



R2.02



Notions de base

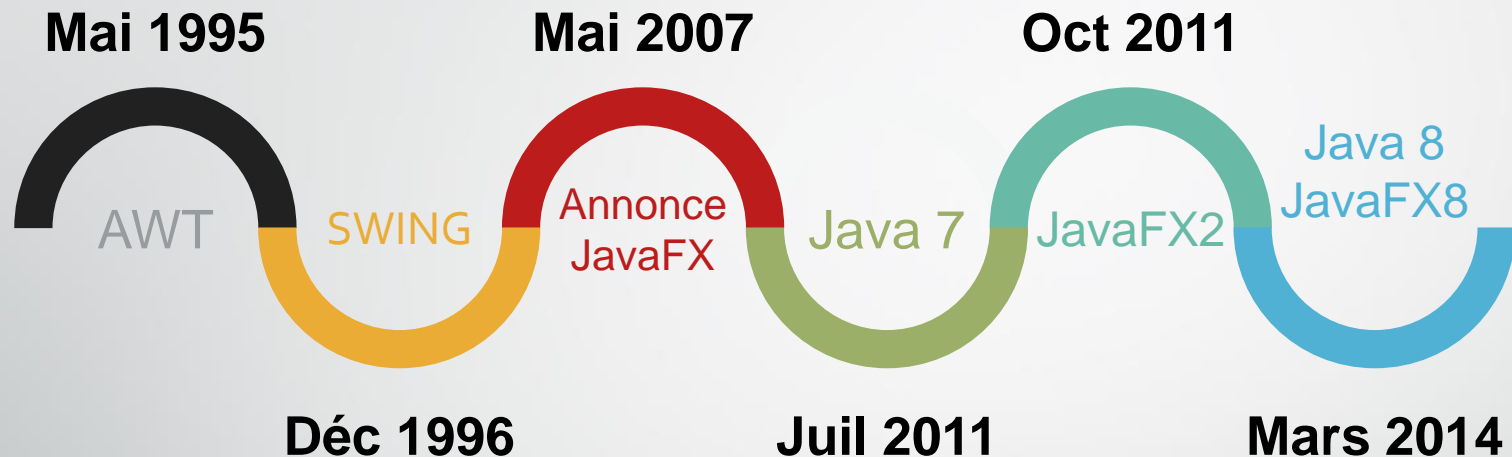
# Sommaire

- Chronologie
- Architecture de Java et de JavaFX
- Architecture applicative
- Le graphe de scène
- Les composants
- Les conteneurs

# JavaFX : chronologie

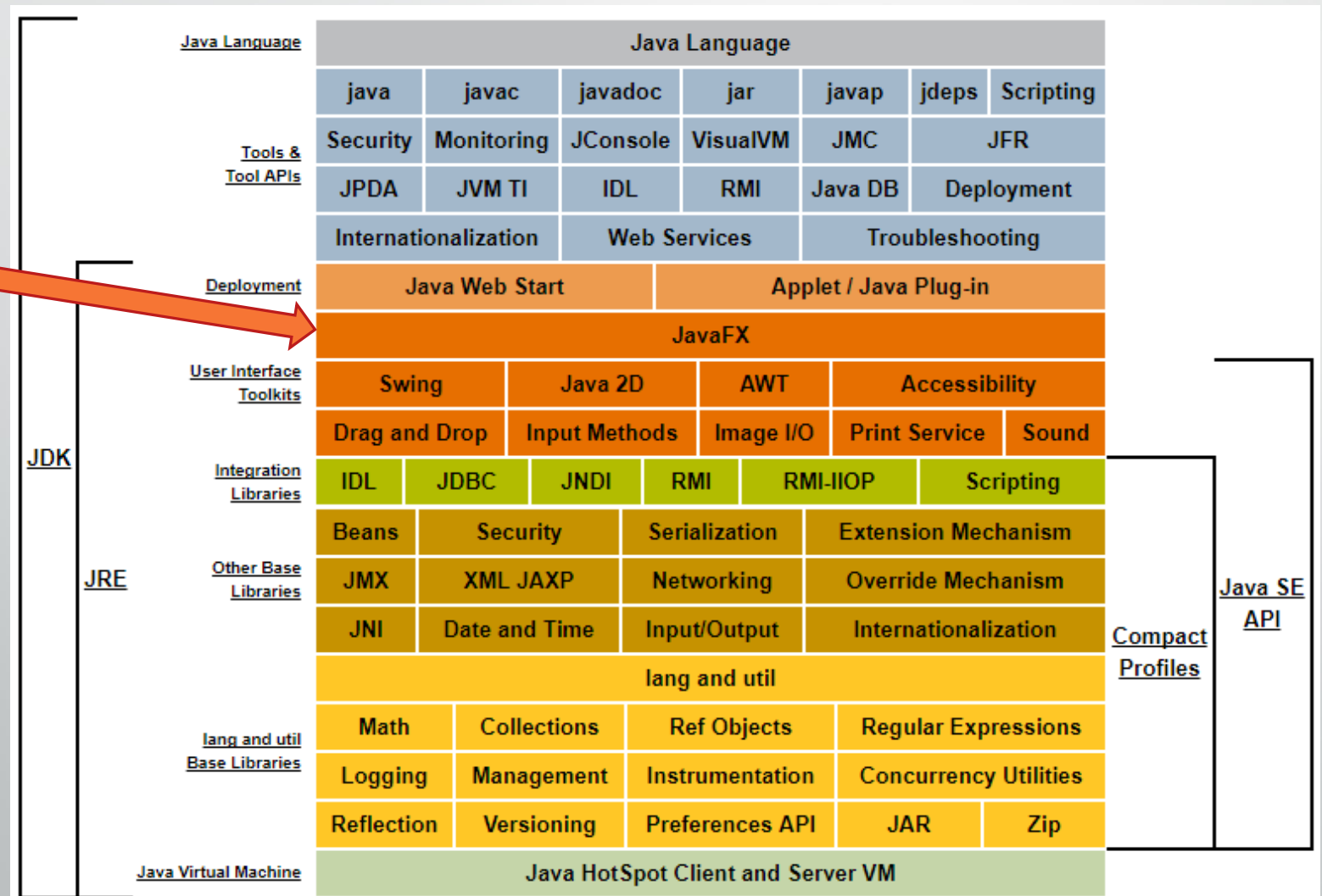
- Mai 1995 : AWT
- Janvier 1996 : Java 1.0
- Décembre 1996 : Swing
- Mai 2007 : annonce JavaFX
- Juillet 2011 : Java 7
- Octobre 2011 : JavaFX 2
- Mars 2014 : Java 8 et JavaFX 8
- Septembre 2017 : Modularisation JavaFX 9
- Septembre 2018 : Module à part JavaFX 11
- Septembre 2021 : JavaFX 17
- Septembre 2023 : JavaFX 21 (oct. 2023 JavaFX 21.0.1)

# JavaFX 8 : chronologie



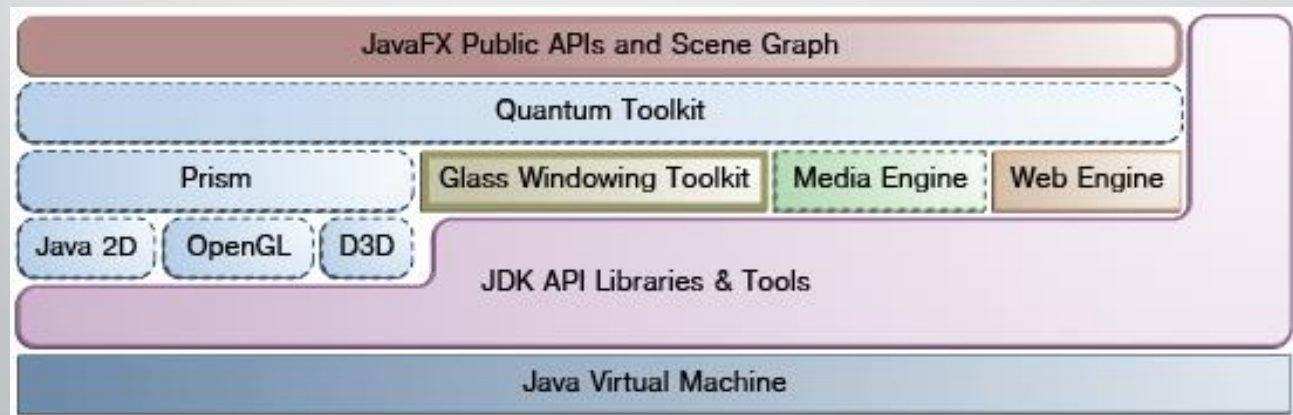
- Dans le cours on va étudier **JavaFX 8** basé sur **Java 8**
- mais on ne va pas utiliser les concepts apportés par Java 8 telles que : les interfaces fonctionnelles, les lambda expressions, l'API des streams

# Architecture de Java



# Architecture de JavaFX

- Prism : moteur de rendu graphique
- Glass : système de gestion des fenêtres, timers et surfaces
- Quantum Toolkit
- Web Engine
- Media Engine

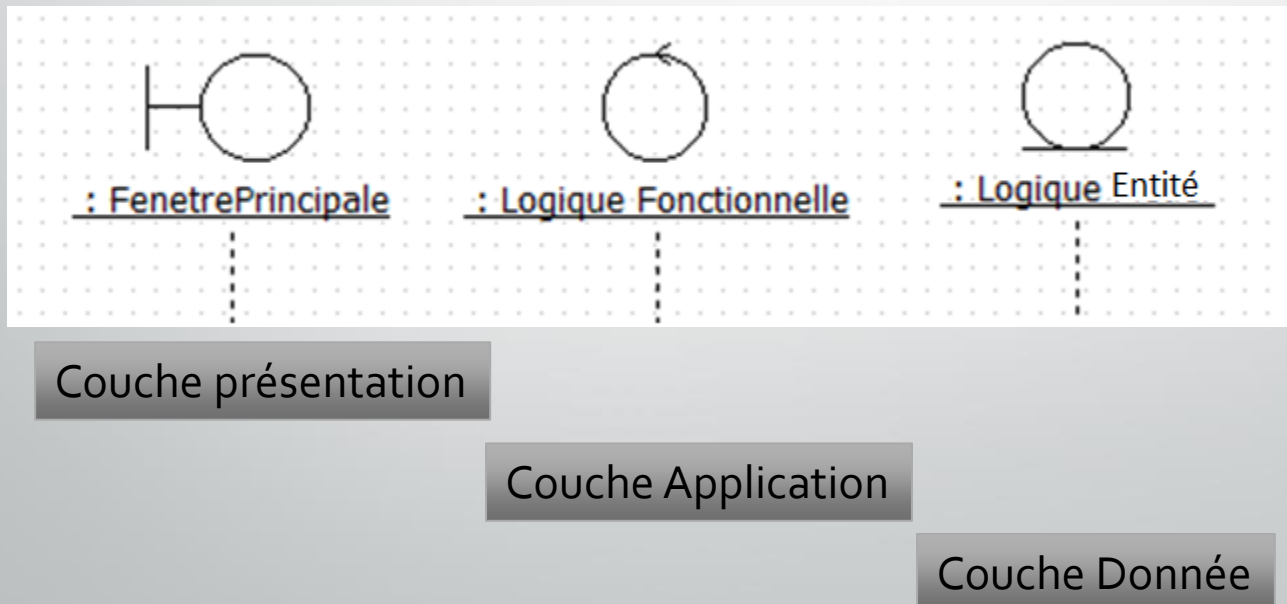


# Le développement logiciel

- Un petit focus sur l'architecture applicative :
  - le découpage en 3 couches logicielles (ou N couches)
  - le modèle de conception MVC

# Le découpage en couches 1/2

- Architecture trois-tiers (ou N-tiers) : dans un environnement client-serveur
- Application composée de plusieurs composants logiciels distincts





# Le découpage en couches 2/2

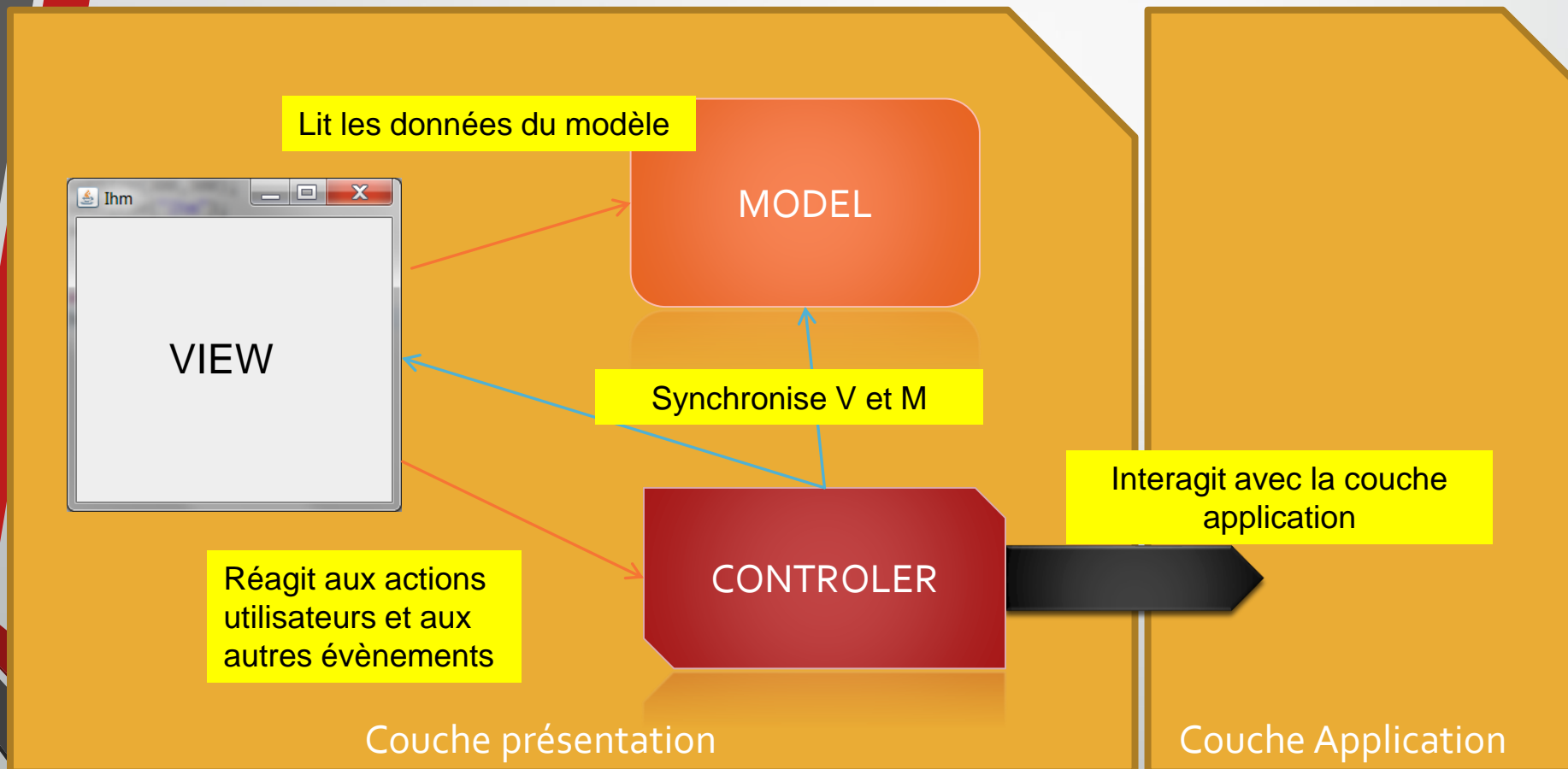
- **Couche présentation :**
  - partie visuelle et interactive de l'application (ihm)
  - interface classique (client lourd) ou web (client léger)
- **Couche application :**
  - partie logique, fonctionnelle et métier (règles de gestion)
- **Couche donnée :**
  - partie chargée de la manipulation des données

# La couche présentation

- Gestion du domaine visuel et interactions avec l'utilisateur
- Logique de présentation
  - navigation (enchaînement des pages)
  - appel aux services métiers (situés sur le serveur)
  - traitement des données retournées par les services métiers
  - mise en forme des résultats et présentation de ceux-ci
- Isolation des responsabilités
  - découplage des couches entre elles
  - séparation des responsabilités
  - minimiser l'impact du changement

# Le modèle de conception : MVC

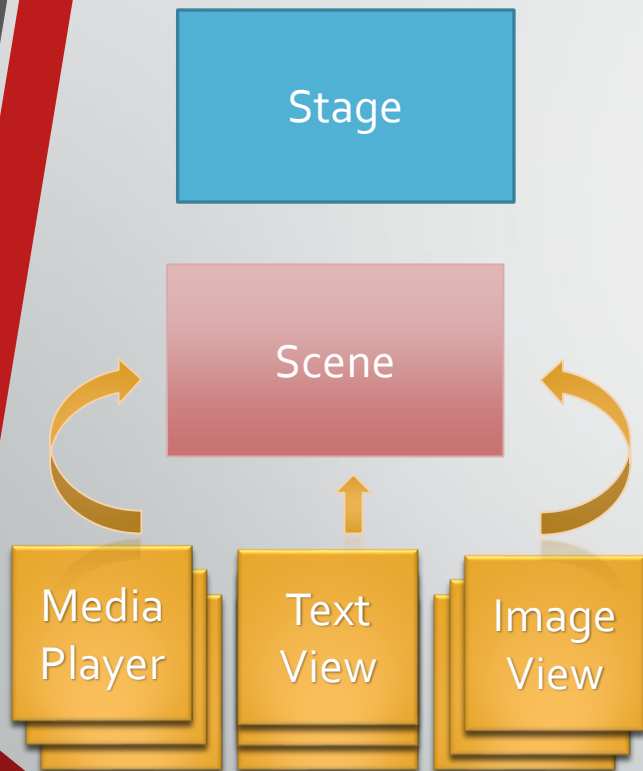
- Model / View / Controller



# La scène JavaFX



# Une représentation de théâtre



**Container de plus haut niveau (fenêtre)**

**Compose une page de l'application :  
Contient tous les éléments de notre  
interface graphique.**

**Nœuds JavaFX (composants ou groupes)  
qui sont des objets de type Node.**

# Structure d'une application JavaFX

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class HelloWorld extends Application {

    @Override
    public void start(Stage primaryStage) {

        // Code pour créer votre application JavaFX
        // instanciation d'une scene
        Scene scene = new Scene(...);
        // on la rattache au container Stage
        primaryStage.setScene(scene);
        // on affiche l'interface
        primaryStage.show();

    }

    /**
     * @param args arguments passés à la JVM
     */
    public static void main(String[] args) {

        // Lancement de l'application
        Application.launch(args);

    }
}
```

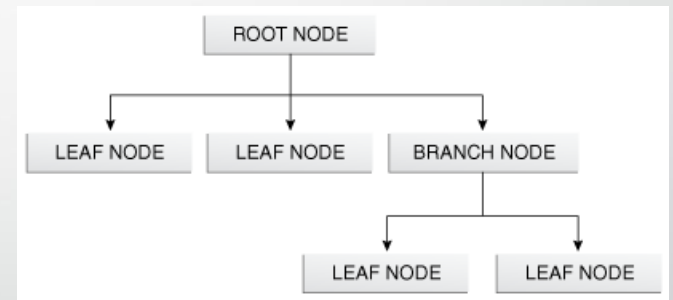
La scène contient tous les composants de votre interface graphique

# Le graphe de scène

- **Structure hiérarchique de l'interface graphique**
  - c'est un graphe orienté acyclique

- **Les nœuds peuvent être de trois types :**

- racine
- nœud intermédiaire (ou branche)
- feuille



- **Généralement :**

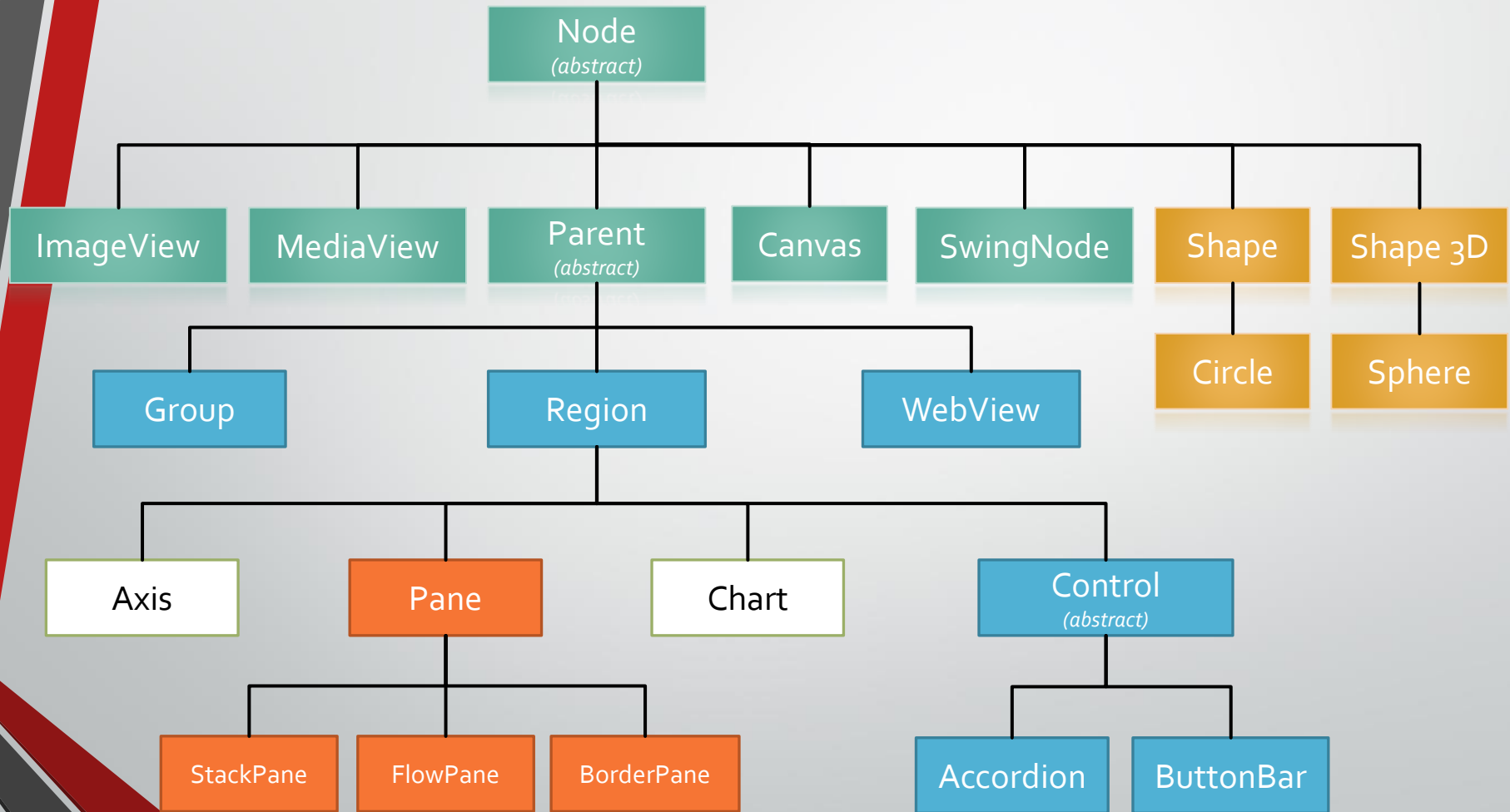
- feuilles : composants visibles (texte, bouton, etc)
- racine et nœuds intermédiaires : éléments de structuration

# La classe Node

- Tous les éléments contenus dans le graphe de scène héritent de la classe `javafx.scene.Node`
- C'est une classe abstraite représentant un nœud
- Elle possède de nombreuses sous-classes :
  - les composants graphiques standards (UI Control)
  - les formes primitives (Shape)
  - les conteneurs (Layout-Pane ou simplement Pane)
  - les composants spécialisés (exemple `MediaView` ou `WebView`)

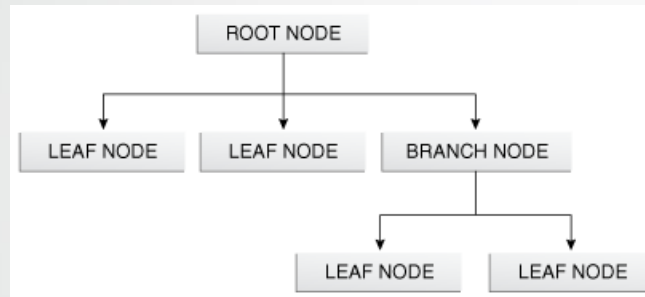


# Sous-classes de Node

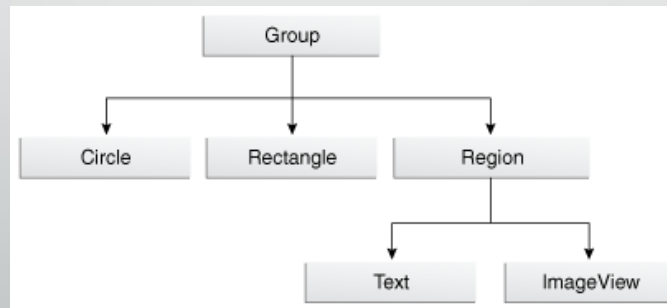


# Illustration d'un graphe

- Arbre comportant : racine, branches et feuilles :



- Exemple avec des classes concrètes :



# Les composants : UI Controls



# Les UI Controls

- Ensemble de composants graphiques fournis par JavaFX
- Héritent de la classe **Control** : `package javafx.scene.control`
- Il faut les distinguer des **conteneurs** (on les voit juste après)
- Certains sont des composants simples (libellé, bouton), d'autres sont complexes (tableau, arbre, etc)
- Certains composants ont un contenu textuel associé : comportement pris en charge par la classe parente **Labeled**
  - <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Labeled.html>
  - exemples de propriétés : *text*, *font*, *alignment*, etc.
- On peut aussi écrire nos propres Controls

# Exemples de Controls

- Label

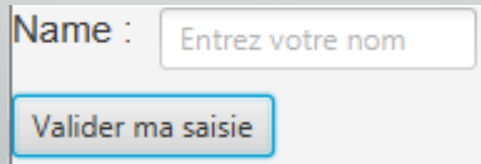
```
final Label label = new Label();  
label.setText("Name :");  
  
// Propriétés de la classe parente Labeled  
label.setFont(new Font("Arial", 15));
```

- TextField

```
final TextField textField = new TextField();  
textField.setPromptText("Entrez votre nom");
```

- Button

```
final Button btn = new Button();  
btn.setText("Valider ma saisie");  
// TODO programmer une action quand on clique sur le bouton
```



Name :

[https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/ui\\_controls.htm](https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/ui_controls.htm)

Label	Button
Checkbox	Radio Button
Choice Box	Toggle Button
Combo Box	Scroll Bar
Text Field	Scroll Pane
Password Field	Separator
List View	Slider
Table View	Progress Bar and Progress Indicator
Tree View	Hyperlink
Tree Table View	Tooltip
HTML Editor	Titled Pane and Accordion
Menu	Color Picker
Pagination Control	Date Picker
	File Chooser

# Les conteneurs

# Positionnement des composants

Après avoir créé tous les nœuds de la scène, nous devons les positionner les uns par rapport aux autres.

Cet agencement au sein du conteneur est appelé le **Layout**.

Avec JavaFX, on peut disposer des composants :

- de manière absolue : utile pour certains cas mais présente de nombreux inconvénients lors du redimensionnement de la zone d'affichage.
- de manière relative : la disposition des éléments est confiée à un layout manager, ce dernier étant intégré dans un conteneur nommé **Layout Pane**.

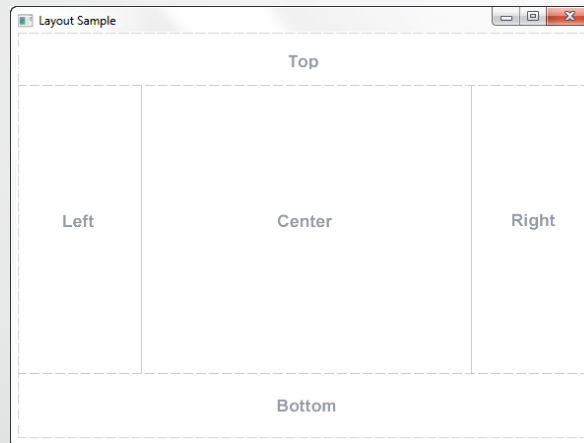


# Les Layout Panes

- Facilite la création et la gestion de votre interface graphique
- Prend en charge l'ajout et la disposition des composants
- Il va placer automatiquement chaque composant à une position particulière au sein du conteneur, selon des règles (ou contraintes) qui lui sont spécifiques
- Lorsque la fenêtre sera redimensionnée, il va repositionner et redimensionner automatiquement les nœuds qu'elle contient, selon les propriétés de chaque nœud
- JavaFX fournit différents Layout Panes pré-définis pour les besoins d'agencement courant : ligne, colonne, tableau, etc.
- On peut aussi écrire nos propres Layout Panes

# Les Layout Panes fournis 1/3

- **BorderPane** : permet de diviser votre espace en cinq zones graphiques : top, down, right, left et center.



- **Hbox** : permet d'aligner horizontalement vos éléments graphiques



# Les Layout Panes fournis 2/3

- **Vbox** : permet d'aligner verticalement vos éléments graphiques
- **StackPane** : permet de ranger vos éléments de façon à ce que chaque nouvel élément inséré apparaisse au-dessus de tous les autres (comme une pile)
- **GridPane** : permet de créer une grille d'éléments organisés en lignes et en colonnes

# Les Layout Panes fournis 3/3

- **FlowPane** : permet de ranger des éléments les uns à la suite des autres, en ligne (ou en colonne) ; ils se positionnent automatiquement en fonction de leur taille et de celle du layout. S'il n'y a pas assez de place dans le conteneur, le layout crée des lignes (ou des colonnes) supplémentaires.
- **TilePane** : est similaire au FlowPane, excepté que chacune de ses cellules fait la même taille.
- **AnchorPane** : permet de fixer (ancrer) un élément graphique par rapport à un des bords de la fenêtre.

Voir doc : [https://docs.oracle.com/javafx/2/layout/builtin\\_layouts.htm](https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm)

# Exemple

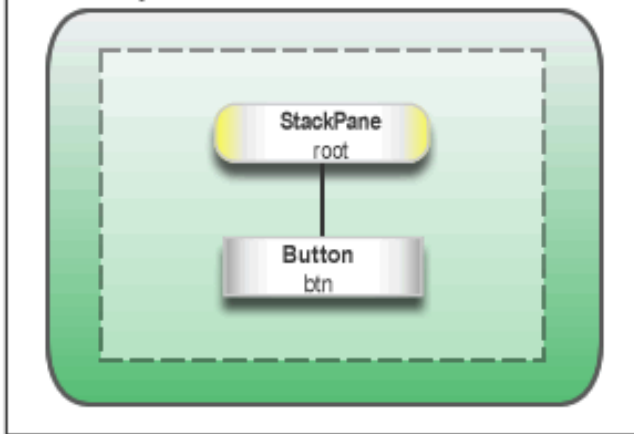
# HelloWorld au style JavaFX



```
public class HelloWorld extends Application {  
  
    @Override  
    public void start(Stage primaryStage) {  
        Button btn = new Button();  
        btn.setText("Say '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) {  
        Application.launch(args);  
    }  
}
```

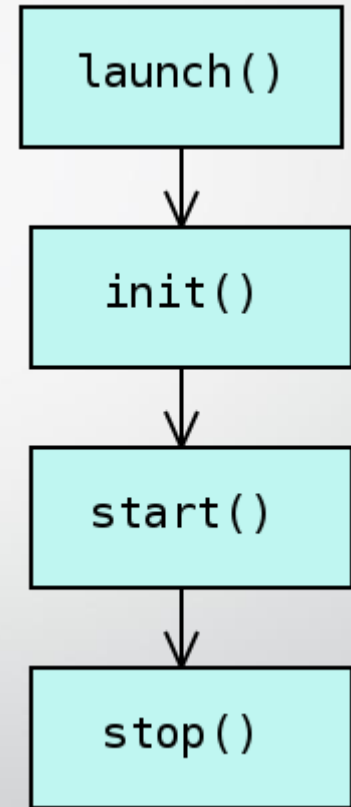
Stage javafx.stage (window)

Scene javafx.scene



# Cycle de vie

- launch : à appeler dans le main
- init :
- start : thread séparé
- stop :



`getParameters()` : pour récupérer les arguments si besoin