**MỤC LỤC**

[LỜI CẢM ƠN 6](#_Toc55316517)

[1. GIỚI THIỆU CHUNG VỀ JAVAFX 7](#_Toc55316518)

[1.1 Tổng quan về JavaFX 7](#_Toc55316519)

[1.2. JavaFX với Swing và AWT 7](#_Toc55316520)

[1.2.1 Swing và AWT 7](#_Toc55316521)

[1.2.2 Ưu điểm của JavaFX so với Swing và AWT 7](#_Toc55316522)

[1.2.3 Lý do nên học JavaFX 8](#_Toc55316523)

[1.3 Cấu trúc cơ bản của chương trình ứng dụng JavaFX 8](#_Toc55316524)

[1.3.1 Các bước cơ bản để tạo chương trình JavaFX 8](#_Toc55316525)

[1.3.2 Giao diện của ứng dụng javaFX 9](#_Toc55316526)

[1.3.3 Sơ đồ UML các thành phần cơ bản trong ứng dụng JavaFX 10](#_Toc55316527)

[2. CÁC NGĂN BỐ CỤC (LAYOUT PANES) TRONG JAVAFX 12](#_Toc55316528)

[2.1 Giới thiệu chung về các ngăn bố cục trong JavaFX 12](#_Toc55316529)

[2.2 Pane 13](#_Toc55316530)

[2.2.1 Tổng quan về lớp Pane 13](#_Toc55316531)

[2.2.2 Chương trình minh họa sử dụng lớp Pane 13](#_Toc55316532)

[2.3 GridPane 14](#_Toc55316533)

[2.3.1 Tổng quan về lớp GridPane 14](#_Toc55316534)

[2.3.2 Chương trình minh họa sử dụng lớp GridPane 15](#_Toc55316535)

[2.4 StackPane 16](#_Toc55316536)

[2.4.1 Tổng quan về lớp StackPane 16](#_Toc55316537)

[2.4.2 Chương trình minh họa sử dụng lớp StackPane 17](#_Toc55316538)

[2.5 BorderPane 18](#_Toc55316539)

[2.5.1 Tổng quan về lớp BorderPane 18](#_Toc55316540)

[2.5.2 Chương trình mình họa sử dụng lớp BorderPane 19](#_Toc55316541)

[2.6 FlowPane 20](#_Toc55316542)

[2.6.1 Tổng quan về lớp FlowPane 20](#_Toc55316543)

[2.6.2 Chương trình mình họa sử dụng lớp FlowPane 22](#_Toc55316544)

[2.7 HBox 23](#_Toc55316545)

[2.7.1 Tổng quan về lớp HBox 23](#_Toc55316546)

[2.7.2 Chương trình minh họa sử dụng lớp HBox 23](#_Toc55316547)

[2.8 VBox 24](#_Toc55316548)

[2.8.1 Tổng quan về lớp VBox 24](#_Toc55316549)

[2.8.2 Chương trình minh họa sử dụng lớp VBox 25](#_Toc55316550)

[3. CÁC UI CONTROL TRONG JAVAFX 27](#_Toc55316551)

[3.1 Giới thiệu chung về các UI Control trong JavaFX 27](#_Toc55316552)

[3.2 Label 27](#_Toc55316553)

[3.2.1 Tổng quan về lớp Label 27](#_Toc55316554)

[3.2.2 Chương trình minh họa sử dụng lớp Label 28](#_Toc55316555)

[3.3 Button 29](#_Toc55316556)

[3.3.1 Tổng quan về lớp Button 29](#_Toc55316557)

[3.3.2 Chương trình minh họa sử dụng lớp Button 30](#_Toc55316558)

[3.4 CheckBox 30](#_Toc55316559)

[3.4.1 Tổng quan về lớp CheckBox 30](#_Toc55316560)

[3.4.2 Chương trình minh họa sử dụng lớp CheckBox 31](#_Toc55316561)

[3.5 RadioButton 32](#_Toc55316562)

[3.5.1 Tổng quan về lớp RadioButton 32](#_Toc55316563)

[3.5.2 Chương trình minh họa sử dụng lớp RadioButton 33](#_Toc55316564)

[3.6 TextField 34](#_Toc55316565)

[3.6.1 Tổng quan về lớp TextField 34](#_Toc55316566)

[3.6.2 Chương trình minh họa sử dụng lớp TextField 35](#_Toc55316567)

[3.7 TextArea 35](#_Toc55316568)

[3.7.1 Tổng quan về lớp TextArea 35](#_Toc55316569)

[3.7.2 Chương trình minh họa sử dụng lớp TextArea 36](#_Toc55316570)

[3.8 ComboBox 37](#_Toc55316571)

[3.8.1 Tổng quan về lớp ComboBox 37](#_Toc55316572)

[3.8.2 Chương trình minh họa sử dụng lớp ComboBox 38](#_Toc55316573)

[3.9 ListView 39](#_Toc55316574)

[3.9.1 Tổng quan về lớp ListView 39](#_Toc55316575)

[3.9.2 Chương trình minh họa sử dụng lớp ListView 39](#_Toc55316576)

[3.10 ScrollBar 40](#_Toc55316577)

[3.10.1 Tổng quan về lớp ScrollBar 40](#_Toc55316578)

[3.10.2 Chương trình minh họa sử dụng lớp ScrollBar 41](#_Toc55316579)

[3.11 Slider 42](#_Toc55316580)

[3.11.1 Tổng quan về lớp Slider 42](#_Toc55316581)

[3.11.2 Chương trình minh họa sử dụng lớp Slider 43](#_Toc55316582)

[4. NHỮNG HÌNH DẠNG HAI CHIỀU (2D SHAPE) TRONG JAVAFX 45](#_Toc55316583)

[4.1 Giới thiệu chung về những hình dạng 2 chiều trong JavaFX 45](#_Toc55316584)

[4.2 Shape 45](#_Toc55316585)

[4.3 Line 46](#_Toc55316586)

[4.3.1 Tổng quan về lớp Line 46](#_Toc55316587)

[4.3.2 Chương trình minh họa sử dụng lớp Line 46](#_Toc55316588)

[4.4 Rectangle 47](#_Toc55316589)

[4.4.1 Tổng quan về lớp Rectangle 47](#_Toc55316590)

[4.4.2 Chương trình minh họa sử dụng lớp Rectangle 48](#_Toc55316591)

[4.5 Circle 49](#_Toc55316592)

[4.5.1 Tổng quan về lớp Circle 49](#_Toc55316593)

[4.5.2 Chương trình minh họa lớp Circle 49](#_Toc55316594)

[4.6 Ellipse 50](#_Toc55316595)

[4.6.1 Tổng quan về lớp Ellipse 50](#_Toc55316596)

[4.6.2 Chương trình minh họa lớp Ellipse 51](#_Toc55316597)

[4.7 Arc 52](#_Toc55316598)

[4.7.1 Tổng quan về lớp Arc 52](#_Toc55316599)

[4.7.2 Chương trình minh họa lớp Arc 52](#_Toc55316600)

[4.8 Polygon và Polyline 53](#_Toc55316601)

[4.8.1 Tổng quan về lớp Polygon và lớp Polyline 53](#_Toc55316602)

[4.8.2 Chương trình minh họa lớp Polygon và lớp Polyline 54](#_Toc55316603)

[5. MỘT SỐ LỚP THÔNG DỤNG TRONG JAVAFX 56](#_Toc55316604)

[5.1 Giới thiệu chung về một số lớp thông dụng trong JavaFX 56](#_Toc55316605)

[5.2 Lớp Color 56](#_Toc55316606)

[5.2.1 Tổng quan về lớp Color 56](#_Toc55316607)

[5.2.2 Chương trình minh họa lớp Color 58](#_Toc55316608)

[5.3 Lớp Font 58](#_Toc55316609)

[5.3.1 Tổng quan về lớp Font 58](#_Toc55316610)

[5.3.2 Chương trình minh họa lớp Font 60](#_Toc55316611)

[5.4 Lớp Image và ImageView 60](#_Toc55316612)

[5.4.1 Tổng quan về lớp Image và ImageView 60](#_Toc55316613)

[5.4.2 Chương trình minh họa lớp Image và ImageView 62](#_Toc55316614)

[5.5 Lớp Media, MediaPlayer, MediaView 63](#_Toc55316615)

[5.5.1 Tổng quan về các lớp Media, MediaPlayer, MediaView 63](#_Toc55316616)

[5.5.2 Chương trình minh họa các lớp Media, MediaPlayer, MediaView 65](#_Toc55316617)

[5.6 Lớp Text 66](#_Toc55316618)

[5.6.1 Tổng quan về lớp Text 66](#_Toc55316619)

[5.6.2 Chương trình minh họa lớp Text 66](#_Toc55316620)

[6. DỰ ÁN MINH HỌA SỬ DỤNG JAVAFX 69](#_Toc55316621)

[KẾT LUẬN 72](#_Toc55316622)

[TÀI LIỆU THAM KHẢO 73](#_Toc55316623)

# LỜI CẢM ƠN

*Lời đầu tiên chúng tôi xin gửi lời cảm ơn chân thành nhất đến thầy Nguyễn Mạnh Sơn – giảng viên môn Lập trình hướng đối tượng đã tận tình hướng dẫn và cung cấp nguồn tài liệu hữu ích phục vụ chúng tôi trong suốt quá trình làm báo cáo này.*

*Để hoàn thành tốt bài báo cáo này, nhóm chúng tôi đã nỗ lực hết sức. Nhưng vì kiến thức của cả nhóm còn hạn chế, trong suốt quá trình làm và hoàn thiện không tránh khỏi những sai sót. Chúng tôi rất mong nhận được sự chỉ bảo, đóng góp ý kiến của thầy và các bạn để có điều kiện bổ xung, nâng cao kiến thức của các thành viên trong nhóm để hoàn thành tốt hơn những bài báo cáo sau này.*

*Cuối cùng, chúng tôi xin kính chúc thầy và các bạn dồi dào sức khỏe và đạt được nhiều thành công tốt đẹp trong công việc.*

*Xin chân thành cảm ơn!*

# 1. GIỚI THIỆU CHUNG VỀ JAVAFX

## 1.1 Tổng quan về JavaFX

JavaFX là một công cụ tuyệt vời để học lập trình hướng đối tượng. JavaFX là một thư viện của Java được sử dụng để phát triển các ứng dụng máy tính cũng như các ứng dụng internet phong phú. Các ứng dụng được xây dựng trong JavaFX có thể chạy trên nhiều nền tảng bao gồm Web, Mobile, Desktop.

JavaFX cung cấp nhiều chức năng hỗ trợ tuyệt vời để phát triển một chương trình Java GUI (Graphical User Interface – Giao diện đồ họa người dùng) hoàn chỉnh bao gồm 2D Shapes, 3D Shapes, Effects, Animation, Text, Layouts, UI Controls, Transformations, Charts ...

## 1.2. JavaFX với Swing và AWT

### 1.2.1 Swing và AWT

Khi Java được giới thiệu, các lớp GUI được đóng gói trong một thư viện được gọi là Abstract Windows Toolkit (AWT). AWT phù hợp để phát triển giao diện người dùng đồ họa đơn giản, nhưng không phù hợp để phát triển các dự án GUI toàn diện. Ngoài ra, AWT dễ gặp các lỗi đặc trưng của nền tảng.

Các thành phần giao diện người dùng của AWT được thay thế bằng một thư viện mạnh mẽ, đa năng và linh hoạt hơn được gọi là Swing. Các thành phần của Swing ít phụ thuộc vào nền tảng và ít sử dụng các tài nguyên tự nhiên của GUI. Swing được thiết kế để phát triển các ứng dụng GUI trên máy tính.

### 1.2.2 Ưu điểm của JavaFX so với Swing và AWT

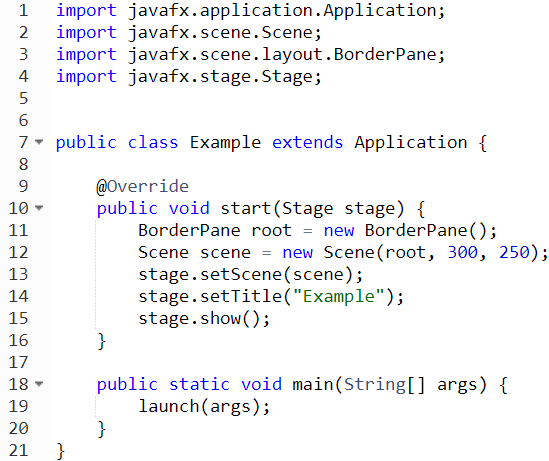
Swing và AWT đang dần được thay thế bằng nền tảng JavaFX để phát triển các ứng dụng GUI phong phú. JavaFX kết hợp các công nghệ GUI hiện đại và cung cấp hỗ trợ cảm ứng đa điểm cho các thiết bị hỗ trợ cảm ứng như máy tính bảng và điện thoại thông minh. JavaFX có hỗ trợ 2D, 3D, hoạt ảnh, video và âm thanh. Với sự hỗ trợ của phần mềm bên thứ ba, chúng ta có thể phát triển các chương trình JavaFX trên các thiết bị chạy iOS hoặc Android.

### 1.2.3 Lý do nên học JavaFX

* JavaFX đơn giản hơn nhiều để học và sử dụng cho các lập trình viên Java mới.
* JavaFX là một công cụ tuyệt vời để phát triển các ứng dụng về lập trình hướng đối tượng hơn Swing.
* Swing về cơ bản đã chết vì nó sẽ không nhận bất cứ sự nâng cấp cao hơn nào.
* JavaFX là công cụ GUI mới để phát triển các ứng dụng GUI phong phú đa nền tảng trên máy tính và trên thiết bị cầm tay.

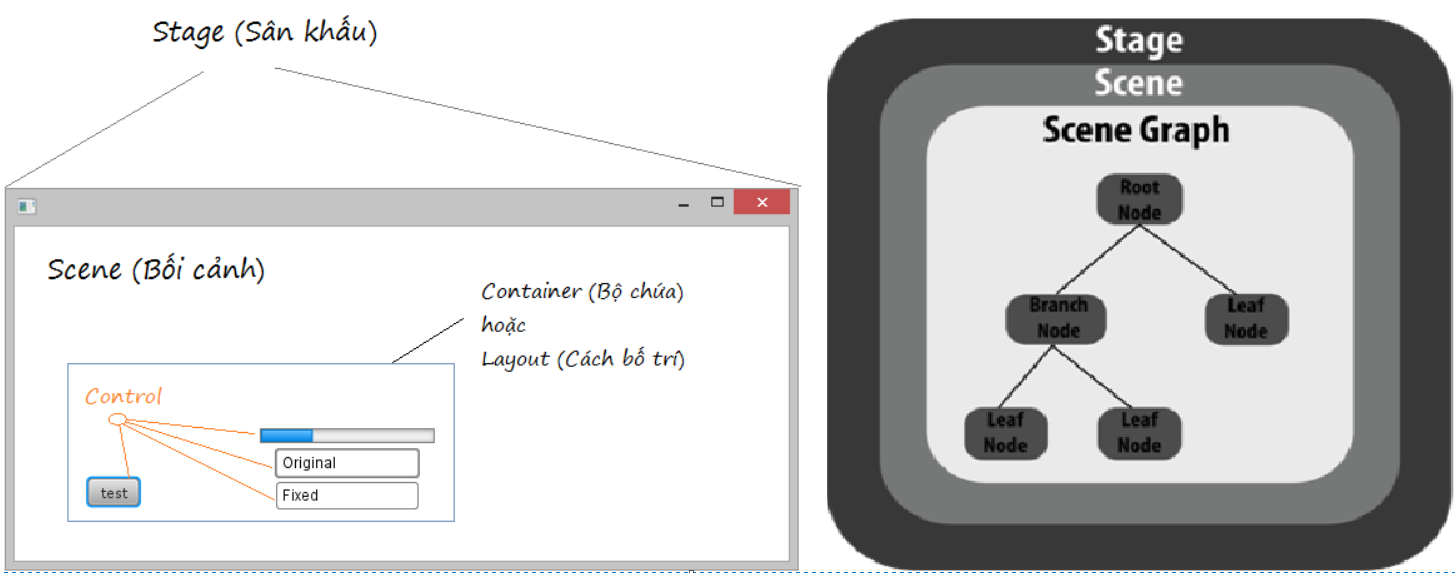
## 1.3 Cấu trúc cơ bản của chương trình ứng dụng JavaFX

### 1.3.1 Các bước cơ bản để tạo chương trình JavaFX

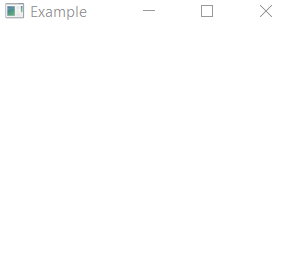
* Bước 1: Inport **javafx.application.Application**cho mọi chương trình JavaFX.
* Bước 2: Lớp của chương trình JavaFX phải kế thừa lớp **Application**.
* Bước 3: Override phương thức **start()** của lớp Application.
* **Bước 4: Gọi phương thức **launch(args)** trong phương thức **main**.

*Hình 1: Cấu trúc cơ bản của một chương trình javaFX*

### 1.3.2 Giao diện của ứng dụng javaFX

 Giao diện của một chương trình JavaFX được cầu tạo gồm ba thành phần chính là **Stage**, **Scene** và **Node**.

*Hình 2: Giao diện của ứng dụng JavaFX*

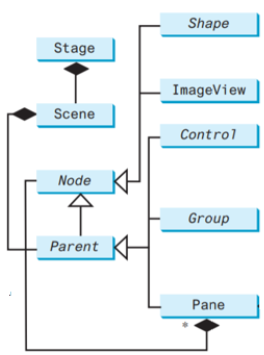
** **Stage** trong ứng dụng JavaFX tương tự như **Frame** trong ứng dụng Swing. Nó là vùng chứa cho tất cả các đối tượng JavaFX. Để sử dụng được đối tượng Stage chúng ta phải import **javafx.stage.Stage**. Một đối tượng stage chính sẽ luôn được tạo sẵn trong nền tảng và được truyền vào với vai trò như một tham số trong phương thừc **start()**. Ngoài ra chúng ta có thể tạo thêm các đối tượng stage khác để phù hợp với nhu cầu ứng dụng. Để hiển thị thì đối tượng stage cần gọi phương thức **show()** để hiện thị stage dưới dạng một cửa số. Ban đầu stage sẽ có dạng như hình 3. Lớp Stage có phương thức **setScene(Scene)** để thêm scene vào stage và **setTitle(String)** để đặt tiêu đề cho stage.

*Hình 3: Hiển thị Stage*

**Scene** là vùng chứa cho tất cả nội dung của scene graph. Chúng ta phải import **Javafx.scene.Scene** để có thể tạo một đối tượng scene và cung cấp tầt cả các phương thức để xử lý cho đối tượng scene. Một đối tượng stage tại một thời điểm chỉ có thể chứa được một đối tượng scene. Lớp Scene cung cấp rất nhiều các phương thức khởi tạo một đối tượng scene. Thông thường ta sử dụng **Scene(Parent root, double width, double height)** để khởi tạo một đối tượng scene với width và height là chiều rộng và cao cúa Scene.

**Scene graph** không phải là tên gọi của mội đối tượng trong JavaFX được chứa trong scene. Nó chỉ là tên gọi ám chỉ cây phân cấp chứa các node. Node là các phần tử được hiển thị trên stage. Các node được hiển thị theo cấu trúc cây. Luôn có một node gốc (cha) trong scene graph. Node cha này chỉ được là các đối tượng của các lớp con thuộc lớp **Parent**.

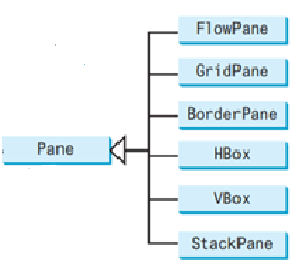
### 1.3.3 Sơ đồ UML các thành phần cơ bản trong ứng dụng JavaFX

** Qua sơ đồ được thể hiện trong hình 3, chúng ta có thể thấy rõ lớp Scene là một thành phần của lớp Stage nên lớp Stage chỉ chứa được đối tượng của lớp Scene. Lớp Parent cũng là một thành phần của lớp Scene nên lớp Scene cũng chỉ chứa được đối tượng của lớp Parent hoặc là các đối tượng của các lớp kế thừa lớp Parent.

*Hình 4: Sơ đồ UML các thành phần cơ bản trong ứng dụng JavaFX*

# 2. CÁC NGĂN BỐ CỤC (LAYOUT PANES) TRONG JAVAFX

## 2.1 Giới thiệu chung về các ngăn bố cục trong JavaFX

**JavaFX cung cấp nhiều loại ngăn bố cục khác nhau để tự động bố trí các node ở vị trí và kích thước mong muốn. Các ngăn bố cục này được dựng sẵn thành các lớp riêng biệt và chúng nằm trong gói **javafx.scene.layout**. Lớp **javafx.scene.layout.Pane** là lớp cơ sở cho tất cả các lớp ngăn bố cục này và chúng được thể hiện cụ thể trong bảng sau.

|  |  |
| --- | --- |
| **Lớp** | **Miêu tả** |
| Pane | Lớp cơ sở cho các ngăn bố cục. |
| GridPane | Tổ chức các node con trong ngăn bố cục dưới dạng lưới ma trận 2 chiều. |
| StackPane | Đặt các node con chồng nên nhau. |
| Hbox | Đặt các node con trong ngăn bố cục thành một hàng duy nhất. |
| Vbox | Đặt các node con trong ngăn bố cục thành một cột duy nhất. |
| BorderPane | Đặt các node con vào vùng trái, phải, trên, dưới và trung tâm trong ngăn bố cục. |
| FlowPane | Đặt các node con trong ngăn bố cục theo một hàng ngang hoặc hàng dọc duy nhất. Tự động đẩy các node con sang hàng ngang hoặc hàng dọc tiếp theo nếu không gian trong hàng hiện thị không đủ chỗ. |

*Hình 5: Sơ đồ UML thể hiện mối quan hệ của các ngăn bố cục*

## 2.2 Pane

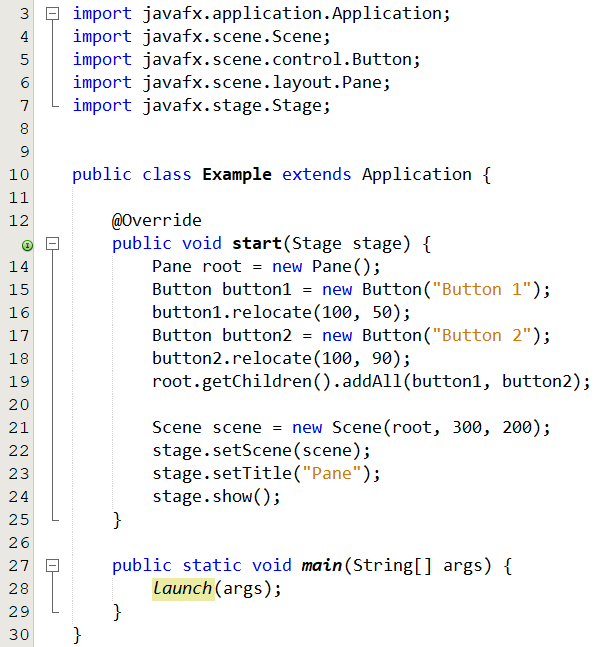
### 2.2.1 Tổng quan về lớp Pane

Lớp Pane là một phần của JavaFX. Nó là một ngăn bố cục cho phép chứa tập hợp các node con và đồng thời nó cũng là lớp cơ sở cho các ngăn bố cục khác. Các phương thức của lớp Pane thường sử dụng được thể hiện qua bảng sau:

|  |  |
| --- | --- |
| Pane() | Tạo một đối tượng Pane mới. |
| Pane(Node… c) | Tạo một đối tượng Pane mới với các node được chỉ định. |
| getChildren() | Trả về danh sách các phần tử con đang được chứa trong Pane. |
| setLayoutX(double v) | Đặt giá trị của thuộc tính layoutX. |
| setLayoutY(double v) | Đặt giá trị của thuộc tính layoutY. |
| getLayoutX() | Trả về thuộc tính layoutX |
| getLayoutY() | Trả về thuộc tính layoutY |
| setPrefSize(double width, double height) | Đặt kích thước ưu thích của Pane. |
| ... | |

### 2.2.2 Chương trình minh họa sử dụng lớp Pane

*Hình 6: Giao diện mình họa sử dụng lớp Pane*

*Hình 7: Chương trình mình họa lớp Pane*

***Chú ý****; Các node con được đặt trong Pane có thể sử dụng phương thức relocate(double x, double y) để di chuyển đến tọa độ xác định trong Pane.*

## 2.3 GridPane

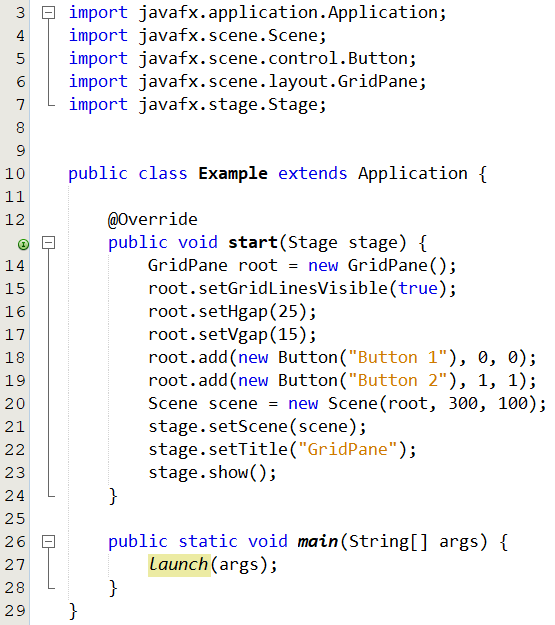
### 2.3.1 Tổng quan về lớp GridPane

GridPane là một là một ngăn bố cục, nó chia bề mặt của nó thành một lưới như một ma trận 2 chiều bao gồm các hàng và các cột. Mỗi thành phần node con có thể được đặt vào ngăn chứa thông qua chỉ số hàng và cột được chỉ định. Biểu đồ lớp cho GridPane được thể hiện trong bảng sau đây.

|  |  |
| --- | --- |
| **javafx.scene.layout.GridPane** | **Miêu tả** |
| - hgap: DoubleProperty | Khoảng cách ngang giữa các node (mặc định: 0). |
| - vgap: DoubleProperty | Khoảng cách dọc giữa các node (mặc định: 0). |
| - alignment: ObjectProperty<Pos> | Căn chỉnh tổng thể nội dung trong ngăn chứa (mặc định: Pos.LEFT). |
| - gridLinesVisible:  BooleanProperty | Đường viền của lưới có được nhìn thấy không (mặc định: false). |
| + GridPane() | Tạo một GridPane. |
| + add(child: Node, columnIndex:  int, rowIndex: int): void | Thêm một node vào một cột, hàng cụ thể. |
| + addColumn(columnIndex: int,  children: Node...): void | Thêm nhiều node vào một cột cụ thể. |
| + addRow(rowIndex: int,  children: Node...): void | Thêm nhiều node vào một hàng cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 2.3.2 Chương trình minh họa sử dụng lớp GridPane

*Hình 8: Giao diện minh họa sử dụng lớp GridPane*

*Hình 9: Chương trình minh họa lớp GridPane*

## 2.4 StackPane

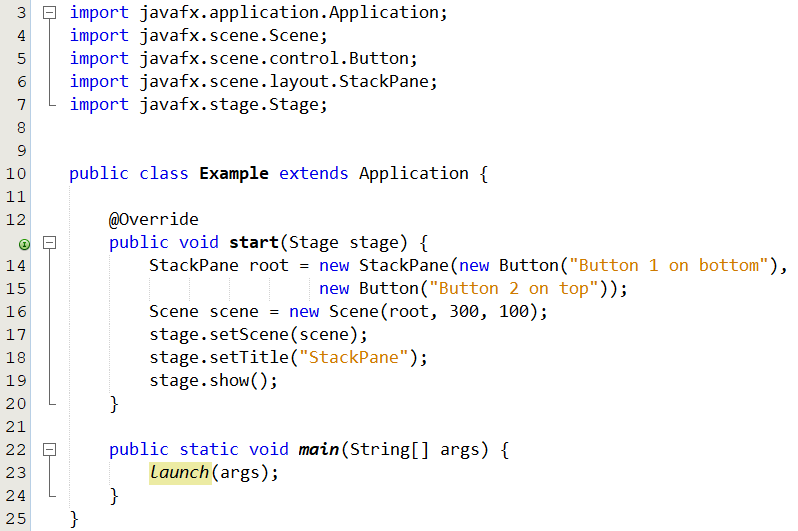
### 2.4.1 Tổng quan về lớp StackPane

Lớp StackPane là một phần của JavaFX, nó bố chí các node con của nó dưới dạng một ngăn xếp duy nhất và node mới sẽ được đặt trên đỉnh của node trước đó trong StackPane. Các phương thức thường dùng trong StackPane được thể hiện trong bảng dưới dây.

|  |  |
| --- | --- |
| StackPane () | Tạo một StackPane |
| StackPane(Node... children) | Tạo một đối tượng StackPane và thêm các node con vào ngăn chứa. |
| getAlignment() | Trả về kiểu căn chỉnh theo Pos của StackPane. |
| getAlignment(Node child) | Là một phương thức static của lớp StackPane, nó trả về kiểu căn chính theo Pos của node con trong StackPane. |
| setAlignment(Node n, Pos v) | Là một phương thức static của lớp StackPane, nó căn chính node con trong StackPane theo Pos. |
| setMargin(Node child, Insets v) | Là một phương thức static của lớp StackPane, nó căn lề cho node con trong StackPane theo Insets. |
| getMargin(Node c) | Là một phương thức static của lớp StackPane, nó trả về kiểu căn lề Insets của node con trong |
| setAlignment(Pos v) | Đặt nội dung cho thuộc tính alignment cúa StackPane theo Pos để căn chỉnh StackPane trong khung hiển thị (mặc định: Pos.CENTER) |

### 2.4.2 Chương trình minh họa sử dụng lớp StackPane

*Hình 10: Giao diện minh họa lớp StackPane*

*Hình 11: Chương trình mình họa sử dụng lớp StackPane*

## 2.5 BorderPane

### 2.5.1 Tổng quan về lớp BorderPane

BorderPane là loại ngăn chứa có 5 vùng riêng biệt là top, left, right, bottom và center. Các thuộc tính và phương thức thường dùng của BorderPaen được thể hiện dưới bảng sau.

|  |  |
| --- | --- |
| **javafx.scene.layout.BorderPane** | **Miêu tả** |
| - top: ObjectProperty<Node> | Node được đặt ở vùng top (mặc định: null). |
| - right: ObjectProperty<Node> | Node được đặt ở vùng right (mặc định: null). |
| - bottom: ObjectProperty<Node> | Node được đặt ở vùng bottom (mặc định: null). |
| - center: ObjectProperty<Node> | Node được đặt ở vùng center (mặc định: null). |
| - left: ObjectProperty<Node> | Node được đặt ở vùng left (mặc định: null). |
| + BorderPane() | Tạo một BorderPane. |
| + BorderPane(node: Node) | Tạo một BorderPane với Node được đặt ở vùng center. |
| + BorderPane(Node center, Node top, Node right, Node bottom, Node left) | Tạo một BorderPane với các 5 Node nhất định được thêm vào năm vùng tương ứng. |
| + setAlignment (Node child, Pos v) | Là một phương thức static của lớp BorderPane. Nó đặt căn chỉnh Node child theo Pos v trong BorderPane. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

Đặc điểm của các vùng:

* Vùng Top/Bottom: Có thể co/giãn theo chiều ngang và giữ nguyên chiều thẳng đứng.
* Vùng Left/Right: Có thể co/giãn theo chiều thẳng đứng và giữ nguyên chiều ngang.
* 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.

### 2.5.2 Chương trình mình họa sử dụng lớp BorderPane

*Hình 12: Giao diện mình họa sử dụng lớp BorderPane*

 *Hình 13: Chương trình minh họa lớp BorderPane*

## 2.6 FlowPane

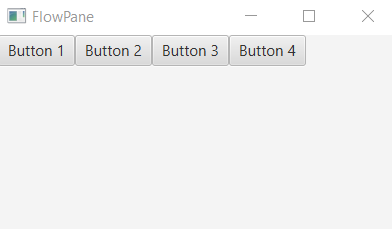
### 2.6.1 Tổng quan về lớp FlowPane

FlowPane sắp xếp các node con trong ngăn chứa theo chiều ngang từ trái sang phải hoặc theo chiều dọc từ trên xuống dưới, theo thứ tự mà chúng đã được thêm vào. Khi không còn không gian hiển thị các node con trên một hàng hoặc một cột, các node con sẽ tự sang một hàng hoặc một cột mới. Các thuộc tính và các phương thức thường dùng được thể hiện trong bảng sau đây.

|  |  |
| --- | --- |
| **javafx.scene.layout.FlowPane** | **Miêu tả** |
| - hgap: DoubleProperty | Khoảng cách ngang giữa các node con (mặc định: 0). |
| - vgap: DoubleProperty | Khoảng cách dọc giữa các node con (mặc định: 0). |
| - alignment: ObjectProperty<Pos> | Căn chỉnh tổng thể các node con trong ngăn chứa (mặc định: Pos.LEFT). |
| - orientation: ObjectProperty<Orientation> | Hướng của ngăn chứa theo chiều ngang hay dọc (mặc định: Orientation.HORIZONTAL). |
| **+** FlowPane() | Tạo một FlowPane mặc định. |
| + FlowPane(Node… c) | Tạo một FlowPane với các node con được chỉ định. |
| + FlowPane(hgap: double, vgap: double) | Tạo một FlowPane với khoảng cách ngang và dọc cụ thể. |
| + FlowPane(double h, double v, Node… c) | Tạo một FlowPane mới, với khoảng cách ngang, dọc và các node con được chỉ định. |
| + FlowPane(orientation:  ObjectProperty<Orientation>) | Tạo một FlowPane với một hướng cụ thể. |
| + FlowPane(orientation:  ObjectProperty<Orientation>, Node...c) | Tạo một FlowPane với một hướng cụ thể và các node con được chỉ định. |
| + FlowPane(orientation:  ObjectProperty<Orientation>,  hgap: double, vgap: double) | Tạo một FlowPane với một hướng, khoảng cách ngang và dọc cụ thể. |
| + FlowPane(orientation:  ObjectProperty<Orientation>,  hgap: double, vgap: double, Node... c) | Tạo một FlowPane với một hướng, khoảng cách ngang, dọc cụ thể và các node con được chỉ định. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

*Chú ý: Việc quy định trình tự các node con theo chiều ngang hoặc dọc thông qua thuộc tính orientation của lớp FlowPane. Ta có thể đặt giá trị cho thuộc tính này thông qua phương thức* ***setOrientation(Orientation o)*** *với* ***o*** *ở đây có thể là một trong hai hằng số* ***Orientation.HORIZONTAL*** *tương ứng với chiều ngang và* ***Orientation.VERTICAL*** *tương ứng với chiều dọc.*

### 2.6.2 Chương trình mình họa sử dụng lớp FlowPane

*Hình 14: Chương trình minh họa lớp FlowPane*

*Hình 15: Giao diện minh họa lớp FlowPane*

## 2.7 HBox

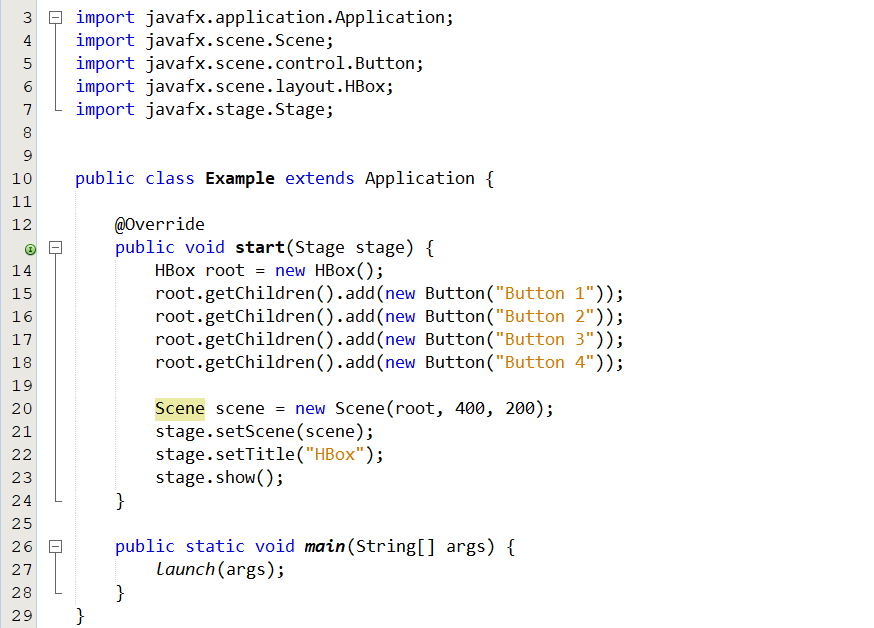
### 2.7.1 Tổng quan về lớp HBox

HBox là một thành phần của JavaFX, nó là một ngăn chứa bố trí các thành phần của nó thành một hàng ngang duy nhất. Càc thuộc tính và các phương thức thường sử dụng được thể hiện dưới bảng sau đây.

|  |  |
| --- | --- |
| **javafx.scene.layout.** **HBox** | **Miêu tả** |
| - spacing: DoubleProperty | Khoảng cách ngang giữa các node con (mặc định: 0). |
| - fillHeight: BooleanProperty | Node con có thể thay đổi kích thước bằng chiều cao của ngăn chứa HBox (mặc định: true). |
| - alignment: ObjectProperty<Pos> | Căn chỉnh tổng thể các node con trong ngăn chứa (mặc định: Pos.LEFT). |
| + HBox() | Tạo một HBox mặc định |
| + HBox(spacing: double) | Tạo một HBox với khoảng cách ngang cụ thể giữa các node. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 2.7.2 Chương trình minh họa sử dụng lớp HBox

*Hình 16: Giao diện minh họa lớp HBox*

*Hình 17: Chương trình minh họa lớp HBox*

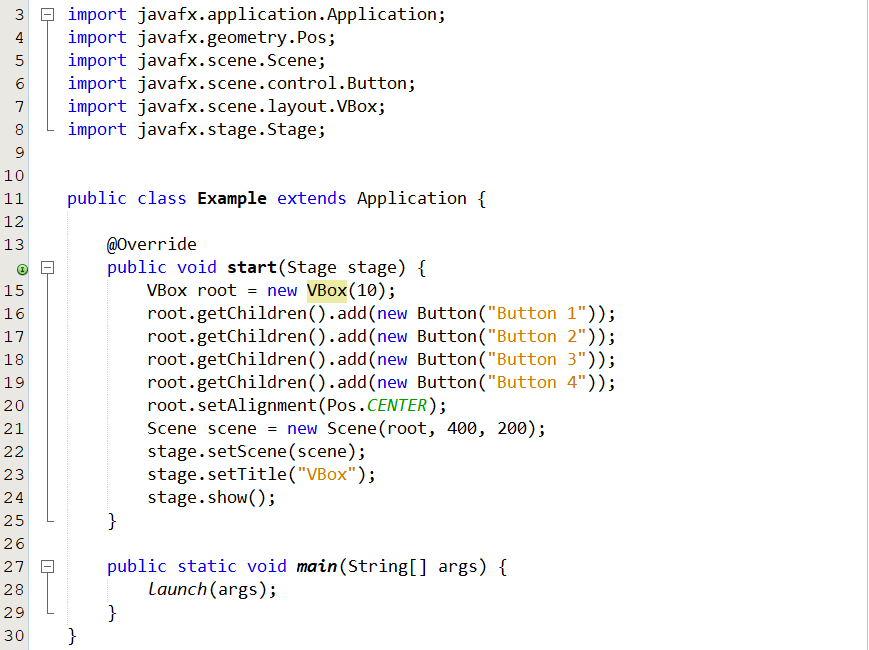
## 2.8 VBox

### 2.8.1 Tổng quan về lớp VBox

VBox là một thành phần của JavaFX, nó là một ngăn chứa bố trí các thành phần của nó thành một hàng dọc duy nhất. Càc thuộc tính và các phương thức thường sử dụng được thể hiện dưới bảng sau đây.

|  |  |
| --- | --- |
| **javafx.scene.layout.** **HBox** | **Miêu tả** |
| - spacing: DoubleProperty | Khoảng cách dọc giữa các node con (mặc định: 0). |
| - fillWidth: BooleanProperty | Node con có thể thay đổi kích thước bằng chiều rộng của ngăn chứa VBox (mặc định: true). |
| - alignment: ObjectProperty<Pos> | Căn chỉnh tổng thể các node con trong ngăn chứa (mặc định: Pos.LEFT). |
| + VBox() | Tạo một VBox mặc định |
| + VBox(spacing: double) | Tạo một VBox với khoảng cách ngang cụ thể giữa các node. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

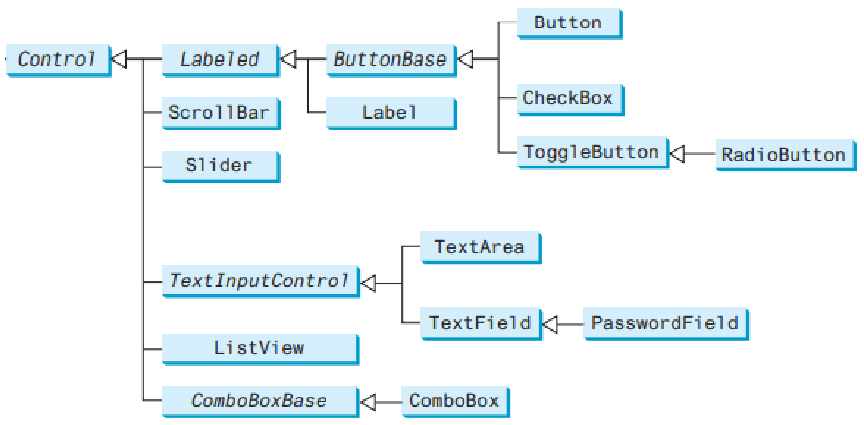
### 2.8.2 Chương trình minh họa sử dụng lớp VBox

*Hình 18: Giao diện minh họa lớp VBox*

*Hình 19: Chương trình minh họa lớp VBox*

# 3. CÁC UI CONTROL TRONG JAVAFX

## 3.1 Giới thiệu chung về các UI Control trong JavaFX

JavaFX cung cấp nhiều các UI Control (điều khiển giao diện người dùng) để phát triển các giao diện người dùng toàn diện. Giao diện người dùng đồ họa (GUI - graphical user interface) làm cho chương trình thân thiện và dễ sử dụng. Tạo GUI đòi hỏi sự sáng tạo và kiến thức về cách hoạt động của các UI control. Vì các UI control trong JavaFX rất linh hoạt và đa dạng, ta có thể tạo nhiều loại giao diện người dùng hữu ích cho các ứng dụng GUI phong phú. Các lớp UI control đều nằm trong gói **javafx.scene.control** và mối quan hệ của chúng được thể hiện trong sơ đồ dưới đây.

*Hình 20: Mối quan hệ của các UI Control trong JavaFX*

## 3.2 Label

### 3.2.1 Tổng quan về lớp Label

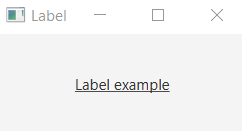
Label là một phần của JavaFX. Label được sử dụng để hiện thị một văn bản ngắn hoặc một hình ảnh hoặc cả hai. Label kế thừa lớp Labeled. Các thuộc tính và phương thức của lớp Labeled và Label thể hiện qua bảng sau.

|  |  |
| --- | --- |
| **javafx.scene.control.Labeled** | **Miêu tả** |
| - underline: BooleanProperty | Văn bản có được gạch chân không. |
| - text: StringProperty | Nội dung của văn bản được hiển thị. |
| - textFill: ObjectProperty<Paint> | Màu của văn bản được hiển thị. |
| - graphic: ObjectProperty<Node> | Hình ảnh được hiển thị. |
| - alignment: ObjectProperty<Pos> | Căn chỉnh của văn bản và node được hiển thị. |
| - graphicTextGap: DoubleProperty | Khoảng cách giữa hình ảnh và văn bản hiển thị |
| - wrapText: BooleanProperty | Văn bản có được bao bọc trong khung hiển thị không nếu nó vượt quá chiều rộng. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

|  |  |
| --- | --- |
| **javafx.scene.** **control.** **Label** | **Miêu tả** |
| + Label() | Tạo một Label trống. |
| + Label(text: String) | Tạo một Label với nội dung văn bản được cụ thể. |
| + Label(text: String, graphic: Node) | Tạo một Label với nội dung văn bản và hình ảnh cụ thể. |

### 3.2.2 Chương trình minh họa sử dụng lớp Label

*Hình 21: Chương trình minh họa sử dụng lớp Label*

* Hình 22: Giao diện minh họa lớp Label*

## 3.3 Button

### 3.3.1 Tổng quan về lớp Button

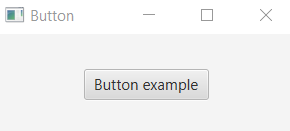
Button là một trình điều khiển kích hoạt sự kiện hành động khi được nhấp vào. Nó đại diện bởi lớp **javafx.scene.control.Button** vàcó thể hiển thị văn bản, hình ảnh hoặc cả hai. Các thuộc tính chung của Button được định nghĩa trong các lớp ButtonBase và Labeled. Các thuộc tính và phương thức của Button được thể hiện qua bảng sau đây.

|  |
| --- |
| **javafx.scene.** **control.** **Labeled** |

|  |  |
| --- | --- |
| **javafx.scene.** **control.** **ButtonBase** | **Miêu tả** |
| - onAction: ObjectProperty<EventHandler  <ActionEvent>> | Dịnh nghĩa một trình xử lý để xử lý hành động của một Button. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

|  |  |
| --- | --- |
| **javafx.scene.** **control.** **Button** | **Miêu tả** |
| + Button() | Tạo một Button trống. |
| + Button(text: String) | Tạo một Button với nội dung căn vản cụ thể. |
| + Button(text: String, graphic: Node) | Tạo một Button với nội dung văn bản và hình ảnh cụ thể. |

### 3.3.2 Chương trình minh họa sử dụng lớp Button

*Hình 23: Chương trình minh họa lớp Button*

*Hình 24: Giao diện minh họa lớp Button*

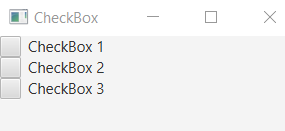
## 3.4 CheckBox

### 3.4.1 Tổng quan về lớp CheckBox

CheckBox được đại diện bởi lớp **javafx.scene.control.CheckBox**,là một phần của JavaFX, nó là một hộp được được đánh dấu khi nó được chọn và trống khi không được chọn. Cũng giống như Button, lớp CheckBox kế thừa từ lớp ButtonBase. Các thuộc tính và phương thức được thể hiện trong hình sau đây.

|  |  |
| --- | --- |
| **javafx.scene.control.CheckBox** | **Miêu tả** |
| - selected: BooleanProperty | Cho biết CheckBox có được chọn hay không. |
| + CheckBox() | Tạo một CheckBox trống. |
| + CheckBox(text: String) | Tạo một CheckBox với một văn bản cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 3.4.2 Chương trình minh họa sử dụng lớp CheckBox

*Hình 25: Chương trình minh họa lớp CheckBox*

*Hình 26: Giao diện minh họa lớp CheckBox*

## 3.5 RadioButton

### 3.5.1 Tổng quan về lớp RadioButton

RadioButton là một lớp kế thừa từ ToggleButton. Nó có hai trạng thái là chọn hoặc không chọn. Khi các RadioButton được kết hợp vào trong một nhóm, tại một thời điểm chỉ có duy nhất một RadioButton được chọn. Các thuộc tính và phương thức của RadioButton và ToggleButton được thể hiện trong bảng sau.

|  |  |
| --- | --- |
| **javafx.scene.** **control.** **ToggleButton** | **Miêu tả** |
| - selected: BooleanProperty | Cho biết ToggleButton có được chọn hay không. |
| - toggleGroup:  ObjectProperty<ToggleGroup> | Kết hợp các node thành một nhóm. |
| + ToggleButton() | Tạo một ToggleButton trống. |
| + ToggleButton(text: String) | Tạo một ToggleButton với một văn bản cụ thể. |
| + ToggleButton(text: String, graphic: Node) | Tạo một ToggleButton với một văn bản và hình ảnh cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

|  |  |
| --- | --- |
| **javafx.scene.** **control.** **Button** | **Miêu tả** |
| + RadioButton() | Tạo một RadioButton trống. |
| + RadioButton(text: String) | Tạo một RadioButton với một văn bản cụ thể. |

*Chú ý: Để kết hợp các RadioButton thành một nhóm, ta cần tạo ra một phiên bản của ToggleGroup và gọi phướng thức setToggleGroup(ObjectProperty<ToggleGroup>) để nhóm các RadioButton với nhau. Nếu không có nhóm thì các RadioButton sẽ độc lập với nhau.*

### 3.5.2 Chương trình minh họa sử dụng lớp RadioButton

*******Hình 27: Giao diện minh họa lớp RadioButton*

*Hình 28: Chương trình minh họa lớp RadioButton*

## 3.6 TextField

### 3.6.1 Tổng quan về lớp TextField

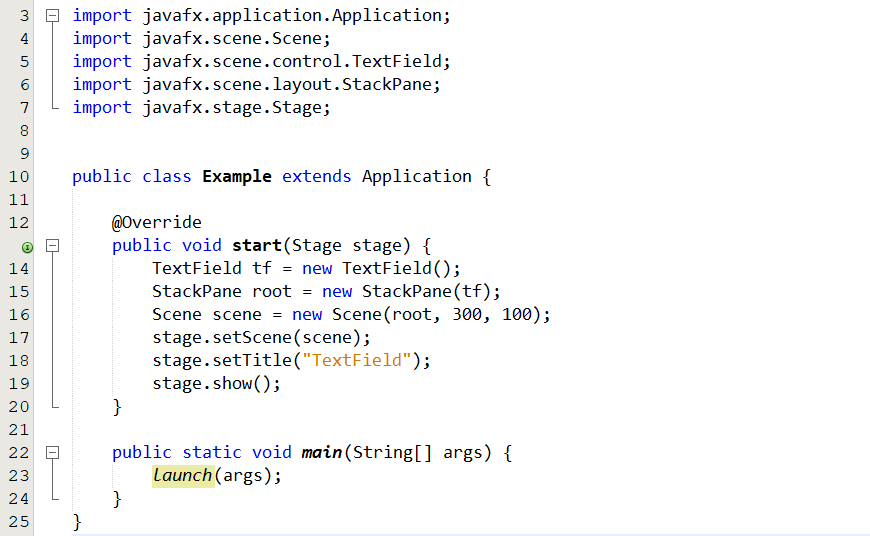
Lớp TextField là một thành phần của JavaFX, nó cho phép người dùng nhập một dòng văn bản vào ô nhập. TextField là một lớp con của TextInputControl và được đại diện bởi lớp **javafx.scene.control.TextField**. Các thuộc tính và phương thức của lớp TextField và TextInputControl được thể hiện trong bảng dưới đây.

|  |  |
| --- | --- |
| **javafx.scene.** **control.** **TextInputControl** | **Miêu tả** |
| - text: StringProperty | Nội dung của văn bản. |
| - editable: BooleanProperty | Cho biết liệu người dùng có thể chỉnh sửa được nội dung của văn bản hay không. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

|  |  |
| --- | --- |
| **javafx.scene.** **control.** **TextField** | **Miêu tả** |
| - alignment: ObjectProperty<Pos> | Cách căn chỉnh văn bản trong TextField. |
| - prefColumnCount: IntegerProperty | Chỉ định số cột văn bản ưu tiên trong TextField. |
| - onAction:  ObjectProperty<EventHandler<ActionEvent>> | Chỉ định trình xử lý để xử lý sự kiện hành động trên TextField |
| + TextField() | Tạo một TextField trống |
| + TextField(text: String) | Tạo một TextField với một văn bản cụ thể |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 3.6.2 Chương trình minh họa sử dụng lớp TextField

*Hình 29: Giao diện mình họa lớp TextField*

*Hình 30: Chương trình minh họa lớp TextField*

## 3.7 TextArea

### 3.7.1 Tổng quan về lớp TextArea

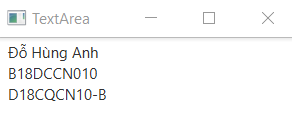
TextArea là một thành phần của JavaFX, nó cho phép người dùng nhập nhiều dòng văn bản trong ô nhập. Cũng như lớp TextField, lớp TextArea cũng kế thừa từ lớp TextInputControl. Các thuộc tính và phương thức của lớp TextArea được thể hiện trong bảng sau.

|  |
| --- |
| **javafx.scene.** **control.** **TextInputControl** |

|  |  |
| --- | --- |
| **javafx.scene.** **control.** **TextArea** | **Miêu tả** |
| - prefRowCount: IntegerProperty | Chỉ định số hàng văn bản được ưu tiên trong TextField. |
| - prefColumnCount: IntegerProperty | Chỉ định số cột văn bản được ưu tiên trong TextField. |
| - wrapText: BooleanProperty | Xác định xem văn bản có được bao quanh dòng tiếp theo hay không. |
| + TextArea() | Tạo một TextArea trống |
| + TextArea(text: String) | Tạo một TextArea với một văn bản cụ thể |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 3.7.2 Chương trình minh họa sử dụng lớp TextArea

*Hình 31: Chương trình minh họa lớp TextArea*

*Hình 32: Giao diện mình họa lớp TextArea*

## 3.8 ComboBox

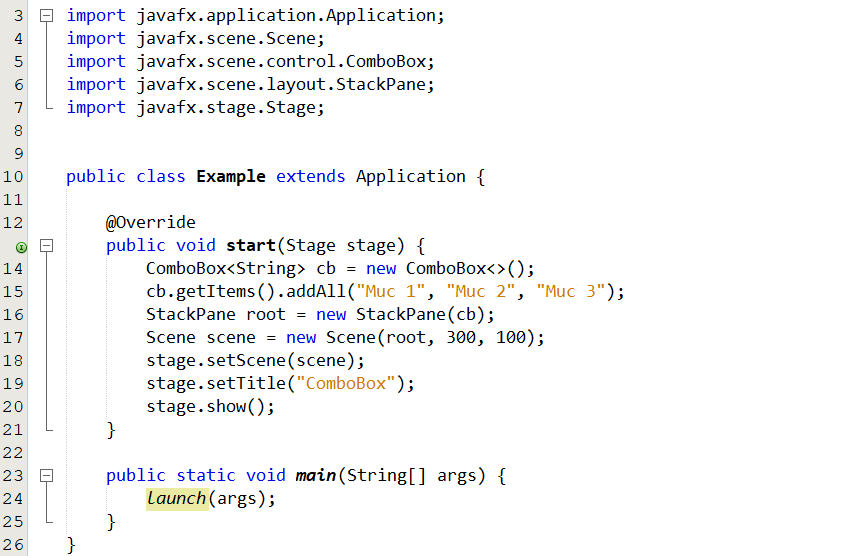
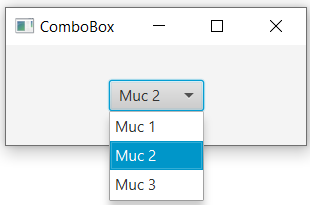
### 3.8.1 Tổng quan về lớp ComboBox

ComboBox là một thành phần điều khiển của JavaFX. Nó cho phép người dùng chọn một trong nhiều lựa chọn. Khi người dùng click vào ComboBox, một danh sách các lựa chọn sẽ hiện ra cho người dùng lựa chọn. ComboBox kế thừa từ lớp ComboBoxBase và nó được định nghĩa là một lớp generic giống như lớp ArrayList. Các thuộc tính và phương thức của lớp ComboBox và lớp ComboBoxBase sẽ được thể hiện trong bảng sau đây.

|  |  |
| --- | --- |
| **javafx.scene.** **control.** **ComboBoxBase<T>** | **Miêu tả** |
| - value: ObjectProperty<T> | Giá trị được chọn trong ComboBox. |
| - editable: BooleanProperty | Chỉ định xem ComboBox có cho phép người dùng nhập hay không. |
| - onAction:  ObjectProperty<EventHandler<ActionEvent>> | Chỉ định trình xử lý để sử lý sự kiện hành động. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

|  |  |
| --- | --- |
| **javafx.scene.** **control** **.ComboBox<T>** | **Miêu tả** |
| - visibleRowCount: IntegerProperty | Số hàng hiển thị tối đa của của các mục khi ComboBox bật ra. |
| - items: ObjectProperty<ObservableList<T>> | Các mục trong ComboBox. |
| + ComboBox() | Tạo một ComboBox trống |
| + ComboBox(items: ObservableList<T>) | Tạo một ComboBox với các mục cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 3.8.2 Chương trình minh họa sử dụng lớp ComboBox

***Hình 33: Chương trình minh họa lớp ComboBox*

*Hình 34: Chương trình minh họa lớp ComboBox*

## 3.9 ListView

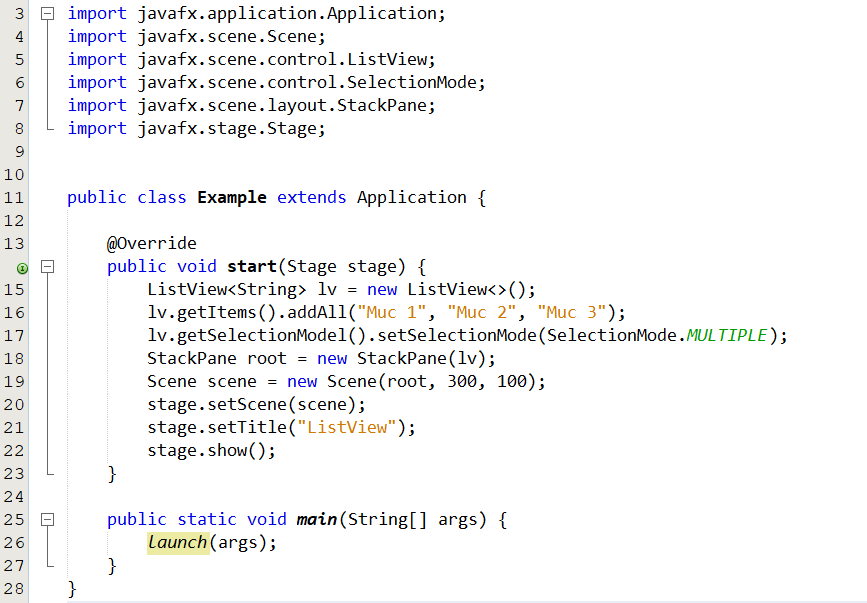
### 3.9.1 Tổng quan về lớp ListView

ListView về cơ bản thực hiện chức năng giống như một ComboBox, nó có hai lựa chọn cho phép người dùng chọn một mục duy nhất hoặc chọn cùng lúc nhiều mục. ListView được định nghĩa là lớp generic giống như lớp ArrayList. Các thuộc tính và phương thức của ListView được thể hiện trong bảng sau.

|  |  |
| --- | --- |
| **javafx.scene.** **control.ListView<T>** | **Miêu tả** |
| - items: ObjectProperty<ObservableList<T>> | Các mục trong ListView. |
| - orientation: BooleanProperty | ListView được hiển thị theo chiều ngang hay chiều dọc. |
| selectionModel: ObjectProperty<MultipleSelectionModel<T>> | Đặt chế độ lựa chọn là chỉ chọn một mục duy nhất (SelectionMode.SINGLE) hay chọn nhiều mục (SelectionMode.MULTIPLE). |
| + ListView() | Tạo một ListView trống. |
| + ListView(items: ObservableList<T>) | Tạo một ListView với các mục cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 3.9.2 Chương trình minh họa sử dụng lớp ListView

*Hình 35: Giao diện minh họa lớp ListView*

*Hình 36: Chương trình minh họa lớp ListView*

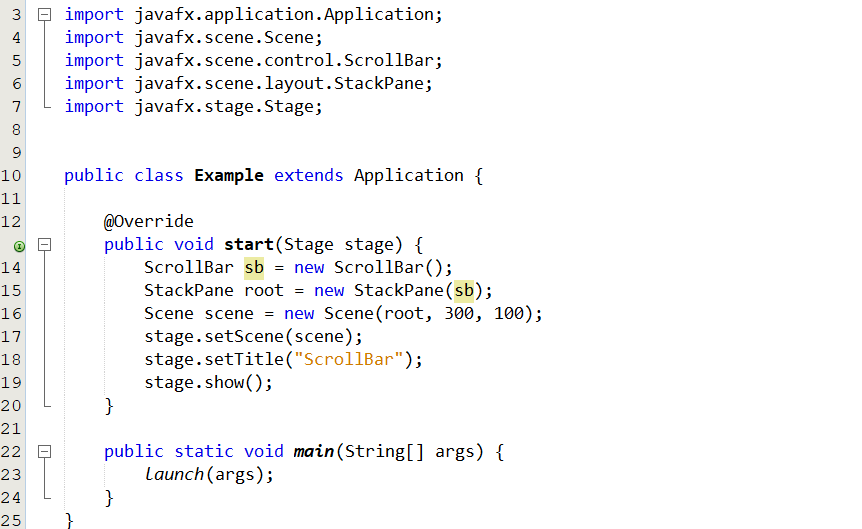
## 3.10 ScrollBar

### 3.10.1 Tổng quan về lớp ScrollBar

ScrollBar là một phần của JavaFX được sử dụng để cung cấp một thanh cuộn cho người dùng để người dùng có thể cuộn xuống các trang ứng dụng. Nó được đại diện bởi lớp javafx.scene.control.ScrollBar. Các phương thức và thuộc tính của ScrollBar được thể hiện dưới bảng sau.

|  |  |
| --- | --- |
| **javafx.scene.** **control.ScrollBar** | **Miêu tả** |
| - max: DoubleProperty | Giá trị lớn nhất của thanh cuộn (mặc định: 100). |
| - min: DoubleProperty | Giá trí nhỏ nhất của thanh cuộn (mặc định: 0). |
| - unitIncrement: DoubleProperty | Đơn vị tăng/giảm của thanh cuộn (mặc định: 1). |
| - value: DoubleProperty | Giá trị hiện tại của thanh cuộn (mặc định: 0). |
| - visibleAmount: DoubleProperty | Chiều rộng của thanh cuộn (mặc định: 15). |
| - orientation: ObjectProperty<Orientation> | Hướng của thanh cuộn ngang hay dọc (mặc định: HORIZONTAL – ngang). |
| - blockIncrement: DoubleProperty | Giá trị tăng thêm khi nhấp vào thanh cuộn (mặc định: 10). |
| + ScrollBar() | Tạo một ScrollBar nằm ngang. |
| + increment() | Tăng giá trị của ScrollBar theo unitIncrement. |
| + decrement() | Giảm giá trị của ScrollBar theo unitIncrement. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 3.10.2 Chương trình minh họa sử dụng lớp ScrollBar

*Hình 37: Giao diện minh họa lớp ScrollBar*

*Hình 38: Chương trình minh họa lớp ScrollBar*

## 3.11 Slider

### 3.11.1 Tổng quan về lớp Slider

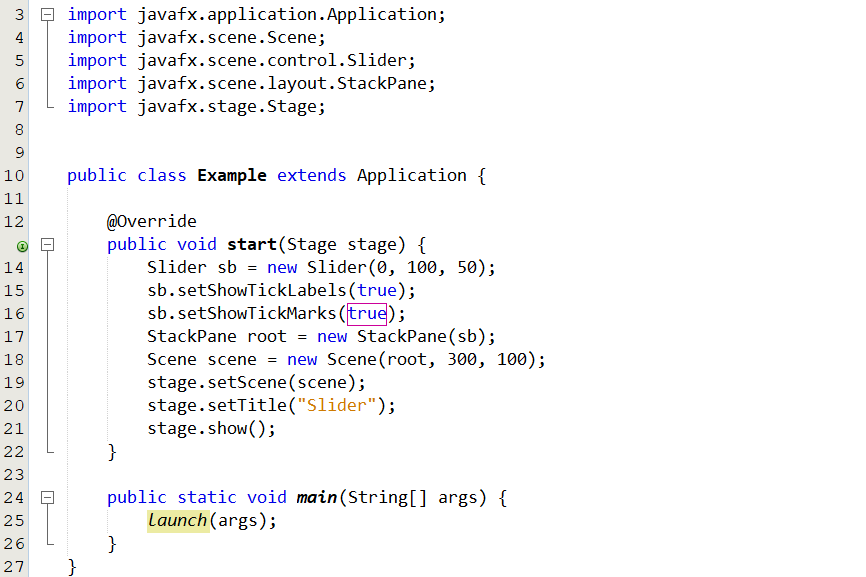
Slider là một thành phần của JavaFX, nó cho phép người dùng di chuyển cần gạt trên thanh giá trị để chọn . Slider được đại diện bởi lớp javafx.scene.control.Slider. Các thuộc tính và phương thức của lớp Slider được thể hiện trong bảng dưới đây.

|  |  |
| --- | --- |
| **javafx.scene.** **control.** **Slider** | **Miêu tả** |
| - max: DoubleProperty | Giá trị lớn nhất của thanh trượt (mặc định: 100). |
| - min: DoubleProperty | Giá trí nhỏ nhất của thanh trượt (mặc định: 0). |
| - value: DoubleProperty | Giá trị hiện tại của thanh trượt (mặc định: 0). |
| - majorTickUnit: DoubleProperty | Khoảng cách đơn vị giữa các dấu tích chính. |
| - minorTickCount: IntegerProperty | Khoảng cách đơn vị giữa các dấu tích phụ. |
| - showTickMarks: BooleanProperty | Chỉ định các dấu tích có được hiển thị không. |
| - showTickLabels: BooleanProperty | Chỉ định các số dưới dấu tích có được hiển thị không. |
| - orientation: ObjectProperty<Orientation> | Hướng của thanh trượt ngang hay dọc (mặc định: HORIZONTAL – ngang). |
| - blockIncrement: DoubleProperty | Giá trị tăng thêm khi nhấp vào thanh trượt (mặc định: 10). |
| + Slider() | Tạo một Slider mặc định theo chiều ngang. |
| + Slider(min: double, max: double,  value: double) | Tạo một Slider với giá trị các thuộc tính min, max, value cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

*Lưu ý: Giá trị cùa ScrollBar theo chiều dọc tăng từ trên xuống dưới, nhưng giá trị của Slider theo chiều dọc giảm từ trên xuống dưới.*

### 3.11.2 Chương trình minh họa sử dụng lớp Slider

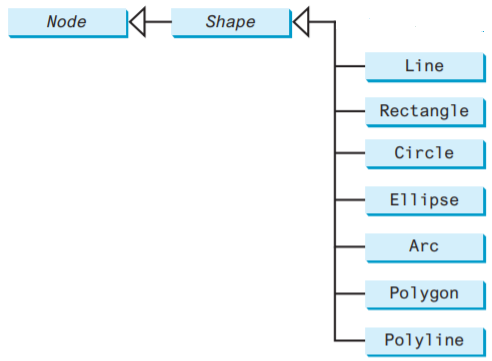
*Hình 39: Giao diện minh họa lớp Slider*



*Hình 40: Chương trình minh họa lớp Slider*

# 4. NHỮNG HÌNH DẠNG HAI CHIỀU (2D SHAPE) TRONG JAVAFX

## 4.1 Giới thiệu chung về những hình dạng 2 chiều trong JavaFX

** Trong một số ứng dụng, ta cần hiển thị hình dạng hai chiều (2D) cho người dùng. JavaFX cung cấp nhiều lớp khác nhau để triển khai hình dạng 2D và các lớp này nằm trong gói **javafx.scene.shape**. Lớp **javafx.scene.shape.Shape** là lớp cơ sở cho tất cả các lớp hình dạng 2D trong JavaFX. Biểu đồ thể hiện các mối quan hệ giữa các lớp trong gói **javafx.scene.shape** được hiển thị trong hình sau.

*Hình 41: Mối quan hệ giữa các lớp trong gói javafx.scene.shape*

## 4.2 Shape

Lớp Shape là một lớp abstract, nó xác định các thuộc tính chung cho tất cả các lớp hình dạng 2D. Các thuộc tính và phương thức của lớp Shape được thể hiện trong bảng dưới đây.

|  |  |
| --- | --- |
| **javafx.scene.** **shape.** **Shape** | **Miêu tả** |
| - fill | Chỉ định một màu tô bên trong hình dạng 2D. |
| - stroke | Chỉ định một màu để vẽ đường viền của hình dạng 2D. |
| - strokeWidth | Chỉ định chiều trộng đường viền của một hình dạng 2D. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

## 4.3 Line

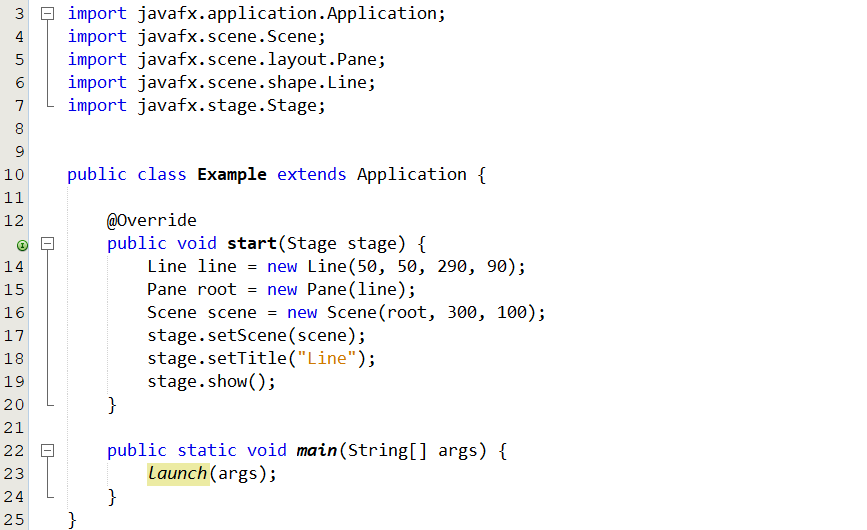
### 4.3.1 Tổng quan về lớp Line

Trong JavaFX, lớp Line sử dụng để vẽ ra một đoạn thằng, nó thế thừa lớp Shape. Các thuộc tính và phương thức của lớp Line được thể hiện trong bảng dưới đây.

|  |  |
| --- | --- |
| **javafx.scene.** **shape.** **Line** | **Miêu tả** |
| - startX: DoubleProperty | Tọa độ X của điểm bắt đầu đoạn thẳng. |
| - startY: DoubleProperty | Tọa độ Y của điểm bắt đầu đoạn thẳng. |
| - endX: DoubleProperty | Tọa độ X của điểm kết thúc đoạn thằng |
| - endY: DoubleProperty | Tọa độ Y của điểm kết thúc đoạn thẳng. |
| + Line() | Tạo một Line trống rỗng. |
| + Line(startX: double, startY:double, endX: double, endY:double) | Tạo một Line với tọa độ điểm bắt đầu và kết thúc cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 4.3.2 Chương trình minh họa sử dụng lớp Line

*Hình 42: Giao diện minh họa lớp Line*

*Hình 43: Chương trình minh họa lớp Line*

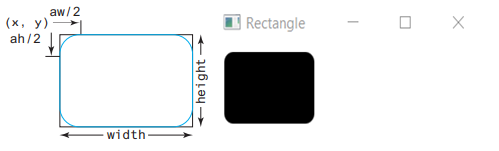
## 4.4 Rectangle

### 4.4.1 Tổng quan về lớp Rectangle

Lớp Rectangle định nghĩa một hình chữ nhật, nó thế thừa lớp Shape. Các thuộc tính và phương thức của lớp Rectangle được thể hiện trong bảng sau.

|  |  |
| --- | --- |
| **javafx.scene.** **shape.** **Rectangle** | **Miêu tả** |
| - width: DoubleProperty | Chiều rộng của hình chữ nhật (mặc định: 0). |
| - height: DoubleProperty | Chiều cao của hình chữ nhật (mặc định: 0). |
| - x: DoubleProperty | Tọa độ x của góc trên bên trái của hình chữ nhật (mặc định: 0). |
| - y: DoubleProperty | Tọa độ y của góc trên bên trái của hình chữ nhật (mặc định: 0). |
| - arcWidth: DoubleProperty | Đường kính ngang của các cung tròn ở góc (mặc định: 0, minh họa hình 44: bên trái). |
| - arcHeight: DoubleProperty | Đường kính dọc của các cung tròn ở góc (mặc định: 0, minh họa hình 44: bên trái). |
| + Rectangle() | Tạo một Rectangle màu đen mặc định. |
| + Rectangle(x: double, y:double, width: double,height: double) | Tạo một Rectangle với tọa độ điểm ở góc bên trái, chiều rộng, chiều cao cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 4.4.2 Chương trình minh họa sử dụng lớp Rectangle

*Hình 44: Chương trình minh họa lớp Rectangle*

*Hình 45: Giao diện minh họa lớp Rectangle*

## 4.5 Circle

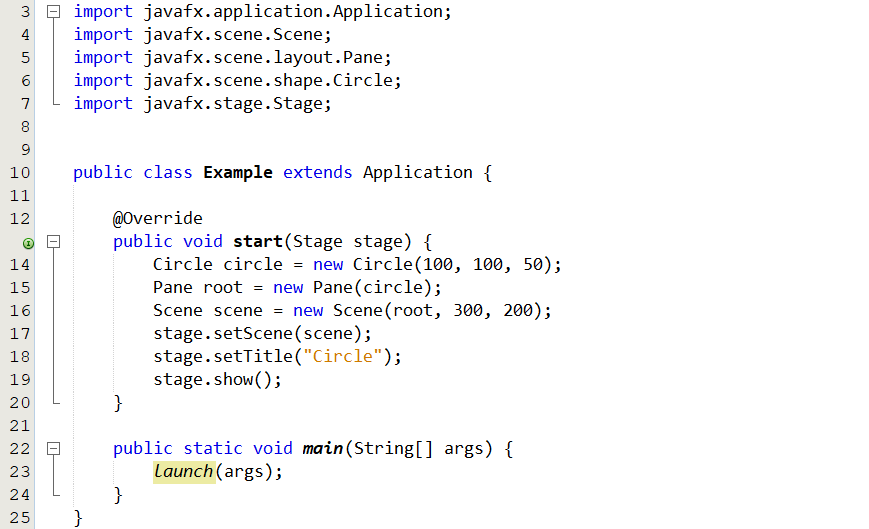
### 4.5.1 Tổng quan về lớp Circle

Lớp Circle định nghĩa một hình tròn, nó thế thừa lớp Shape. Các thuộc tính và phương thức của lớp Circle được thể hiện trong bảng sau.

|  |  |
| --- | --- |
| **javafx.scene.** **shape.** **Circle** | **Miêu tả** |
| - centerX: DoubleProperty | Tọa độ x của tâm hình tròn (mặc định: 0). |
| - centerY: DoubleProperty | Tọa độ y của tâm hình tròn (mặc định: 0). |
| - radius: DoubleProperty | Bán kính của hình tròn (mặc định: 0). |
| + Circle() | Tạo một Circle màu đen trống. |
| + Circle(x: double, y: double) | Tạo một Circle với tọa độ tâm cụ thể. |
| + Circle(x: double, y: double, radius: double) | Tạo một Circle với tọa độ tâm, độ dài bán kính cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 4.5.2 Chương trình minh họa lớp Circle

*Hình 46: Giao diện minh họa lớp Circle*

*Hình 47: Chương trình minh họa lớp Circle*

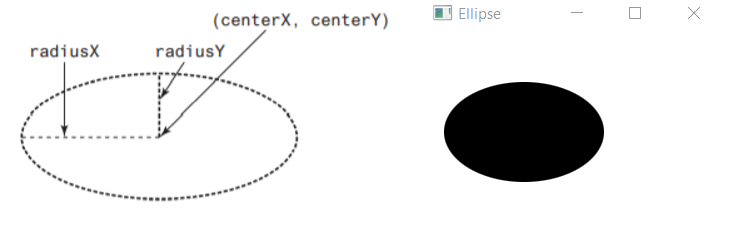
## 4.6 Ellipse

### 4.6.1 Tổng quan về lớp Ellipse

Lớp Ellipse định nghĩa một hình elip, nó thế thừa lớp Shape. Các thuộc tính và phương thức của lớp Ellipse được thể hiện trong bảng sau.

|  |  |
| --- | --- |
| **javafx.scene.** **shape.** **Ellipse** | **Miêu tả** |
| - centerX: DoubleProperty | Tọa độ x của tâm hình elip (mặc định: 0). |
| - centerY: DoubleProperty | Tọa độ y của tâm hình elip (mặc định: 0). |
| - radiusX: DoubleProperty | Bán kính ngang của hình elip (mặc định: 0). |
| - radiusY: DoubleProperty | Bán kính dọc của hình elip (mặc định: 0). |
| + Ellipse() | Tạo một Ellipse màu đen trống. |
| + Ellipse(x: double, y: double) | Tạo một Ellipse với tọa độ tâm cụ thể. |
| + Ellipse(x: double, y: double, radiusX: double, radiusY: double) | Tạo một Ellipse với tọa độ tâm, độ dài bán kính ngang và dọc cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 4.6.2 Chương trình minh họa lớp Ellipse

*Hình 48: Chương trình minh họa lớp Ellipse*

*Hình 49: Giao diện minh họa lớp Ellipse*

## 4.7 Arc

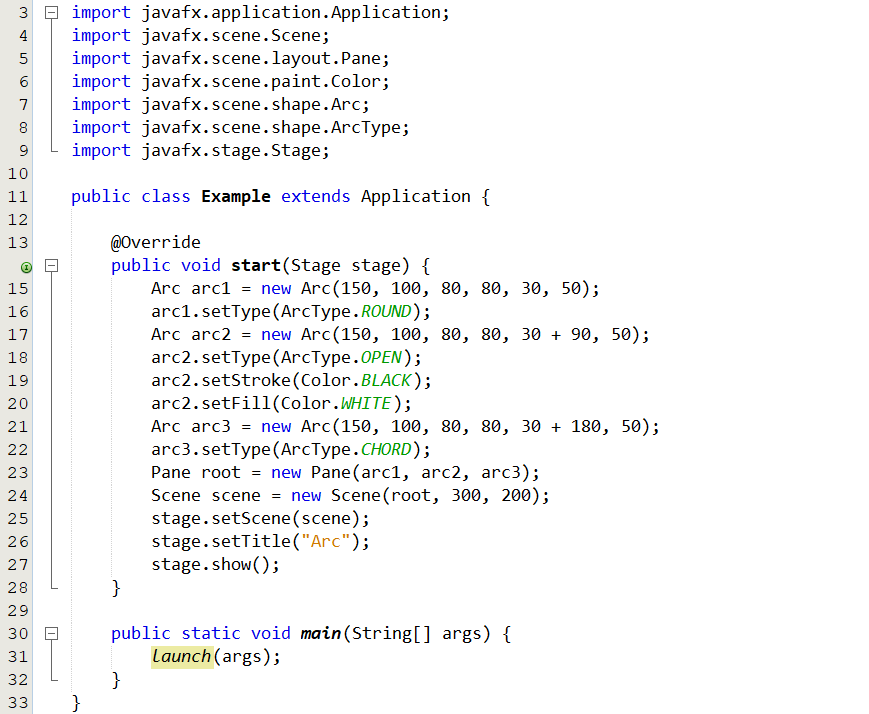
### 4.7.1 Tổng quan về lớp Arc

Lớp Arc định nghĩa một cung, nó được hình thành như một phần của hình elip, nó thế thừa lớp Shape. Các thuộc tính và phương thức của lớp Arc được thể hiện trong bảng sau.

|  |  |
| --- | --- |
| **javafx.scene.** **shape.** **Arc** | **Miêu tả** |
| - centerX: DoubleProperty | Tọa độ x của tâm elip (mặc định: 0). |
| - centerY: DoubleProperty | Tọa độ y của tâm elip (mặc định: 0). |
| - radiusX: DoubleProperty | Bán kính ngang của hình elip (mặc định: 0). |
| - radiusY: DoubleProperty | Bán kính dọc của hình elip (mặc định: 0). |
| - startAngle: DoubleProperty | Góc bắt đầu cung (tính bằng độ). |
| - length: DoubleProperty | Góc mở rộng (góc được bao phủ bởi cung). |
| - type: ObjectProperty<ArcType> | Kiểu đóng của cung xác định bởi các hằng số (ArcType.OPEN, ArcType.CHORD, ArcType.ROUND). Minh họa hình 49. |
| + Arc() | Tạo một Arc trống. |
| + Arc(x: double, y: double, radiusX: double, radiusY: double, startAngle: double, length: double) | Tạo một Arc với các đối số cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 4.7.2 Chương trình minh họa lớp Arc

*Hình 50: Giao diện minh họa lớp Arc*

*Hình 51: Chương trình minh họa lớp Arc*

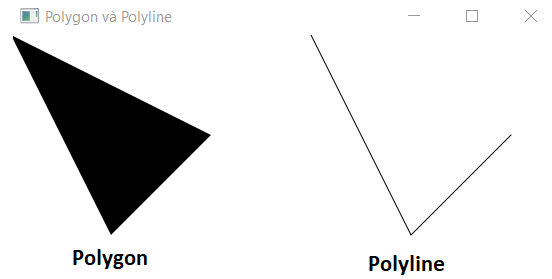
## 4.8 Polygon và Polyline

### 4.8.1 Tổng quan về lớp Polygon và lớp Polyline

Lớp Polygon và lớp Polyline thế thừa lớp Shape. Lớp Polygon định nghĩa một đa giác nối chuổi các điếm và nó tự động đóng. Lớp Polyline tương tự như lớp Polygon ngoại trừ việc nó không tự động đóng. Các phương thức thường dùng của lớp Polygon và Polyline được thể hiện trong bảng sau.

|  |  |
| --- | --- |
| **javafx.scene.** **shape. Polygon/Polyline** | **Miêu tả** |
| + Polygon()  + Polyline() | Tạo một Polygon.  Tạo một Polyline trống |
| + Polygon(double... points)  + Polyline(double... points) | Tạo một Polygon với các điểm đã cho.  Tạo một Polyline với các điểm đã cho. |
| + getPoints(): ObservableList<Double> | Trả về danh sách các giá trị double dưới dạng tọa độ x và y của các điểm. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 4.8.2 Chương trình minh họa lớp Polygon và lớp Polyline

*Hình 52: Chương trình minh họa lớp Polygon và lớp Polyline*

*Hình 53: Giao diện minh họa lớp Polygon và lớp Polyline*

# 5. MỘT SỐ LỚP THÔNG DỤNG TRONG JAVAFX

## 5.1 Giới thiệu chung về một số lớp thông dụng trong JavaFX

JavaFX cung cấp rất nhiều các gói cũng như các lớp để có thể hỗ trợ tốt nhất việc lập trình giao diện. Sau đây là một số lớp tiêu biểu và thường được sử dụng.

|  |  |
| --- | --- |
| **Lớp** | **Miêu tả** |
| Lớp Color | Được sử dụng để tạo màu sắc. |
| Lớp Text | Được sử dụng để hiển thị một văn bản ngắn. |
| Lớp Font | Được sử dụng để tạo phông chữ như kiểu, kích thước chữ. |
| Lớp Image | Được sử dụng để tải hình ảnh từ một file hoặc một URL. |
| Lớp ImageView | Được sử dụng để hiển thị hình ảnh lên cửa sổ giao diện. |
| Lớp Media | Được sử dụng để tải các video và audio từ file hoặc URL. |
| Lớp MediaPlayer | Được sử dụng để phát và điều khiển video hoặc audio. |
| Lớp MediaView | Được sử dụng để hiển thị và chạy video và audio. |

## 5.2 Lớp Color

### 5.2.1 Tổng quan về lớp Color

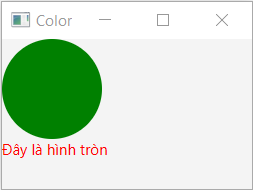
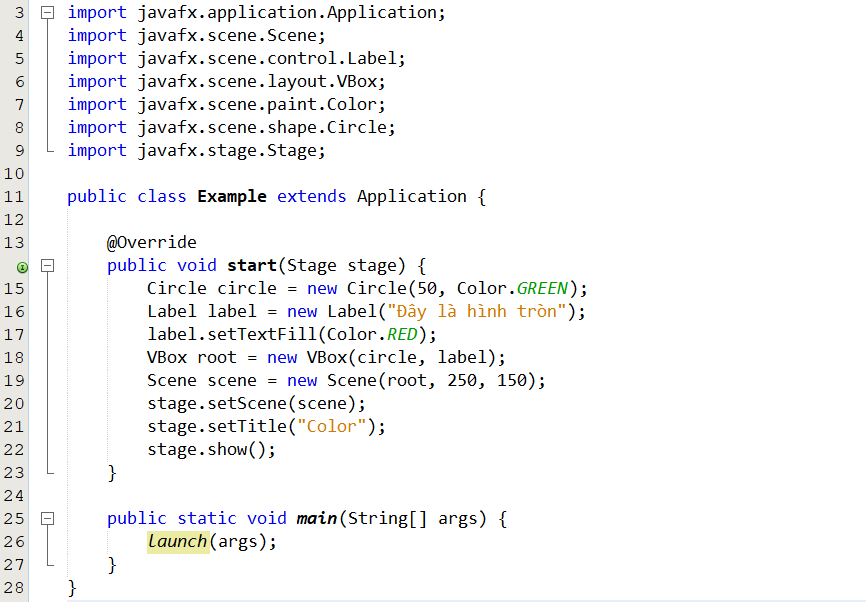
Lớp Color được sử dụng để tạo màu. JavaFX định nghĩa một lớp abstract Paint để tô màu một node. Lớp javafx.scene.paint.Color là một lớp con của Paint, được sử dụng để đóng gói các màu. Các phương thức và thuộc tính của lớp Color được thể hiện trong bảng sau đây.

|  |  |
| --- | --- |
| **javafx.scene.** **paint.** **Color** | **Miêu tả** |
| - red: double | Giá trị màu đỏ của màu này (từ 0.0 🡪 1.0). |
| - green: double | Giá trị màu xanh lá cây của màu này (từ 0.0 🡪 1.0). |
| - blue: double | Giá trì màu xanh lam của màu này (từ 0.0 🡪 1.0). |
| - opacity: double | Độ mờ của màu này (từ 0.0 🡪 1.0). |
| + Color(r: double, g: double, b:  double, opacity: double) | Tạo một Color với các giá trị red, green, blue và opacity cụ thể. |
| + brighter(): Color | Trả vể một Color mới là phiên bản sáng hơn của Color hiện tại (giá trị các thuộc tính lớn hơn). |
| + darker(): Color | Trả về một Color mới là phiêm bản tối hơn của Color hiện tại (giá trị các thuộc tính lớn hơn). |
| + Các phương thức getter cho các thuộc tính.  + ... | |
| **Các phương thức static** | |
| + color(r: double, g: double, b: double): Color | Trả về một Color mờ đục với các giá trị red, green, blue cụ thể. |
| + color(r: double, g: double, b: double, opacity: double): Color | Trả về một Color với các giá trị màu red, green, blue và opacity cụ thể. |
| + rgb(r: int, g: int, b: int): Color | Trả về một Color với các giá trị red, green, blue cụ thể trong phạm vi từ 0 🡪 255. |
| + rgb(r: int, g: int, b: int, opacity: double): Color | Trả về một Color với các giá trị red, green, blue cụ thể trong phạm vi từ 0 🡪 255 và độ mờ nhất định. |

Trong đó r, b, g là chỉ định màu cho các thuộc tính red, green và blue. Giá trị của các thuộc tính từ 0.0 (vùng tối nhất) đến 1.0 (vùng sáng nhất). Giá trị opacity xác định độ trong suốt của một màu trong phạm vi từ 0.0 (hoàn toàn trong suốt) đến 1.0 (hoàn toàn không trong suốt). Đây gọi là mô hình RBGA, trong đó RBGA là viết tắt của red, green, blue và alpha. Giá trị alpha cho biết độ mờ.

Lớp Color là bất biến. Khi một đối tượng Color được tạo, các thuộc tính của nó không thể thay đổi. Ngoài ra, ta có thể sử dụng một trong các màu tiêu chuẩn được xác định bởi các hằng số trong lớp Color như BEIGE, BLACK, BLUE, BROWN, CYAN, DARKGRAY, GOLD, GRAY, GREEN, LIGHTGRAY, MAGENTA, NAVY, ORANGE, PINK, RED, SILVER, WHITE và YELLOW.

### 5.2.2 Chương trình minh họa lớp Color

*Hình 54: Chương trình minh họa lớp Color*

*Hình 55: Giao diện minh họa lớp Color*

## 5.3 Lớp Font

### 5.3.1 Tổng quan về lớp Font

Lớp Font định nghĩa tên, kiểu và kích thước của phông chữ. Ta có thể đặt phông chữ để hiển thị văn bản. Font được đại diện bởi lớp javafx.scene.text.Font. Một đối tượng Font là bất tiến. Khi một đối tượng Font được tạo nó sẽ không thể thay đổi các giá trị thuộc tính. Các thuộc tính và phương thức của lớp Font được thể hiện trong bảng sau đây.

|  |  |
| --- | --- |
| **javafx.scene.** **text.** **Font** | **Miêu tả** |
| - size: double | Kích thước của phông chữ |
| - name: String | Tên của phông chữ. |
| - family: String | Họ của phông chữ. |
| + Font(size: double) | Tạo Font với kích thước chữ cụ thể. |
| + Font(name: String, size: double) | Tạo một Font với tên và kích thước của phông chữ. |
| + Các phương thức getter cho các thuộc tính.  + ... | |
| **Các phương thức static** | |
| + font(family: String, size: double): Font | Trả về một Font với họ và kích thước phông chữ cụ thể. |
| + font(family: String, w: FontWeight, size: double) | Trả về một Font với họ, độ rộng của nét chữ, kích thước cụ thể của phông chữ. |
| + font(family: String, w: FontWeight, p: FontPosture, size: double) | Trả về một Font với họ, độ rộng của nét chữ, kiểu dáng chữ, kích thước cụ thể của phông chữ. |
| + getFontNames(): List<String> | Trả về danh sách tất cả các tên phông chữ được cài đặt trên hệ thống người dùng. |

Trong đó thuộc tính name của phông chữ có thể xem đầy đủ tên các phông chữ bằng cách gọi phương thức getFontNames(). Thuộc tính family có thể xem đầy đủ các họ kiểu chữ bằng cách gọi phương thức getFamilies(). Kiểu dáng phông chữ (FontPosture) có hai loại là in nghiêng và thằng đứng được xác định thông qua các hằng số FontPosture.ITALIC, FontPosture.REGULAR. Độ rộng của nét chữ (FontWeight) có nhiều kiểu và chúng cũng được xác định thông qua các hằng số như: FontWeight.BLACK, FontWeight.BOLD, FontWeight.LIGHT, FontWeight.THIN, FontWeight.NORMAL, FontWeight.MEDIUM, FontWeight.EXTRA\_BOLD, FontWeight.SEMI\_BOLD.

### 5.3.2 Chương trình minh họa lớp Font

*Hình 56: Chương trình minh họa lớp Font*

*Hình 57: Giao diện minh họa lớp Font*

## 5.4 Lớp Image và ImageView

### 5.4.1 Tổng quan về lớp Image và ImageView

Lớp Image được sử dụng để tải lên một hình ảnh từ một URL hoặc từ một file.

ImageView được sử dụng để hiển hiển thị hình ảnh của một đối tượng Image. Các phương thức và thuộc tính của lớp Image và ImageView được thể hiện trong bảng sau.

|  |  |
| --- | --- |
| **javafx.scene.** **image.** **Image** | **Miêu tả** |
| - error: ReadOnlyBooleanProperty | Cho biết liệu hình ảnh có được tải đúng không ? |
| - height: ReadOnlyDoubleProperty | Chiều cao của hình ảnh. |
| - width: ReadOnlyDoublePropert | Chiều rộng của hình ảnh. |
| - progress: ReadOnlyDoubleProperty | Tỷ lệ gần đúng quá trình tải hình ảnh. |
| + Image(filename/URL: String) | Tạo hình ảnh từ một file hoặc một URL. |
| + Các phương thức getter cho các thuộc tính.  + ... | |

|  |  |
| --- | --- |
| **javafx.scene.** **image.** **ImageView** | **Miêu tả** |
| - x: DoubleProperty | Tọa độ x của điểm gốc ImageView. |
| - y: DoubleProperty | Tọa độ y của điểm gốc ImageView. |
| - image: ObjectProperty<Image> | Hình ảnh cần được hiển thị. |
| - fitWidth: DoubleProperty | Chiều rộng giới hạn của khung hiển thị trong đó hình ảnh sẽ phải thay đổi kích thước để vừa với khung hình. |
| - fitHeight: DoubleProperty | Chiều cao giới hạn của khung hiển thị trong đó hình ảnh sẽ phải thay đổi kích thước để vừa với khung hình. |
| + ImageView() | Tạo một ImageView. |
| + ImageView(image: Image) | Tạo một ImageView với một hình ảnh cụ thể. |
| + ImageView(filename/URL: String) | Tạo một ImageView với một hỉnh ảnh được tải từ một file hoặc một URL. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

*Chú ý: Một đối tượng Image có thể được chia sẻ cho nhiều ImageView. Chúng ta không thể đặt một đối tượng ImageView nhiều lần vào một ngăn bố cục hoặc một Scene. Nếu sử dụng URL để định vị file hình ảnh, thì giao thức URL http:// luôn phải có.*

### 5.4.2 Chương trình minh họa lớp Image và ImageView

*Hình 58: Chương trình minh họa lớp Image và ImageView*

*Hình 59: Giao diện minh họa lớp Image và ImageView*

## 5.5 Lớp Media, MediaPlayer, MediaView

### 5.5.1 Tổng quan về các lớp Media, MediaPlayer, MediaView

Video và audio là những thành phần cần thiết, không thể thiết trong việc phát triển các ứng dụng GUI phong phú. JavaFX cung cấp các lớp Media, MediaPlayer, MediaView để làm việc video và audio. Hiện tại, JavaFX hỗ trợ các định dạng audio như MP3, AIFF, WAV, MPEG-4 và các định dạng video như FLV, MPEG-4.

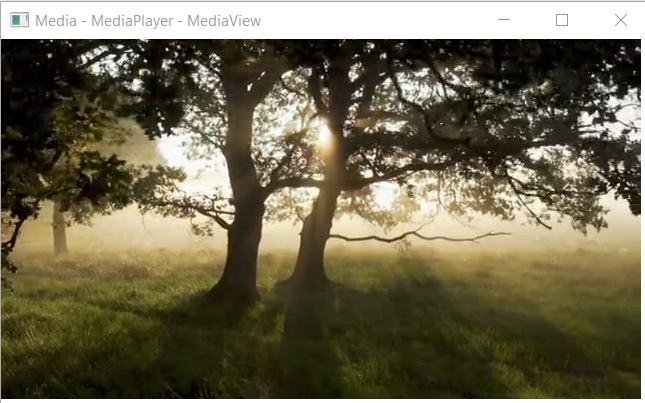
Lớp Media được sử dụng để tải nguồn video hoặc audio chuỗi URL. Lớp MediaPlayer phát và điểu khiển video hoặc audio. Lớp MediaView tạo ra chế độ xem video và audio trên khung. Một đối tượng Media có thể được chia sẻ cho nhiều MediaPlayer. Một MediaPlayer có thể chia sẻ cho nhiều MediaView. Các thuộc tính và phương thức, mối quan hệ của cả ba lớp được thể hiện cụ thể trong bảng dưới đây.

|  |  |
| --- | --- |
| **javafx.scene.** **media.Media** | **Miêu tả** |
| - duration: ReadOnlyObjectProperty<Duration> | Thời lượng (tính bằng giây) một video hoặc audio nguồn. |
| - width: ReadOnlyIntegerProperty | Chiều rộng (tính bằng pixel) của video nguồn. |
| - height: ReadOnlyIntegerProperty | Chiều cao (tính bằng pixel) của video nguồn. |
| + Media(source: String) | Tạo một Media từ nguồn URL. |
| + Các phương thức getter cho các thuộc tính.  + ... | |

|  |  |
| --- | --- |
| **javafx.scene.** **media.MediaPlayer** | **Miêu tả** |
| - mute: BooleanProperty | Âm thanh có được tắt tiếng hay không. |
| - volume: DoubleProperty | Âm lượng cho âm thanh. |
| - media: ObjectProperty<Media> | Đối tượng media chứa video hoặc audio để phát. |
| - cycleCount: IntegerProperty | Số thời lần video và audio sẽ được phát lại. |
| - totalDuration:  ReadOnlyObjectProperty<Duration> | Thời lượng của video và audio. |
| - currentCount: ReadOnlyIntegerProperty | Số chu kì đã hoàn thành phát lại. |
| - autoPlay: BooleanProperty | Chỉ định video hoặc audio tự động chạy hay không. |
| + MediaPlayer(media: Media) | Tạo một MediaPlayer từ một đối tượng Media. |
| + play(): void | Chạy video hoặc audio. |
| + pause(): void | Tạm dừng video hoặc audio. |
| + seek(): void | Đặt video hoặc audio muốn phát lại từ thời gian nào của video và audio. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

|  |  |
| --- | --- |
| **javafx.scene.** **media.** **MediaView** | **Miêu tả** |
| - x: DoubleProperty | Tọa độ x hiện tại của MediaView. |
| - y: DoubleProperty | Tọa độ y hiện tại của MediaView. |
| - mediaPlayer: ObjectProperty | Một mediaPlayer đẻ điều khiển video và audio được hiển thị. |
| - fitWidth: DoubleProperty | Chiều rộng của khung xem, video sẽ tự động căn chỉnh theo khung đó. |
| - fitHeight: DoubleProperty | Chiều cao của khung xem, video sẽ tự động căn chỉnh theo khung đó. |
| + MediaView() | Tạo một MediaView trống. |
| + MediaView(mediaPlayer: MediaPlayer) | Tạo một MediaView với MediaPlayer được chỉ định. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 5.5.2 Chương trình minh họa các lớp Media, MediaPlayer, MediaView

*****Hình 60: Chương trình minh họa lớp Media, MediaPlayer, MediaView*

*Hình 61: Giao diện minh họa lớp Media, MediaPlayer, MediaView*

## 5.6 Lớp Text

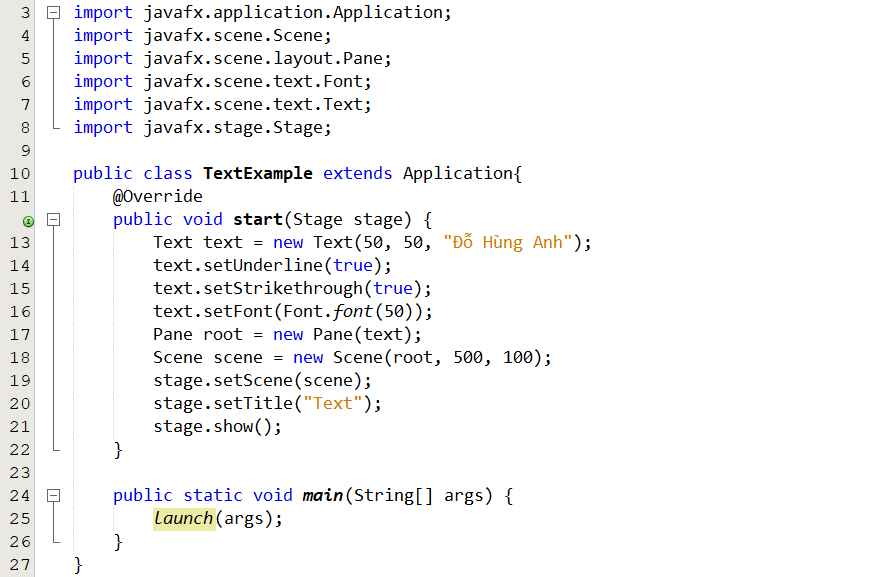
### 5.6.1 Tổng quan về lớp Text

Lớp Text định nghĩa một node để hiển thị một chuỗi tại điểm bắt đầu (x, y). Một chuối có thể được hiển thị trong nhiều dòng cách nhau bởi \n. Các thuộc tính và phương thức của lớp Text được thể hiện trong bảng dưới đây.

|  |  |
| --- | --- |
| **javafx.scene.** **text.Text** | **Miêu tả** |
| - text: StringProperty | Nội dung văn bản sẽ được hiển thị. |
| - x: DoubleProperty | Xác định tọa độ x bắt đầu của văn bản (mặc định: 0). |
| - y: DoubleProperty | Xác định tọa độ y bắt đầu của văn bản (mặc định: 0). |
| - font: ObjectProperty<Font> | Định nghĩa phông chữ cho văn bản. |
| - strikethrough: BooleanProperty | Văn bản có được gạch ngang không (mặc định: false). |
| - underline: BooleanProperty | Văn bản có được gạch chân bên dưới không (mặc định: false). |
| + Text() | Tạo một Text trống. |
| + Text(text: String) | Tạo một Text với nội dung văn bản được chỉ định. |
| + Text(x: double, y: double, text: String) | Tạo một Text với tọa độ bắt đầu (x, y) và nội dung văn bản cụ thể. |
| + Các phương thức getter và setter cho các thuộc tính.  + ... | |

### 5.6.2 Chương trình minh họa lớp Text

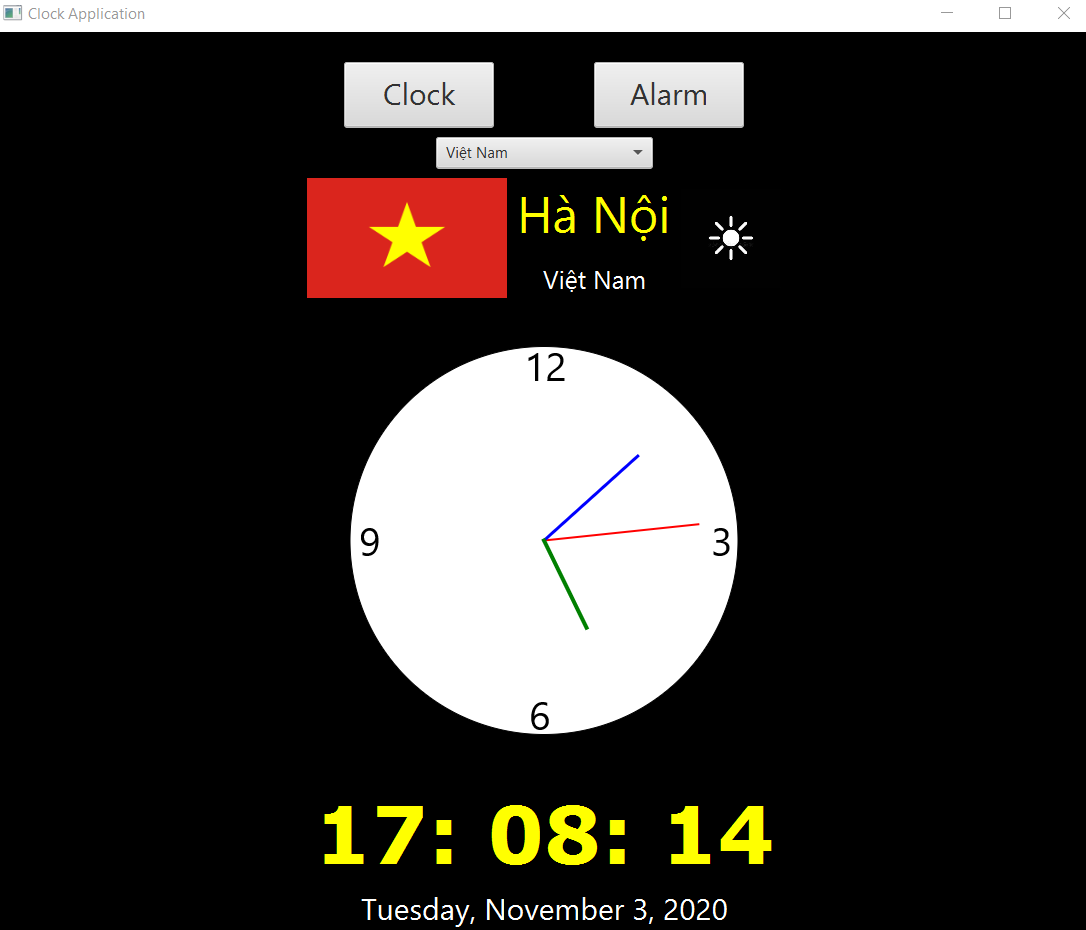
*Hình 62: Giao diện minh họa lớp Text*

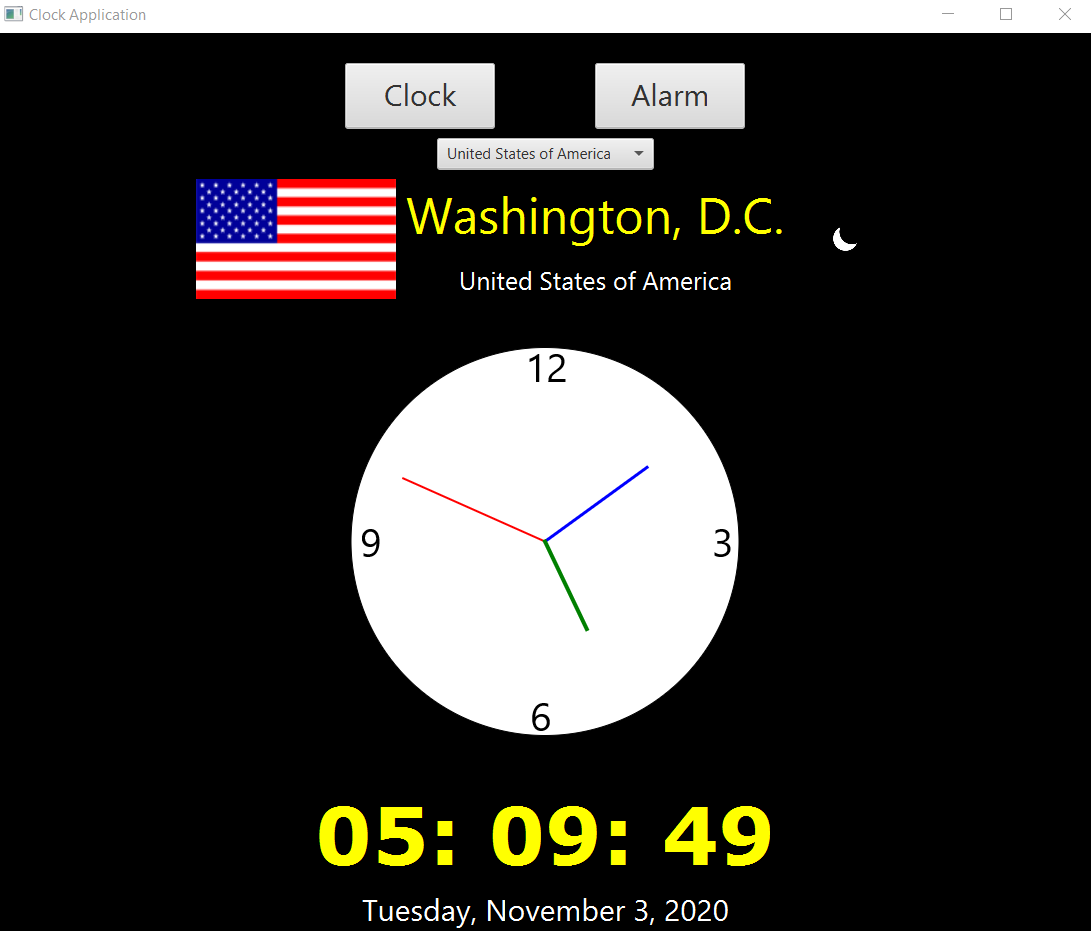
*Hình 63: Chương trình minh họa lớp Text*

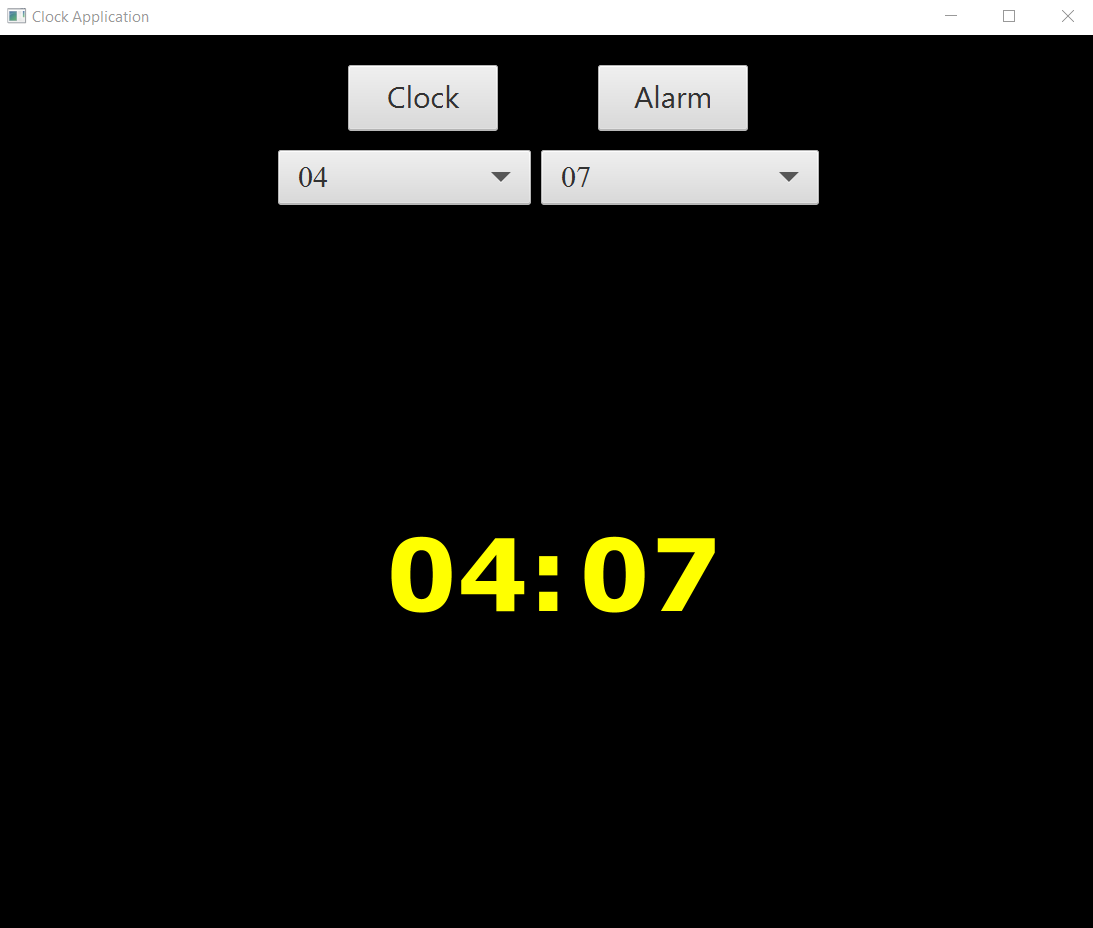
# 6. DỰ ÁN MINH HỌA SỬ DỤNG JAVAFX

Các kiến thức trong bài báo cáo này đểu là những kiến thức cơ bản về JavaFX. Tuy nhiên vẫn đủ để chúng ta vẫn có thể áp dụng để tạo một dự án nhỏ. Sau đây là dự án viết một ứng dụng đồng hồ có hai chức năng là xem giở của một số quốc gia và đặt báo thức. Các kiến thức của JavaFX được vận dụng trong dự án bao gồm:

* Các hình dạng 2D.
* Các UI Control: Label, Button, ComboBox.
* Các ngăn bố cục: BorderPane, HBox, Vbox, Pane.
* Các lớp Image, ImageView, Media, MediaPlayer, MediaView, Color, Text, Font.
* Một số kiến thức khác như: JavaFX với CSS, Lập trình hướng sự kiện, vào ra file.

 Sau đây là một số hình ảnh minh họa về dự án.





Vì mã nguồn của dự án rất dài, không thể đưa hết vào bài báo cáo. Thầy và các bạn có thể xem mã nguồn ở thư mục kèm theo hoặc truy cập trực tiếp vào liên kết dưới đây để lấy mã nguồn. <https://drive.google.com/file/d/18vBDPOSSThjiitrGSrNbw6Aqbt8zvZ8E/view?usp=sharing>

# KẾT LUẬN

JavaFX là một framework mới để phát triển các chương trình GUI, nó cung cấp đầy đủ các thành phần để phát triển một giao diện ứng dụng hoàn hảo. Kiến thức về JavaFX rất rộng, bài báo cáo này chỉ tậm trung vào những vấn đề chung và cơ bản cốt lõi nhất của JavaFX để phát triển một giao diện ứng dụng đơn giản. Có hai cách để phát triển một giao diện ứng dụng bằng JavaFX. Một là viết toàn bộ bằng mã Java như trong bài báo cáo này. Hai là sử dụng giao diện kéo thả. Làm theo cách hai thì chương trình sẽ trở lên gọn gàng và dễ hình dung vì nó tách riêng file giao diện và file logic của mã Java. Mặc dù cách thứ hai tiện lợi nhưng chúng ta vẫn phải đi từ những thứ cơ bản nhất để có thể hiểu được cách hoạt động của một chương trình JavaFX bất kì. Ngoài ra ta cũng có thể xử dụng CSS để làm cho giao diện ứng dụng JavaFX trở lên đẹp hơn rẩt nhiều.

# TÀI LIỆU THAM KHẢO

[1] Y. D. Liang, “Introduction to Java Programming and Data Structures, Comprehensive Version 11th Edition,” 2018.

[2] <https://o7planning.org/vi/11009/javafx>

[3] <https://www.javatpoint.com/javafx-tutorial>

[4] <https://www.geeksforgeeks.org>