

Bài 12

# Giao diện đồ họa

Trịnh Thành Trung

[trungtt@soict.hust.edu.vn](mailto:trungtt@soict.hust.edu.vn)

# Nội dung

1. Giao diện đồ họa người sử dụng
2. JavaFX
3. Các thành phần của JavaFX
4. UI Control
5. Layout Pane
6. Xử lý sự kiện
7. SceneBuilder

The image features a large, stylized Java logo on the right side. It consists of a teal and orange vertical bar on the left, a blue coffee cup with steam rising from it, and the word "java" in orange lowercase letters with a trademark symbol (TM) at the bottom right.

java™

# 1

## Giao diện đồ họa người sử dụng

Graphical User Interface (GUI)

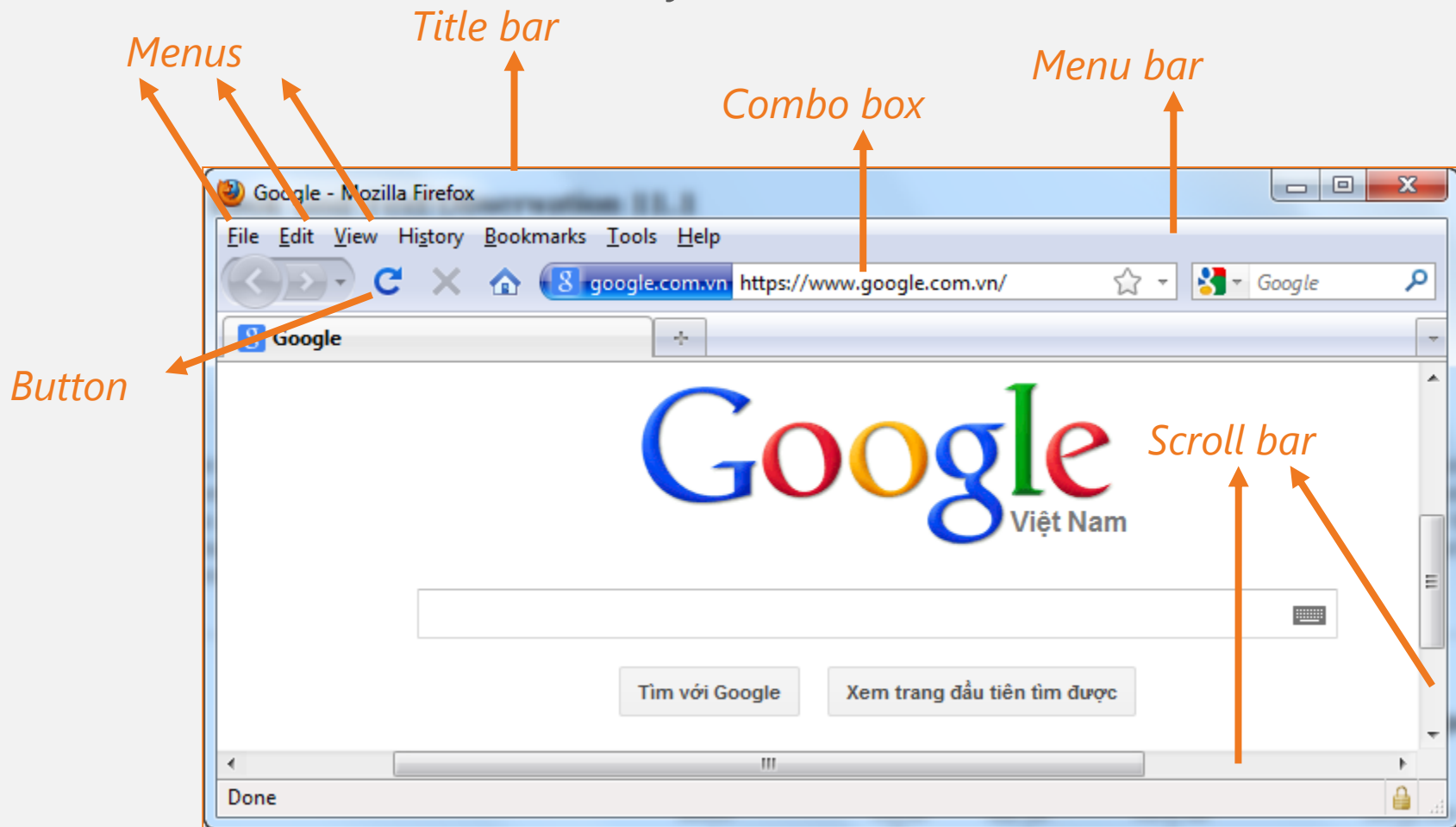


# Giao diện đồ họa người dùng

- Giao diện đồ họa người sử dụng (Graphical user interface – GUI)
- Giúp tạo ra các ứng dụng có giao diện đồ họa với nhiều các điều khiển như: Button, Textbox, Label, Checkbox, List, Tree...

# Ví dụ

- Giao diện trình duyệt web



# Lập trình GUI trong Java

- AWT
  - Được cung cấp trong Java 1.0
- Swing
  - Nâng cấp các thành phần giao diện của AWT
  - Được tích hợp trong Java 1.2
- JavaFX
  - Thư viện Java, phát triển ứng dụng đa nền tảng (Desktop, mobile, TV, tablet)
- Các thư viện khác:
  - Eclipse's Standard Widget Toolkit (SWT)
  - Google Web Toolkit (GWT)
  - 3D Graphics API: Java OpenGL (JOGL), Java3D.

# 2

## JavaFX

Các tính năng và hướng dẫn cài đặt



# Các tính năng của JavaFX

- Viết bằng Java, dùng được trong các ngôn ngữ thực thi trên máy ảo Java (Java, Groovy và JRuby)
- Hỗ trợ FXML (tương tự HTML), giúp dễ dàng định nghĩa giao diện người dùng
- Scene Builder: JavaFX cung cấp ứng dụng Scene Builder trên các nền tảng khác nhau, cho phép LTV kéo thả khi thiết kế giao diện
- Tương thích với Swing: trong ứng dụng JavaFX có thể nhúng các thành phần Swing
- Built-in UI controls: JavaFX cung cấp các control đa dạng để phát triển ứng dụng
- CSS like Styling: thiết kế giao diện với các tính năng giống như trong CSS
- ...



# Lịch sử JavaFX

- JavaFX được phát triển bởi Chris Oliver khi ông làm trong tập đoàn See Beyond Technology Corporation (Được Sun Microsystems mua lại vào 2005)
- 2007: Được giới thiệu chính thức ở hội nghị Java One
- 2008: Được tích hợp vào NetBean. JavaFX 1.0 được ban hành
- 2014: JavaFX được tích hợp vào Java SDK 8
- 2018: JavaFX được tách ra khỏi Java SDK 11

# Cài đặt JavaFX

- Trang chủ JavaFX: <https://openjfx.io/>
- Trang download thư viện JavaFX: <https://gluonhq.com/products/javafx/>
- Download, giải nén, copy các file trong thư mục lib, add vào build path của project
- Lưu ý khi chạy chương trình trên IDE Eclipse
  - Vào runtime configuration, cấu hình VM arguments:
    - + --module-path  
\${project\_classpath:REPLACE\_ME\_WITH\_YOUR\_PROJECT\_NAME} --add-modules javafx.controls,javafx.fxml
  - Bỏ chọn: "Use the -XstartOnFirstThread argument when launching with SWT"

# HelloWorld

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

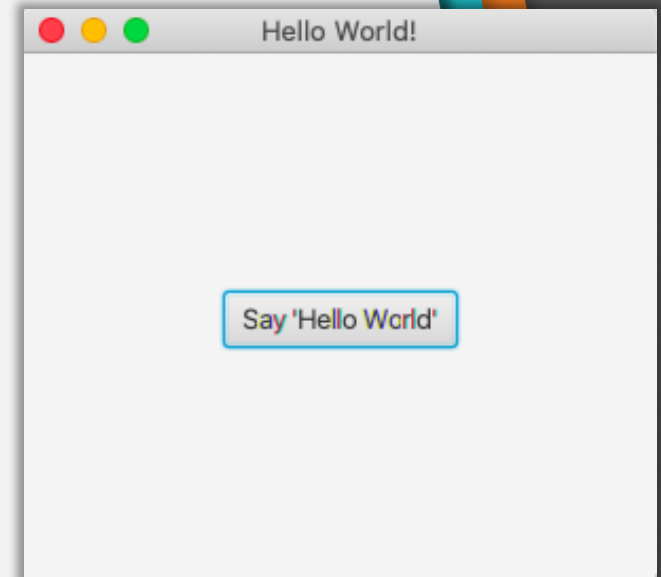
public class HelloWorld extends Application {
    @Override
    public void start(Stage primaryStage) {
        Button btn = new Button();
        btn.setText("Say 'Hello World'");
        btn.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                System.out.println("Hello World!");
            }
        });

        StackPane root = new StackPane();
        root.getChildren().add(btn);

        Scene scene = new Scene(root, 300, 250);

        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

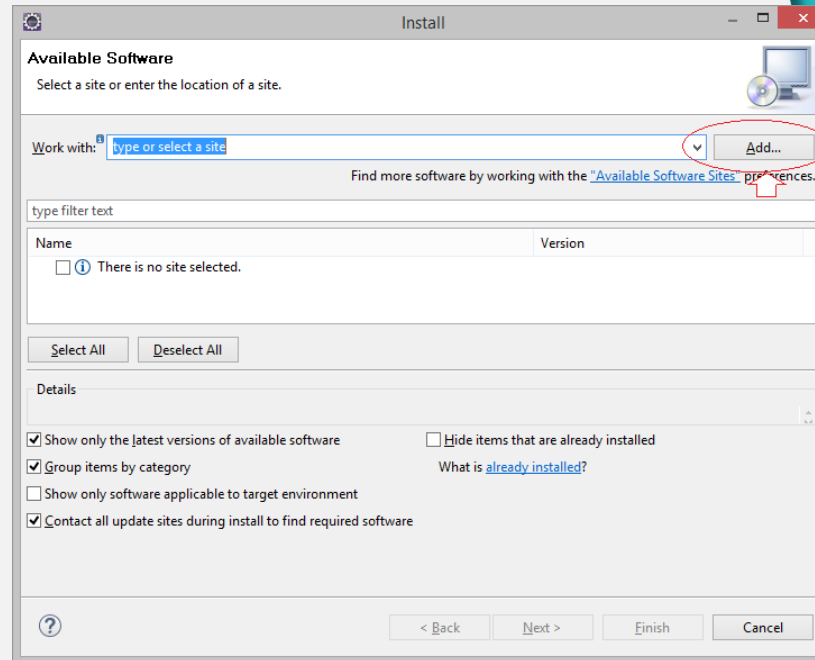
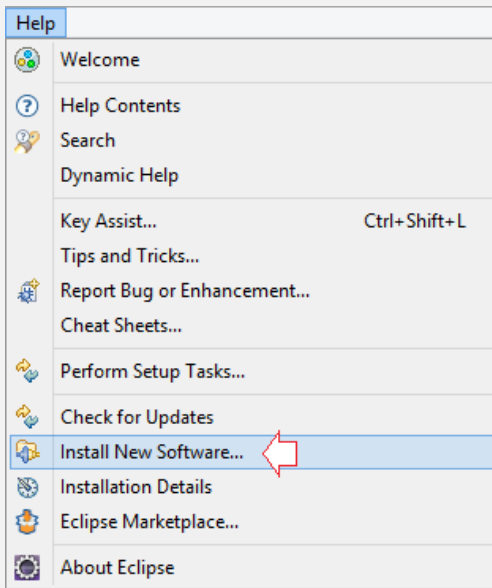
    public static void main(String[] args) {
        Launch(args);
    }
}
```



# Tiện ích JavaFX trên Eclipse

- e(fx)clipse
  - <https://www.eclipse.org/efxclipse/releases.html>
  - Là một Eclipse plugin
  - Công cụ hỗ trợ lập trình JavaFX trên Eclipse
- JavaFX Scene Builder
  - <https://www.oracle.com/java/technologies/javafxscenebuilder-1x-archive-downloads.html>
  - Công cụ độc lập, đa nền tảng, thiết kế trực quan giao diện cho ứng dụng **JavaFX**.
  - Cho phép kéo thả các thành phần giao diện người dùng, thay đổi thuộc tính, áp dụng style
  - Đầu ra: file FXML dùng trong ứng dụng JavaFX

# Cài đặt e(fx)clipse



Nhập vào:

<http://download.eclipse.org/efxclipse/updates-released/3.0.0/site>

Xem các Phiên bản mới nhất tại:

<https://www.eclipse.org/efxclipse/releases.html>

# Cài đặt e(fx)clipse

The screenshot shows the Eclipse IDE installation wizard in three overlapping windows. The top window is the 'Available Software' dialog, the middle is another 'Available Software' dialog, and the bottom is the 'Install Details' dialog. The 'Install Details' dialog shows a list of software components to be installed, with the 'Next >' button circled in red.

**Available Software**  
Select a site or enter the location of a site.

Work with: type or select a site

**Available Software**  
Check the items that you wish to install.

Work with: e(fx)clipse - http://download.eclipse.org/efxclipse/updates-released/3.0.0/site

**Install Details**  
Review the items to be installed.

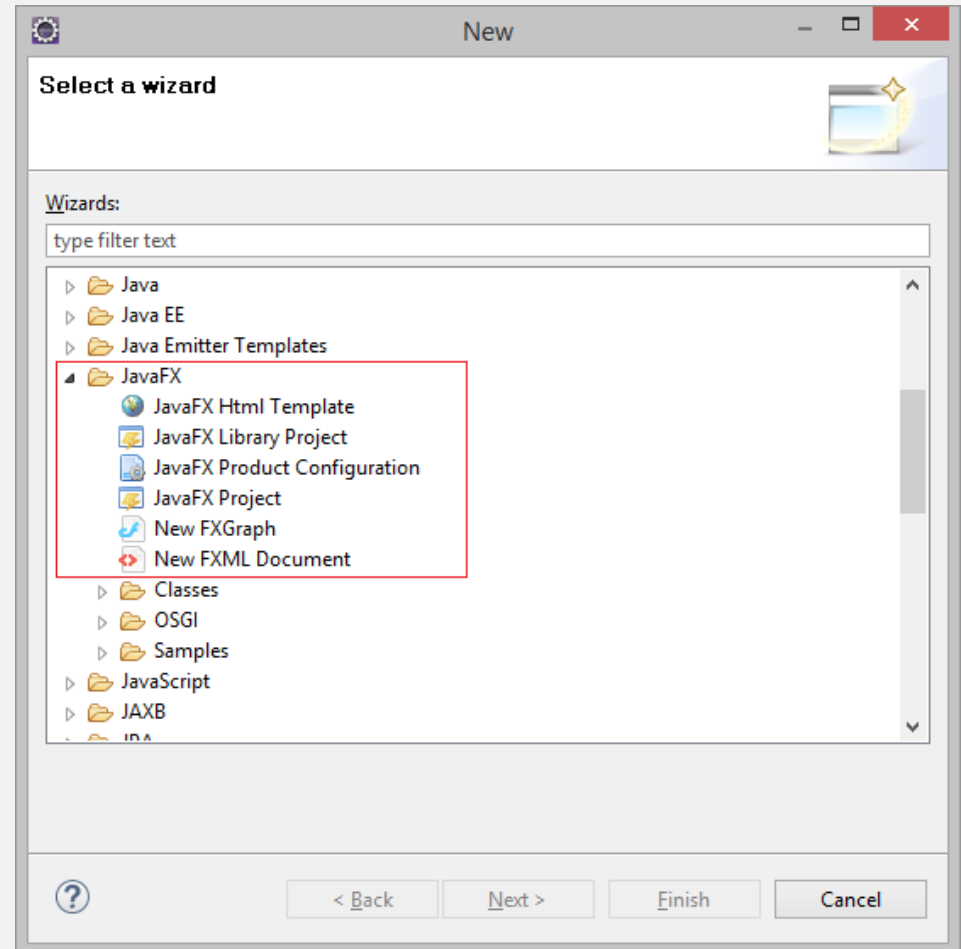
Name	Version	Id
▶ e(fx)clipse - IDE	3.0.0.201705220750	org.eclipse.fx.ide.feature.feature.gr...
▶ e(fx)clipse - IDE - Basic	3.0.0.201705220750	org.eclipse.fx.ide.basic.feature.featur...
▶ e(fx)clipse - IDE - Converter	3.0.0.201705220750	org.eclipse.fx.ide.converter.feature.featur...
▶ e(fx)clipse - IDE - CSS	3.0.0.201705220750	org.eclipse.fx.ide.css.feature.featur...
▶ e(fx)clipse - IDE - DSL to setup JavaFX bas...	3.0.0.201705220750	org.eclipse.fx.ide.edef.feature.featur...
▶ e(fx)clipse - IDE - FXGraph	3.0.0.201705220750	org.eclipse.fx.ide.fxgraph.feature.f...
▶ e(fx)clipse - IDE - FXML	3.0.0.201705220750	org.eclipse.fx.ide.fxml.feature.featur...
▶ e(fx)clipse - IDE - GModel Feature	3.0.0.201705220750	org.eclipse.fx.ide.gmod.feature.featur...
▶ e(fx)clipse - IDE - I10n support	3.0.0.201705220750	org.eclipse.fx.ide.i10n.feature.featur...
▶ e(fx)clipse - IDE - PDE	3.0.0.201705220750	org.eclipse.fx.ide.pde.feature.featur...
▶ e(fx)clipse - IDE - RRobot	3.0.0.201705220750	org.eclipse.fx.ide.robot.feature.featur...

Size: Unknown  
Details

< Back **Next >** Finish Cancel

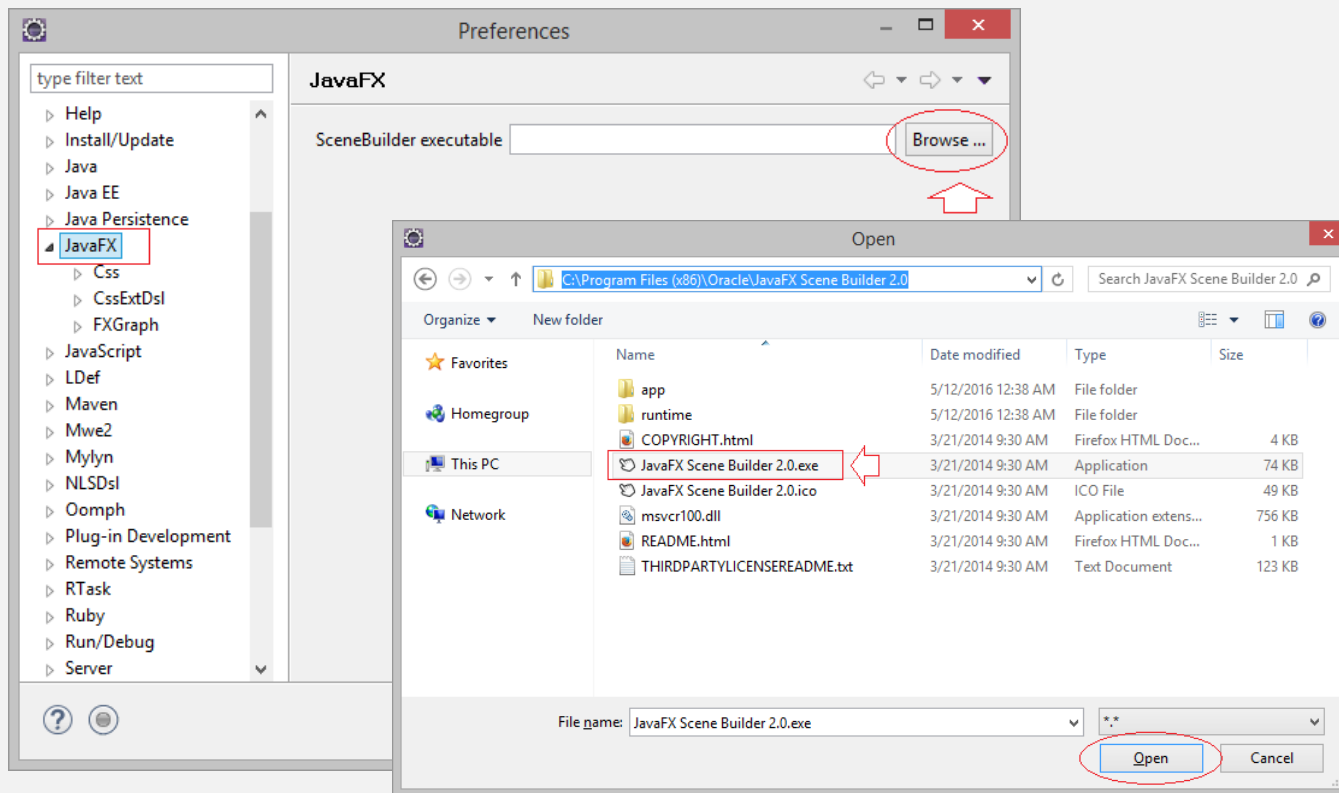
# Cài đặt e(fx)clipse

- Sau khi cài đặt và khởi động lại **Eclipse**, vào menu **File/New/Others ...** sẽ thấy các **Wizard** cho phép lập trình **JavaFX**



# Tích hợp JavaFX Scene Builder vào Eclipse

- Download, cài đặt JavaFX Scene Builder
- Trên eclipse, vào Window/Preferences

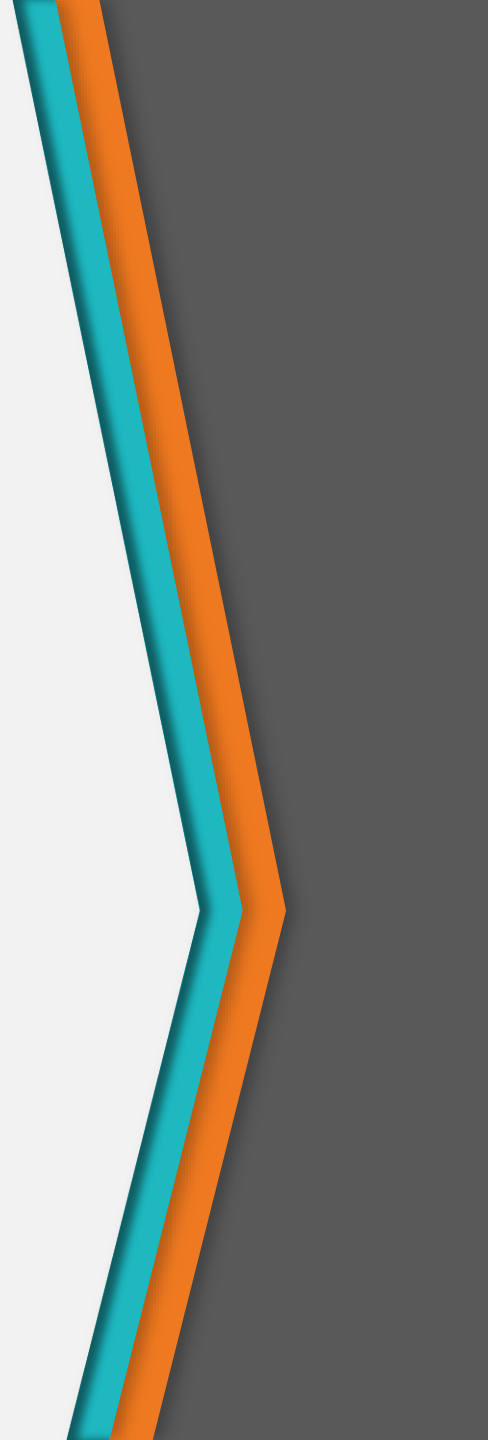




# 3

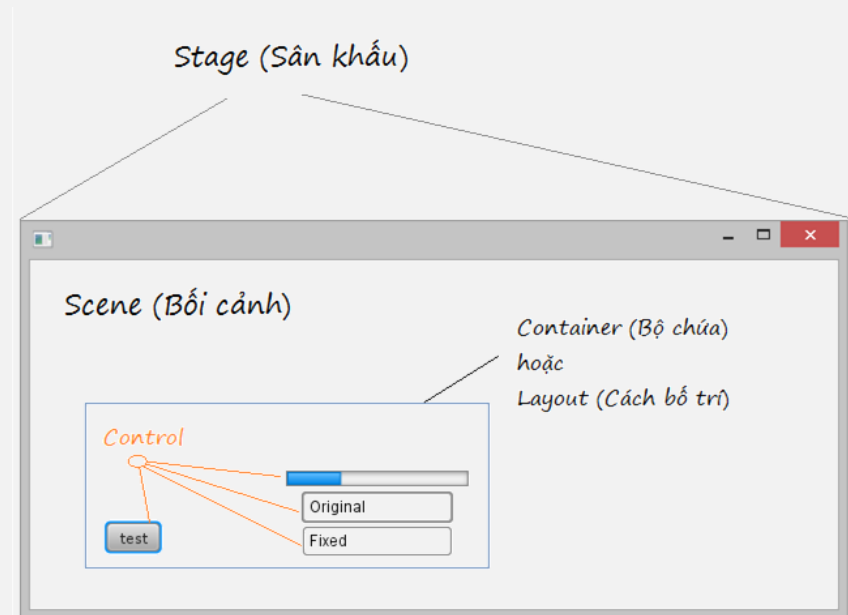
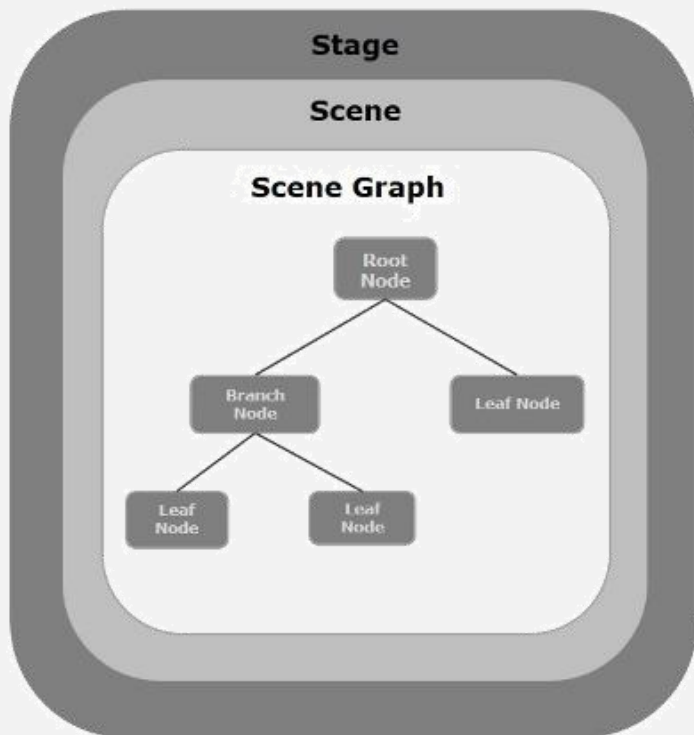
## Các thành phần của JavaFX

Stage, scene, node



# 3. Các thành phần giao diện JavaFX

- Cấu trúc ứng dụng JavaFX gồm 3 thành phần chính: Stage, Scene và Nodes



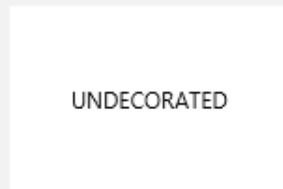
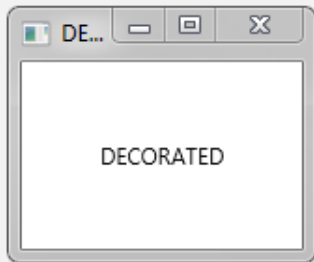
# Stage

- Đối tượng Stage (Window) chứa tất cả các đối tượng khác trong ứng dụng JavaFX
- Là đối tượng của lớp `javafx.stage.Stage`
- Đối tượng Stage sẽ truyền làm tham số cho phương thức `start()` của lớp `Application` (Xem lại ví dụ `HelloWorld JavaFX`)
- Có 2 tham số `width` và `height`
- Được chia làm 2 phần: `Content Area` và `Decorations` (`Title bar` và `Borders`)
- Để hiển thị Stage, gọi phương thức `show()`

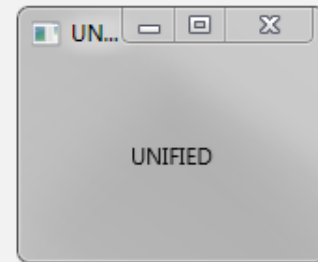
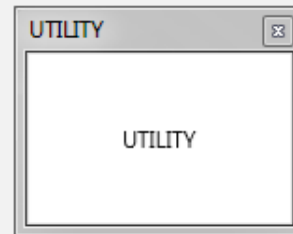
# Stage – thiết lập style

- Có 5 style cho Stage: Decorated, Undecorated, Transparent, Unified, Utility

```
stage.initStyle(StageStyle.DECORATED);  
stage.intStyle(StageStyle.UNDECORATED);  
stage.initStyle(StageStyle.TRANSPARENT);  
stage.initStyle(StageStyle.UNIFIED);  
stage.initStyle(StageStyle.UTILITY);
```



TRANSPARENT



# Scene

- Scene chứa tất cả các nội dung trình bày của một scene graph
- Là đối tượng của lớp `javafx.scene.Scene`
- Một Scene được thêm vào duy nhất một Stage
- Một số phương thức khởi dựng:
  - `Scene(Parent root)`
  - `Scene(Parent root, double width, double height)`
  - ...

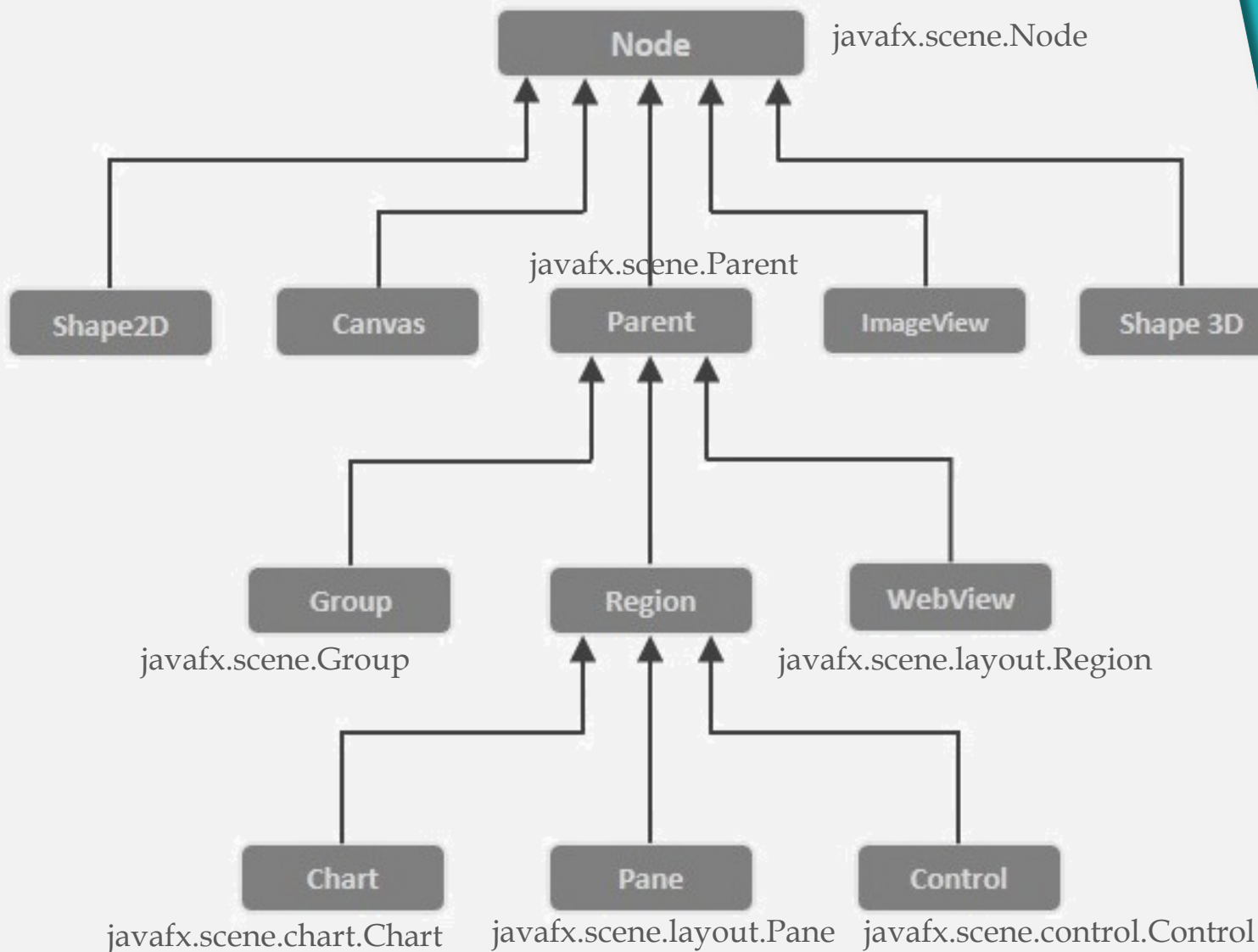
# Scene Graph và Nodes

- **Scene graph**: là cấu trúc dữ liệu phân cấp dạng tree biểu diễn nội dung một **Scene**, bao gồm tất cả các controls, layout
- Node: là một đối tượng đồ họa của một Scene graph, bao gồm
  - Đối tượng hình học (2D và 3D) như: Circle, Rectangle, Polygon, ...
  - Đối tượng điều khiển UI như: Button, Checkbox, TextArea, ...
  - Phần tử đa phương tiện Media như: Audio, Video, Image
- Lớp cơ sở cho tất cả các loại Node: `javafx.scene.Node`

# Scene Graph và Nodes

- Có 2 loại Node:
  - Branch Node/Parent Node: là các node có các node con, lớp cơ sở là lớp `javafx.scene.Parent` (lớp trừu tượng). Có 3 loại:
    - + **Group**: là một node tổng hợp, chứa một list các node con. Khi render node Group, tất cả các node con sẽ lần lượt được render. Các chuyển đổi hiệu ứng áp dụng cho một Group được áp dụng cho tất cả node con
    - + **Region**: là lớp cơ sở cho các UI Controls, bao gồm **Chart** (AreaChart, BarChart, BubbleChart, ...), **Pane** (AnchorPane, BorderPane, DialogPane, FlowPane, HBox, VBox ...), **Control** (Accordion, ButtonBar, ChoiceBox, ComboBoxBase, HTML editor, ...)
    - + **WebView**: tương tự như Browser
  - Leaf Node: là node không có node con. Ví dụ: Rectangle, Ellipse, Box, ImageView, MediaView
- Lưu ý: Root node là một branch/parent node, nhưng root node không có node cha.

# Cây phân cấp kế thừa Node





# Cách tạo ứng dụng JavaFX

- Viết lớp kế thừa lớp `javafx.application.Application`, thực thi phương thức trừu tượng `start`
- Trong phương thức `main`, gọi phương thức `static launch()`. Phương thức `launch` đã tự động gọi phương thức `start()`

```
public class JavafxSample extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        /*
         * Code for JavaFX application.
         * (Stage, scene, scene graph)
         */
    }

    public static void main(String args[]){
        launch(args);
    }
}
```

# Vòng đời ứng dụng JavaFX

- Có 3 phương thức trong vòng đời ứng dụng JavaFX: start(), stop(), init()
- Cài đặt mặc định là phương thức rỗng, có thể override khi muốn làm gì đó
- Thứ tự hành động
  - Tạo thể hiện của lớp application
  - Gọi phương thức init (không tạo stage hoặc scene trong phương thức này)
  - Gọi phương thức start
  - Khi ứng dụng kết thúc, gọi phương thức stop
- Khi cửa sổ (window) cuối cùng của ứng dụng JavaFX được đóng, ứng dụng tự động kết thúc. Có thể gọi tường minh với phương thức Platform.exit() hoặc System.exit(int)

# Cài đặt phương thức start

- 3 bước:
  - Tạo một Scene graph với các Node
  - Tạo một Scene với kích thước mong muốn và thêm vào root node của scene graph
  - Tạo một Stage, thêm Scene vào Stage, và hiển thị nội dung của Stage

# Tạo scene graph

- Cần tạo node gốc, có thể là Group, Region hoặc WebView

- VD: `Group root = new Group();`

- Thêm các node vào root node theo 2 cách

- Cách 1:

```
//Retrieving the observable list object  
ObservableList list = root.getChildren();
```

```
//Setting a node object as a node  
list.add(NodeObject);
```

- Cách 2:

```
Group root = new Group(NodeObject);
```

# Tạo Scene

- Khởi tạo đối tượng Scene, bắt buộc phải truyền tham số là root object

```
Scene scene = new Scene(root);
```

- Có thể vừa khởi tạo vừa thiết lập kích thước của Scene

```
Scene scene = new Scene(root, 600, 300);
```

# Tạo Stage

- Đối tượng Stage được truyền làm tham số cho phương thức start() của lớp Application → không cần khởi tạo

- Thao tác cơ bản

```
//Setting the title to Stage.  
primaryStage.setTitle("Sample application");
```

```
//Setting the scene to Stage  
primaryStage.setScene(scene);
```

```
//Displaying the stage  
primaryStage.show();
```

# Ví dụ: Tạo ứng dụng với cửa sổ JavaFX rỗng

```
public class JavafxSample extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        //creating a Group object
        Group group = new Group();

        //Creating a Scene
        Scene scene = new Scene(group ,600, 300);

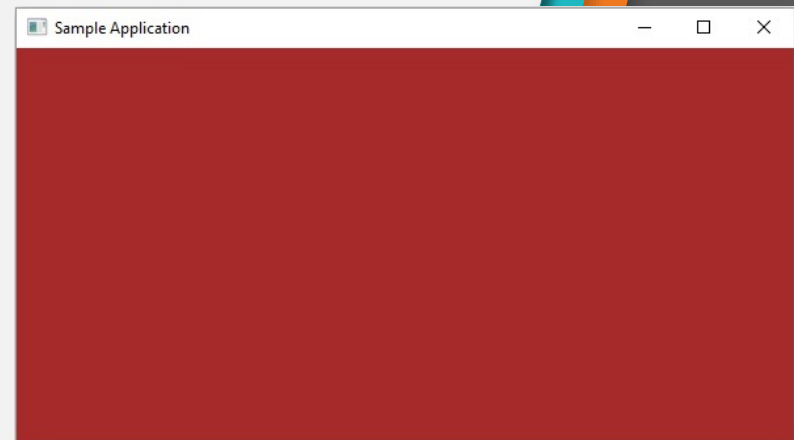
        //setting color to the scene
        scene.setFill(Color.BROWN);

        //Setting the title to Stage.
        primaryStage.setTitle("Sample Application");

        //Adding the scene to Stage
        primaryStage.setScene(scene);

        //Displaying the contents of the stage
        primaryStage.show();
    }

    public static void main(String args[]) {
        launch(args);
    }
}
```



# VD: Vẽ đường thẳng

```
public class DrawingLine extends Application{
    @Override
    public void start(Stage stage) {
        //Creating a line object
        Line line = new Line();

        //Setting the properties to a line
        line.setStartX(100.0);
        line.setStartY(150.0);
        line.setEndX(500.0);
        line.setEndY(150.0);

        //Creating a Group
        Group root = new Group(line);

        //Creating a Scene
        Scene scene = new Scene(root, 600, 300);

        //Setting title to the scene
        stage.setTitle("Sample application");

        //Adding the scene to the stage
        stage.setScene(scene);

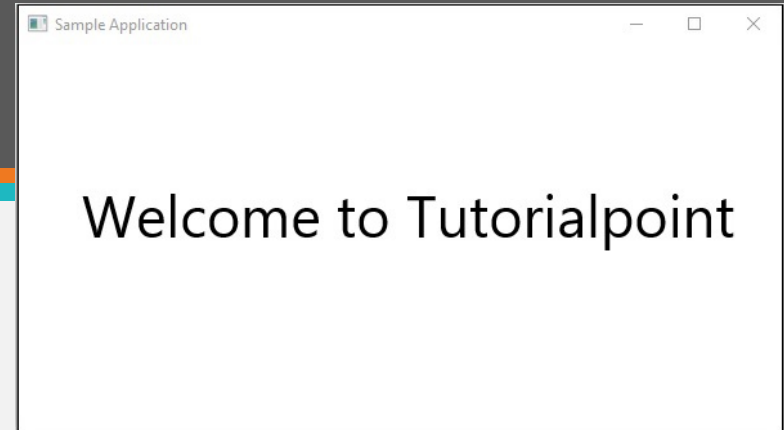
        //Displaying the contents of a scene
        stage.show();
    }
    public static void main(String args[]){
        launch(args);
    }
}
```





# VD: Hiển thị chữ

```
public class DisplayingText extends Application {  
    @Override  
    public void start(Stage stage) {  
        //Creating a Text object  
        Text text = new Text();  
  
        //Setting font to the text  
        text.setFont(new Font(45));  
  
        //setting the position of the text  
        text.setX(50);  
        text.setY(150);  
  
        //Setting the text to be added.  
        text.setText("Welcome to Tutorialpoint");  
  
        //Creating a Group object  
        Group root = new Group();  
  
        //Retrieving the observable list object  
        ObservableList list = root.getChildren();  
  
        //Setting the text object as a node to the group object  
        list.add(text);  
    }  
}
```



```
        //Creating a scene object  
        Scene scene = new Scene(root, 600, 300);  
  
        //Setting title to the Stage  
        stage.setTitle("Sample Application");  
  
        //Adding scene to the stage  
        stage.setScene(scene);  
  
        //Displaying the contents of the stage  
        stage.show();  
    }  
    public static void main(String args[]){  
        launch(args);  
    }  
}
```

# Ví dụ: Hiển thị 2 dòng text

```
public class DecorationsExample extends Application {
    @Override
    public void start(Stage stage) {
        //Creating a Text_Example object
        Text text1 = new Text("Hi how are you");

        //Setting font to the text
        text1.setFont(
            Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 20)
        );

        //setting the position of the text
        text1.setX(50);
        text1.setY(75);

        //Striking through the text
        text1.setStrikethrough(true);

        //Creating a Text_Example object
        Text text2 = new Text("Welcome to Tutorialspoint");

        //Setting font to the text
        text2.setFont(
            Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 20)
        );
    }
}
```

# Ví dụ: Hiển thị 2 dòng text

```
//setting the position of the text
text2.setX(50);
text2.setY(150);

//underlining the text
text2.setUnderline(true);

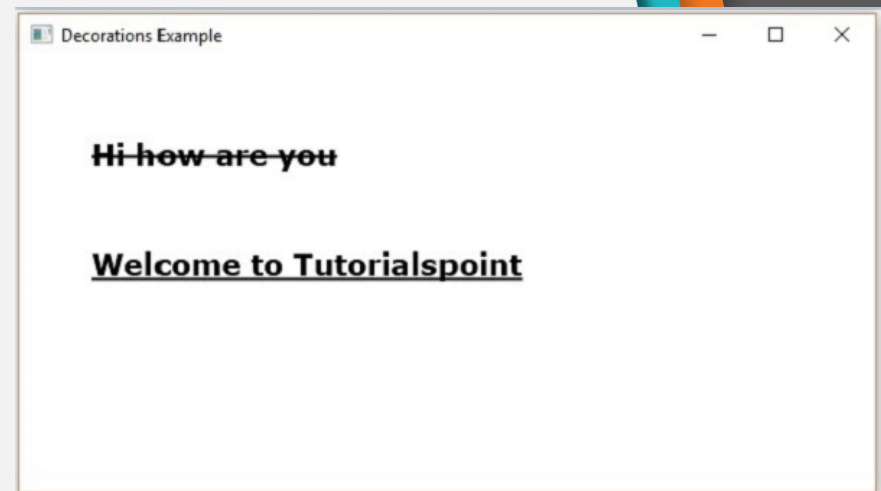
//Creating a Group object
Group root = new Group(text1, text2);

//Creating a scene object
Scene scene = new Scene(root, 600, 300);

//Setting title to the Stage
stage.setTitle("Decorations Example");

//Adding scene to the stage
stage.setScene(scene);

//Displaying the contents of the stage
stage.show();
}
public static void main(String args[]){
    launch(args);
}
}
```



# 4

## JavaFX UI Control

Các điều khiển giao diện



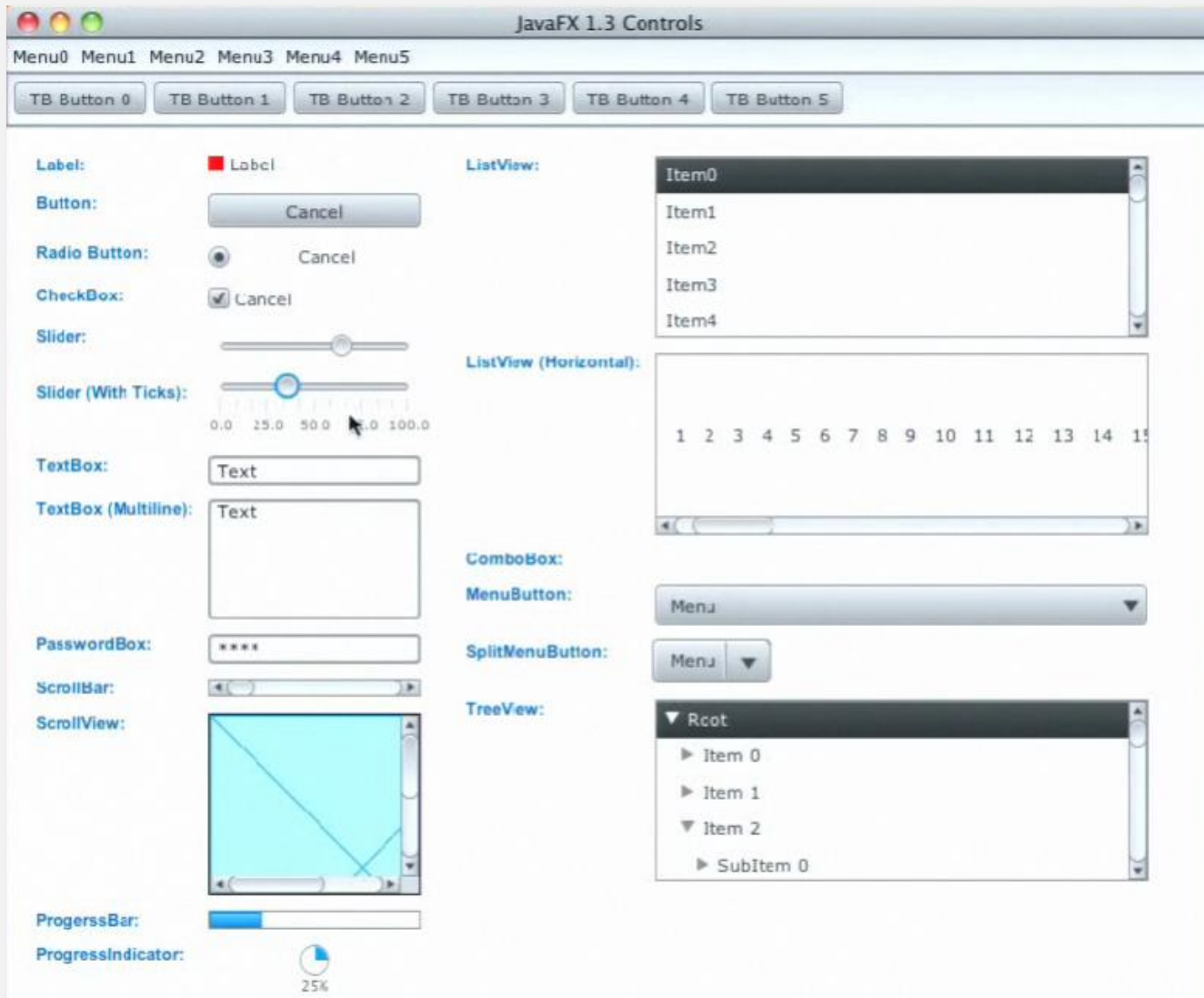
# Java FX - UI Controls

- Một giao diện người dùng gồm 3 thành phần chính
  - **UI elements** – Là các phần tử người dùng thấy sau cùng và trực tiếp tương tác với (Button, Label, Checkbox, ...)
  - **Layouts** – Định nghĩa cách thức sắp xếp các UI elements trên màn hình
  - **Behavior** – Các sự kiện xảy ra khi người dùng tương tác với các UI elements (Event Handling)

# Java FX - UI Controls

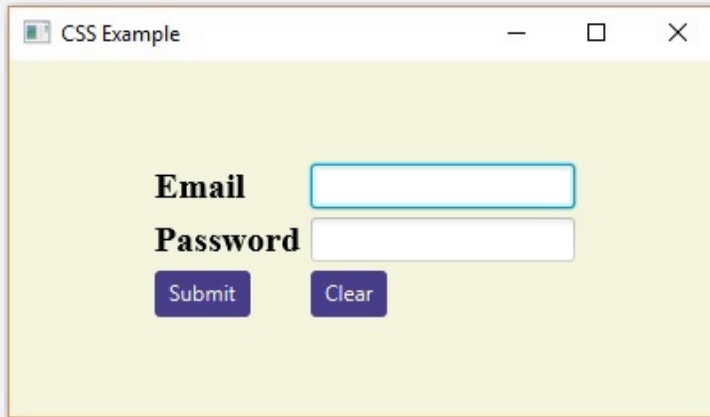


# 4. Java FX - UI Controls



# Ví dụ

- Viết ứng dụng với giao diện như sau



The image shows a browser window titled "CSS Example" with a light yellow background. It contains a form with two input fields: "Email" and "Password". Below the fields are two buttons: "Submit" and "Clear".

**Email**

**Password**



# Ví dụ

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.geometry.Insets;
import javafx.geometry.Pos;

import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.layout.GridPane;
import javafx.scene.text.Text;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

public class LoginPage extends Application {
    @Override
    public void start(Stage stage) {
        //creating label email
        Text text1 = new Text("Email");

        //creating label password
        Text text2 = new Text("Password");

        //Creating Text Filed for email
        TextField textField1 = new TextField();

        //Creating Text Filed for password
        PasswordField textField2 = new PasswordField();

        //Creating Buttons
        Button button1 = new Button("Submit");
        Button button2 = new Button("Clear");
    }
}
```

# Ví dụ

```
//Creating a Grid Pane
GridPane gridPane = new GridPane();

//Setting size for the pane
gridPane.setMinSize(400, 200);

//Setting the padding
gridPane.setPadding(new Insets(10, 10, 10, 10));

//Setting the vertical and horizontal gaps between the columns
gridPane.setVgap(5);
gridPane.setHgap(5);

//Setting the Grid alignment
gridPane.setAlignment(Pos.CENTER);

//Arranging all the nodes in the grid
gridPane.add(text1, 0, 0);
gridPane.add(textField1, 1, 0);
gridPane.add(text2, 0, 1);
gridPane.add(textField2, 1, 1);
gridPane.add(button1, 0, 2);
gridPane.add(button2, 1, 2);
```

# Ví dụ

```
//Styling nodes
button1.setStyle("-fx-background-color: darkslateblue; -fx-text-fill: white;");
button2.setStyle("-fx-background-color: darkslateblue; -fx-text-fill: white;");

text1.setStyle("-fx-font: normal bold 20px 'serif' ");
text2.setStyle("-fx-font: normal bold 20px 'serif' ");
gridPane.setStyle("-fx-background-color: BEIGE;");

//Creating a scene object
Scene scene = new Scene(gridPane);

//Setting title to the Stage
stage.setTitle("CSS Example");

//Adding scene to the stage
stage.setScene(scene);

//Displaying the contents of the stage
stage.show();
}
public static void main(String args[]){
    launch(args);
}
}
```

# 5

## JavaFX Layout Pane



# JavaFX - Layout Panes (Container)

- Sau khi tạo các node trong 1 scene, cần sắp xếp trình bày các node
- Layout của 1 container: là cách sắp xếp các node nằm trong container đó
- Các loại layout trong JavaFX: HBox, VBox, Border Pane, Stack Pane, Text Flow, Anchor Pane, Title Pane, Grid Pane, Flow Panel, ...
- Mỗi loại layout ứng với 1 class, tất cả các class này nằm trong gói `javafx.layout`, lớp Pane là lớp cơ sở của tất cả các lớp layout

# Các bước tạo Layout

- Tạo các nodes
- Khởi tạo đối tượng của lớp layout mong muốn
- Thiết lập các thuộc tính cho layout
- Thêm tất cả các nodes đã tạo vào trong layout

# Ví dụ với layout HBox

- Đặc điểm: các node xếp theo hàng ngang
- Một số thuộc tính quan trọng:
  - alignment: giống hàng các node
  - spacing: khoảng cách giữa các node
- Khởi tạo HBox

```
// Khởi tạo rỗng  
HBox hbox = new HBox();
```

```
// Khởi tạo với các node  
Button button1 = new Button("Button Number 1");  
Button button2 = new Button("Button Number 2");  
HBox hbox = new HBox(button1, button2);
```

# Ví dụ với layout HBox

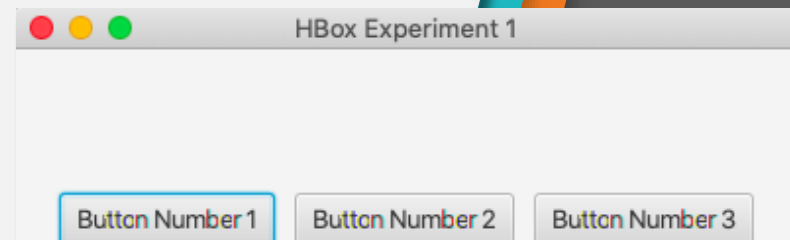
```
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
public class HBoxExperiments extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("HBox Experiment 1");

        Button button1 = new Button("Button Number 1");
        Button button2 = new Button("Button Number 2");
        Button button3 = new Button("Button Number 3");

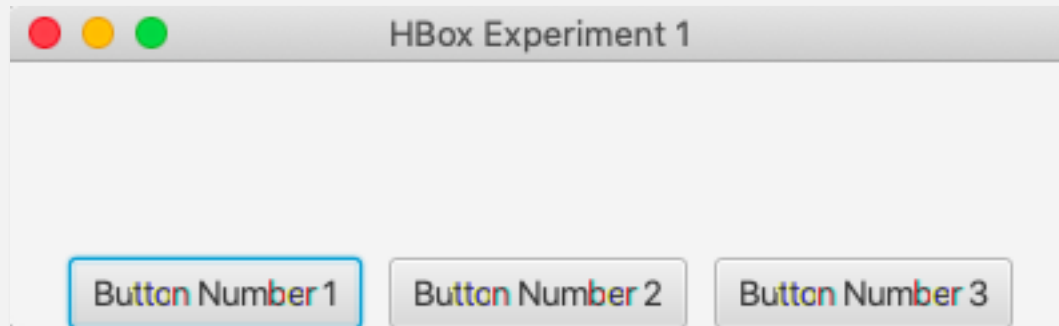
        HBox hbox = new HBox(button1, button2);
        hbox.setSpacing(10);
        hbox.setAlignment(Pos.BOTTOM_CENTER);
        hbox.getChildren().add(button3);

        Scene scene = new Scene(hbox, 400, 100);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

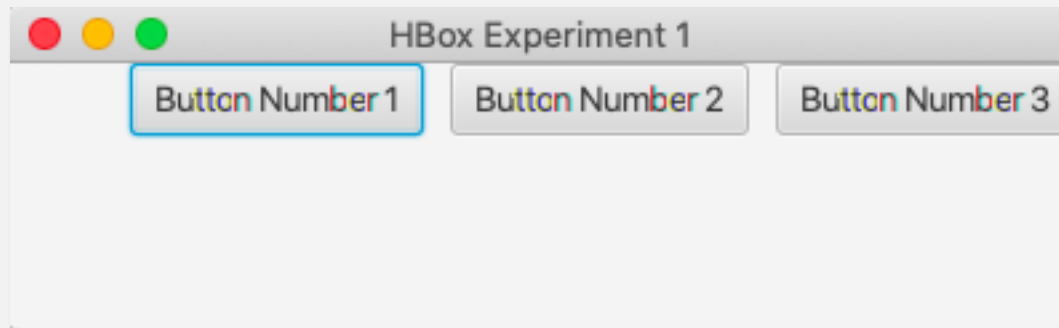




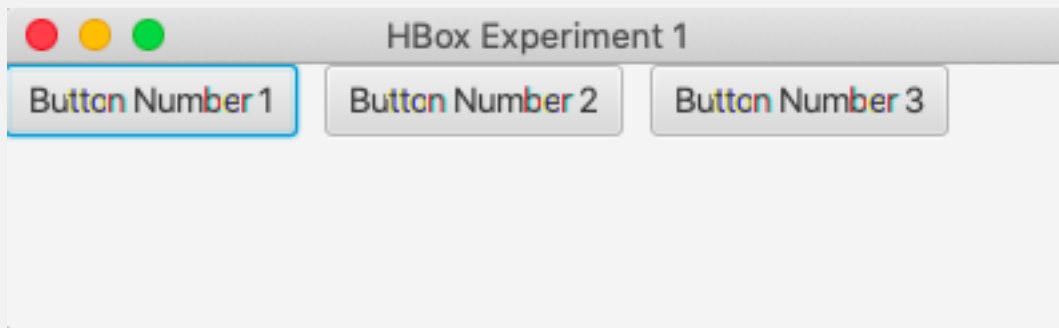
# Ví dụ với layout HBox



***BOTTOM\_CENTER***



***BASELINE\_RIGHT***



***TOP\_LEFT***

# Ví dụ với layout Group

- Group: không sắp xếp các component trong nó, tất cả đều ở tọa độ (0, 0)

```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;
public class GroupExperiments extends Application {

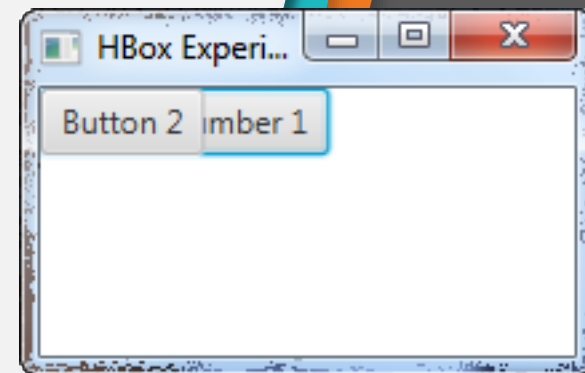
    @Override
    public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("HBox Experiment 1");

        Button button1 = new Button("Button Number 1");
        Button button2 = new Button("Button 2");

        Group group = new Group();
        group.getChildren().add(button1);
        group.getChildren().add(button2);

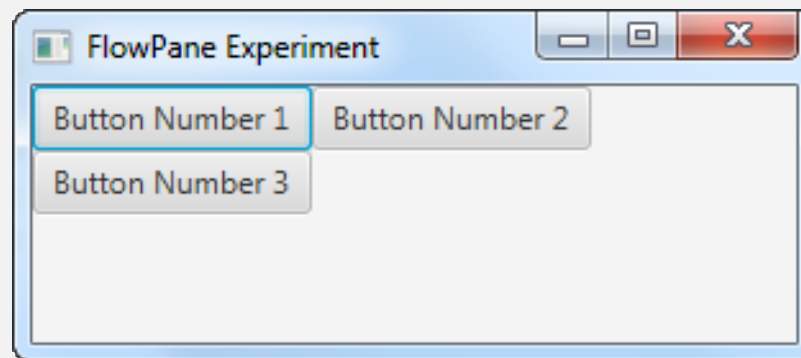
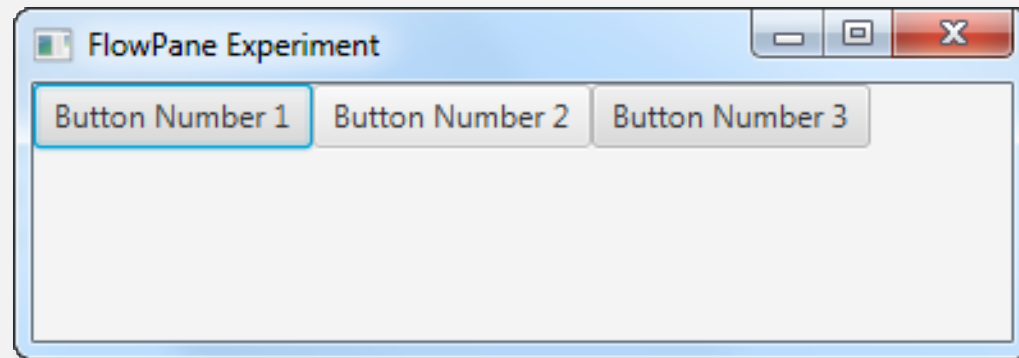
        Scene scene = new Scene(group, 200, 100);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

2 button đều ở tọa độ  
(0, 0), đè lên nhau



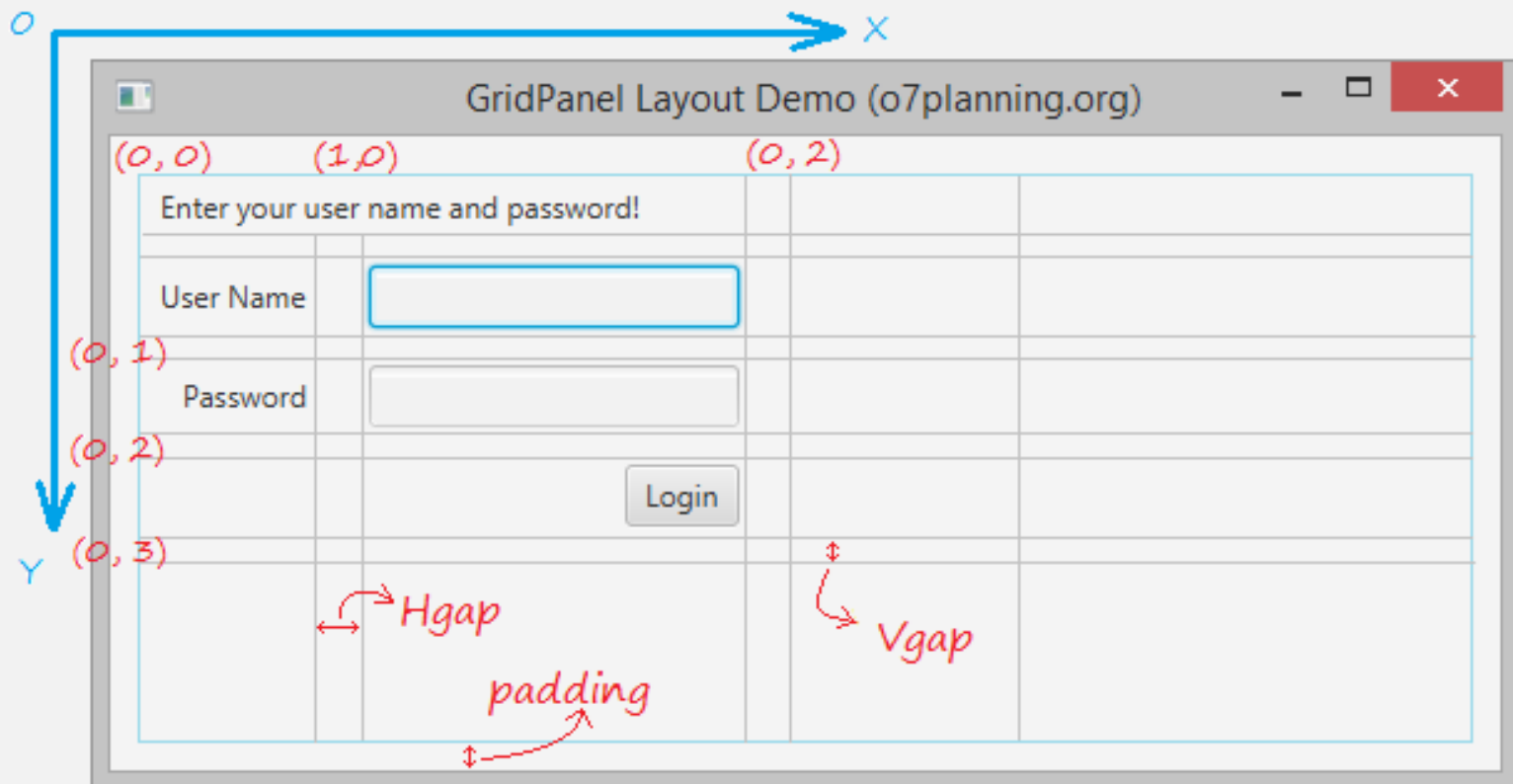
# Một số layout khác

- FlowPane: sắp xếp các thành phần con liên tiếp nhau trên một dòng, và tự động đẩy phần tử con xuống dòng tiếp theo nếu dòng hiện tại không còn chỗ trống



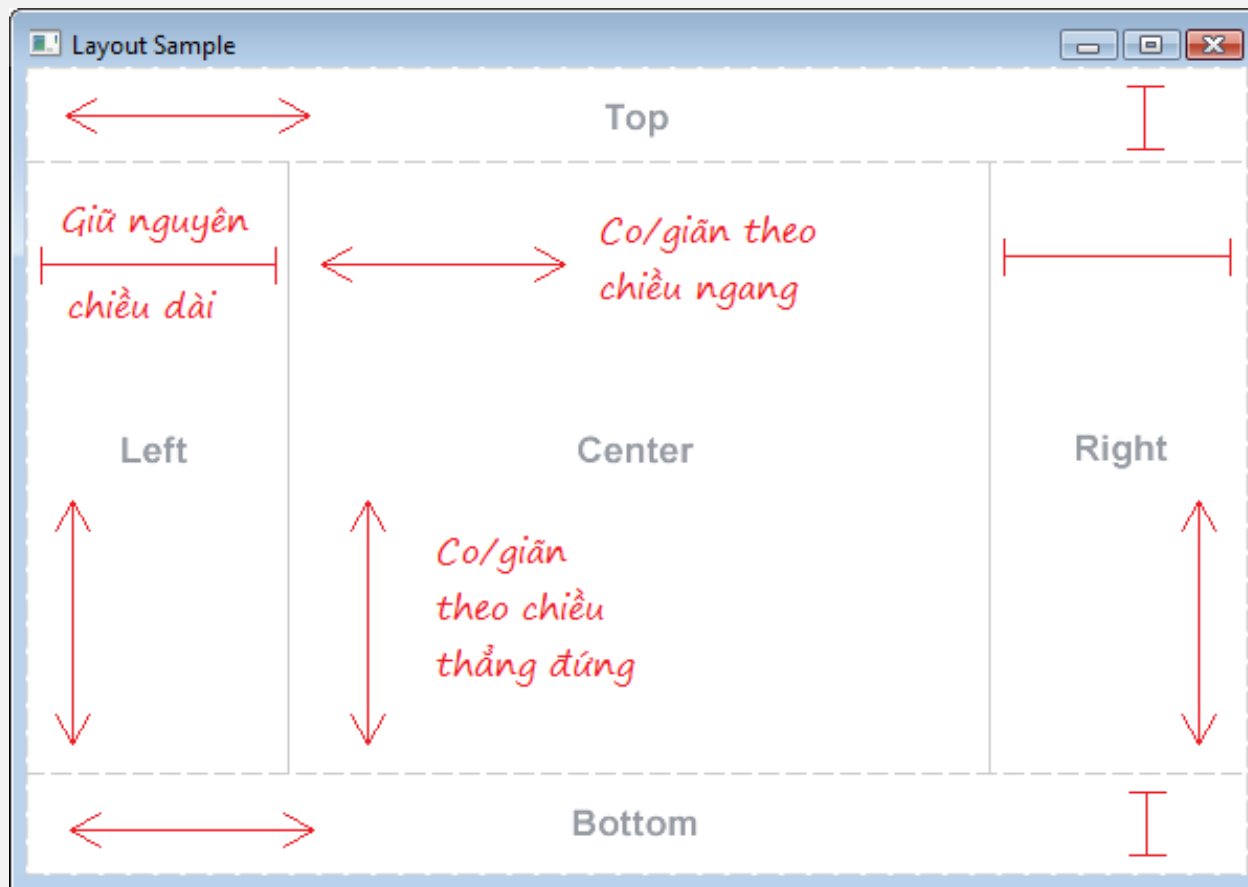
# Một số layout khác

- GridPane: chia thành lưới gồm các hàng và các cột. Một thành phần con có thể nằm trên một ô lưới hoặc nằm trên một ô hợp nhất từ các ô gần nhau



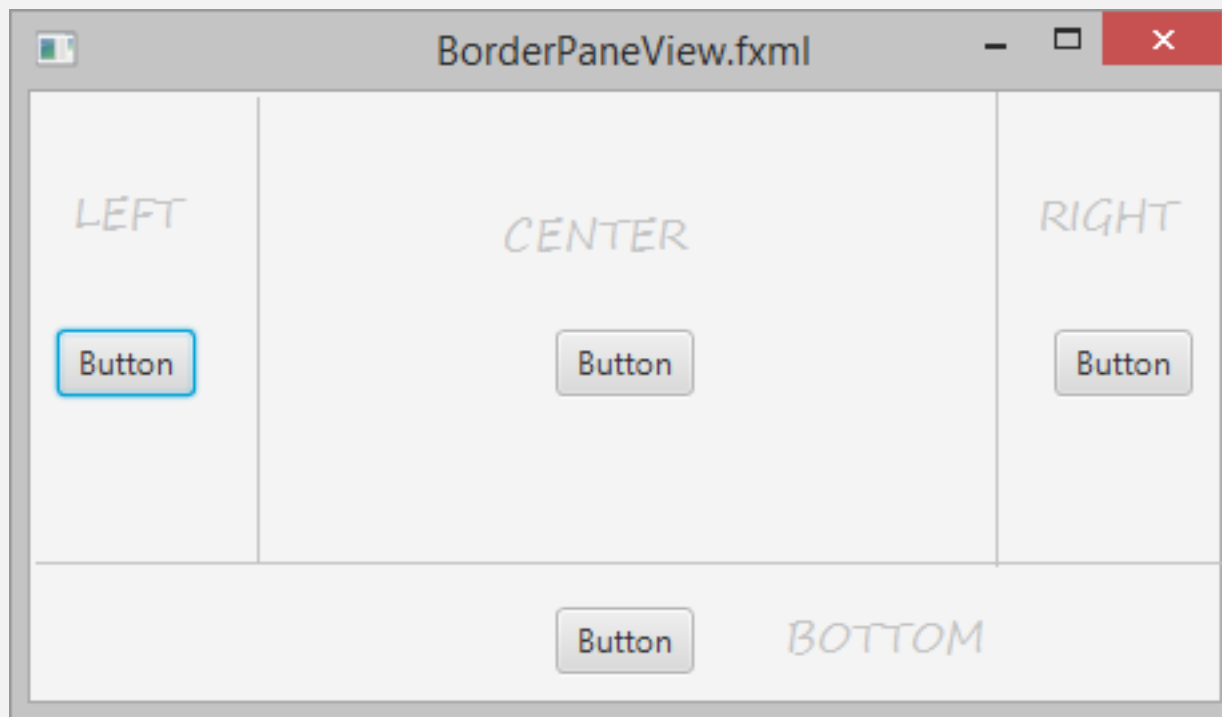
# Một số layout khác

- BorderLayout: chia thành 5 vùng riêng biệt, mỗi vùng có thể chứa được một thành phần con.



# Một số layout khác

- **BorderPane:** Nếu một vùng nào đó không chứa thành phần con, các vùng khác sẽ chiếm lấy không gian của nó.
- Ví dụ: Vùng TOP không có thành phần con, không gian của nó sẽ bị các thành phần khác chiếm chỗ:



# 6

## Mô hình xử lý sự kiện

Event Handler



# Mô hình xử lý sự kiện

- Các sự kiện được chia làm hai loại:
  - **Foreground Events** – Là sự kiện cần người dùng tương tác trực tiếp. VD: nhấn chuột vào button, di chuyển chuột, gõ ký tự, chọn 1 item trong list, cuộn trang, ...
  - **Background Events** – VD: can thiệp của hệ điều hành, lỗi phần mềm/phần cứng, hết giờ, hoàn thiện 1 thao tác gì đó, ...



# Mô hình xử lý sự kiện

- Lớp cơ sở cho các loại sự kiện: `javafx.event.Event`
- JavaFX hỗ trợ xử lý nhiều loại sự kiện
  - **Mouse Event** – Sự kiện xảy ra khi nhấn chuột (mouse clicked, mouse pressed, mouse released, mouse moved, mouse entered target, mouse exited target). Lớp tương ứng là **MouseEvent**.
  - **Key Event** – Sự kiện xảy ra khi nhấn phím (key pressed, key released and key typed). Lớp tương ứng là **KeyEvent**.
  - **Drag Event** – Sự kiện xảy ra khi rê chuột (drag entered, drag dropped, drag entered target, drag exited target, drag over). Lớp tương ứng là **DragEvent**.
  - **Window Event** – Sự kiện xảy ra khi hiện/ẩn cửa sổ (window hiding, window shown, window hidden, window showing). Lớp tương ứng là **WindowEvent**.

# Mô hình xử lý sự kiện

- Xử lý sự kiện (Event Handling): cài đặt code - sẽ được thực thi khi một sự kiện xác định nào đó xảy ra
- JavaFX cung cấp các handlers và các filters để xử lý sự kiện. Mỗi sự kiện sẽ có 3 thuộc tính:
  - **Event Target** – Node xảy ra sự kiện. Target có thể là stage, scene, hoặc một node
  - **Event Source** – Là đối tượng có trạng thái thay đổi, nó sinh ra sự kiện. Ví dụ: chuột, bàn phím, ...
  - **Event Type** – Kiểu của sự kiện. Ví dụ, với nguồn sự kiện là chuột, kiểu của sự kiện có thể là mouse pressed, mouse released
- Khi sự kiện xảy ra, event source tạo một đối tượng event và chuyển đối tượng này đến bộ xử lý sự kiện

# Ví dụ

- Với ứng dụng JavaFX như trong hình, nếu nhấp chuột vào nút play, source sẽ là chuột, target là nút play, kiểu của sự kiện sinh ra là mouse click



# Các bước xử lý sự kiện trong JavaFX

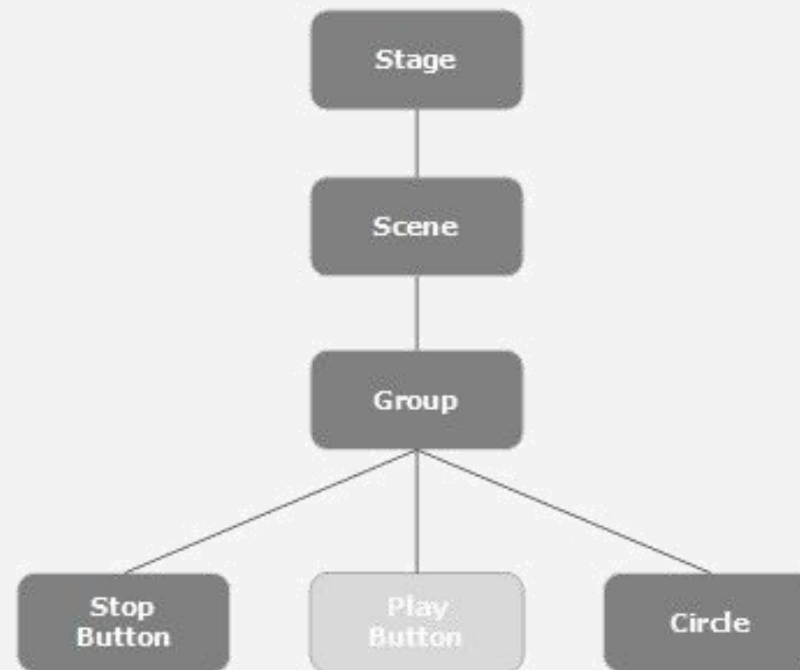
- 4 bước:
  1. Target selection
  2. Route construction
  3. Event capturing
  4. Event bubbling

# Các bước xử lý sự kiện trong JavaFX

- Bước 1: Target selection (xác định **target node**). Khi một hành động xảy ra, hệ thống xác định **target node** theo các luật sau:
  - Với sự kiện nhấn phím, **target node** là node đang được focus
  - Với sự kiện nhấn chuột, **target node** là node ứng với vị trí hiện tại của chuột
  - ... (một số sự kiện khác trên thiết bị cảm ứng)

# Các bước xử lý sự kiện trong JavaFX

- Bước 2: Route Construction – Tạo chuỗi sự kiện phát sinh (Event Dispatch chain): là đường đi từ stage tới target node



# Các bước xử lý sự kiện trong JavaFX

- Bước 3: Event Capturing (bắt sự kiện)
  - Sau khi tạo chuỗi sự kiện, root node của ứng dụng sẽ gửi đi sự kiện (dispatch event).
  - Sự kiện này sẽ đi dọc theo các node từ trên xuống dưới (top to bottom). Nếu một node nào đó đăng ký một **filter** cho sự kiện sinh ra, **filter** đó sẽ được thực thi.
  - Nếu một filter nào đó **consume** event bằng cách gọi phương thức consume() từ đối tượng event tạo ra, quá trình xử lý sự kiện lập tức kết thúc
  - Nếu event chưa được consume, cuối cùng sự kiện sẽ được chuyển tới cho **target node**

# Các bước xử lý sự kiện trong JavaFX

- Bước 4: Nổi bọt sự kiện (Event Bubbling)
  - Sự kiện sẽ đi ngược lên trên, từ **target node** tới **root node** (bottom to top).
  - Nếu bất kỳ một node nào đó trong **event dispatch chain** đăng ký một **handler** cho sự kiện sinh ra, **handler** sẽ được thực thi.
  - Nếu không handler nào consume event, sự kiện sẽ chuyển tới root node, và hoàn thành việc xử lý



# Các bước xử lý sự kiện trong JavaFX

- Event Handlers và Event Filters: chứa logic ứng dụng để xử lý một sự kiện
- Một node có thể đăng ký nhiều handler/filter.
- filter/handler cho parent node có thể được cài đặt như xử lý mặc định cho tất cả các node con của nó
- Tất cả các handlers và filters đều thực thi giao diện **`javafx.event.EventHandler`**

# Thêm/bỏ filter

- Thêm filter

```
//Creating the mouse event handler
EventHandler<MouseEvent> eventHandler = new EventHandler<MouseEvent>() {

    @Override
    public void handle(MouseEvent e) {
        System.out.println("Hello World");
        circle.setFill(Color.DARKSLATEBLUE);
    }
};

//Adding event Filter
Circle.addEventFilter(MouseEvent.MOUSE_CLICKED, eventHandler);
```

- Bỏ filter

```
circle.removeEventFilter(MouseEvent.MOUSE_CLICKED, eventHandler);
```

```
import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.FlowPane;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;

public class EventFiltersExample extends Application {

    @Override
    public void start(Stage stage) {
        Button button = new Button("Button");
        TextArea text = new TextArea();
        Circle circle = new Circle(25.0f);

        FlowPane fp = new FlowPane(button, text, circle);

        fp.addEventFilter(MouseEvent.MOUSE_CLICKED,
            new EventHandler<MouseEvent>() {
                @Override
                public void handle(MouseEvent arg0) {
                    text.appendText("Filter in flow pane\n");
                }
            });
        fp.addEventHandler(MouseEvent.MOUSE_CLICKED,
            new EventHandler<MouseEvent>() {
                @Override
                public void handle(MouseEvent arg0) {
                    text.appendText("Handler in flow pane\n");
                }
            });
    }
}
```

```
button.addEventFilter(MouseEvent.MOUSE_CLICKED,
                        new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent arg0) {
        text.appendText("Filter in button\n");
    }
});
button.addEventHandler(MouseEvent.MOUSE_CLICKED,
                        new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent arg0) {
        text.appendText("Handler in button\n");
    }
});

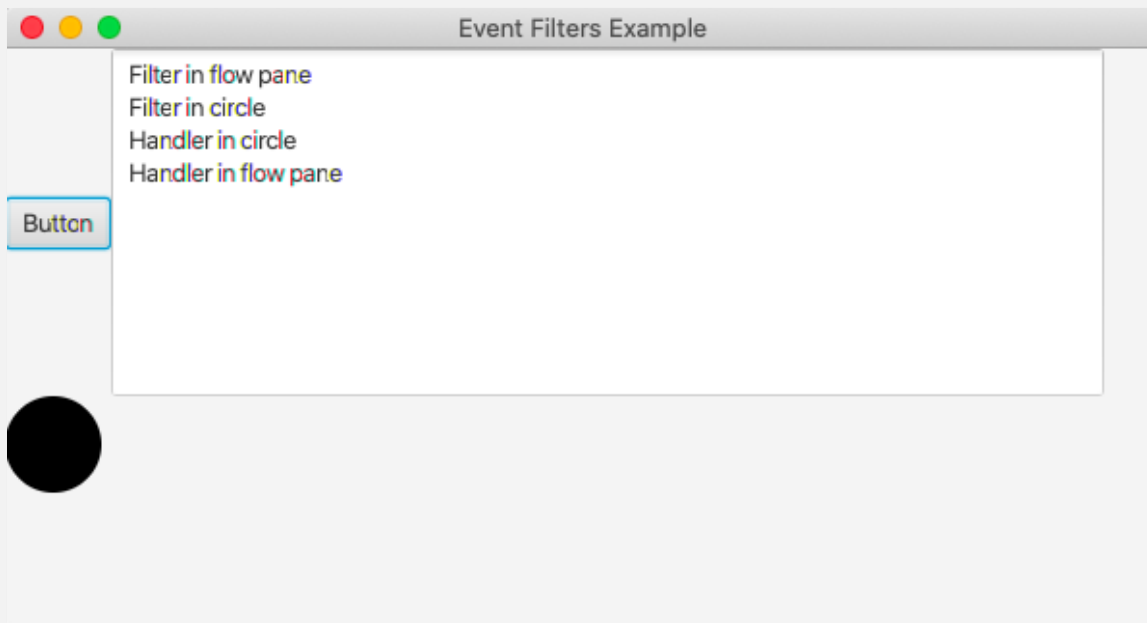
circle.addEventFilter(MouseEvent.MOUSE_CLICKED,
                       new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent arg0) {
        text.appendText("Filter in circle\n");
    }
});
circle.addEventHandler(MouseEvent.MOUSE_CLICKED,
                       new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent arg0) {
        text.appendText("Handler in circle\n");
    }
});

// Creating a scene object
Scene scene = new Scene(fp, 600, 300);
stage.setTitle("Event Filters Example");
stage.setScene(scene);
stage.show();
}
```

```
import javafx.application.Application;

public final class Main {
    public static void main(final String[] args) {
        Application.Launch(EventFiltersExample.class, args);
    }
}
```

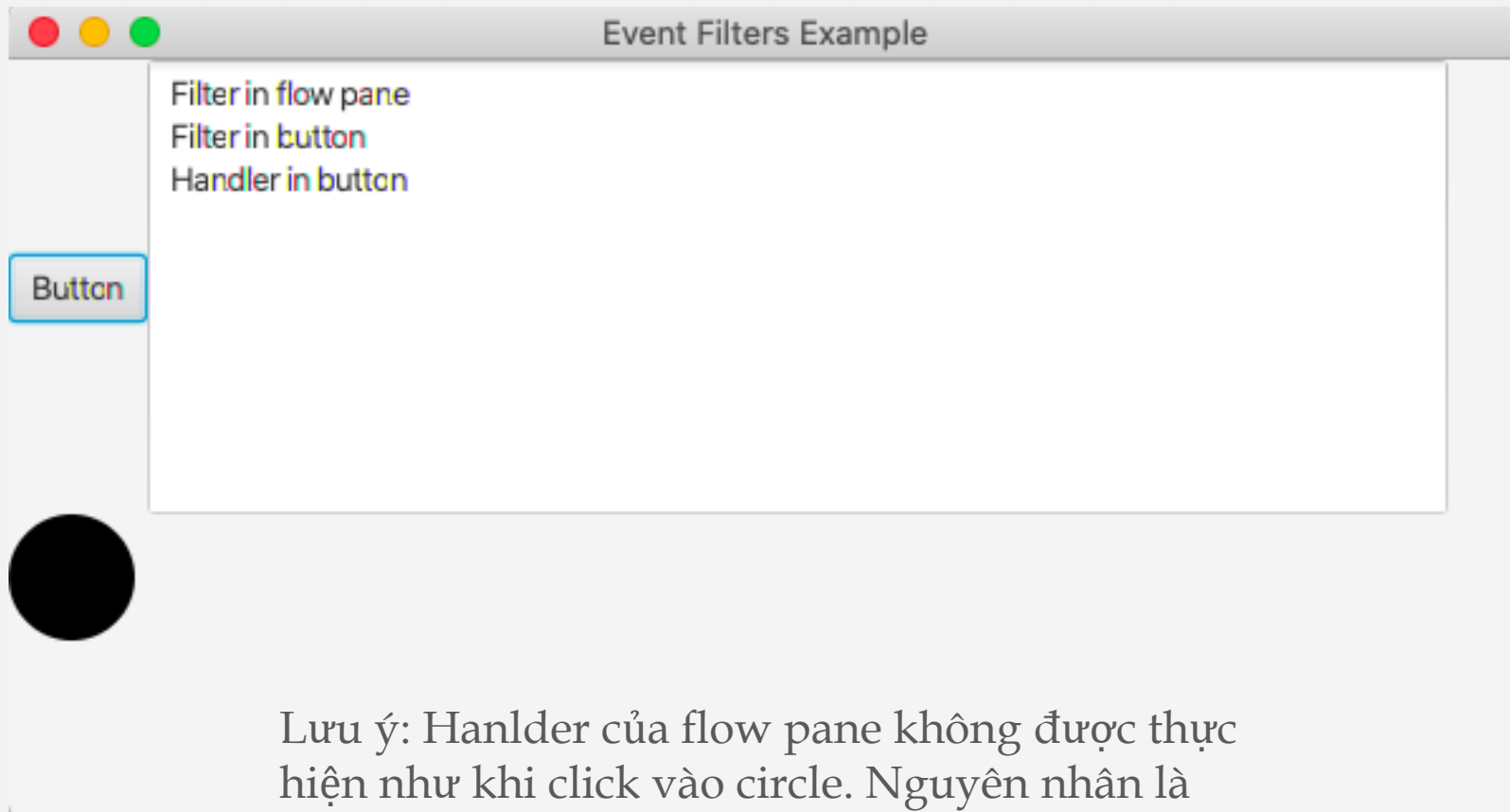
Khi nhấn chuột vào Circle



Lưu ý thứ tự thực hiện.

1. Filter thực hiện trước, Handler thực hiện sau
2. Filter của flow pane thực hiện trước Filter của circle
3. Handler của circle thực hiện trước Handler của flow pane

# Ví dụ

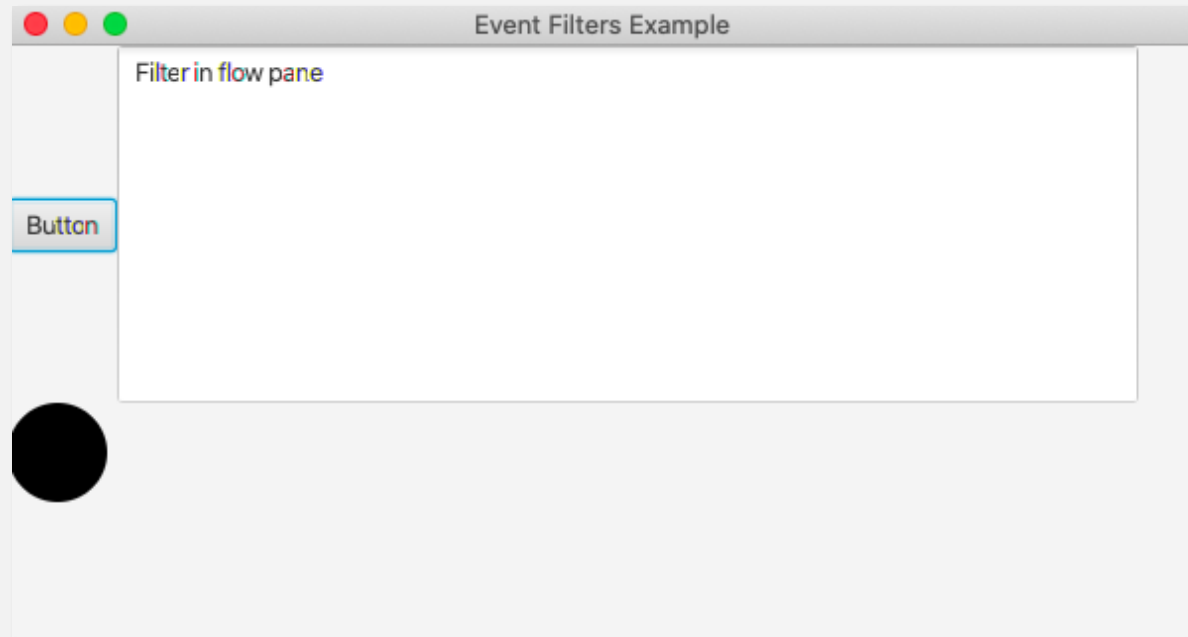


Lưu ý: Handler của flow pane không được thực hiện như khi click vào circle. Nguyên nhân là handler của button mặc định đã consume event

# Ví dụ

- Sửa đổi lại filter của flow pane như sau

```
fp.addEventFilter(MouseEvent.MOUSE_CLICKED, new EventHandler<MouseEvent>() {  
    @Override  
    public void handle(MouseEvent arg0) {  
        text.appendText("Filter in flow pane\n");  
        arg0.consume();  
    }  
});
```



# 7

## SceneBuilder tool

Công cụ kéo thả giao diện cho JavaFX



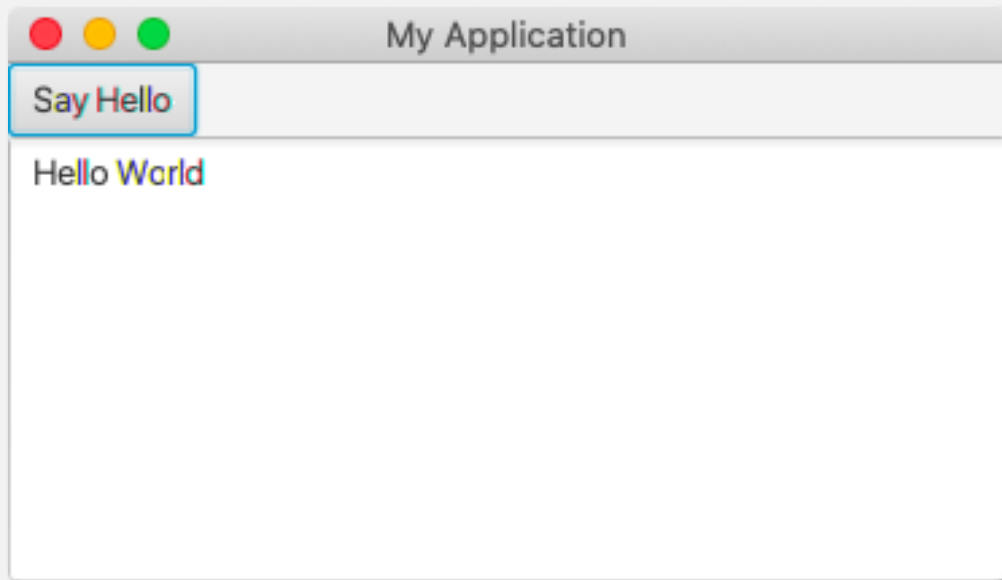


# Kéo thả giao diện với SceneBuilder

- Ý tưởng: tách biệt giao diện với logic xử lý nghiệp vụ
  - Giao diện ứng dụng: thiết kế trong file fxml
  - Logic xử lý (controller): tách biệt riêng trong file mã nguồn Java
- Các bước thực hiện:
  - Cài đặt SceneBuilder
  - Tạo giao diện (file fxml), định nghĩa các thuộc tính cho các component (tên component, các phương thức xử lý sự kiện)
  - Tạo JavaFX project
  - Copy file giao diện fxml vào JavaFX project
  - Cài đặt controller
  - Kết nối file giao diện fxml với controller
  - Tạo ứng dụng JavaFX, load file fxml

# Ví dụ

- Viết ứng dụng: Khi nhấn button "Say Hello", in dòng chữ Hello World ra textbox



# Tạo file SayHello.fxml với SceneBuilder

Kéo thả container VBox

Tương tự, kéo thả Button và Textarea trong mục "Controls"

Library

Containers

- TabPane
- TabPane (empty)
- TextFlow (FX8)
- TilePane
- TitledPane
- TitledPane (empty)
- ToolBar
- VBox**

Controls

Glue

Menu

Miscellaneous

Shapes

Charts

3D

Document

Hierarchy

- VBox**
  - Button Say Hello
  - TextArea

Inspector

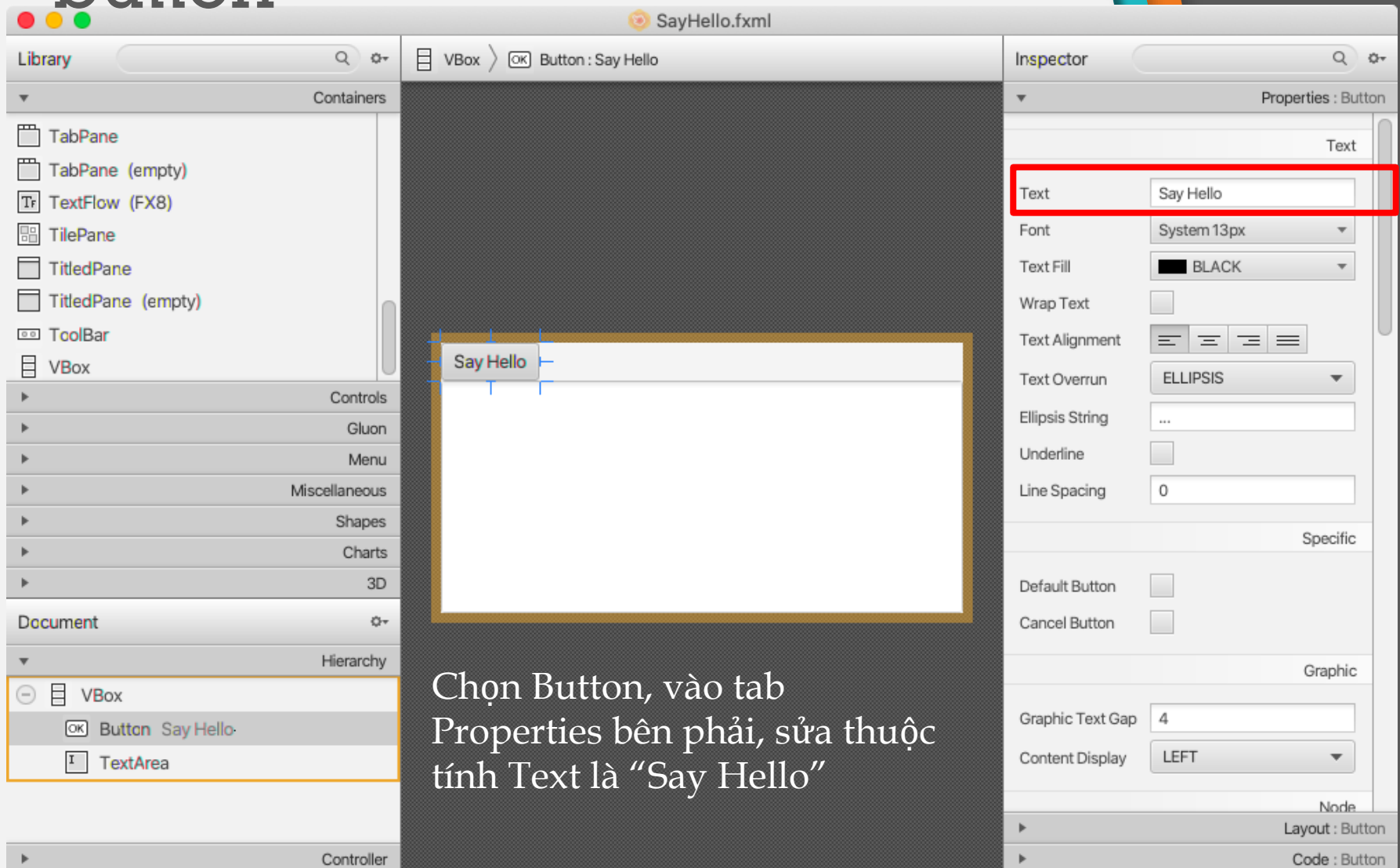
Properties

No Selection

Layout

Code

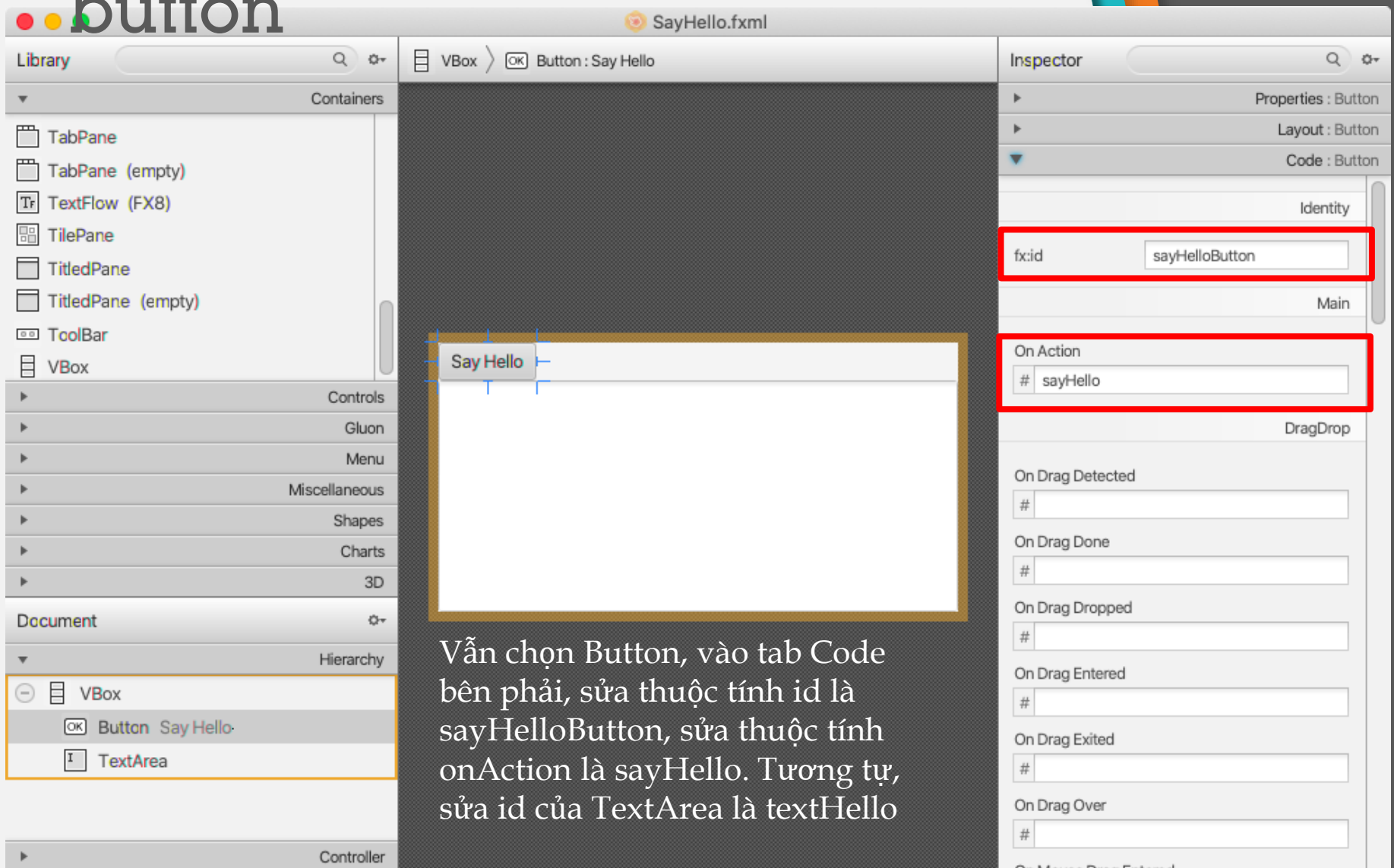
# Chỉnh sửa thuộc tính của button



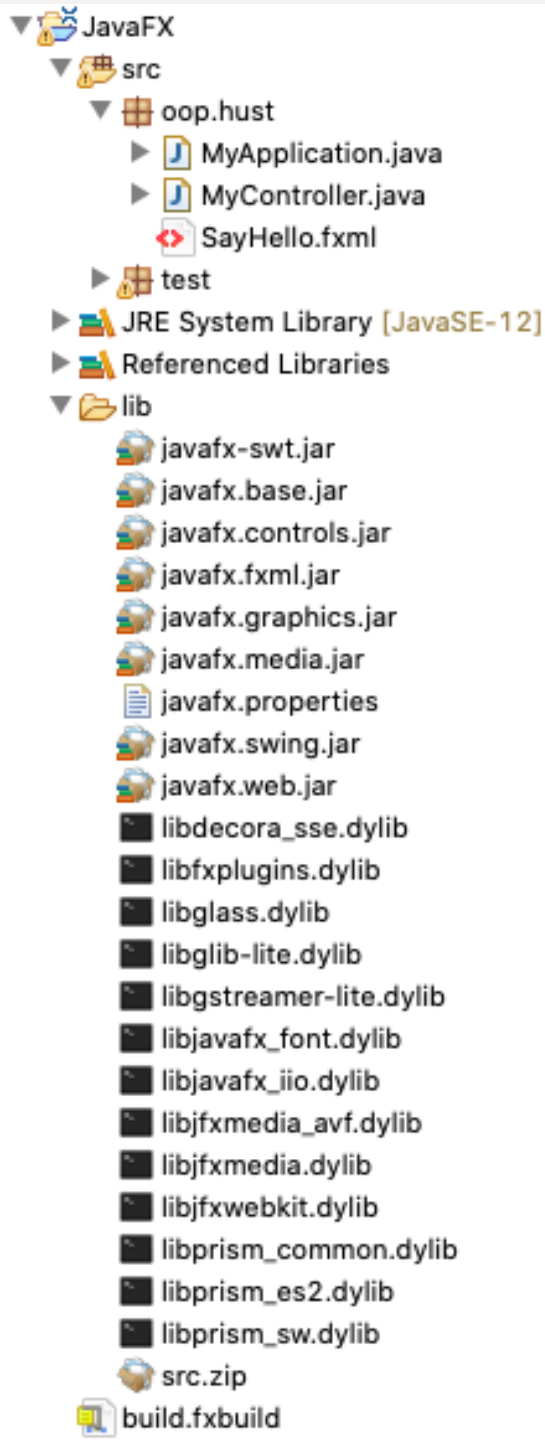
The screenshot displays the JavaFX IDE interface for a project named "SayHello.fxml". The central workspace shows a button with the text "Say Hello" and a yellow selection border. The Inspector panel on the right is open to the "Text" properties of the selected button. The "Text" property is highlighted with a red box and is set to "Say Hello". Other visible properties include "Font" (System 13px), "Text Fill" (BLACK), "Text Alignment" (left), "Text Overrun" (ELLIPSIS), "Ellipsis String" (...), "Underline" (unchecked), "Line Spacing" (0), "Default Button" (unchecked), "Cancel Button" (unchecked), "Graphic Text Gap" (4), and "Content Display" (LEFT).

Chọn Button, vào tab Properties bên phải, sửa thuộc tính Text là "Say Hello"

# Chỉnh sửa thuộc tính của button



Vẫn chọn Button, vào tab Code bên phải, sửa thuộc tính id là sayHelloButton, sửa thuộc tính onAction là sayHello. Tương tự, sửa id của TextArea là textHello



# Tạo project JavaFX

- Tạo project JavaFX như bình thường, copy file SayHello.fxml vào project

# Cài đặt Controller: tạo lớp MyController

```
import java.net.URL;
import java.util.ResourceBundle;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;

public class MyController implements Initializable {
    @FXML
    private Button sayHelloButton;

    @FXML
    private TextArea textHello;

    @Override
    public void initialize(URL location, ResourceBundle resources) {

    }

    public void sayHello(ActionEvent event) {
        textHello.setText("Hello World");
    }
}
```

Lưu ý: tên Button và tên  
TextArea phải khớp với các id  
tạo trong SceneBuilder

# Kết nối file giao diện fxml với controller

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.layout.VBox?>

<VBox prefHeight="192.0" prefWidth="371.0"
xmlns="http://javafx.com/javafx/11.0.1"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="oop.hust.MyController">
<children>
<Button fx:id="sayHelloButton" mnemonicParsing="false"
onAction="#sayHello" text="Say Hello" />
<TextArea fx:id="textHello" prefHeight="173.0" prefWidth="162.0"/>
</children>
</VBox>
```

- Sửa lại file SayHello.fxml: thêm thuộc tính fx:controller cho thẻ VBox, trỏ tới lớp MyController vừa tạo (dùng full name)



# Tạo ứng dụng JavaFX, load file fxml

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class MyApplication extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
            // Đọc file fxml và vẽ giao diện.
            Parent root = FXMLLoader.load(getClass()
                .getResource("/oop/hust/SayHello.fxml"));

            primaryStage.setTitle("My Application");
            primaryStage.setScene(new Scene(root));
            primaryStage.show();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

**Thank you!**

Any questions?

