LẬP TRÌNH JAVA
Kết nối cơ sở dữ liệu (JDBC)

Nguyễn Thị Hồng Anh - Đỗ Ngọc Như Loan

Nội dung

- Giới thiệu về JDBC
- Các bước làm việc với CSDL
- PreparedStatement
- CallableStatement
- ResultSetMetaData
- Transactions

Giới thiệu về JDBC

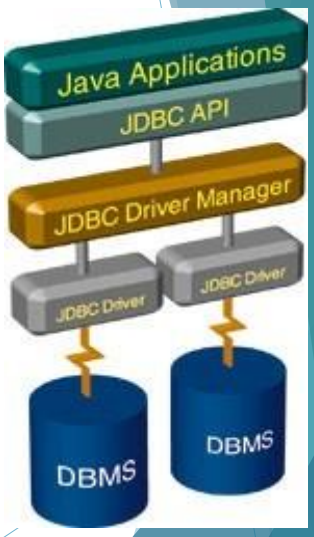
- JDBC (Java DataBase Connectivity)
- Là Java API (Application Programming Interface) chứa tập hợp các class và interface hỗ trợ xây dựng các ứng dụng Java truy cập đến các CSDL khác nhau (Access, SQL Server, MySQL, Oracle, …)
- Bao gồm 2 gói thư viện chính:
  - **java.sql.*:** chứa các lớp và giao diện cơ sở
  - **javax.sql.*:** chứa các lớp và giao diện mở rộng

Giới thiệu về JDBC

- JDBC giúp lập trình viên tạo nên các ứng dụng Java truy xuất CSDL mà không cần phải tìm hiểu và sử dụng các API độc quyền do các công ty sản xuất phần mềm khác nhau bên thứ ba cung cấp.
- JDBC API sử dụng JDBC Driver (trình điều khiển JDBC) để kết nối với CSDL.

Kiến trúc JDBC

- **JDBC API:** là 1 Java API giúp truy xuất CSDL từ Java.
- **JDBC Driver Manager:** class giúp kết nối ứng dụng Java với các JDBC Driver.
- **JDBC Driver:** phần mềm hiện thực các interface cho phép ứng dụng Java tương tác với các CSDL khác nhau.



Một số lớp và giao diện chính của JDBC

- Class **DriverManager**: cung cấp các phương thức để quản lý các Driver.
- Interface **Driver**: cung cấp các phương thức để xử lý các giao tiếp với CSDL, thường ứng dụng không giao tiếp trực tiếp với Driver mà thông qua DriverManager.
- Interface **Connection**: cung cấp các phương thức để kết nối với CSDL.

Một số lớp và giao diện chính của JDBC

- Interface **Statement**: cung cấp các phương thức để gửi các câu lệnh SQL tới CSDL và thực thi.
- Interface **ResultSet**: cung cấp các phương thức để truy cập dữ liệu trả về từ CSDL sau khi thực thi một truy vấn.
- Class **SQLException**: cung cấp các phương thức để xử lý các ngoại lệ xảy ra trong quá trình thao tác với CSDL.

JDBC Drivers

- JDBC Driver (trình điều khiển JDBC) là tập hợp các class hiện thực các JDBC interface đối với 1 hệ quản trị CSDL (DBMS) cụ thể.
- Do nhà sản xuất DBMS hoặc một đơn vị thứ ba cung cấp.
- Để truy cập các DBMS khác nhau từ chương trình viết bằng Java thì ta cần có các JDBC driver tương ứng.
- Các JDBC driver có nhiệm vụ yêu cầu DBMS thực hiện các câu lệnh SQL.

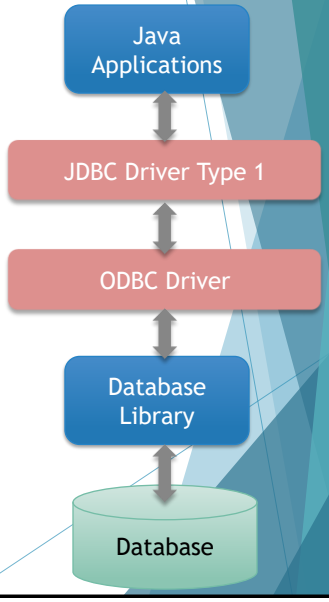
JDBC Drivers

Các JDBC Drivers được chia ra làm 4 loại:

- **Type 1:** JDBC-ODBC Bridge driver
- **Type 2:** Native-API driver
- **Type 3:** Network-Protocol driver
- **Type 4:** Native-Protocol driver / Thin driver

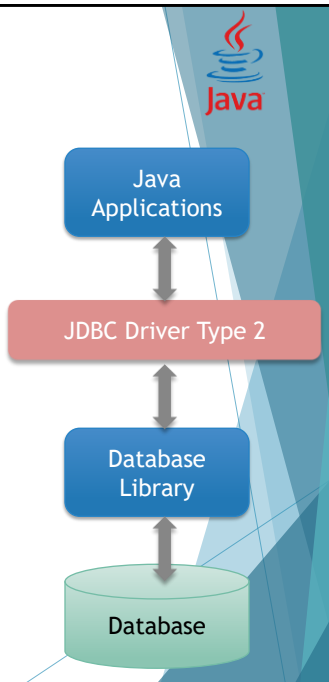
Type 1: JDBC-ODBC Bridge Driver

- Chuyển đổi các lời gọi JDBC thành lời gọi ODBC (Open Database Connectivity), ODBC là chuẩn kết nối mở có thể truy xuất DBMS.
- ODBC driver cần được cài đặt trên máy client.
- Chậm nhất trong các JDBC driver, hiện không còn được khuyến khích sử dụng.



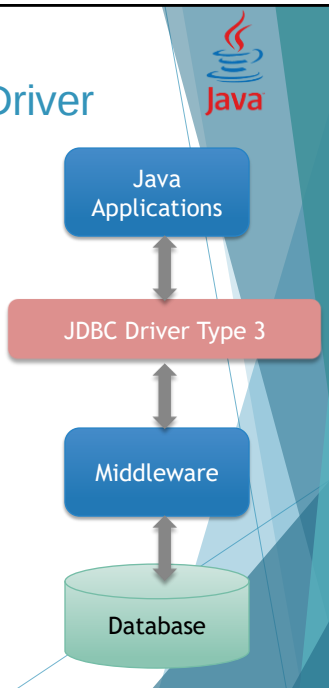
Type 2: Native-API Driver

- Chuyển các lời gọi JDBC sang thư viện hàm tương ứng với DBMS cụ thể (database API).
- Thường do nhà xây dựng DBMS cung cấp.
- Không viết hoàn toàn bằng Java.
- Thư viện client-side của database cần được cài đặt trên máy client.



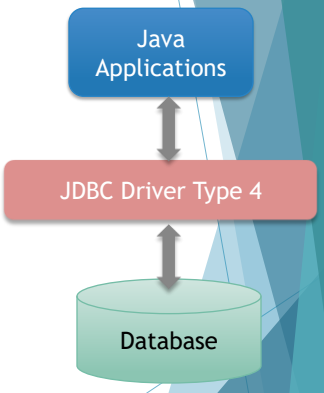
Type 3: Network-Protocol Driver

- Chuyển các lời gọi JDBC thành giao thức mạng độc lập với các DBMS.
- Sau đó 1 phần mềm trung gian (middleware) chạy trên server chuyển đổi giao thức mạng thành giao thức DBMS đặc thù.
- Có thể giao tiếp với nhiều loại CSDL.
- Thuần Java, do bên thứ 3 cung cấp.



Type 4: Native-Protocol Driver/ Thin Driver

- Chuyển các lời gọi JDBC sang các lời gọi giao thức DBMS đặc thù, cho phép giao tiếp trực tiếp với CSDL.
- Thuần Java, thường do nhà xây dựng DBMS cung cấp.
- Nhanh nhất trong các JDBC driver, được khuyến khích sử dụng.



Các bước làm việc với CSDL

1. Cài đặt JDBC Driver
2. Nạp JDBC Driver
3. Thiết lập kết nối
4. Tạo đối tượng thực hiện câu lệnh
5. Thực hiện câu truy vấn SQL
6. Xử lý kết quả trả về
7. Đóng kết nối

Ví dụ Quản lý sinh viên

Database **QLSinhVien** trong SQL Server có table **SinhVien** như sau:

Tên cột	Kiểu dữ liệu
MSSV	int
HoTen	varchar(50)
Namsinh	int

Cài đặt JDBC Driver

- Download JDBC Driver ứng với loại database sử dụng, driver được đóng gói trong JAR file.

  Ví dụ: Tải JDBC Driver SQL Server, nếu dùng JDK 11->14 thì sử dụng file **mssql-jdbc-9.2.1.jre11.jar**
- Thêm file JAR vào project.

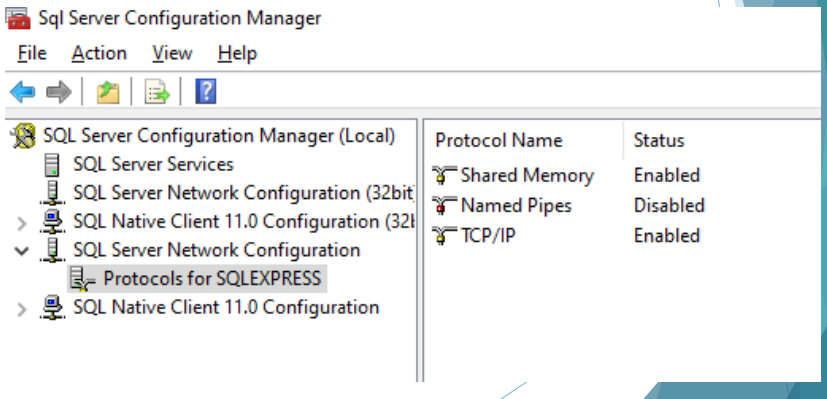
Ví dụ: Trong Netbeans project (Ant project) -> Libraries -> Click phải chọn Add JAR/Folder -> Chọn file JAR đã tải

Cài đặt JDBC Driver

- Nếu là Maven project thì có thể cài đặt SQL server JDBC driver theo cách sau trong Netbeans (*cần có internet, không cần tải trước driver*):
  - Dependencies -> Click phải chọn Add Dependency
  - Nhập các thông tin tương ứng sau để thêm:
    - groupId: com.microsoft.sqlserver
    - artifactId: mssql-jdbc
    - version: 9.2.1.jre11

Kết nối SQL Server với Java

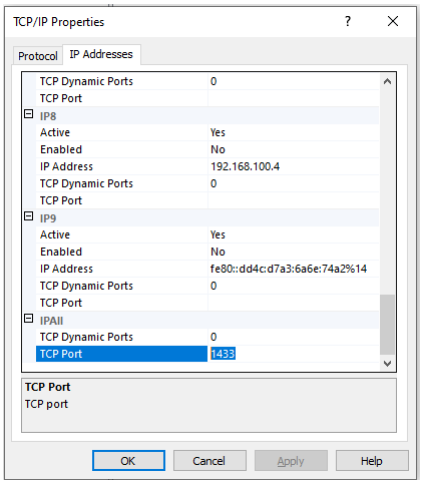
- Mở cổng kết nối SQL:
  - SQL Server Management Studio



Kết nối SQL Server với Java

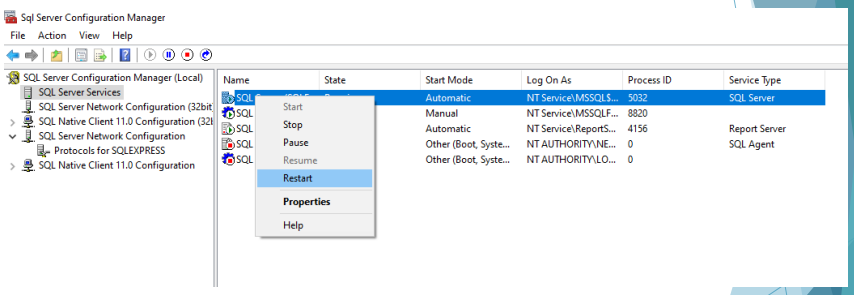
- Enable TCP/IP
- Click phải chuột lên TCP/IP chọn Properties → vào tab IP Adrresses → Thiết lập IPALL:
  - TCP Port: **1433**

Kết nối SQL Server với Java



Kết nối SQL Server với Java

- Restart SQL Server



Kết nối SQL Server với Java

- Khởi động SQL Server Management Studio
  - Nếu không mở được tài khoản sa thì có thể thực hiện đăng nhập bình thường
  - Thay đổi password của tài khoản sa: Security/ Logins/sa /Properties → thay đổi pass
  - Enable login theo sa: status → Login: enable → OK
- Ngắt kết nối → Thử kết nối lại
  - Nếu thành công : OK
  - Nếu không: Mở lại SQL → Chọn Server → Properties → Security → Đánh dấu chọn mục SQL Server and Windows Authentication mode → OK

Cài đặt JDBC Driver cho SQL Server

1. Tải JDBC Driver SQL Server
2. Giải nén thư mục
3. Tìm đường dẫn đến **JAR file** ứng với bản JDK/JRE đang dùng. Ví dụ nếu dùng JDK/JRE 11->14 thì sử dụng file **mssql-jdbc-9.2.1.jre11.jar**
4. Thêm đường dẫn đến JAR file vào CLASSPATH

Nạp JDBC Driver

- Sử dụng phương thức tĩnh **forName()** của lớp **Class** như sau:

  Class.forName(driverName);

- Trong đó **driverName** là tên trình điều khiển JDBC.
- Phương thức này ném ra ClassNotFoundException.
- JDBC Driver Type 3 tự động nạp.
- Kể từ Java 1.6 JDBC Drivers được tự động nạp.

Nạp JDBC Driver

Trình điều khiển của **MySQL**

Class.forName(“com.mysql.jdbc.Driver”);

Trình điều khiển của **Oracle**

Class.forName(“oracle.jdbc.driver.OracleDriver”);

Trình điều khiển của **Microsoft SQL Server**

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

Ví dụ

```
import java.sql.*;
public class JDBCexample{
    public static void main(String args[]){
        try{
          Class.forName(
          “com.microsoft.sqlserver.jdbc.SQLServerDriver");
        } catch(Exception e){
          System.out.println(e); }
    }
}
```

Thiết lập kết nối

- Để thiết lập kết nối ta gọi phương thức tĩnh **getConnection()** của lớp **DriverManager**, phương thức này trả về một thể hiện của lớp Connection, theo dạng như sau:
  Connection con = DriverManager.getConnection(dbUrl, username, password);
- Trong đó:
  - **dbUrl**: là chuỗi kết nối đến CSDL
  - **username** : tên người dùng đăng nhập
  - **password** : mật khẩu đăng nhập
- SQLException sẽ được ném ra nếu có lỗi trong quá trình kết nối đến CSDL.

Thiết lập kết nối

Chuỗi kết nối cho **MySQL**
jdbc:mysql://hostname:portNumber/databaseName
Chuỗi kết nối cho **Oracle**
jdbc:oracle:thin:@hostname:portNumber:databaseName
Chuỗi kết nối đến **Microsoft SQL Server**
jdbc:sqlserver://hostname:portNumber;databaseName=dataBaseName

Ví dụ

```
import java.sql.*;
public class JDBCexample{
    public static void main(String args[]){
        try{
          Class.forName(
          “com.microsoft.sqlserver.jdbc.SQLServerDriver");
          String dbUrl =  "jdbc:sqlserver://localhost:1433;
DatabaseName=QLSinhvien” ;
          String username = “sa”; String password= “sa”;
          Connection con=DriverManager.getConnection(
dbUrl, username, password);
        } catch(Exception e){
          System.out.println(e); }
} }
```


Tạo đối tượng thực hiện câu lệnh

- Sử dụng phương thức **createStatement()** của đối tượng Connection để tạo đối tượng Statement.

Ví dụ:

Statement s = con.createStatement();

- Đối tượng này có nhiệm vụ gửi các câu lệnh SQL đến CSDL.
- Cùng một đối tượng Statement có thể sử dụng cho nhiều câu lệnh SQL khác nhau.

Ví dụ

```
import java.sql.*;
public class JDBCexample{
    public static void main(String args[]){
        try{
          …
          Connection con=DriverManager.getConnection(dbUrl,
username, password);
          Statement stmt=con.createStatement();
        } catch(Exception e){
          System.out.println(e); }
    }
}
```


Thực hiện câu truy vấn SQL

- Có 3 phương thức thực thi
  - executeQuery()
  - executeUpdate()
  - execute()

Phương thức executeQuery()

- Nhận câu lệnh truy vấn SQL SELECT làm đối số
- Trả về đối tượng ResultSet
- Ví dụ:

```
ResultSet rs =
    stmt.executeQuery("SELECT * FROM SinhVien");
```


Ví dụ

```
import java.sql.*;
public class JDBCexample{
    public static void main(String args[]){
        try{
          …
          Connection con=DriverManager.getConnection(dbUrl,
username, password);
          Statement stmt=con.createStatement();
          ResultSet rs = stmt.executeQuery("SELECT *
FROM SinhVien");
        } catch(Exception e){
          System.out.println(e); }
  } }
```

Phương thức executeUpdate()

- Nhận các câu lệnh SQL dạng cập nhật làm đối số: CREATE, ALTER, DROP, UPDATE, INSERT, DELETE.
- Trả về số nguyên biểu thị số hàng được cập nhật/thêm/xóa.
- Ví dụ:

int i = s.executeUpdate("DELETE FROM SinhVien WHERE MSSV=1");

Ví dụ

```
import java.sql.*;
public class JDBCexample{
    public static void main(String args[]){
        try{
            …
            Connection con=DriverManager.getConnection(dbUrl, username, password);
            Statement stmt=con.createStatement();
            int t = stmt.executeUpdate("DELETE FROM SinhVien WHERE MSSV=1");
            System.out.println("So dong da xoa: " + t);
        } catch(Exception e){
            System.out.println(e); }
}}
```

Phương thức execute()

- Được áp dụng cho trường hợp không rõ loại SQL nào được thực hiện.
- Có thể trả về nhiều kết quả.
- Trả về kiểu boolean:
  - **true** nếu trả về ResultSet, dùng phương thức getResultSet() để lấy kết quả
  - **false** nếu trả về số dòng được cập nhật hoặc không trả về kết quả, dùng phương thức getUpdateCount() để lấy số dòng được cập nhật

Ví dụ

```
import java.sql.*;
public class JDBCexample{
    public static void main(String args[]){
        try{
            …
            Statement stmt=con.createStatement();
            boolean result  = stmt.execute("SELECT * FROM SinhVien");
            if (result) {
                ResultSet rs = stmt.getResultSet(); }
            else {
                int i = stmt.getUpdateCount(); }
    } catch(Exception e){
            System.out.println(e); } } }
```

Xử lý kết quả trả về

- ResultSet là một đối tượng dạng bảng được trả về khi truy vấn (query) dữ liệu.
- executeUpdate() không trả về ResultSet.
- Có thể di chuyển tới lui trong ResultSet để lấy dữ liệu ra.
- Kiểu và chế độ hoạt động đồng thời cho ResultSet có thể được truyền vào làm đối số cho phương thức createStatement:

  createStatement(int resultSetType, int resultSetConcurrency);

Các kiểu ResultSet

- **TYPE_FORWARD_ONLY (mặc định):** Con trỏ của ResultSet kiểu này chỉ được di chuyển theo một hướng từ đầu đến cuối
- **TYPE_SCROLL_INSENSITIVE:** Con trỏ có thể di chuyển tới lui tương đối với vị trí hiện tại của nó, và cũng có thể di chuyển đến một vị trí cụ thể, không bị ảnh hưởng nếu kết quả được thay đổi ở nơi khác
- **TYPE_SCROLL_SENSITIVE:** Con trỏ có thể di chuyển tới lui tương đối với vị trí hiện tại của nó, và cũng có thể di chuyển đến một vị trí cụ thể, sẽ bị ảnh hưởng nếu kết quả bị thay đổi nơi khác

Các chế độ hoạt động đồng thời của ResultSet

- **CONCUR_READ_ONLY (mặc định):** Xác định chế độ hoạt động đồng thời, kết quả lưu trong đối tượng ResultSet không được thay đổi
- **CONCUR_UPDATABLE:** Xác nhận chế độ hoạt động đồng thời, kết quả lưu trong đối tượng ResultSet được thay đổi

Các phương thức của ResultSet

- **next():** di chuyển con trỏ đến dòng kế, trả về true nếu có dòng kế tiếp, false nếu đến cuối ResultSet
- **previous():** di chuyển con trỏ đến dòng trước
- **first():** di chuyển con trỏ đến dòng đầu tiên
- **last():** di chuyển con trỏ đến dòng cuối cùng

Các phương thức của ResultSet

- **beforeFirst():** di chuyển con trỏ đến vị trí trước dòng đầu tiên
- **afterLast():** di chuyển con trỏ đến sau dòng cuối cùng
- **relative(int rows):** di chuyển con trỏ tương đối với vị trí hiện tại của nó với số dòng là rows
- **absolute(int row):** di chuyển con trỏ đến dòng thứ row

Lấy dữ liệu từ ResultSet

Các phương thức để lấy dữ liệu từ cột:

- **get<Type>(String col_name)**
- **get<Type>(int col_index)**
- Trong đó:
  - **<Type>** là kiểu dữ liệu được trả về
  - **col_name** là tên của cột cần lấy dữ liệu
  - **col_index** là chỉ số của cột cần lấy dữ liệu, **chỉ số bắt đầu từ 1** (không phải từ 0)
- Ví dụ:

```
String name = rs.getString("NAME");
double val = rs.getDouble(“2");
```

Ví dụ

```
import java.sql.*;
public class JDBCexample{
    public static void main(String args[]){
        try{
          …
          ResultSet rs = stmt.executeQuery("SELECT * FROM
SinhVien");
          while(rs.next())
            System.out.println("MSSV: " + rs.getInt(1) +
          "  Ho Ten: " + rs.getString(2) + "  Nam sinh: " +
rs.getInt(3));
        } catch(Exception e){
          System.out.println(e); }
 } }
```


Đóng kết nối

- Để đóng kết nối ta sử dụng phương thức **close()** của đối tượng connection:

  Ví dụ: **con.close()**

- Đóng đối tượng Connection thì đối tượng Statement và ResultSet sẽ tự động đóng.
- Nên đóng connection sau khi hoàn tất.

Ví dụ

```
import java.sql.*;
public class JDBCexample{
    public static void main(String args[]){
        try{
          …
          ResultSet rs = stmt.executeQuery(“SELECT * FROM SinhVien”);
          while(rs.next())
            System.out.println(“MSSV: “ + rs.getInt(1) +
          "  Ho Ten: “ + rs.getString(2) + "  Nam sinh: “ + rs.getInt(3));
          con.close();
        } catch(Exception e){
          System.out.println(e); } } }
```


Ví dụ 1 – JDBC

```
import java.sql.*;
class JDBCexample1 {
   public static void main(String args[]) {
      try {
        Class.forName(
         “com.microsoft.sqlserver.jdbc.SQLServerDriver");
        String dbUrl =  "jdbc:sqlserver://localhost:1433;
        DatabaseName=QLSinhvien” ;
        String username = “sa”; String password= “sa”;
        Connection con=DriverManager.getConnection( dbUrl,
        username, password);
        Statement stmt = con.createStatement();
```

Ví dụ 1 - JDBC

```
            ResultSet rs = stmt.executeQuery("SELECT * FROM
SinhVien WHERE NamSinh = 2000");
            while(rs.next())
              System.out.println("MSSV: " +
rs.getInt("MSSV") +  "  Ho Ten: " + rs.getString("HoTen") +
"  Nam sinh: " + rs.getInt("NamSinh"));
             con.close();
          } catch(Exception e){
            System.out.println(e); }
      }
}
```


Ví dụ 2 – JDBC

```
import java.sql.*;
class JDBCexample2 {
   public static void main(String args[]) {
      try {
        Class.forName(
         "com.microsoft.sqlserver.jdbc.SQLServerDriver");
        String dbUrl =  "jdbc:sqlserver://localhost:1433;
        DatabaseName=QLSinhvien" ;
        String username = "sa"; String password= "sa";
        Connection con=DriverManager.getConnection( dbUrl,
        username, password);
```

Ví dụ 2 - JDBC

```
        Statement stmt=con.createStatement(
ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
        ResultSet rs=stmt.executeQuery(“SELECT * FROM
SinhVien");
        rs.absolute(3);       //getting the record of 3rd row
        rs.updateInt(2, “Hung”);  //update the second column
        rs.updateInt(“Namsinh", 1999);  //update Namsinh
        rs.updateRow();
        con.close();
    } catch (Exception e) {
        System.out.println(e); }
}
```


Ví dụ 3 - JDBC

```
import java.sql.*;
public class BasicJDBCDemo {
  Connection con;
  public static void main(String[] args) {
    new BasicJDBCDemo(); }
  public BasicJDBCDemo() {
    try {
      Class.forName( “com.microsoft.sqlserver.jdbc.SQLServerDriver");
      String dbUrl = "jdbc:sqlserver://localhost:1433;
DatabaseName=QLSinhvien” ;
      String username = “sa”; String password= “sa”;
      Connection con=DriverManager.getConnection(dbUrl,
username, password);
```

Ví dụ 3 - JDBC

```
        doSelectTest();
        doInsertTest();
        doUpdateTest();
        doDeleteTest();
        doSelectTest();
        con.close();
      }
      catch (Exception ex) {
        System.err.println(ex.getMessage());
      }
}
```


Ví dụ 3 – doSelectTest()

```
private void doSelectTest() {
   String query = "SELECT * FROM SINHVIEN";
   try {
     Statement st = con.createStatement();
     ResultSet rs = st.executeQuery(query);
     while (rs.next()) {
      int mssv = rs.getInt(“MSSV");
       String hoten = rs.getString(“HoTen");
       int  namsinh = rs.getInt(“Namsinh");
       System.out.println(mssv+“  “+hoten+"    "+namsinh);
     }
   } catch (SQLException ex) {
     System.err.println(ex.getMessage()); }
}
```

Ví dụ 3 - doInsertTest()

```
private void doInsertTest() {
    try {
      Statement st = con.createStatement();
      st.executeUpdate("INSERT INTO SinhVien VALUES
(20, 'Phuong', 1998)");
      st.executeUpdate("INSERT INTO SinhVien VALUES
(21, 'An', 2000)");
     } catch (SQLException ex) {
      System.err.println(ex.getMessage());
    }
  }
```


Ví dụ 3 – doUpdateTest()

```
private void doUpdateTest() {
    try {
      Statement st = con.createStatement();
      st.executeUpdate("UPDATE SinhVien SET Namsinh=1999
WHERE MSSV=20");
    } catch (SQLException ex) {
      System.err.println(ex.getMessage());
    }
}
```

Ví dụ 3 - doDeleteTest()

```
private void doDeleteTest() {
    try {
      Statement st = con.createStatement();
     st.executeUpdate("DELETE FROM SinhVien WHERE
MSSV=21");
    } catch (SQLException ex) {
      System.err.println(ex.getMessage());
    }
}
}
```


Bài tập 1

- Tạo form và viết các chức năng thêm, xóa, sửa, hiển thị

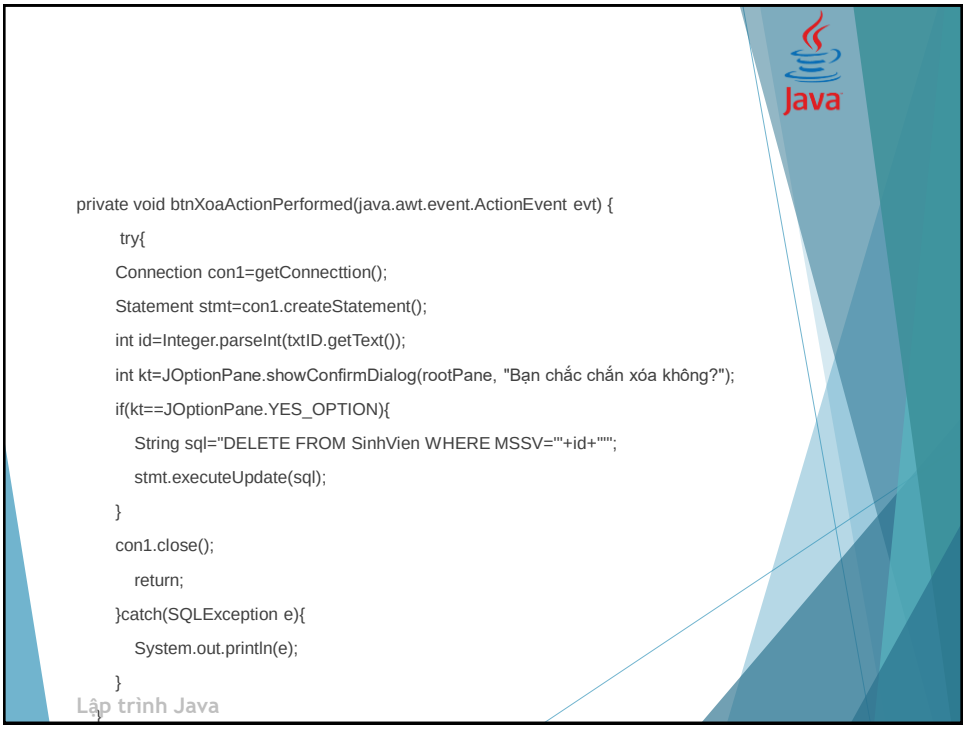
Bài tập 1

- Tạo phương thức kết nối:

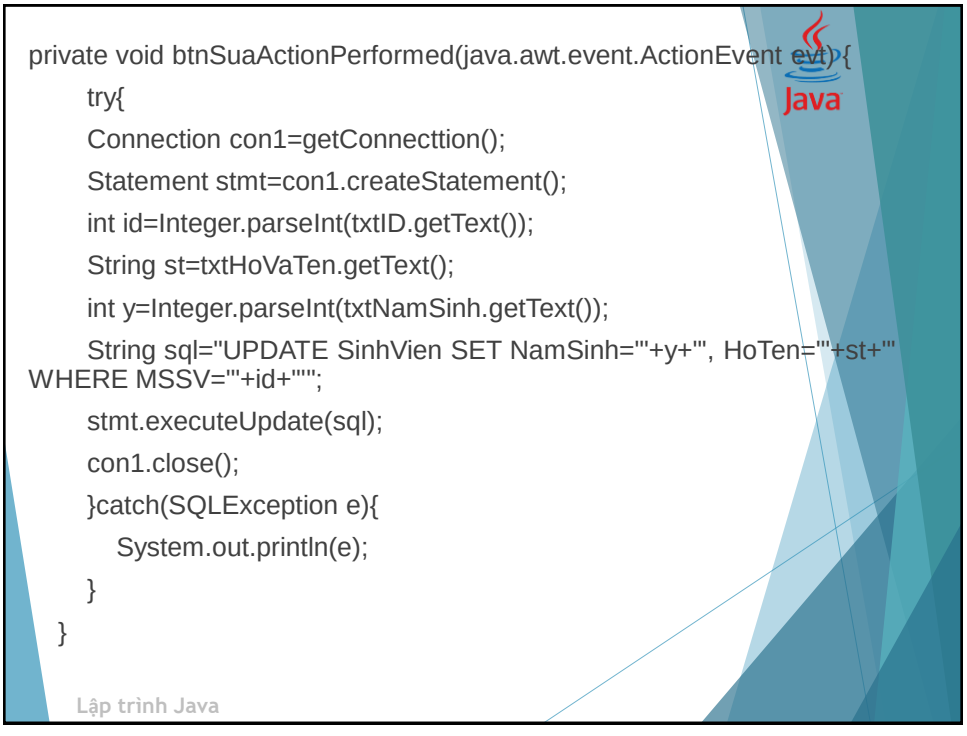
```
public Connection getConnecttion() throws Exception{

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        String dbUrl = "jdbc:sqlserver://localhost:1433;
DatabaseName=JavaSinhvien" ;
        String username = "sa"; String password= "123";
        Connection con=DriverManager.getConnection( dbUrl,
username, password);
        return con;
  }
```



```
private void btnThemActionPerformed(java.awt.event.ActionEven
      try{
      Connection con1=getConnecttion();
      Statement stmt=con1.createStatement();
      int id=Integer.parseInt(txtID.getText());
      String st=txtHoVaTen.getText();
      int y=Integer.parseInt(txtNamSinh.getText());
      String sql="INSERT INTO SinhVien VALUES('"+id+"','"+st+"','"+y+"
      stmt.executeUpdate(sql);
      con1.close();
      }catch(SQLException e){
          System.out.println(e);
      }
  }
```



```
private void btnXoaActionPerformed(java.awt.event.ActionEvent evt) {
        try{
        Connection con1=getConnecttion();
        Statement stmt=con1.createStatement();
        int id=Integer.parseInt(txtID.getText());
        int kt=JOptionPane.showConfirmDialog(rootPane, "Bạn chắc chắn xóa không?");
        if(kt==JOptionPane.YES_OPTION){
           String sql="DELETE FROM SinhVien WHERE MSSV='"+id+"'";
           stmt.executeUpdate(sql);
        }
        con1.close();
           return;
        }catch(SQLException e){
           System.out.println(e);
        }
    }
```

```
private void btnSuaActionPerformed(java.awt.event.ActionEvent evt){
    try{
    Connection con1=getConnecttion();
    Statement stmt=con1.createStatement();
    int id=Integer.parseInt(txtID.getText());
    String st=txtHoVaTen.getText();
    int y=Integer.parseInt(txtNamSinh.getText());
    String sql="UPDATE SinhVien SET NamSinh='"+y+"', HoTen='"+st+"'
WHERE MSSV='"+id+"'";
    stmt.executeUpdate(sql);
    con1.close();
    }catch(SQLException e){
       System.out.println(e);
    }
  }
```

Bài tập 2

a) Tạo CSDL trong SQL Server bao gồm 2 bảng:

SanPham(MaSP, TenSP, Gia, MaLoaiSP)

LoaiSanPham(MaLoaiSP, TenLoaiSP)

b) Viết chương trình cho phép hiển thị danh sách các sản phẩm (gồm các cột MaSP, TenSP, Gia, TenLoaiSP)

c) Cho phép thêm, sửa, xóa sản phẩm

d) Cho phép tìm sản phẩm theo loại sản phẩm

Lưu ý: Làm bài tập với giao diện Console trước, sau đó xây dựng giao diện đồ họa cho ứng dụng.

PreparedStatement

- PreparedStatement kế thừa từ Statement.
- Sử dụng để thực hiện các câu **truy vấn SQL động hoặc có tham số**,
- Thường dùng khi cần thực hiện nhiều câu lệnh SQL có cấu trúc tương tự nhau, chỉ có giá trị là thay đổi. Khi đó PreparedStatement được dùng để soạn trước câu lệnh có sẵn cấu trúc cần thiết, giá trị sẽ được đưa vào như những đối số khi câu lệnh được thực thi.
- Giúp tránh SQL Injection.

Ví dụ 1 - PreparedStatement

```
public void prepareStatement1(){
    String insertQuery = "INSERT INTO SinhVien
VALUES(?,?,?)";
    if (con != null){
        try{
            PreparedStatement prest =
    con.prepareStatement(insertQuery);
            prest.setInt(1, 30);
            prest.setString(2, “Minh");
            prest.setInt(3, 2001);
            prest.executeUpdate();
        } catch(SQLException e){
            System.out.println(e); }
} }
```


Ví dụ 2 - PreparedStatement

```
public void prepareStatement2(){
    String insertQuery = "INSERT INTO SinhVien
VALUES(?,?,?)";
    int n = 4;
    int mssv[] = {“40”,”41”,”42”,”43”};
    String hoten[] = {“Lan”,”Hong”,”Son”,”Duy”};
    int namsinh[] = {“2000”,”2002”,”2001”,”2002”};
    if (con != null){
        try{
            PreparedStatement prest =
    con.prepareStatement(insertQuery);
```

Ví dụ 2 - PreparedStatement

```
            for (int i = 0; i<n; i++) {
                prest.setInt(1, mssv[i]);
                prest.setString(2, hoten[i]);
                prest.setInt(3, namsinh[i]);
                prest.executeUpdate();
            }
        } catch(SQLException e){
            System.out.println(e);}
    }
}
```


CallableStatement

- CallableStatement kế thừa từ PreparedStatement.
- CallableStatement được sử dụng để thực thi **stored procedures**.
- Stored procedures là một tập hợp các câu lệnh SQL dùng thực hiện 1 công việc nào đó, được gọi như 1 phương thức hay hàm.
- Stored procedure là khái niệm khá phổ biến và được hầu hết các DBMS hỗ trợ.

Ví dụ - CallableStatement

```
CREATE PROCEDURE insertStudent
    @MSSV INT,
    @HoTen VARCHAR(50),
    @NamSinh INT
AS
BEGIN
    INSERT INTO SinhVien
        VALUES(@MSSV,@HoTen,@NamSinh);
END
```


Ví dụ - CallableStatement

```
public void CallableStatement {
    if (con != null){
        try{
            CallableStatement callSt =
con.prepareCall("{call insertStudent(?,?,?)}");
            callSt.setInt(1,50);
            callSt.setString(2, “Nam”);
            callSt.setInt(3, 1999);
            callSt.execute();
        } catch (SQLException e) {
            System.out.println(e);}
} }
```

ResultSetMetaData

- ResultSetMetaData cung cấp các thông tin về cấu trúc cụ thể của ResultSet, ví dụ số lượng cột, tên và kiểu của chúng.
- Phương thức **getMetaData()** của ResultSet trả về đối tượng ResultSetMetaData.

Ví dụ:

ResultSetMetaData rsmd = rs.getMedaData();

Ví dụ - ResultSetMetaData

```
import java.sql.*;
public class ResultSetMetaDataExample {
   public static void main(String[] args) {
     try {
          … // connect database
          Statement stmt = con.createStatement(SELECT * FROM
SinhVien);
          ResultSet rs = stmt.executeQuery();
          ResultSetMetaData rsmd = rs.getMetaData();
          System.out.println("Tong so cot cua bang: "
               + rsmd.getColumnCount());
```

Ví dụ - ResultSetMetaData

```
            System.out.println("Ten cot thu 2: "
                + rsmd.getColumnName(2));
            System.out.println("Kieu du lieu cua cot thu 2: "
                + rsmd.getColumnTypeName(2));
            con.close();
        } catch (Exception e) {
            System.out.println(e);}
    }
}
```

Kết quả:

Tong so cot cua bang: 3

Ten cot thu 2: HoTen

Kieu du lieu cua cot thu 2: VARCHAR

Transactions

- Mặc định sau khi mỗi câu lệnh SQL được thực thi qua JDBC, dữ liệu sẽ được cập nhật ngay vào CSDL (AutoCommit).
- Có những trường hợp, ta muốn dữ liệu chỉ được cập nhật vào CSDL sau khi 1 nhóm câu lệnh SQL được thực hiện.
- Một nhóm các câu lệnh như thế được gọi là một giao dịch (transaction).

Transactions

- Ví dụ:

Chuyển 5 triệu từ tài khoản A sang tài khoản B, các bước thực hiện:

1) Trừ 5 triệu trong tài khoản A

2) Cộng 5 triệu trong tài khoản B

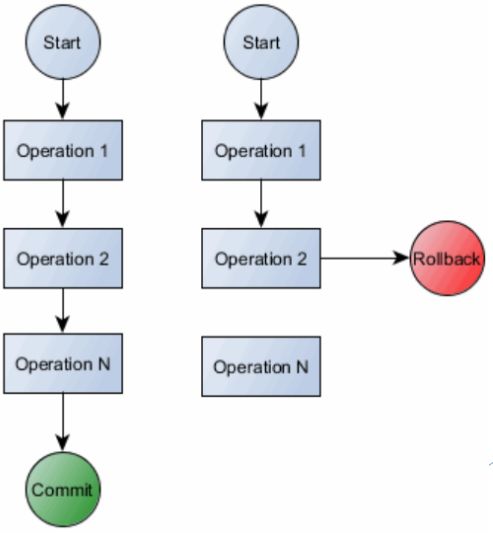
3) Lưu lại thông tin giao dịch để rà soát sau

Nếu bị lỗi giữa chừng thì sao?

Transactions

- Transaction là cơ chế đảm bảo tính toàn vẹn của dữ liệu trong CSDL.
- Transaction kết thúc với **commit** hoặc **rollback.**
  - **commit** (thực hiện giao dịch) nếu thực hiện thành công tất cả các lệnh trong giao dịch
  - **rollback** (hủy giao dịch) nếu có lỗi xảy ra trong quá trình thực hiện giao dịch

Transactions



Transactions

- Khi rollback sẽ hủy dữ liệu đến thời điểm nó được commit gần nhất.
- Tùy thuộc vào hệ quản trị CSDL mà cách rollback có thể khác nhau.
- Dữ liệu sau khi được commit sẽ không hủy được bằng rollback.
- Dùng phương thức **commit()** và **rollback()** của đối tượng Connection để thực thi / hủy giao dịch.

Ví dụ: **con.commit()** //thực thi giao dịch

Transactions

- Dùng phương thức **setAutoCommit(boolean)** của đối tượng Connection để bật/tắt chế độ tự động commit.

Ví dụ: Tắt chế độ tự động commit để thực hiện giao dịch

con.setAutoCommit(false);

- Nếu không cần dùng ở chế độ giao dịch nữa, ta nên trả lại chế độ commit tự động

Ví dụ - Transactions

```
public void transaction(){
    String insertQuery = "INSERT INTO SinhVien
VALUES(?,?,?)";
    if (con != null){
        try{
            con.setAutoCommit(false);
            PreparedStatement prest =
    con.prepareStatement(insertQuery);
            prest.setInt(1, 60);
            prest.setString(2, “Hai");
            prest.setInt(3, 2001);
            prest.executeUpdate();
            prest.setInt(1, 61);
```

Ví dụ - Transactions

```
            prest.setString(2, “Linh");
            prest.setInt(3, 2002);
            prest.executeUpdate();
            con.commit();
            con.setAutoCommit(true);
        } catch(SQLException e){
            try {
                con.rollback();
            } catch (SQLException e1) {
                System.out.println(e1); }
            System.out.println(e); }
    }
}
```


Bài tập

1) Tạo bảng Users trong CSDL gồm các cột ID (int, auto-increment, primary key), Username (varchar), Password (varchar).

2) Xây dựng form **Đăng nhập**, kết nối CSDL để kiểm tra thông tin đăng nhập (sử dụng PreparedStatement). Nếu thông tin đăng nhập đúng thì tắt cửa sổ đăng nhập, mở ra cửa sổ **Home** (frame) với dòng chữ Hello + username.

4) Xây dựng form **Đăng ký**: Kiểm tra username không được trùng, không chứa khoảng trắng, password ít nhất 6 ký tự, nếu hợp lệ thì thêm vào CSDL (sử dụng stored procedure và CallableStatement). Sau khi thêm, hiện dialog thông báo, khi click OK sẽ chuyển đến form Đăng nhập.

Bài tập

5) Màn hình Home có nút Đăng xuất, click vào chuyển đến form Đăng nhập.

6) Thêm nút Đăng ký vào form đăng nhập, click vào chuyển đến form Đăng ký.

7) Khi bắt đầu chạy ứng dụng sẽ hiển thị form Đăng nhập.

8) Khi tắt bất kỳ cửa sổ làm việc nào ứng dụng sẽ kết thúc.