# BỘ QUẢN LÝ TRÌNH BÀY TRONG JAVAFX

Các bộ quản lý trình bày trong JavaFX là các vùng chứa được sử dụng để sắp xếp linh hoạt và tương thích động các thành phần giao diện đồ họa tương tác trong vùng chứa Scene của JavaFX. Khi một cửa sổ được thay đổi kích thước, khung bố trí sẽ tự động định vị lại và thay đổi kích thước các thành phần chứa bên trong nó.

javafx.scene.Node

javafx.scene.Parent

javafx.scene.layout.Region

javafx.scene.layout.Pane

JavaFX cung cấp một số lớp có chức năng của một bộ quản lý trình bày như Hbox, Vbox, FlowPane, StackPane, BorderPane, GridPane, TilePane, TextFlow, đây là các lớp thừa kế từ lớp javafx.scene.layout.Pane, ngoài ra có thể kể thêm một số lớp khác như ScrollPane, Accordion... Trong giáo trình này, chúng tôi giới thiệu một số lớp thông dụng thừa kế từ lớp Pane. Các lớp này thừa kế các thuộc tính, phương thức chung từ các lớp cha Pane, Region, Parent, Node như border, padding, margin, spacing, alignment...

## ▪ Pane

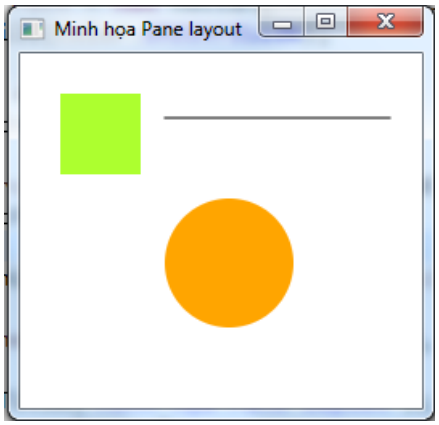
Pane là lớp cha của các lớp quản lý trình bày cơ bản, Pane cũng có thể được sử dụng để định vị các thành phần theo tọa độ tuyệt đối. Các bố cục phức tạp phải luôn được tạo bằng cách kết hợp sử dụng các bộ quản lý trình bày khác nhau, cách bố cục tuyệt đối được sử dụng trong các tình huống cụ thể như biểu đồ định vị hoặc hình ảnh.

Ví dụ:

```
@Override
    public void start(Stage stage) {

        Pane root = new Pane();

        Rectangle rect = new Rectangle(25, 25, 50, 50);
        rect.setFill(Color.GREENYELLOW);

        Line line = new Line(90, 40, 230, 40);
        line.setStroke(Color.BLACK);

        Circle circle = new Circle(130, 130, 30);
        circle.setFill(Color.ORANGE);

        root.getChildren().addAll(rect, line, circle);

        Scene scene = new Scene(root, 250, 220, Color.WHITE);

        stage.setTitle("Minh họa Pane layout");
        stage.setScene(scene);
        stage.show();
```

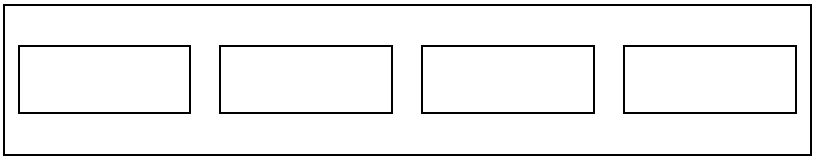
```
}
```

Kết quả chạy chương trình:




- **Hbox**

Hbox là một bộ quản lý trình bày, nó sắp xếp các thành phần con theo chiều ngang từ trái sang phải theo một dòng duy nhất.



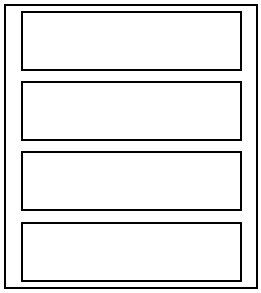
**Hình 7.4. Cách sắp xếp các thành phần con của Hbox**

Hbox đã được sử dụng trong các ví dụ về các thành phần UI.

- **Vbox**

Vbox là một bộ quản lý trình bày, nó sắp xếp các thành phần con theo chiều dọc từ trên xuống dưới theo một cột duy nhất.

Vbox đã được sử dụng trong các ví dụ về các thành phần UI.



**Hình 7.5. Cách sắp xếp các thành phần con của Vbox**

## ▪ FlowPane

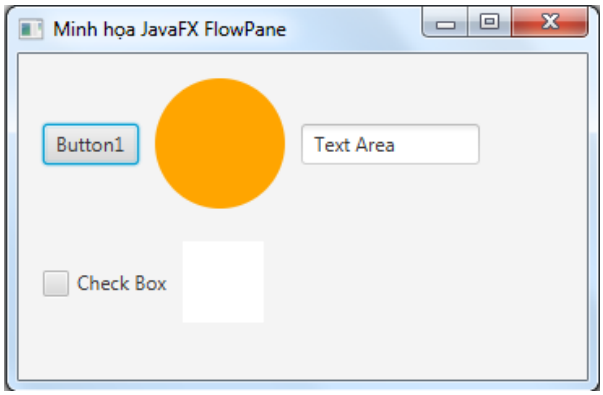
**FlowPane** là một bộ quản lý trình bày, nó sắp xếp các thành phần con liên tiếp nhau trên một dòng theo mặc định hoặc cột tùy thuộc việc thiết lập hướng, và tự động đẩy phần tử con xuống dòng/cột tiếp theo nếu dòng/cột hiện tại không còn chỗ trống.

Ví dụ:

```
@Override
    public void start(Stage primaryStage) throws Exception {
        FlowPane root = new FlowPane();

        // Khoảng cách giữa các UI theo chiều ngang
        root.setHgap(10);
```

```
        // Khoảng cách giữa các UI theo chiều dọc
        root.setVgap(20);
        root.setPadding(new Insets(15,15,15,15));

        Button button1= new Button("Button1");
        Circle circle = new Circle(130, 130, 40);
        circle.setFill(Color.ORANGE);

        TextField textField = new TextField("Text Area");
        textField.setPrefWidth(110);

        CheckBox checkBox = new CheckBox("Check Box");
        Rectangle rect = new Rectangle(25, 25, 50, 50);
        rect.setFill(Color.WHITE);

        root.getChildren().addAll(button1, button2,
                                  textField, checkBox, rect);

        Scene scene = new Scene(root, 350, 200);

        primaryStage.setTitle("Minh họa JavaFX FlowPane");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```

Kết quả chạy chương trình:




## ▪ StackPane

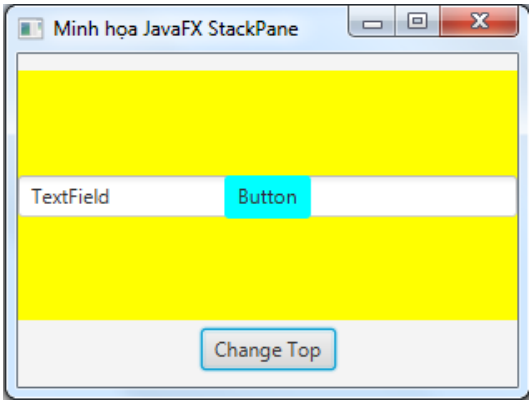
**StackPane** là một quản lý trình bày, nó sắp xếp các thành phần con chồng lên nhau. Các thành phần con được thêm vào mới nhất sẽ nằm ở phía trên cùng của **Stack**. Ta có thể đưa một thành phần con bất kỳ nên phía trước của **Stack** thông qua phương thức **toFront()**, hoặc đưa thành phần con xuống phía cuối của **stack** thông qua phương thức **toBack()**.

Ví dụ:

```
@Override
    public void start(Stage primaryStage) throws Exception {
        TextField textField = new TextField("TextField");

        Button button = new Button("Button");
```

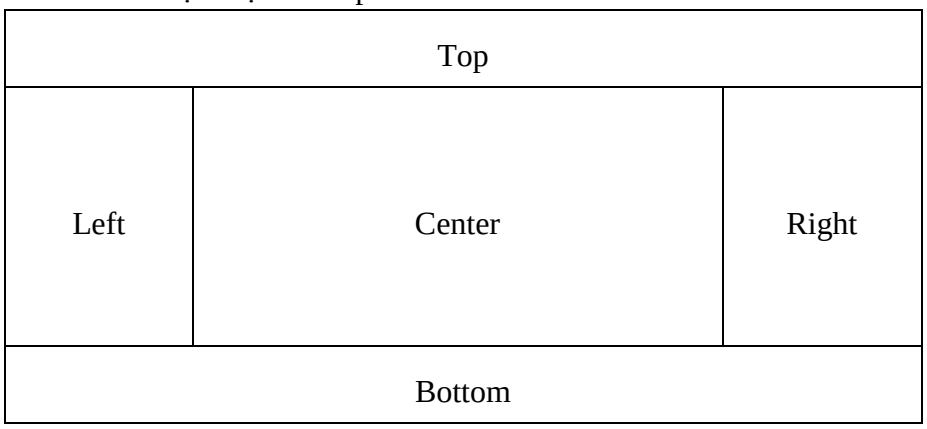
```
        button.setStyle("-fx-background-color: cyan;");

        CheckBox checkBox = new CheckBox("CheckBox");

        StackPane stackPane = new StackPane();
        stackPane.setPrefSize(150, 150);
        stackPane.setStyle("-fx-background-color: yellow;");

        VBox root = new VBox();
        stackPane.getChildren().addAll(textField, button, checkBox);
        root.getChildren().add(stackPane);

        Button changeButton = new Button("Change Top");
        root.getChildren().add(changeButton);
        root.setAlignment(Pos.CENTER);
        root.setSpacing(5);

        Scene scene = new Scene(root, 300, 200);

        primaryStage.setTitle("Minh họa JavaFX StackPane");
        primaryStage.setScene(scene);

        changeButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                ObservableList<Node> childNodes = stackPane.getChildren();
                if (childNodes.size() > 1) {
                    //
                    Node bottomNode = childNodes.get(0);
                    bottomNode.toFront();
                }
            }

        });

        primaryStage.show();
    }
```

Kết quả chạy chương trình:




## ▪ BorderPane

**BorderPane** là một bộ quản lý trình bày, nó được phân chia thành 5 vùng riêng biệt, mỗi vùng có thể chứa được một thành phần con.




**Hình 7.6. Cách tổ chức layout của BorderPane**

Cơ chế co giãn của các vùng trong BorderPane tuân theo quy tắc:

+ Vùng Top/Bottom: Có thể co/giãn theo chiều ngang và giữ nguyên chiều cao.

+ Vùng Left/Right: Có thể co/giãn theo chiều thẳng đứng và giữ nguyên độ dài.

+ Vùng Center: Có thể co/giãn theo cả 2 chiều.

Nếu một vùng nào đó không chứa thành phần con, các vùng khác sẽ chiếm lấy không gian của nó.

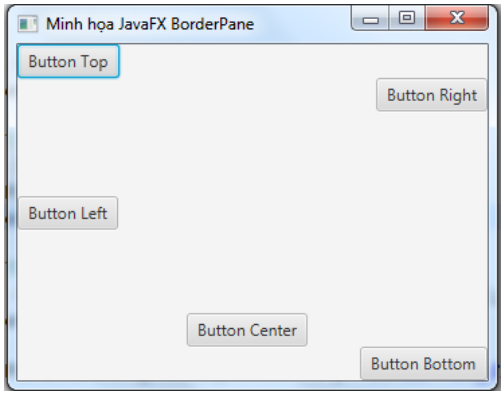
Trong **JavaFX**, các thành phần con nằm trong một vùng nào đó của **BorderPane** có thể không chiếm đầy không gian của vùng đó. Ví dụ, nếu **Button** nằm trong một vùng của **BorderPane** mặc định nó sẽ không dãn đầy vùng này. Tuy nhiên nếu một Hbox hay Vbox nằm trong một vùng con nào đó của BorderPane, mặc định nó sẽ chiếm đầy vùng đó.

Ví dụ:

```
@Override
  public void start(Stage primaryStage) throws Exception {
      BorderPane root = new BorderPane();
      root.setPadding(new Insets(15, 20, 10, 10));

      // TOP
      Button btnTop = new Button("Top");
```

```
        root.setTop(btnTop);

        // LEFT
        Button btnLeft = new Button("Left");
        root.setLeft(btnLeft);
        // Set margin cho vùng left.
        BorderPane.setMargin(btnLeft, new Insets(10, 10, 10, 10));

        // CENTER
        Button btnCenter = new Button("Center");
        root.setCenter(btnCenter);
        // Căn lề.
        BorderPane.setAlignment(btnCenter, Pos.BOTTOM_CENTER);

        // RIGHT
        Button btnRight = new Button("Right");
        root.setRight(btnRight);

        // BOTTOM
        Button btnBottom = new Button("Bottom");
        btnBottom.setPadding(new Insets(5, 5, 5, 5));
        root.setBottom(btnBottom);
        // Căn lề
        BorderPane.setAlignment(btnBottom, Pos.TOP_RIGHT);

        Scene scene = new Scene(root, 350, 250);

        primaryStage.setTitle("Minh họa JavaFX BorderPane");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```

Kết quả chạy chương trình:




## ▪ AnchorPane

**AnchorPane** là một bộ quản lý trình bày, nó cho phép ta định nghĩa các điểm neo (anchor) cho các thành phần con. Có 4 loại điểm neo là: top (trên), bottom (dưới), left (trái), right (phải). Mỗi thành phần con có thể không chọn hoặc lựa chọn nhiều nhất 4 điểm neo.

Mỗi điểm neo được hiểu theo nghĩa khoảng cách thiết lập từ một thành phần con tới một cạnh nào đó (trên, dưới, trái, phải) sẽ không đổi cho dù kích thước khung thay đổi.

Một thành phần con muốn cố định ở một góc, nó có thể chọn 2 điểm neo ứng với 2 cạnh của góc đó.

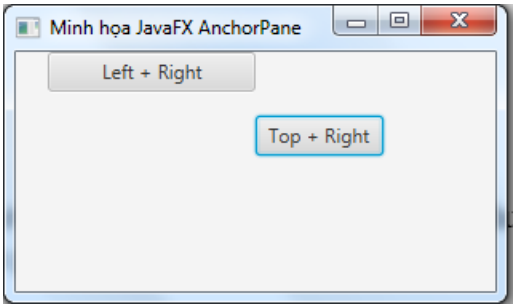
**Hình 7.7. Minh họa cách neo một thành phần con vào các góc của AnchorPane**

Một thành phần con nếu thiết lập điểm neo ở 2 cạnh đối diện nhau thì khi kích thước khung chứa thay đổi theo chiều đó, kích thước thành phần con cũng bị co giãn theo.

Ví dụ:

```
@Override
    public void start(Stage primaryStage) throws Exception {
        AnchorPane root = new AnchorPane();
        Button button1 = new Button("Top + Right");

        Button button2 = new Button("Left + Right");

        // Neo vào Top + Right
        AnchorPane.setTopAnchor(button1, 40.0);
        AnchorPane.setRightAnchor(button1, 70.0);

        // Neo vào Left + Right
        AnchorPane.setLeftAnchor(button2, 20.0);
        AnchorPane.setRightAnchor(button2, 150.0);

        // Thêm vào AnchorPane
        root.getChildren().addAll(button1, button2);

        Scene scene = new Scene(root, 300, 150, Color.WHITE);

        primaryStage.setTitle("Minh họa JavaFX AnchorPane");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```

Kết quả chạy chương trình (co giãn khung để quan sát thay đổi):




## ▪ GridPane

**GridPane** là một bộ quản lý trình bày, bề mặt của nó được chia thành một lướt, bao gồm các hàng và các cột. Một thành phần con (child node), có thể nằm trên một ô lưới hoặc nằm trên một ô hợp nhất từ các ô gần nhau.

Để thêm một thành phần con vào một ô lưới ta dùng phương thức:

*grid.add(component, columnIndex, rowIndex);*

Để thêm một thành phần con vào một ô hợp nhất từ các ô liền nhau ta dùng phương thức: *grid.add(component, columnIndex, rowIndex, columnSpan, rowSpan);*

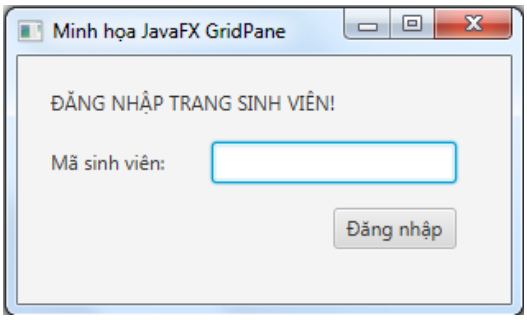
Độ rộng một cột nào đó trên GridPane được quyết định bởi độ rộng lớn nhất của thành phần con chứa trong cột đó.

Độ cao của một dòng nào đó trên GridPane được quyết định bởi độ cao lớn nhất của thành phần con chứa trong dòng đó.

```
@Override
    public void start(Stage primaryStage) throws Exception {
        GridPane root = new GridPane();

        root.setPadding(new Insets(20));
        root.setHgap(25); // khoảng cách giữa các cột
        root.setVgap(15); // khoảng cách giữa các hàng

        Label titleLabel = new Label("ĐĂNG NHẬP TRANG SINH VIÊN!");

        // Đặt vào ô lưới (0, 0), rộng bằng 2 ô, cao bằng 1 ô
        root.add(titleLabel, 0, 0, 2, 1);

        Label codeLabel = new Label("Mã sinh viên: ");
        GridPane.setHalignment(codeLabel, HPos.RIGHT);
        // Đặt vào ô lưới (0,1)
        root.add(codeLabel, 0, 1);

        TextField codeTextField = new TextField();
        // Căn lề trái cho User Name
        GridPane.setHalignment(codeTextField, HPos.LEFT);
        root.add(codeTextField, 1, 1);


        Button loginButton = new Button("Đăng nhập");
```

```
        // Căn lề phải cho button Login.
        GridPane.setHalignment(loginButton, HPos.RIGHT);
        root.add(loginButton, 1, 2);

        Scene scene = new Scene(root, 300, 150, Color.WHITE);
        primaryStage.setTitle("Minh họa JavaFX GridPanel");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```

Kết quả chạy chương trình:




## ▪ TilePane

**TilePane** là một bộ quản lý trình bày, tương tự như FlowPane, nó sắp xếp các thành phần con liên tiếp nhau trên một dòng theo mặc định hoặc cột tùy thuộc việc thiết lập hướng, và tự động đẩy phần tử con xuống dòng/cột tiếp theo nếu dòng/cột hiện tại không còn chỗ trống.

TilePane khác FlowPane ở chỗ các thành phần con được sắp xếp nằm trên các ô có kích thước giống nhau. Kích thước của các ô tùy thuộc vào kích thước của thành phần lớn nhất.

Trong ví dụ về FlowPane, thực hiện thay lớp FlowPane bởi TilePane ta có kết quả chạy chương trình: