

JAVA SWING

Lập trình JAVA



NỘI DUNG

- Giao diện đồ họa với Swing
- Xử lý sự kiện trên giao diện
- Kết nối cơ sở dữ liệu

JAVA SWING LÀ GÌ

- Là bộ công cụ giao diện người dùng đồ họa (GUI) cho ứng dụng desktop
- Là một phần của JFC (Java Foundation Classes),
 - JFC là API cung cấp GUI để lập trình Java GUI
- Cung cấp một bộ widget và các package phong phú để tạo ra các thành phần GUI phù hợp dành cho các ứng dụng Java.
- Package javax.swing cung cấp các lớp cho Java Swing API: JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser...

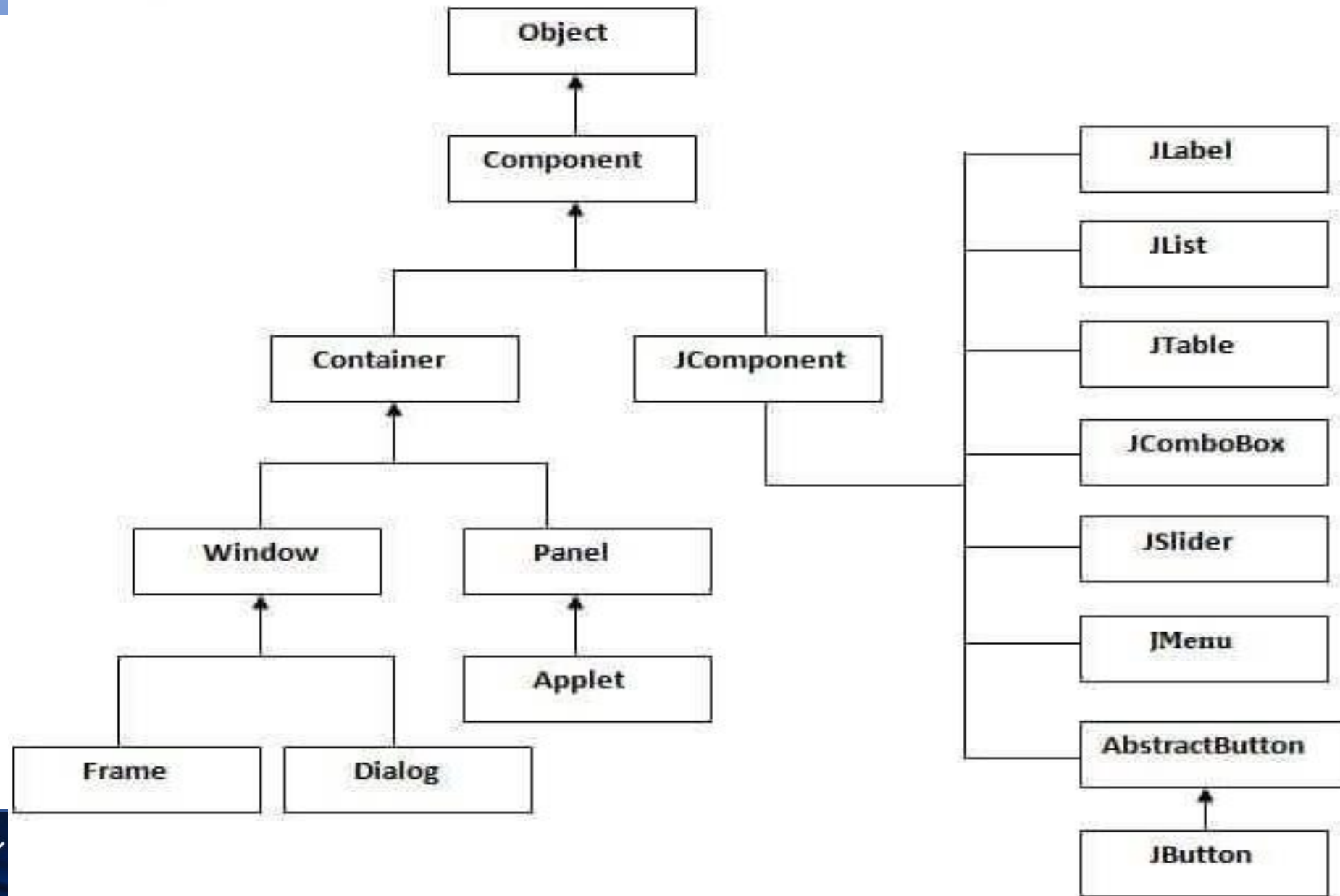
JAVA SWING

- Là bộ công cụ giao diện người dùng đồ họa (GUI) cho ứng dụng desktop
- Là một phần của JFC (Java Foundation Classes),
 - JFC là API cung cấp GUI để lập trình Java GUI
- Cung cấp một bộ widget và các package phong phú để tạo ra các thành phần GUI phù hợp dành cho các ứng dụng Java.
- Package `javax.swing` cung cấp các lớp cho Java Swing API: `JButton`, `JTextField`, `JTextArea`, `JRadioButton`, `JCheckbox`, `JMenu`, `JColorChooser`...

JAVA SWING: ĐẶC ĐIỂM

- Độc lập nền tảng.
- Có thể tùy chỉnh, mở rộng.
- Khá nhẹ.
- Có thể cấu hình.
- Tuân theo cấu trúc MVC (Model View Controller)
- Được định nghĩa bên trong package java.awt

CÁU TRÚC PHÂN CẤP LỚP



"Component" CLASS

- Là lớp gốc của tất cả các thành phần Swing
 - Tất cả các thành phần giao diện trong Swing đều kế thừa từ lớp `java.awt.Component`.
- Định nghĩa các thuộc tính và phương thức cơ bản của các thành phần đồ họa (như kích thước, vị trí, sự kiện, v.v.).

Phương thức	Mô tả
<code>add(Component c)</code>	thêm component con
<code>setSize(int width, int height)</code>	thiết lập kích thước
<code>setLayout(LayoutManager m)</code>	thiết lập trình quản lý bố cục (layout)
<code>setVisible(boolean b)</code>	thiết lập khả năng hiển thị (mặc định là false)

"Container" CLASS

- Có thể chứa các thành phần giao diện khác
- Được sử dụng để tạo bố cục (layout)

Các lớp con:

- JFrame: Cửa sổ chính của ứng dụng
- JWindow: Cửa sổ độc lập (không đủ các thành phần như cửa sổ chính)
- JDialog: Hộp thoại (thường dùng để yêu cầu người dùng nhập liệu hoặc xác nhận)
- JPanel: Một vùng, thường dùng để định bố cục.
- JScrollPane: Thanh cuộn.

JComponent CLASS (1)

- Là lớp con của lớp Container
- Là lớp cơ sở cho tất cả các thành phần Swing như button, label, text field,

JButton

JLabel

JTextField

JTextArea

JPasswordField

JCheckBox

JRadioButton

JList

JComboBox

JMenu

JTable

JTree

JComponent CLASS (2)

- XEM TRONG TÀI LIỆU (*Java_Swing_2nd_Edition.pdf*)

JButton

JLabel

JTextField

JTextArea

JPasswordField

JCheckBox

JRadioButton

JList

JComboBox

JMenu

JTable

JTree

SỰ KIỆN (EVENT)

- Là sự thay đổi trạng thái của một đối tượng
- Thường được tạo ra do tương tác của người dùng với các thành phần giao diện
 - Click vào nút, di chuyển chuột, nhập ký tự từ bàn phím, chọn một mục từ danh sách, cuộn trang, ...
- Được xử lý bởi 1 hành động nào đó

CÁC LOẠI SỰ KIỆN

- **Foreground Events:**

- Xảy ra khi có sự tương tác trực tiếp của người dùng trên giao diện của ứng dụng
- Ví dụ: nhấn vào nút, di chuyển chuột, nhập ký tự từ bàn phím, chọn một mục từ danh sách, cuộn trang, ...

- **Background Events:**

- Xảy ra khi một điều kiện nào đó thỏa mãn hoặc có sự thay đổi trạng thái trong hệ thống.
- Ví dụ: lỗi phần cứng/phần mềm, hết hạn bộ đếm thời gian, hoàn tất thao tác, đến một thời điểm định trước, hết thời gian định trước, có sự thay đổi trên dữ liệu, ...

XỬ LÝ SỰ KIỆN

- Là cơ chế kiểm soát sự kiện và quyết định điều gì sẽ xảy ra nếu sự kiện xảy ra.
 - Khi 1 sự kiện xảy ra → xử lý như thế nào
- Trình xử lý sự kiện (Event Handler):
 - Chịu trách nhiệm xử lý sự kiện
 - Là đoạn mã lệnh sẽ được thực thi khi sự kiện xảy ra.
- Java sử dụng Delegation Event Model (Mô hình sự kiện ủy quyền) để xử lý các sự kiện.

DELEGATION EVENT MODEL

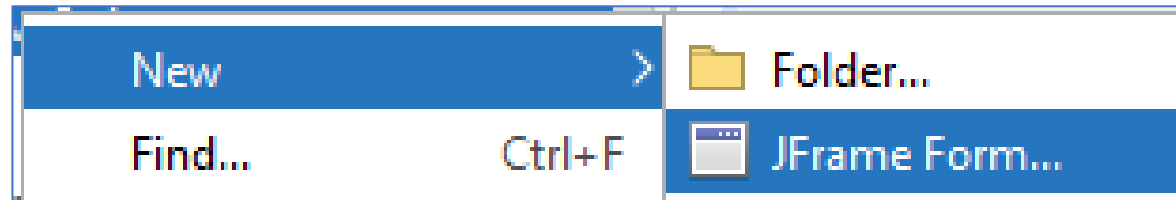
- Định nghĩa cơ chế sinh và xử lý sự kiện
- Các phần tử giao diện người dùng (UI) có thể ủy quyền xử lý sự kiện cho một đoạn mã lệnh nào đó.
- Có 2 thành phần tham gia trong mô hình:
 - Source: đối tượng mà sự kiện xảy ra. Source có trách nhiệm cung cấp thông tin về sự kiện đã xảy ra cho trình xử lý của nó.
 - Listener: trình xử lý sự kiện, chịu trách nhiệm phản hồi/xử lý cho một sự kiện (thực thi đoạn mã lệnh)

CÁC LỚP SỰ KIỆN

- Các sự kiện (như click chuột, nhập từ bàn phím, thay đổi, ...) được xử lý thông qua các lớp riêng biệt
- Các lớp sự kiện:
 - `ActionEvent`: Sự kiện chung về các hành động
 - `MouseEvent`: Sự kiện chuột.
 - `KeyEvent`: Sự kiện bàn phím.
 - `WindowEvent`: Sự kiện cửa sổ.

JFrame

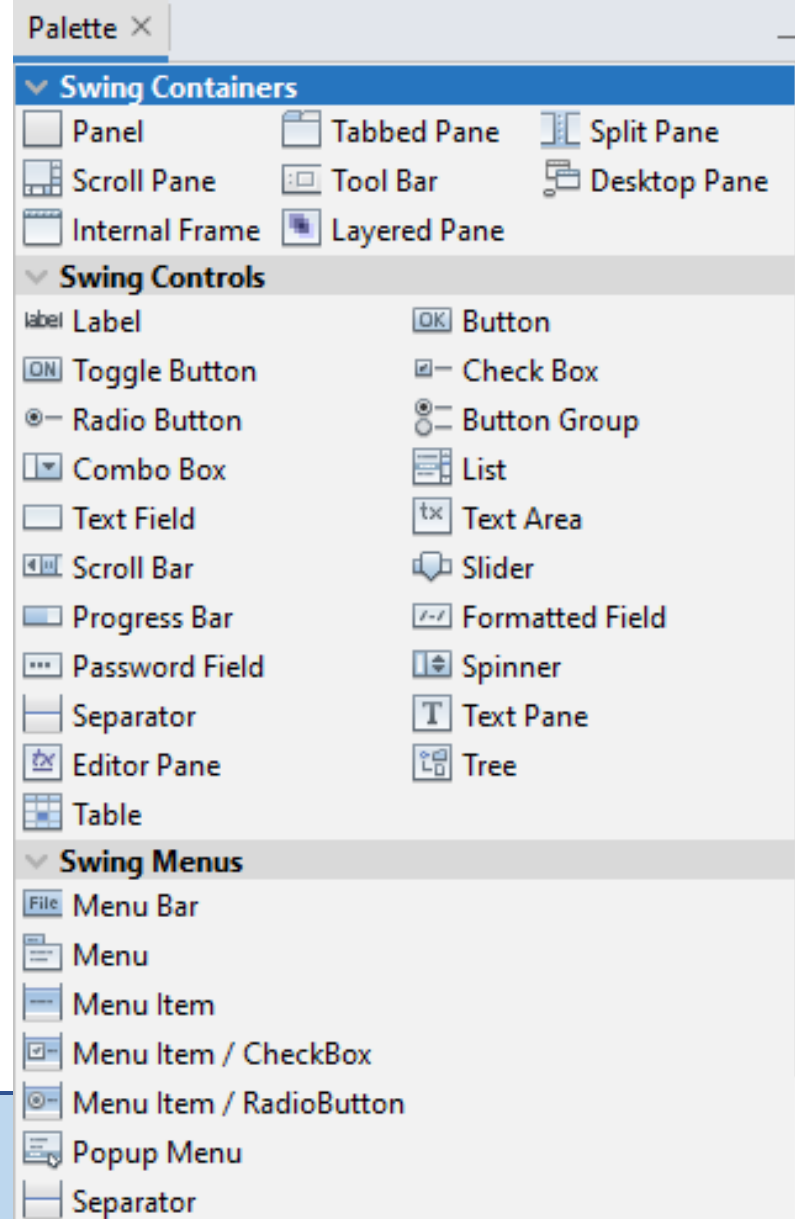
- Tạo cửa sổ trong ứng dụng GUI
- Chứa các thành phần giao diện khác: label, button, textfield, ...
- Kế thừa từ lớp java.awt.Frame, nhưng nó cung cấp nhiều tính năng và khả năng mở rộng hơn



```
class Swing1Dialog extends javax.swing.JFrame { }
```


CÁC THÀNH PHẦN GIAO DIỆN (1)

- Được thêm vào JFrame bằng cách:
 - Khai báo trong code
 - Kéo-thả từ thanh Palette vào JFrame



CÁC THÀNH PHẦN GIAO DIỆN (2)

The screenshot displays an IDE window titled "Calculator.java" with the "Design" tab selected. On the left, the "Other Components - Navigator" pane shows a tree structure under "Form Calculator" > "Other Components" > "[JFrame]". The components listed are:

- label jLabel1 [JLabel]
- label jLabel2 [JLabel]
- txtNumber1 [JTextField]
- jSeparator1 [JSeparator]
- jSeparator2 [JSeparator]
- label jLabel7 [JLabel]
- txtNumber2 [JTextField]
- label jLabel8 [JLabel]
- txtResult [JTextField]
- btnMultiple [JButton]
- btnAdd [JButton]
- btnDevide [JButton]
- btnSubtract [JButton]

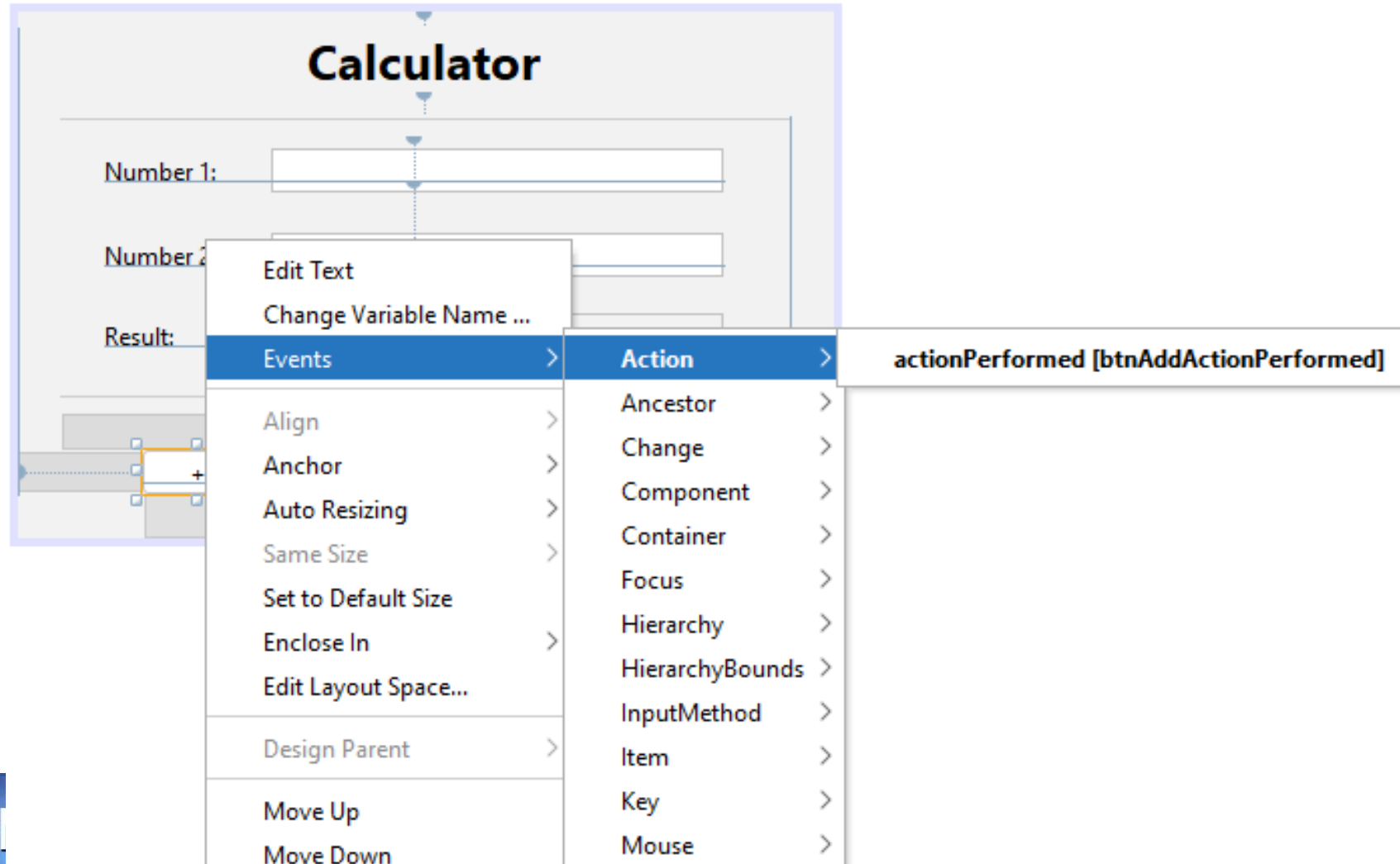
The main design area shows a visual representation of the calculator. It features a title bar, a title "Calculator", and a light gray background. The layout includes:

- Two input fields labeled "Number 1:" and "Number 2:".
- A result field labeled "Result:".
- A row of four buttons: "+", "-", "*", and "/".

A lightbulb icon and a text box above the design area state: "The Preview Design button (in the toolbar) enables you to test the design of the form."

SỰ KIỆN (1)

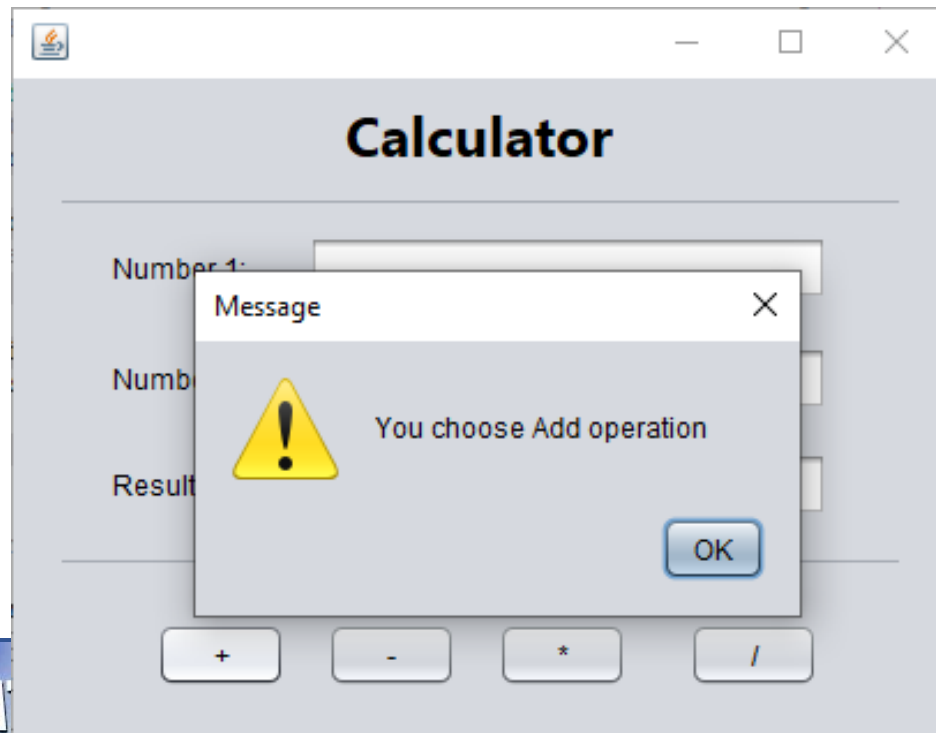
- Định nghĩa Event Handler cho các sự kiện trên giao diện



SỰ KIỆN (2)

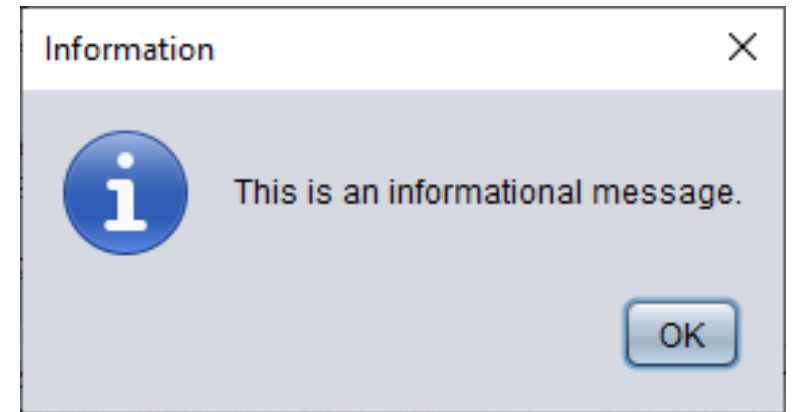
- Định nghĩa Event Handler cho các sự kiện trên giao diện

```
private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {  
    JOptionPane.showMessageDialog(this, "You choose Add operation",  
        "Message", JOptionPane.WARNING_MESSAGE);  
}
```



“JOptionPane” CLASS (1)

- Được sử dụng để hiển thị các hộp thoại (dialog boxes) (thông báo, yêu cầu nhập dữ liệu, xác nhận, ...)
- `showMessageDialog(Component parentComponent, Object message, String title, int messageType)`
 - Hiển thị một hộp thoại thông báo
 - `messageType`: kiểu thông báo
 - `JOptionPane.INFORMATION_MESSAGE`
 - `JOptionPane.WARNING_MESSAGE`
 - `JOptionPane.ERROR_MESSAGE`
 - `JOptionPane.QUESTION_MESSAGE`



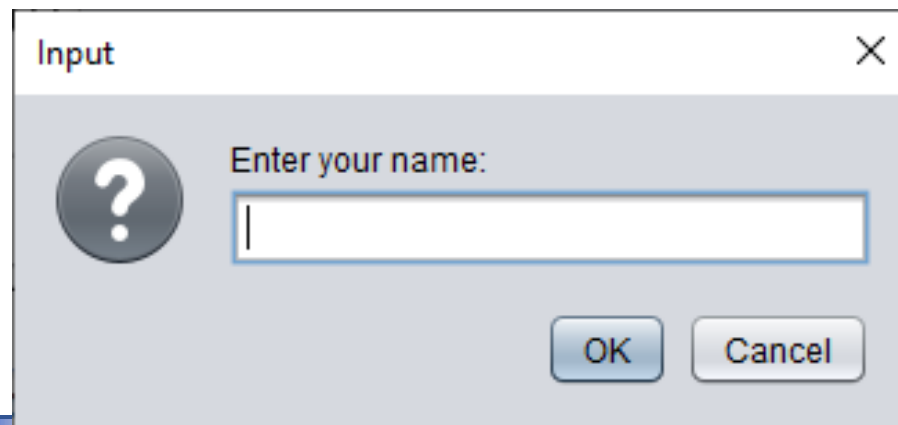
```
JOptionPane.showMessageDialog(null, "This is an informational message.",  
                                "Information", JOptionPane.INFORMATION_MESSAGE);
```

“JOptionPane” CLASS (2)

`showInputDialog(Component parentComponent, Object message, String title, int messageType)`

- Hiện thị một hộp thoại yêu cầu nhập dữ liệu

```
String name = JOptionPane.showInputDialog(null, "Enter your name:", "Input",  
                                           JOptionPane.QUESTION_MESSAGE);
```



“JOptionPane” CLASS (3)

```
showConfirmDialog(Component parentComponent, Object  
message, String title, int optionType)
```

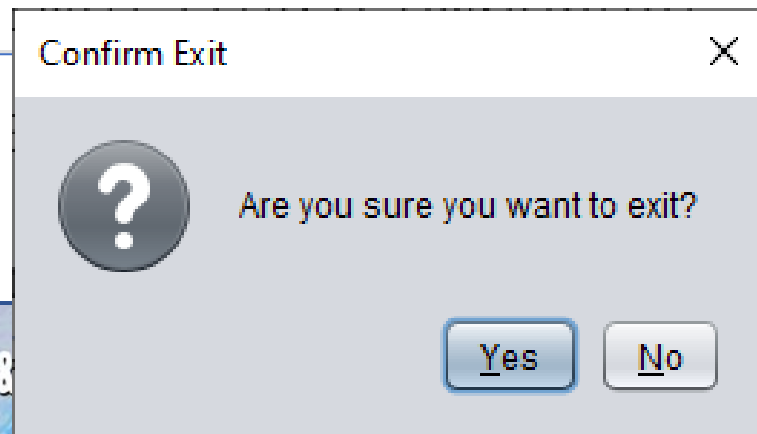
Hiển thị một hộp thoại yêu cầu xác nhận

- optionType: kiểu xác nhận
 - JOptionPane.YES_NO_OPTION
 - JOptionPane.YES_NO_CANCEL_OPTION
 - JOptionPane.OK_CANCEL_OPTION

“JOptionPane” CLASS (4)

`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)`

```
int response = JOptionPane.showConfirmDialog(null, "Are you sure you want to exit?",  
                                             "Confirm Exit", JOptionPane.YES_NO_OPTION);  
if (response == JOptionPane.YES_OPTION) {  
    JOptionPane.showMessageDialog(null, "You chose to exit.", "Goodbye",  
                                  JOptionPane.INFORMATION_MESSAGE);  
    System.exit(0);  
} else {  
    JOptionPane.showMessageDialog(null, "You chose to stay.", "Stay",  
                                  JOptionPane.INFORMATION_MESSAGE);  
}
```



“JOptionPane” CLASS (5)

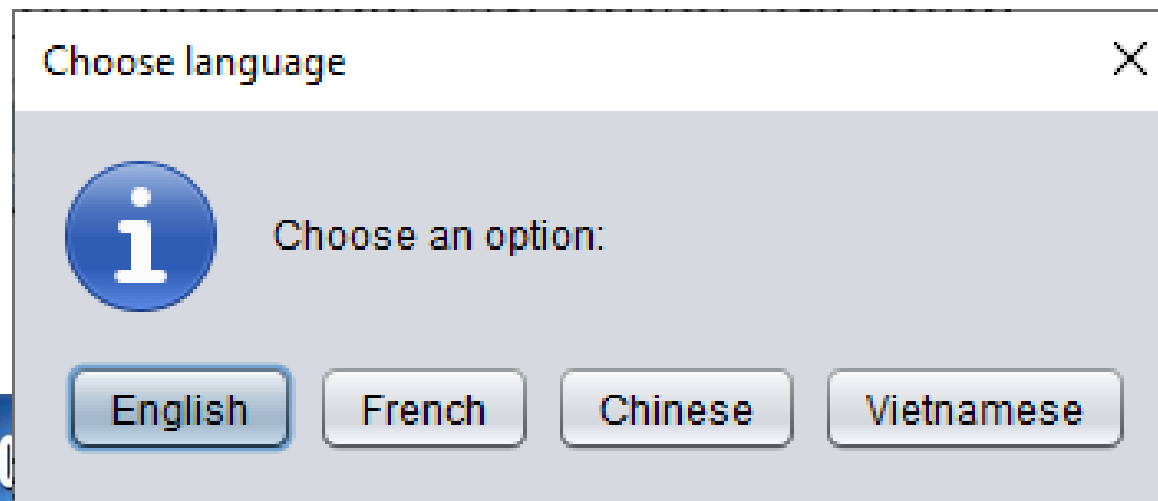
```
showOptionDialog(Component parentComponent, Object  
message, String title, int optionType, int  
messageType, Icon icon, Object[] options, Object  
initialValue)
```

Hiển thị một hộp thoại với nhiều lựa chọn

“JOptionPane” CLASS (6)

Hiển thị một hộp thoại với nhiều lựa chọn

```
String[] options = {"English", "French", "Chinese", "Vietnamese"};  
int choice = JOptionPane.showOptionDialog(null, "Choose an option: ",  
    "Choose language", JOptionPane.DEFAULT_OPTION,  
    JOptionPane.INFORMATION_MESSAGE, null, options, options[0]);  
  
JOptionPane.showMessageDialog(null, "You chose: " + options[choice],  
    "Your Choice", JOptionPane.INFORMATION_MESSAGE);
```



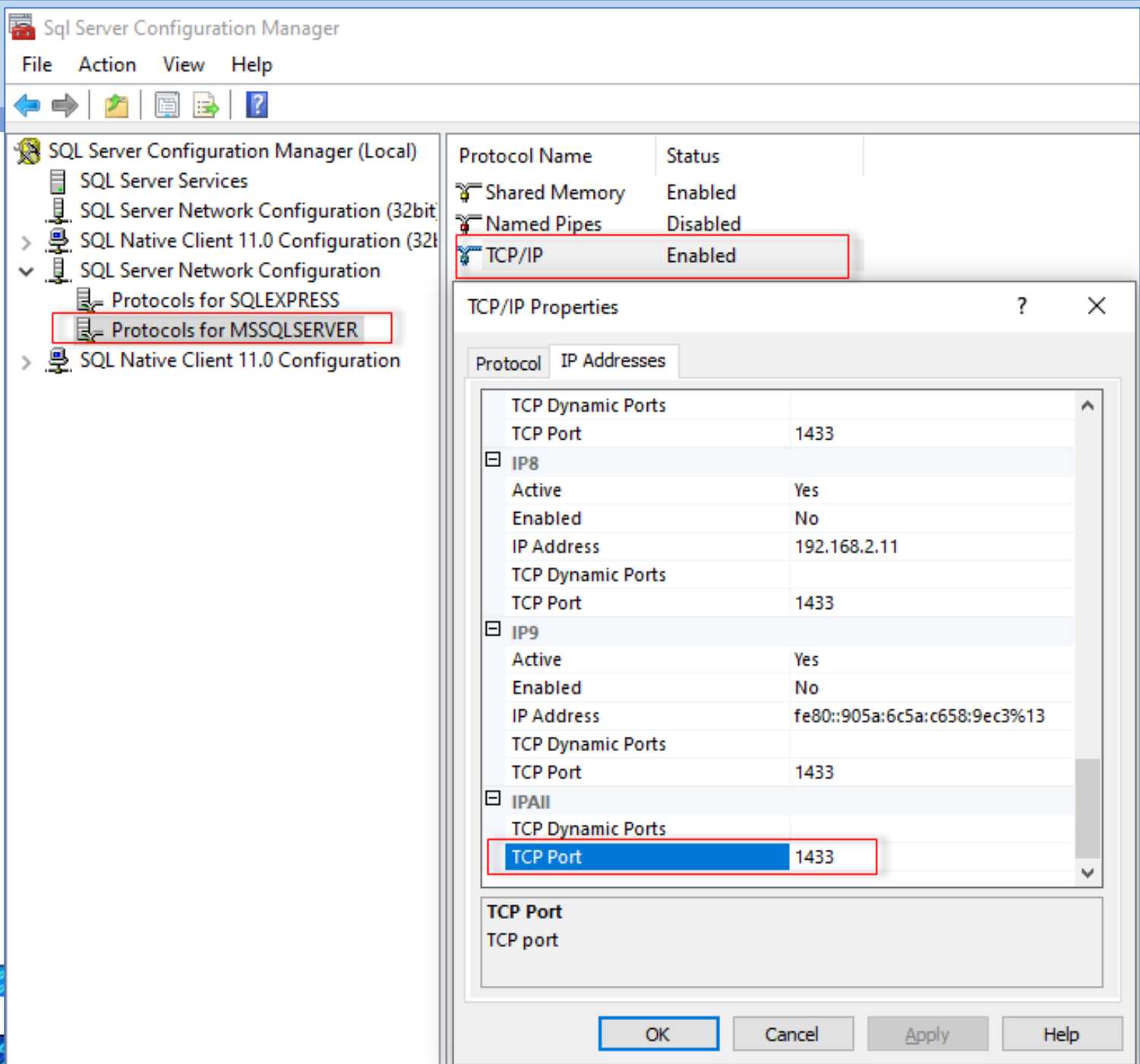
HIỂN THỊ JFRAME

- Mở Jframe khi đang ở trên 1 Jframe
- Cú pháp:

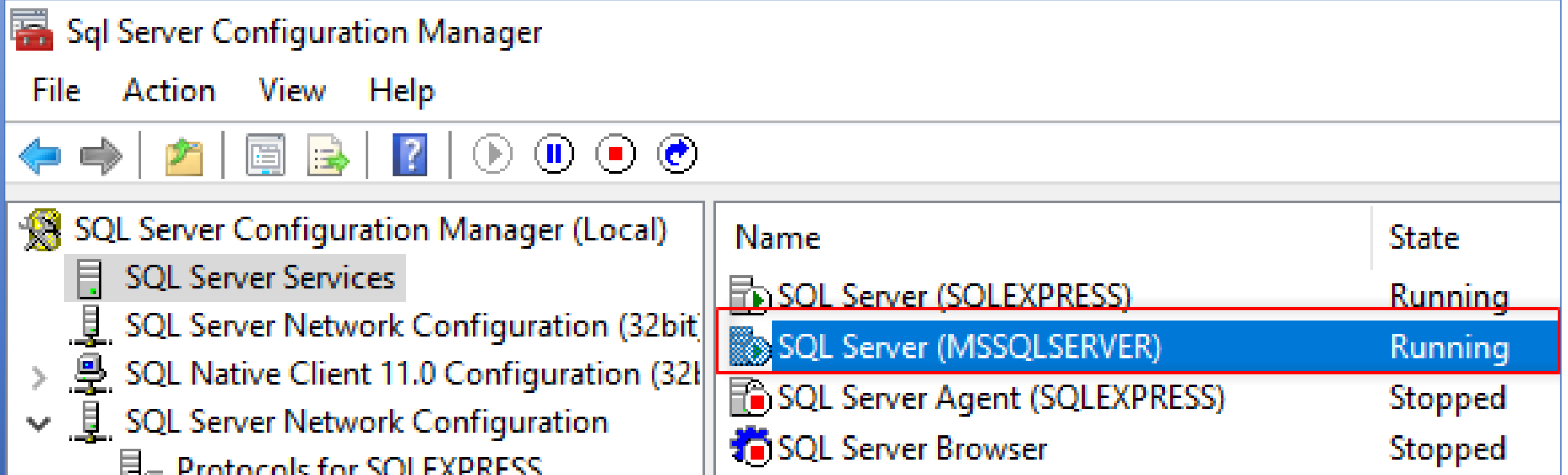
```
JFrameName f = new JFrameName();  
f.setVisible(true);  
  
this.dispose(); // Đóng frame hiện tại
```

KẾT NỐI CƠ SỞ DỮ LIỆU (1)

Bước 1. Cấu hình
kết nối trên SQL
Server



KẾT NỐI CƠ SỞ DỮ LIỆU (2)



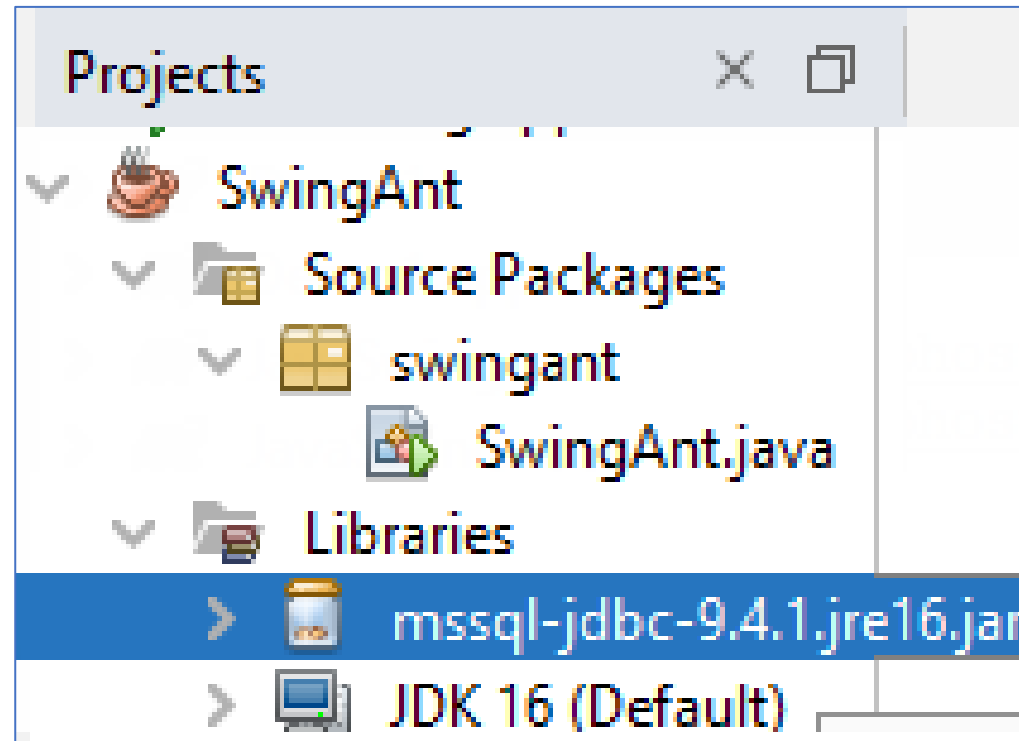
The screenshot shows the SQL Server Configuration Manager interface. The left pane displays the tree view with 'SQL Server Services' expanded. The right pane shows a list of services with their names and states. The 'SQL Server (MSSQLSERVER)' service is highlighted with a blue selection bar and a red border, indicating it is the current focus. The 'SQL Server (SOLEXPRESS)' service is also running, while 'SQL Server Agent (SOLEXPRESS)' and 'SQL Server Browser' are stopped.

Name	State
SQL Server (SOLEXPRESS)	Running
SQL Server (MSSQLSERVER)	Running
SQL Server Agent (SOLEXPRESS)	Stopped
SQL Server Browser	Stopped

Bước 2. Khởi động dịch vụ SQL Server (sau khi cấu hình)

KẾT NỐI CƠ SỞ DỮ LIỆU (3)

- 3. Cài JDBC (.JAR) vào project (phiên bản phù hợp với JDK và SQL Server)



“java.sql” PACKAGE (1)

- Cung cấp các class và interface để làm việc với cơ sở dữ liệu quan hệ thông qua JDBC (Java Database Connectivity)
 - Kết nối với cơ sở dữ liệu
 - Thực thi các câu lệnh SQL
 - Quản lý các kết quả truy vấn

“java.sql” PACKAGE (2)

- Các thành phần chính:

Connection

Statement

PreparedStatement

CallableStatement

ResultSet

ResultSetMetaData

SQLException

"Connection" CLASS (1)

- Thực hiện kết nối và truy xuất cơ sở dữ liệu
 - Thiết lập kết nối với cơ sở dữ liệu
 - Tạo các đối tượng Statement để thực thi các câu lệnh SQL
 - Yêu cầu thực thi các câu lệnh SQL được gửi đến Database Server
 - Quản lý giao dịch.
- Một khi bạn có đối tượng Connection, bạn có thể gửi yêu cầu đến cơ sở dữ liệu, thực thi các truy vấn và cập nhật dữ liệu.

"Connection" CLASS (2)

- Khởi tạo:

`DriverManager.getConnection(connectionstring)`

- Connection String: gồm thông tin server, port, database name, username (user), password

```
jdbc:sqlserver://<host>:<port>;databaseName=<dbName>;user=<username>;password=<password>;
```

```
String url = "jdbc:sqlserver://localhost:1433;databaseName=Northwind;user=sa;password=sa;"  
           + "encrypt=true;trustServerCertificate=true;sslProtocol=TLSv1.2";  
try (Connection conn = DriverManager.getConnection(url)) {  
    System.out.println("Kết nối thành công!");  
}
```

"Connection" CLASS (2)

- Khởi tạo:

`DriverManager.getConnection(connectionstring)`

- Connection String: gồm thông tin server, port, database name, username (user), password

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
String url = "jdbc:sqlserver://localhost:1433;databaseName=Northwind;user=sa;password=sa;"  
           + "encrypt=true;trustServerCertificate=true;sslProtocol=TLSv1.2";  
try (Connection conn = DriverManager.getConnection(url)) {  
    System.out.println("Kết nối thành công!");  
}
```

"Statement" CLASS

- Thực thi các câu lệnh SQL
- Các phương thức:
 - `executeQuery()` : thực thi các câu lệnh SELECT. Kết quả trả về là `ResultSet`
 - `executeUpdate()` : Thực thi các câu lệnh DML như INSERT, UPDATE, DELETE. Kết quả trả về là số lượng dòng bị ảnh hưởng
 - `execute()` : Dùng để thực thi câu lệnh SQL tổng quát

```
Statement stmt = conn.createStatement();  
ResultSet rs = stmt.executeQuery("SELECT * FROM Users");
```

“PreparedStatement” CLASS (1)

- Kế thừa “Statement” class
- Thực thi các câu lệnh SQL có tham số.
- Cải thiện hiệu suất khi thực thi các câu lệnh SQL nhiều lần
- Tránh SQL injection.

"PreparedStatement" CLASS (2)

- Phương thức:

- `setString(int parameterIndex, String value)` : Thiết lập giá trị tham số kiểu `String` của câu lệnh SQL
- `setInt(int parameterIndex, int value)` : Thiết lập giá trị tham số kiểu `int` của câu lệnh SQL

```
String sql = "INSERT INTO Users (name, age) VALUES (?, ?)";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, "John");
pstmt.setInt(2, 30);
pstmt.executeUpdate();
```

"CallableStatement" CLASS

- Kế thừa "PreparedStatement" class
- Gọi thực thi các thủ tục lưu trữ (stored procedures).

```
CallableStatement cstmt = conn.prepareCall("{call my_stored_procedure(?, ?)}");  
cstmt.setInt(1, 10);  
cstmt.setString(2, "abc");  
cstmt.execute();
```

"ResultSet" CLASS (1)

- Đại diện cho một bảng dữ liệu trong cơ sở dữ liệu
- Chứa kết quả trả về của câu lệnh SELECT.
- Cho phép đọc từng dòng dữ liệu. Trên mỗi dòng có thể đọc các cột.
- Phương thức:
 - `next()` : Di chuyển con trỏ đến dòng tiếp theo trong ResultSet
 - `getString(int columnIndex) / getString(String columnName)`: Lấy dữ liệu String của cột
 - `getInt(int columnIndex) / getInt(String columnName)`: Lấy dữ liệu int của cột

"ResultSet" CLASS (2)

```
ResultSet rs = stmt.executeQuery("SELECT id, name FROM Users");  
while (rs.next()) {  
    int id = rs.getInt("id");  
    String name = rs.getString("name");  
    System.out.println("ID: " + id + ", Name: " + name);  
}
```

"ResultSetMetaData" CLASS

- Mô tả thông tin về cấu trúc của một ResultSet
 - Ví dụ: số lượng cột, tên cột, kiểu dữ liệu của các cột, ...

```
ResultSetMetaData rsMetaData = rs.getMetaData();  
int columnCount = rsMetaData.getColumnCount();
```

"SQLException" CLASS

- Là một lớp ngoại lệ (Exception) trong JDBC.
- Được ném ra khi có lỗi xảy ra trong quá trình làm việc với cơ sở dữ liệu (kết nối, thực thi truy vấn, ...)

```
try {  
    // Kết nối và thực hiện các truy vấn  
} catch (SQLException e) {  
    e.printStackTrace();  
}
```

```
import java.sql.*;

public class JDBCExample {
    public static void main(String[] args) {
        // Thông tin kết nối cơ sở dữ liệu
        String url = "jdbc:mysql://localhost:3306/mydb";
        String user = "root";
        String password = "password";
        // Kết nối cơ sở dữ liệu
        try (Connection conn = DriverManager.getConnection(url, user, password)) {
            System.out.println("Kết nối thành công!");

            // Tạo Statement để thực thi truy vấn
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT id, name FROM Users");
            // Duyệt qua kết quả của truy vấn
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                System.out.println("ID: " + id + ", Name: " + name);
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```



LƯU Ý

- try-with-resources: tự động đóng tài nguyên khi không còn sử dụng nữa
- Khi làm việc với các đối tượng như `Connection`, `Statement`, và `ResultSet`: nên sử dụng try-with-resources để tự động đóng các đối tượng khi không còn sử dụng.

```
try (Connection conn = DriverManager.getConnection(url);
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT id, name FROM Users")) {
    while (rs.next()) {
        System.out.println(rs.getInt("id") + " - " + rs.getString("name"));
    }
} catch (SQLException e) {
    e.printStackTrace();
}
```