). Tại một bản ghi nào đó bạn sử dụng các method ***ResultSet.getXxx()*** để lấy ra các giá trị tại các cột. Các cột được đánh với thứ tự 1,2,3,...

* Interface ***ResultSet***

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public String getString(int columnIndex) throws SQLException;  public boolean getBoolean(int columnIndex) throws SQLException;  public int getInt(int columnIndex) throws SQLException;  public double getDouble(int columnIndex) throws SQLException;    ...    public String getString(String columnLabel) throws SQLException;  public boolean getBoolean(String columnLabel) throws SQLException;  public int getInt(String columnLabel) throws SQLException;  public double getDouble(String columnLabel) throws SQLException;    .... |

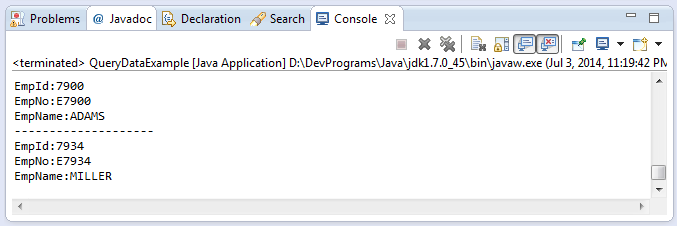
Ví dụ minh họa:

**QueryDataExample.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | package org.o7planning.tutorial.jdbc.basic;    import java.sql.Connection;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.sql.Statement;    import org.o7planning.tutorial.jdbc.ConnectionUtils;    public class QueryDataExample {      public static void main(String[] args) throws ClassNotFoundException,            SQLException {          // Lấy ra đối tượng Connection kết nối vào DB.        Connection connection = ConnectionUtils.getMyConnection();          // Tạo đối tượng Statement.        Statement statement = connection.createStatement();          String sql = "Select Emp\_Id, Emp\_No, Emp\_Name from Employee";          // Thực thi câu lệnh SQL trả về đối tượng ResultSet.        ResultSet rs = statement.executeQuery(sql);          // Duyệt trên kết quả trả về.        while (rs.next()) {// Di chuyển con trỏ xuống bản ghi kế tiếp.            int empId = rs.getInt(1);            String empNo = rs.getString(2);            String empName = rs.getString("Emp\_Name");            System.out.println("--------------------");            System.out.println("EmpId:" + empId);            System.out.println("EmpNo:" + empNo);            System.out.println("EmpName:" + empName);        }        // Đóng kết nối        connection.close();    }    } |

Kết quả chạy ví dụ:



8- Các kiểu ResultSet

Bạn đã làm quen với ***ResultSet***với các ví dụ phía trên. Mặc định các ***ResultSet***khi duyệt dữ liệu chỉ có thể chạy từ trên xuống dưới, từ trái sang phải. Điều đó có nghĩa là với các ***ResultSet***mặc định bạn không thể gọi:

* ***ResultSet.previous()***  : Lùi lại một bản ghi.
* Trên cùng một bản ghi không thể gọi ***ResultSet.getXxx(4)*** rồi mới gọi ***ResultSet.getXxx(2)***.

Việc cố tình gọi sẽ bị một **Exception**.

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public  Statement createStatement(int resultSetType, int resultSetConcurrency)         throws SQLException;    // Ví dụ:  Statement statement = connection.createStatement(                 ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_READ\_ONLY);    // ResultSet có thể cuộn (tiến lùi, sang trái sang phải).  ResultSet rs = statement.executeQuery(sql); |
| **resultSetType** | **Ý nghĩa** |
| TYPE\_FORWARD\_ONLY | - ResultSet chỉ cho phép duyệt từ trên xuống dưới, từ trái sang phải. Đây là kiểu mặc định của các ResultSet. |
| TYPE\_SCROLL\_INSENSITIVE | - ResultSet cho phép cuộn tiến lùi, sang trái, sang phải, nhưng không nhạy với các sự thay đổi dữ liệu dưới DB. Nghĩa là trong quá trình duyệt qua một bản ghi và lúc nào đó duyệt lại bản ghi đó, nó không lấy các dữ liệu mới nhất của bản ghi mà có thể bị ai đó thay đổi. |
| TYPE\_SCROLL\_SENSITIVE | - ResultSet cho phép cuộn tiến lùi, sang trái, sang phải, và nhạy cảm với sự thay đổi dữ liệu. |

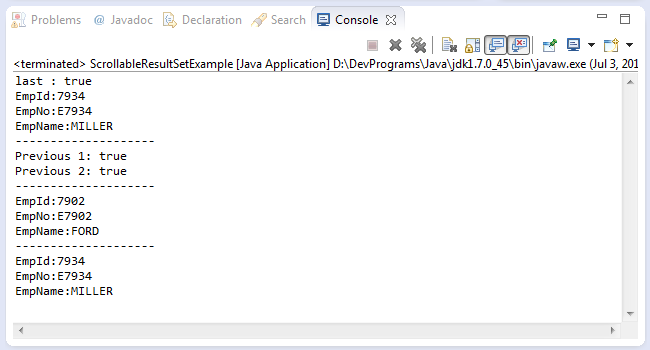
|  |  |
| --- | --- |
| **resultSetConcurrency** | **Ý nghĩa** |
| CONCUR\_READ\_ONLY | - Khi duyệt dữ liệu với các ResultSet kiểu này bạn chỉ có thể đọc dữ liệu. |
| CONCUR\_UPDATABLE | - Khi duyệt dữ liệu với các ResultSet kiểu này bạn chỉ có thể thay đổi dữ liệu tại nơi con trỏ đứng, ví dụ update giá trị cột nào đó. |

**ScrollableResultSetExample.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73 | package org.o7planning.tutorial.jdbc.basic;    import java.sql.Connection;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.sql.Statement;    import org.o7planning.tutorial.jdbc.ConnectionUtils;    public class ScrollableResultSetExample {      public static void main(String[] args) throws ClassNotFoundException,            SQLException {          // Lấy ra đối tượng Connection kết nối tới DB.        Connection connection = ConnectionUtils.getMyConnection();          // Tạo một đối tượng Statement        // Có thể cuộn dữ liệu, nhưng không nhậy với các thay đổi dưới DB.        // Con trỏ chỉ có khả năng đọc, không có khả năng update dữ liệu trong quá trình duyệt.        Statement statement = connection.createStatement(                ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_READ\_ONLY);          String sql = "Select Emp\_Id, Emp\_No, Emp\_Name from Employee";          // Thực thi câu lệnh SQL trả về đối tượng ResultSet.        ResultSet rs = statement.executeQuery(sql);            // Nhẩy con trỏ tới cuối        boolean last = rs.last();          System.out.println("last : "+ last);            if(last) {            // Ghi ra thông tin bản ghi cuối.            System.out.println("EmpId:" + rs.getInt(1));            System.out.println("EmpNo:" + rs.getString(2));            System.out.println("EmpName:" + rs.getString(3));        }            System.out.println("--------------------");          // Nhẩy con trỏ lùi lại lần 1        boolean previous =rs.previous();        System.out.println("Previous 1: "+ previous);          // Nhẩy lùi con trỏ lần 2        previous =rs.previous();        System.out.println("Previous 2: "+ previous);            // Duyệt trên kết quả trả về.        while (rs.next()) {            // Lấy dữ liệu cột 2            String empNo = rs.getString(2);            // Rồi mới lấy dữ liệu cột 1.            int empId = rs.getInt(1);              String empName = rs.getString("Emp\_Name");              System.out.println("--------------------");            System.out.println("EmpId:" + empId);            System.out.println("EmpNo:" + empNo);            System.out.println("EmpName:" + empName);        }        // Đóng kết nối        connection.close();    }    } |

Kết quả chạy ví dụ:



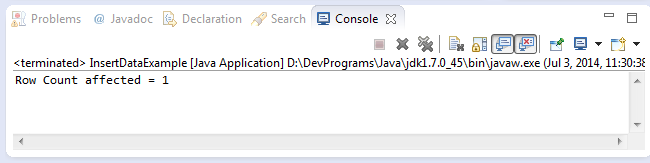
9- Ví dụ Insert dữ liệu

**InsertDataExample.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | package org.o7planning.tutorial.jdbc.basic;    import java.sql.Connection;  import java.sql.SQLException;  import java.sql.Statement;    import org.o7planning.tutorial.jdbc.ConnectionUtils;    public class InsertDataExample {      public static void main(String[] args) throws ClassNotFoundException,            SQLException {          // Lấy ra kết nối tới cơ sở dữ liệu.        Connection connection = ConnectionUtils.getMyConnection();          Statement statement = connection.createStatement();          String sql = "Insert into Salary\_Grade (Grade, High\_Salary, Low\_Salary) "                + " values (2, 20000, 10000) ";          // Thực thi câu lệnh.        // executeUpdate(String) sử dụng cho các loại lệnh Insert,Update,Delete.        int rowCount = statement.executeUpdate(sql);          // In ra số dòng được trèn vào bởi câu lệnh trên.        System.out.println("Row Count affected = " + rowCount);      }  } |

Kết quả chạy ví dụ:



10- PreparedStatement

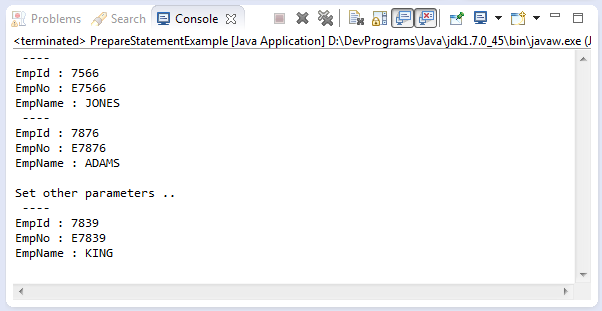
***PreparedStatement*** là một Interface con của ***Statement***.   
***PreparedStatement***sử dụng để chuẩn bị trước các câu lệnh SQL, và tái sử dụng nhiều lần, giúp cho chương trình thực hiện nhanh hơn.

**PrepareStatementExample.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | package org.o7planning.tutorial.jdbc.pareparedstatement;    import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;    import org.o7planning.tutorial.jdbc.ConnectionUtils;    public class PrepareStatementExample {      public static void main(String[] args) throws ClassNotFoundException,            SQLException {        // Lấy ra kết nối tới cơ sở dữ liệu.        Connection connection = ConnectionUtils.getMyConnection();          // Tạo một câu SQL có 2 tham số (?)        String sql = "Select emp.Emp\_Id, emp.Emp\_No, emp.Emp\_Name, emp.Dept\_Id from Employee emp "                + " where emp.Emp\_Name like ? and emp.Dept\_Id = ? ";          // Tạo một đối tượng PreparedStatement.        PreparedStatement pstm = connection.prepareStatement(sql);          // Sét đặt giá trị tham số thứ nhất (Dấu ? thứ nhất)        pstm.setString(1, "%S");        // Sét đặt giá trị tham số thứ hai (Dấu ? thứ hai)        pstm.setInt(2, 20);          ResultSet rs = pstm.executeQuery();          while (rs.next()) {            System.out.println(" ---- ");            System.out.println("EmpId : " + rs.getInt("Emp\_Id"));            System.out.println("EmpNo : " + rs.getString(2));            System.out.println("EmpName : " + rs.getString("Emp\_Name"));        }          System.out.println();        System.out.println("Set other parameters ..");          // Tái sử dụng PreparedStatement.        // Sét đặt các tham số khác.        pstm.setString(1, "KI%");        pstm.setInt(2,10);          // Thực thi câu lệnh truy vấn.        rs = pstm.executeQuery();          while (rs.next()) {            System.out.println(" ---- ");            System.out.println("EmpId : " + rs.getInt("Emp\_Id"));            System.out.println("EmpNo : " + rs.getString(2));            System.out.println("EmpName : " + rs.getString("Emp\_Name"));        }    }  } |

Kết quả chạy ví dụ:



11- CallableStatement

**CallableStatement** được xây dựng để gọi một thủ tục (procedure) hoặc hàm (function) của SQL.

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5 | // Câu lệnh gọi thủ tục SQL trên Java  String sql = "{call procedure\_name(?,?,?)}";    // Câu lệnh gọi một hàm SQL trên Java  String sql ="{? = call function\_name(?,?,?)}"; |

Để làm ví dụ với **CallableStatement** chúng ta cần một hàm hoặc một thủ tục trong DB. Với **Oracle**, **MySQL** hoặc **SQLServer**bạn có thể tạo nhanh một thủ tục như dưới đây:

* **ORACLE**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | -- Thủ tục lấy ra thông tin của một nhân viên,  -- Truyền vào tham số p\_Emp\_ID  (Integer)  -- Có 4 tham số đầu ra v\_Emp\_No, v\_First\_Name, v\_Last\_Name, v\_Hire\_Date    Create Or Replace Procedure Get\_Employee\_Info(p\_Emp\_Id     Integer                                              ,v\_Emp\_No     Out Varchar2                                              ,v\_First\_Name Out Varchar2                                              ,v\_Last\_Name  Out Varchar2                                              ,v\_Hire\_Date  Out Date) Is  Begin    v\_Emp\_No := 'E' || p\_Emp\_Id;    --    v\_First\_Name := 'Michael';    v\_Last\_Name  := 'Smith';    v\_Hire\_Date  := Sysdate;  End Get\_Employee\_Info; |

* **MySQL**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | -- Thủ tục lấy ra thông tin của một nhân viên,  -- Truyền vào tham số p\_Emp\_ID  (Integer)  -- Có 4 tham số đầu ra v\_Emp\_No, v\_First\_Name, v\_Last\_Name, v\_Hire\_Date    CREATE PROCEDURE get\_Employee\_Info(p\_Emp\_ID     Integer,                                      out       v\_Emp\_No        Varchar(50) ,                                      out       v\_First\_Name    Varchar(50) ,                                      Out       v\_Last\_name    Varchar(50) ,                                      Out       v\_Hire\_date      Date)  BEGIN    set v\_Emp\_No  = concat( 'E' , Cast(p\_Emp\_Id as char(15)) );  --    set v\_First\_Name = 'Michael';    set v\_Last\_Name  = 'Smith';    set v\_Hire\_date  = curdate();  END |

* **SQLServer**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

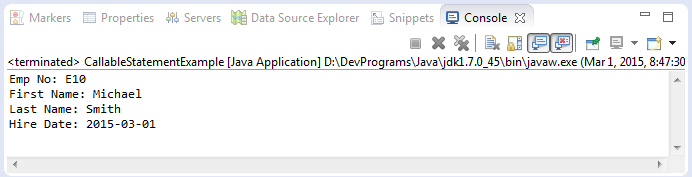
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | -- Thủ tục lấy ra thông tin của một nhân viên,  -- Truyền vào tham số p\_Emp\_ID  (Integer)  -- Có 4 tham số đầu ra v\_Emp\_No, v\_First\_Name, v\_Last\_Name, v\_Hire\_Date    CREATE PROCEDURE Get\_Employee\_Info       @p\_Emp\_Id       Integer ,       @v\_Emp\_No      Varchar(50)   OUTPUT,       @v\_First\_Name  Varchar(50)   OUTPUT,       @v\_Last\_Name  Varchar(50)   OUTPUT,       @v\_Hire\_Date    Date             OUTPUT  AS  BEGIN       set @v\_Emp\_No  =   'E' + CAST( @p\_Emp\_Id as varchar)  ;       --       set @v\_First\_Name = 'Michael';       set @v\_Last\_Name  = 'Smith';       set @v\_Hire\_date  = getdate();  END |

**CallableStatementExample.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | package org.o7planning.tutorial.jdbc.callablestatement;    import java.sql.CallableStatement;  import java.sql.Connection;  import java.sql.Date;  import java.sql.SQLException;    import org.o7planning.tutorial.jdbc.ConnectionUtils;    public class CallableStatementExample {      public static void main(String[] args) throws ClassNotFoundException,            SQLException {        // Lấy ra kết nối tới cơ sở dữ liệu.        Connection connection = ConnectionUtils.getMyConnection();          // Câu lệnh gọi thủ tục (\*\*\*)        String sql = "{call get\_Employee\_Info(?,?,?,?,?)}";          // Tạo một đối tượng CallableStatement.        CallableStatement cstm = connection.prepareCall(sql);          // Truyền tham số vào hàm (p\_Emp\_ID)        // (Là dấu chấm hỏi thứ 1 trên câu lệnh sql \*\*\*)        cstm.setInt(1, 10);          // Đăng ký nhận giá trị trả về tại dấu hỏi thứ 2        // (v\_Emp\_No)        cstm.registerOutParameter(2, java.sql.Types.VARCHAR);          // Đăng ký nhận giá trị trả về tại dấu hỏi thứ 3        // (v\_First\_Name)        cstm.registerOutParameter(3, java.sql.Types.VARCHAR);          // Đăng ký nhận giá trị trả về tại dấu hỏi thứ 4        // (v\_Last\_Name)        cstm.registerOutParameter(4, java.sql.Types.VARCHAR);          // Đăng ký nhận giá trị trả về tại dấu hỏi thứ 5        // (v\_Hire\_Date)        cstm.registerOutParameter(5, java.sql.Types.DATE);          // Thực thi câu lệnh        cstm.executeUpdate();          String empNo = cstm.getString(2);        String firstName = cstm.getString(3);        String lastName = cstm.getString(4);        Date hireDate = cstm.getDate(5);          System.out.println("Emp No: " + empNo);        System.out.println("First Name: " + firstName);        System.out.println("Last Name: " + lastName);        System.out.println("Hire Date: " + hireDate);      }  } |

Kết quả chạy ví dụ:



12- Điều khiển giao dịch (Transaction)

Giao dịch (Transaction) là một khái niệm quan trọng trong SQL.   
  
Ví dụ người A chuyển một khoản tiền 1000$ vào tài khoản người B như vậy trong Database diễn ra 2 quá trình:

* Trừ số dư tài khoản của người A đi 1000$
* Thêm vào số dư tài khoản của người B 1000$.

Và giao dịch được gọi là thành công nếu cả 2 bước kia thành công. Ngược lại chỉ cần 1 trong hai bước hỏng là coi như giao dịch không thành công, phải rollback lại trạng thái ban đầu.

**TransactionExample.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | package org.o7planning.tutorial.transaction;    import java.sql.Connection;  import java.sql.SQLException;    import org.o7planning.tutorial.jdbc.ConnectionUtils;    public class TransactionExample {      private static void doJob1(Connection conn) {        // Làm gì đó tại đây.        // Insert update dữ liêu.    }      private static void doJob2(Connection conn) {        // Làm gì đó tại đây.        // Insert update dữ liêu.    }      public static void main(String[] args) throws ClassNotFoundException,            SQLException {        // Lấy ra kết nối tới cơ sở dữ liệu.        Connection connection = ConnectionUtils.getMyConnection();          // Sét đặt chế độ tự động Commit thành false        // Để tự quản lý việc commit trên chương trình.        connection.setAutoCommit(false);          try {            // Làm một việc gì đó liên quan tới DB.            doJob1(connection);            // Lamf nhiệm vụ thứ 2            doJob2(connection);              // Gọi method commit dữ liệu xuống DB.            connection.commit();          }        // Có vấn đề gì đó lỗi xẩy ra.        catch (Exception e) {            e.printStackTrace();            // Rollback dữ liệu            connection.rollback();        }          // Đóng Connection.        connection.close();    }    } |

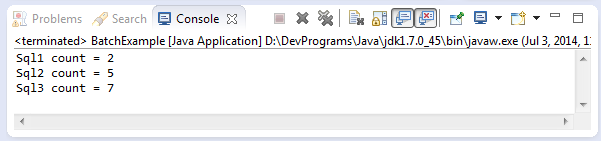
13- Thực thi một lô lệnh (Batch)

**BatchExample.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60 | package org.o7planning.tutorial.transaction;    import java.sql.Connection;  import java.sql.SQLException;  import java.sql.Statement;    import org.o7planning.tutorial.jdbc.ConnectionUtils;    public class BatchExample {      public static void main(String[] args) throws SQLException,            ClassNotFoundException {          Connection conn = ConnectionUtils.getMyConnection();          try {            // Create statement object            Statement stmt = conn.createStatement();              // Set auto-commit to false            conn.setAutoCommit(false);              // Create SQL statement            // Tạo câu lệnh Insert dữ liệu vào bảng Employee            String sql1 = "Update Employee emp set emp.Salary = emp.Salary + 100 "                    + " where emp.Dept\_Id = 10 ";            // Add above SQL statement in the batch.            // Thêm câu lệnh SQL trên vào lô            stmt.addBatch(sql1);              // Create one more SQL statement            String sql2 = "Update Employee emp set emp.Salary = emp.Salary + 20 "                    + " where emp.Dept\_Id = 20 ";            // Add above SQL statement in the batch.            // Thêm vào lô            stmt.addBatch(sql2);              // Create one more SQL statement            String sql3 = "Update Employee emp set emp.Salary = emp.Salary + 30 "                    + " where emp.Dept\_Id = 30 ";            // Add above SQL statement in the batch.            // Thêm vào lô            stmt.addBatch(sql3);              // Create an int[] to hold returned values            int[] counts = stmt.executeBatch();              System.out.println("Sql1 count = " + counts[0]);            System.out.println("Sql2 count = " + counts[1]);            System.out.println("Sql3 count = " + counts[2]);              // Explicitly commit statements to apply changes            conn.commit();        } catch (Exception e) {            e.printStackTrace();            conn.rollback();        }    }    } |

Kết quả chạy ví dụ:

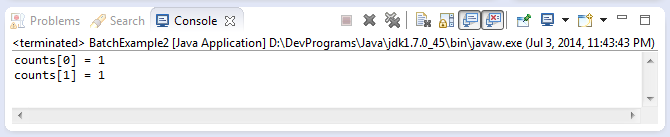


**BatchExample2.java**

[?](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | package org.o7planning.tutorial.transaction;    import java.sql.Connection;  import java.sql.Date;  import java.sql.PreparedStatement;  import java.sql.SQLException;  import java.util.UUID;    import org.o7planning.tutorial.jdbc.ConnectionUtils;    public class BatchExample2 {      public static void main(String[] args) throws ClassNotFoundException,            SQLException {          Connection conn = ConnectionUtils.getMyConnection();          try {            String sql = "Insert into Timekeeper(Timekeeper\_Id, Date\_Time, In\_Out, Emp\_Id) "                    + " values (?,?,?,?) ";            // Create statement object            PreparedStatement stmt = conn.prepareStatement(sql);              // Set auto-commit to false            conn.setAutoCommit(false);              // Sét đặt các tham số.            stmt.setString(1, UUID.randomUUID().toString());            stmt.setDate(2, new Date(System.currentTimeMillis()));            stmt.setString(3, "I");            stmt.setInt(4, 7839);            // Thêm vào lô.            stmt.addBatch();              // Sét đặt các giá trị tham số khác            stmt.setString(1, UUID.randomUUID().toString());            stmt.setDate(2, new Date(System.currentTimeMillis()));            stmt.setString(3, "I");            stmt.setInt(4, 7566);            // Thêm vào lô.            stmt.addBatch();                // Create an int[] to hold returned values            int[] counts = stmt.executeBatch();              System.out.println("counts[0] = " + counts[0]);            System.out.println("counts[1] = " + counts[1]);              // Explicitly commit statements to apply changes            conn.commit();        } catch (Exception e) {            e.printStackTrace();            conn.rollback();        }    }    } |

Kết quả chạy ví dụ:



**Kết nối Java với SQL Server**

**public** **class** SinhVienDAO {

String url = "jdbc:sqlserver://localhost:1433;instance=SQLEXPRESS;databaseName=JavaEE\_Example;integratedSecurity=True";

String userName = "sa";

String password = "123";

Connection connection;

**void** connect(){

**try** {

Class.*forName*("com.microsoft.sqlserver.jdbc.SQLServerDriver");

connection = DriverManager.*getConnection*(url, userName, password);

} **catch** (SQLException e) {

e.printStackTrace();

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

}

}

**public** ArrayList<SinhVien> getListSinhVien() {

connect();

String sql= "SELECT sv.msv, sv.HoTen, sv.GioiTinh, k.TenKhoa"+

" FROM SinhVien sv INNER JOIN Khoa AS k ON k.MaKhoa = sv.MaKhoa";

ResultSet rs = **null**;

**try** {

Statement stmt = connection.createStatement();

rs = stmt.executeQuery(sql);

} **catch** (SQLException e) {

e.printStackTrace();

}

ArrayList<SinhVien> list = **new** ArrayList<SinhVien>();

SinhVien sinhVien;

**try** {

**while**(rs.next()){

sinhVien = **new** SinhVien();

sinhVien.setMsv(rs.getString("msv"));

sinhVien.setHoTen(rs.getString("HoTen"));

sinhVien.setGioiTinh(rs.getString("GioiTinh"));

sinhVien.setTenKhoa(rs.getString("TenKhoa"));

list.add(sinhVien);

}

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** list;

}

**public** ArrayList<SinhVien> getListSinhVien(String maKhoa) {

connect();

String sql= String.*format*("SELECT sv.msv, sv.HoTen, sv.GioiTinh, k.TenKhoa"+

" FROM SinhVien sv INNER JOIN Khoa AS k ON k.MaKhoa = sv.MaKhoa"+

" WHERE sv.MaKhoa = '%s'", maKhoa);

ResultSet rs = **null**;

**try** {

Statement stmt = connection.createStatement();

rs = stmt.executeQuery(sql);

} **catch** (SQLException e) {

e.printStackTrace();

}

ArrayList<SinhVien> list = **new** ArrayList<SinhVien>();

SinhVien sinhVien;

**try** {

**while**(rs.next()){

sinhVien = **new** SinhVien();

sinhVien.setMsv(rs.getString("msv"));

sinhVien.setHoTen(rs.getString("HoTen"));

sinhVien.setGioiTinh(rs.getString("GioiTinh"));

sinhVien.setTenKhoa(rs.getString("TenKhoa"));

list.add(sinhVien);

}

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** list;

}

**public** **void** themSinhVien(String msv, String hoTen, String gioiTinh, String maKhoa) {

connect();

String sql= String.*format*("INSERT INTO SinhVien(msv,HoTen,GioiTinh,MaKhoa) "+

" VALUES ( '%s',N'%s','%s','%s' )", msv, hoTen, gioiTinh, maKhoa);

**try** {

Statement stmt = connection.createStatement();

stmt.executeUpdate(sql);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**public** SinhVien getThongTinSinhVien(String msv) {

connect();

String sql= String.*format*("SELECT HoTen, GioiTinh, MaKhoa, msv "+

" FROM SinhVien WHERE msv = '%s'", msv);

ResultSet rs = **null**;

**try** {

Statement stmt = connection.createStatement();

rs = stmt.executeQuery(sql);

} **catch** (SQLException e) {

e.printStackTrace();

}

SinhVien sinhVien = **new** SinhVien();

**try** {

**while**(rs.next()){

sinhVien.setMsv(msv);

sinhVien.setHoTen(rs.getString("HoTen"));

sinhVien.setGioiTinh(rs.getString("GioiTinh"));

sinhVien.setMaKhoa(rs.getString("MaKhoa"));

}

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** sinhVien;

}

**public** **void** suaSinhVien(String msv, String hoTen, String gioiTinh, String maKhoa) {

connect();

String sql= String.*format*("UPDATE SinhVien "+

" SET HoTen = N'%s', GioiTinh = %s, MaKhoa = '%s' " +

" WHERE msv = '%s'", hoTen, gioiTinh, maKhoa, msv);

**try** {

Statement stmt = connection.createStatement();

stmt.executeUpdate(sql);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**public** **void** xoaSinhVien(String msv) {

connect();

String sql= String.*format*("DELETE FROM SinhVien WHERE msv = '%s'", msv);

**try** {

Statement stmt = connection.createStatement();

stmt.executeUpdate(sql);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

}