Application, "scene Graph", Nodes et Pane

19 janvier 2021

L'application

Hello World

```
(2)   public class HelloWorld extends Application {

        @Override
(3)     public void start(Stage primaryStage) {
            Button btn = new Button();
            btn.setText("Say 'Hello World'");
            btn.setOnAction(new EventHandler<ActionEvent>() {
                ……
        });

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

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

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

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

L'application: exécution

La méthode statique **Application.launch** (1) lance l'application. Dans l'ordre, elle :

1. crée une instance **insApp** de l'Application (ici `HelloWorld`) (2)
2. appelle **insApp.init()**
   on ne l'a pas redéfinie, alors elle ne fait rien.
3. lance la méthode **insApp.start()** (3) en lui passant une instance de `Stage` qui sera la fenêtre principale.
   -`start()` construit l'application, la fenêtre et la montre.
4. attend que l'application se termine.
   - soit par fermeture de toutes les fenêtres
     (sauf si `Platform.isImplicitExit()` retourne false)
   - soit par appel de `Platform.exit()`
5. appelle la méthode **iApp.stop()**
   on ne l'a pas redéfinie, alors elle ne fait rien.

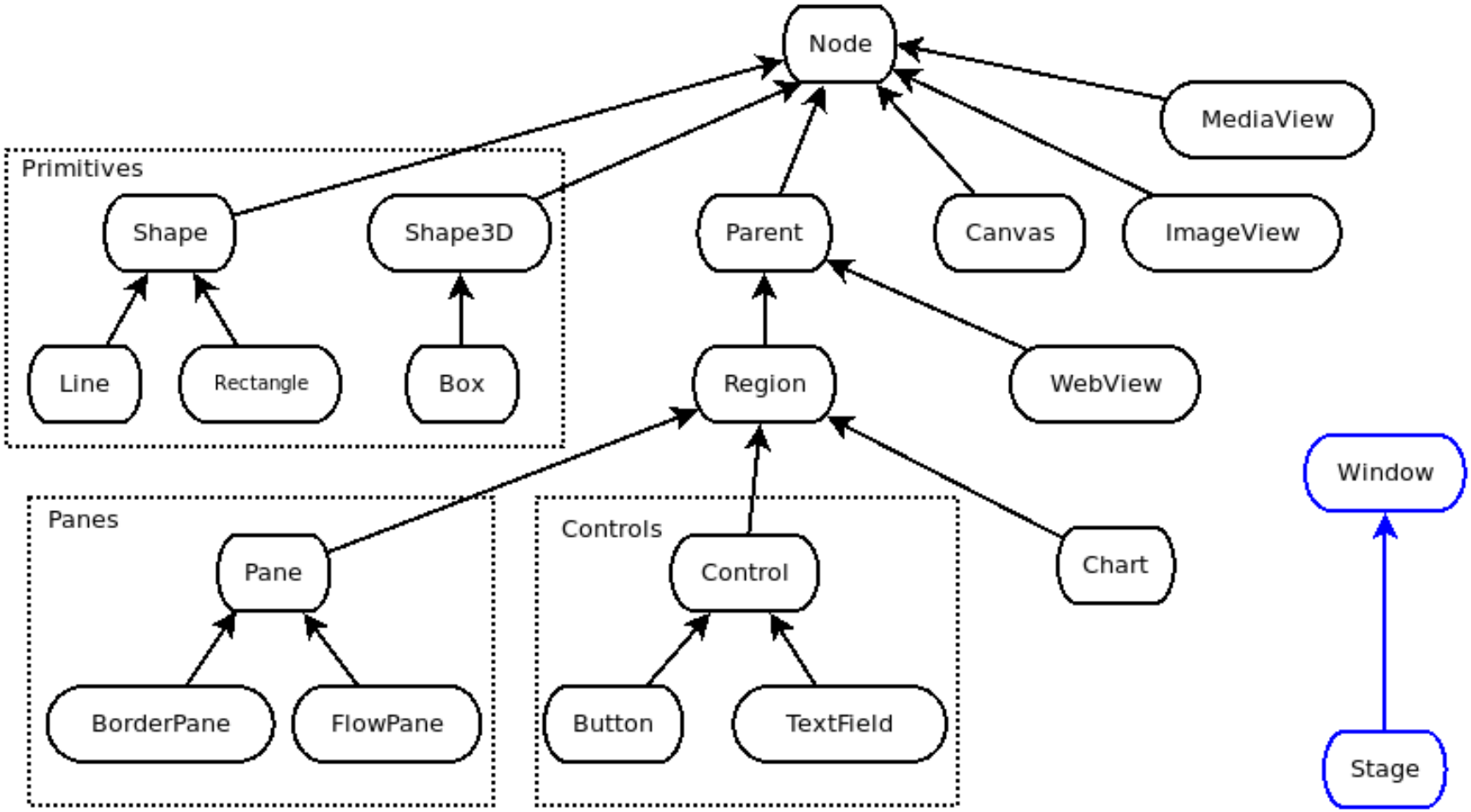
L'application: exécution

- si pas de `main`, il est défini par défaut, appelle `launch`
- `start()` est abstraite, et doit être redéfinie.
  - C'est ici qu'on construit la fenêtre.
    4. crée le panneau racine `root`, avec son bouton
    5. crée la scene en lui affectant le panneau racine.
    6. met la scène dans la fenêtre principale

  - start() s'execute dans le thread *JavaFX Application Thread*, le thread où s'execute tout ce qui manipule la fenêtre.
- `init()` et `stop()` ne font rien par défaut.
- `getParameters()` permet d'accéder (dans `init()`, `start()` ou `stop()` aux paramètres de la ligne de commande

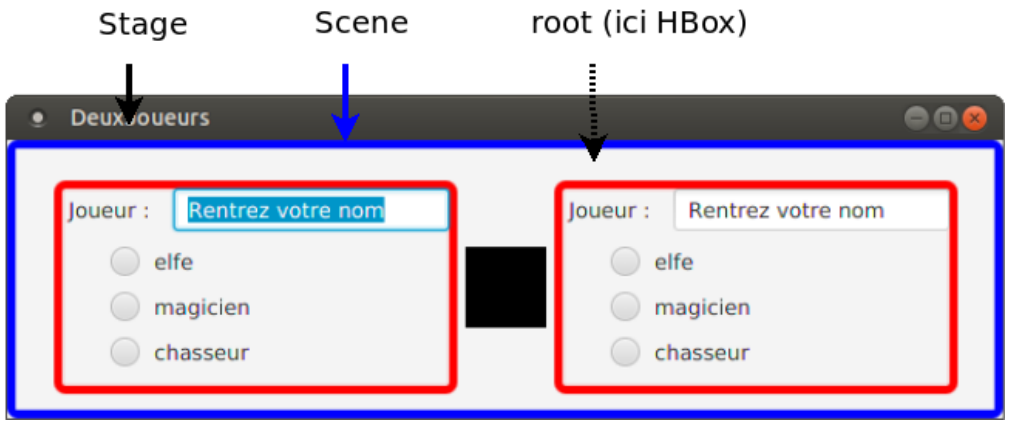
L'application: les différents composants

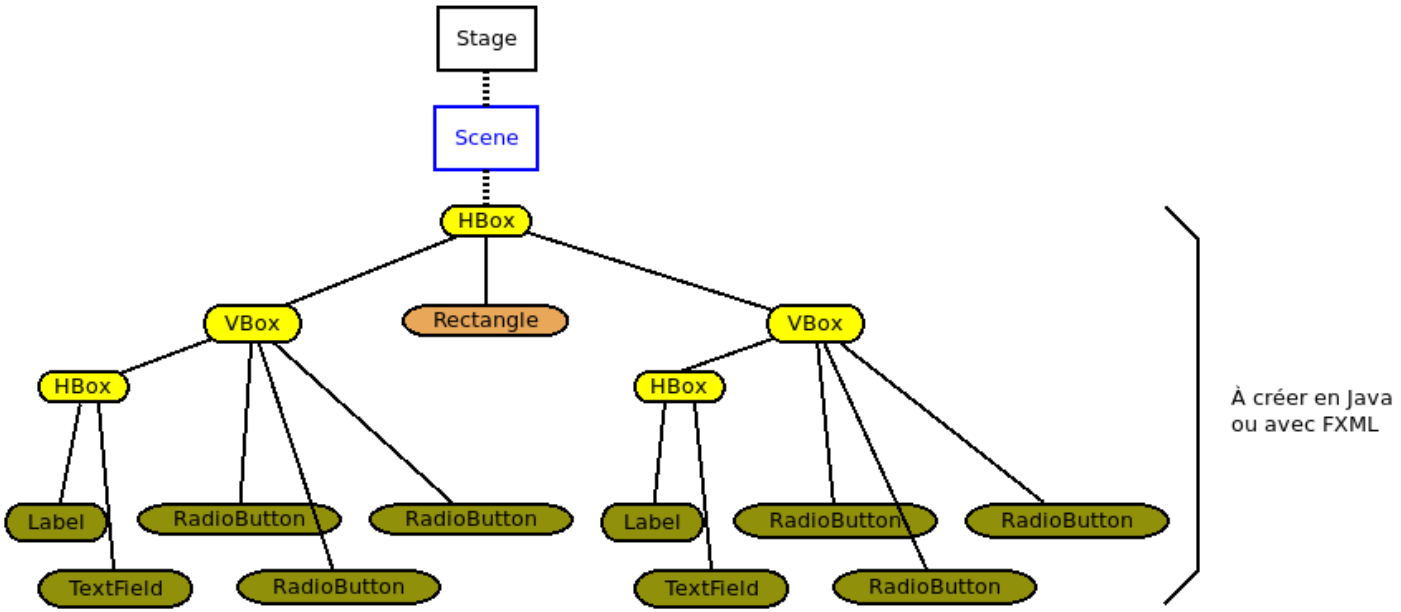
Arborescence des classes javaFX :



L'application: le graphe de scène

Exemple :





L'application: le graphe de scène

Pour faire le "Scene graph" ?

Les `Panneaux` : `Pane` et leurs classes filles `AnchorPane` `StackPane` `FlowPane`, `BorderPane` et... sont les noeuds internes (y compris la racine comprise)

- ajout et suppression `Pane pere` et `Node fils` :
  pere.getChildren().add(fils...);
  `pere.getChildren().remove(...)` Remarque : l'ajout dans un `GridPane` est un peu différent.
- **Scene scene = new Scene(root,...)**
- **stage.setScene(scene)**

Stage: placement et dimension

- Par défaut, la fenêtre principale est centrée sur l'écran.
- Méthodes utilisables pour changer le comportement d'un `Stage`.
  - `setX()` et `setY()` par rapport au bord coin supérieur gauche de l'écran
  - `centerOnScreen()`
  - `setMinWidth()setMinHeight()setMaxHeighth()setMaxWidth()`
  - `setResizable`
  - `sizeToScene()` comportement par défaut, adapte la taille à la taille de la scène.

Possibilité de récupérer les dimensions de l'écran avec la classe `Screen`

Stage: Apparence

- **setTitle()**,
- **setScene()**,
- **show()**,
- `setFullScreen()` met en plein écran,
- `showAndWait()` affiche la fenêtre à l'écran, et ne retourne que lorsqu'elle est fermée.
- `getIcons().add()`
- etc

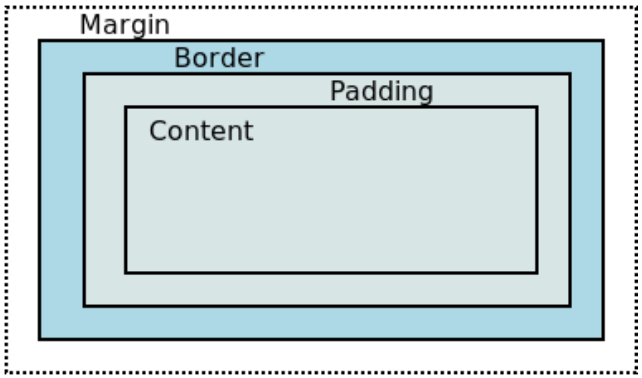
Propriétés des nodes

(Rappel : tout ce que l'on peut mettre dans le graphe de scene)

- **layoutX**, **layoutY** : position par rapport au coin supérieur gauche du composant. *Remarque : outes les coordonnées, sont en fait en X/Y/Z, et en double... cela simplifie les transformation géométriques.*
- d'autres propriétés géométriques (`rotate`, `scale`, `translate` suivant tous les axes)
- des propriétés d'apparence, `opacity`, `visible` par exemple,
- des propriétés contenant les méthodes appelées lors d'un événement `onMousePressed`, `onScroll`, etc...
- etc...

Propriétés des Regions : Control et Pane

- Les propriétés
  - les dimentions `height` `width` `minHeight`, `minWidth`, `maxHeight`, `maxWidth`, `prefHeight`, `prefWidth`
  - des bordures et marges `Padding`, `Border`, `Margin`
  - apparence : `background` par exemple
  - etc
- Les zones d'une région



Placement dans un Pane

Dans un `Pane` (Panneau de base), il faut tout placer à la main. On utilisant :

- `setLayoutX`, `setLayoutY`
- `setWidth`, `setHeight`.

Problèmes :

- Très compliqué de calculer la position en pixel de tout les noeuds d'une interface complète. Si on en agrandit un, ou en rajoute un, on va avoir besoin de recalculer la position de tous les autres.
- Les positions des composants de l'interface sont figées. Que se passe-t-il si on redimensionne la fenêtre ? Au choix de l'utilisateur, ou parce que la taille du terminal est différente (smartphone, tablette, desktop ?).