

P00 Avancé

JavaFx-Les Concepts de Base

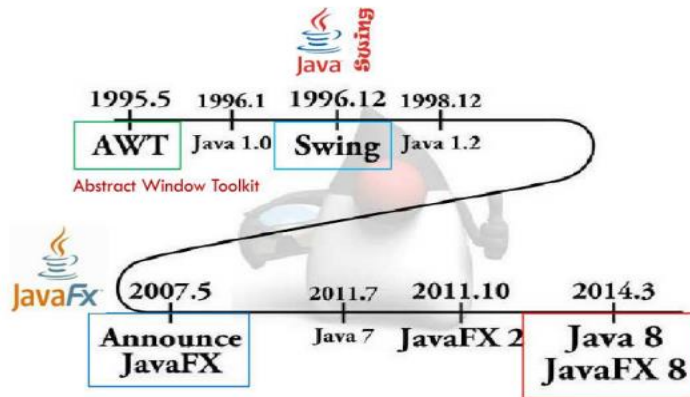


MEJDOUB Safa

Interfaces graphiques

- Les interfaces graphiques **assurent le dialogue** entre les utilisateurs et une application.
- Une interface graphique est **formée** d'une ou plusieurs **fenêtres** qui contiennent divers composants graphiques (**widgets**) tels que:
 - Boutons
 - listes déroulantes
 - Menus
 - champ texte, ...etc.
- Les interfaces graphiques sont souvent appelés **GUI** d'après l'anglais **Graphical User Interface**.

Les APIs



Safa MEJDOUB

3

Les APIs

- **AWT (java.awt):** première librairie de création des interfaces graphiques.
 - Composants "lourds" (heavyweight)
 - Difficulté de créer des applications multiplateformes
- **Swing (javax.swing) :** est venue compléter (et partiellement remplacer) la librairie AWT.
 - Composants "légers" (lightweight) dessinés par la librairie;
 - Tout les composants de Swing exceptés JApplet, JDialog, JFrame, Jwindow sont des composants légers
 - Offre plus de composants
 - Crée des applications multiplateformes

Safa MEJDOUB

4

Les APIs

- **JavaFX :**

- Créer des interfaces graphiques pour toutes les sortes d'applications (mobiles, web, sur poste de travail, etc);
- 2007-2011: il a été basé sur un langage script spécifique (**JavaFX Script**)
- Essayer de concurrencer Flex (qui se base sur Flash +MXML)
- A partir de 2011, elle est **incluse par défaut dans Java** et devient la **bibliothèque** de création d'interfaces graphiques **officielle** du langage Java ,
- à partir de Java8 (en 2014) elle est directement **accessible via un IDE** (Netbeans, Eclipse, JDeveloper, etc)..
- JavaFX contient des outils très divers, notamment pour les **médias audio et vidéo**, le **graphisme 2D et 3D**, l'**animation**, la **réalité virtuelle**...

Safa MEJDOUB

5

JavaFx-Caractéristiques

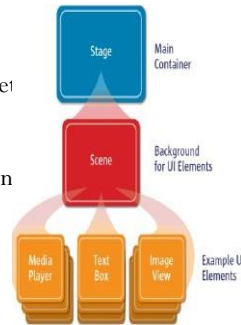
- Abandon du langage de script;
- Choix de deux modes : interfaces basées sur du code Java (**API**) et/ou sur un langage descriptif utilisant une syntaxe XML : **FXML**;
- Création d'un outil interactif **Scene Builder** pour créer graphiquement des interfaces et générer automatiquement du code FXML;
- Utilisation possible de **feuilles de styles CSS** pour adapter la séparation entre le design graphique et les traitements effectués à l'aide de code Java (créer des thèmes, des skins, etc.)

Safa MEJDOUB

6

Comment se présente une application JavaFX ?

- ❑ Une application JavaFx est une classe qui doit hériter de la classe **Application** qui se trouve dans le package `javafx.application`;
- ❑ La **fenêtre** principale d'une application est représentée par un objet de type **Stage**;
en AWT ou Swing cela correspond à une Frame ou JFrame.
C'est le lien avec le système de fenêtre des systèmes d'exploitation
- ❑ L'interface est représentée par un objet de type **Scene** qu'il faut créer et associer à la fenêtre (Stage);
- ❑ La scène est composée des différents éléments de l'interface graphique qui sont des objets de type **Node**;

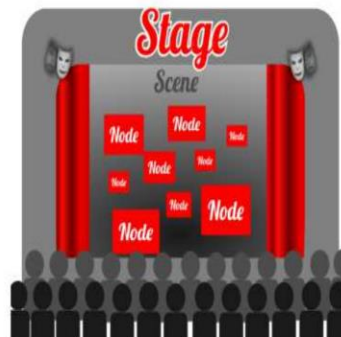


Safa MEJDOUB

7

JavaFx-Eléments principaux

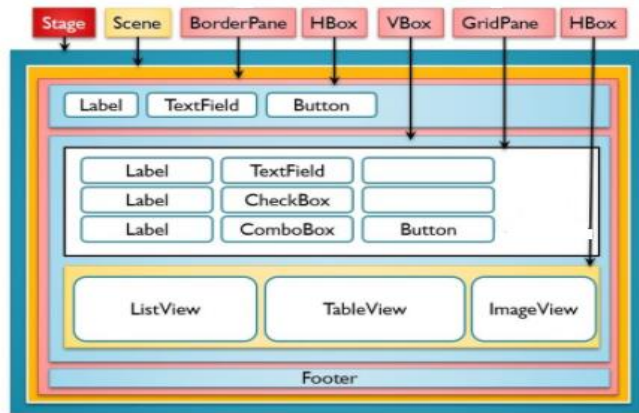
- **Stage** : Endroit où se déroule la scène
- **Scène** : Tableau ou séquence faisant intervenir les acteurs
- **Groupe d'acteurs** : éléments du décor qui font partie de la scène.
Ils représentent les composants graphiques qui peuvent s'animer selon une chronologie
(`javafx.animation.Timeline`)



Safa MEJDOUB

8

Structure d'une Interface JavaFx



Safa MEJDOUB

9

Ma première application JAVA FX

Safa MEJDOUB

10

Stage

- Une application JAVAFX doit **hériter de la classe Application** et **redéfinir** la méthode abstraite **start()** héritée de Application.

```
public abstract void start (Stage stage) throws java.lang.Exception
```

- La méthode start() prend en paramètre un objet de type **Stage**, qui représente la fenêtre de notre application.

Safa MEJDOUB

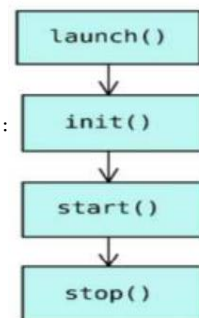
11

Stage

- Une instance de la classe Application constitue le point d'entrée d'une application JavaFX.
- Une fois, l'application se lance par la méthode statique **Application. Launch ()**,

le runtime JavaFX effectue les opérations suivantes :

1. Crée une instance de la classe qui hérite de Application;
2. Appelle la méthode **init()**
3. Appelle la méthode **start()** et lui passe en paramètre une instance de **Stage** (qui représente la fenêtre principale [primary stage])
1. Attend ensuite que l'application se termine; cela se produit lorsque :
 - La dernière fenêtre de l'application a été fermée
 - L'application appelle **Platform.exit()** (ne pas utiliser System.Exit())
5. Appelle la méthode **stop()**



Safa MEJDOUB



```

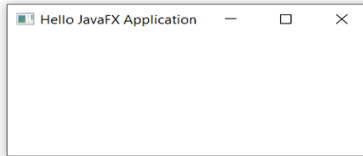
package iset.intro;
import javafx.application.Application;
import javafx.stage.Stage;

public class Hello extends Application {

    public static void main(String[] args) {
        // lancer votre application JavaFX
        Application.launch(args);
    }
    @Override
    public void start(Stage stage) {
        stage.setTitle("Hello JavaFX Application");
        // définir optionnellement les dimensions du
        stage
        stage.setHeight(300);
        stage.setWidth(500);

        //rendre votre fenêtre visible en appelant la méthode show()
        stage.show();
    }
}

```



Safa MEJDOUB


13

Scene

- L'interface est représentée par un objet de type **Scene** qu'il faut créer et associer à la fenêtre (**Stage**) en faisant appel à la méthode **setScene()** de l'objet **Stage**

Safa MEJDOUB

14

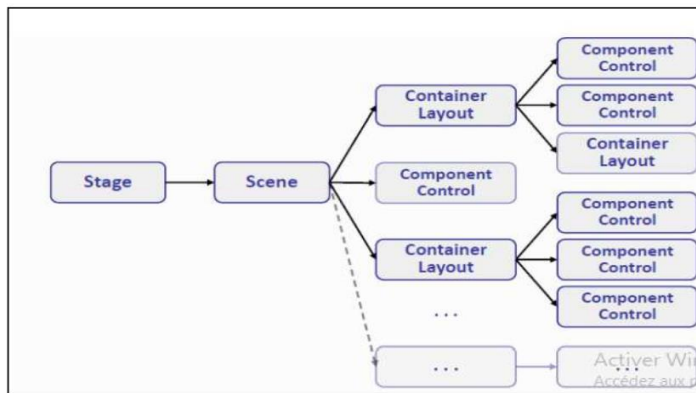


First JavaFX application

Bienvenu dans le monde de JavaFX !!

15

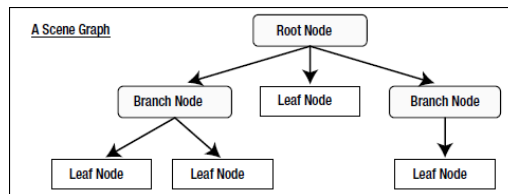
15



16

Graphe de Scène

- ❑ Le graphe de scène (scene graph) est une notion importante qui représente **la structure hiérarchique** de l'interface graphique.
- ❑ Techniquement, c'est un graphe acyclique orienté (arbre orienté) avec :
 - une racine (root)
 - des noeuds (nodes)
 - des arcs qui représentent les relations parent-enfant
- ❑ Les noeuds (nodes) peuvent être de trois types :
 - Racine
 - Noeud intermédiaire
 - Feuille (leaf)

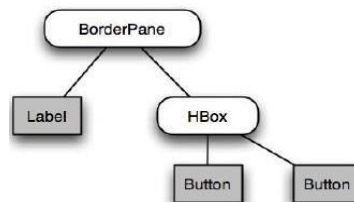


Safa MEJDOUB

Graphe de Scène

- ❑ Les feuilles de l'arbre sont généralement constitués de **composants visibles** (**boutons, champs texte, ...**)
- ❑ les nœuds intermédiaires et le nœud racine sont généralement des éléments de structuration (souvent **invisibles**), typiquement **des conteneurs** de différents types (HBox, VBox, BorderPane, ...).

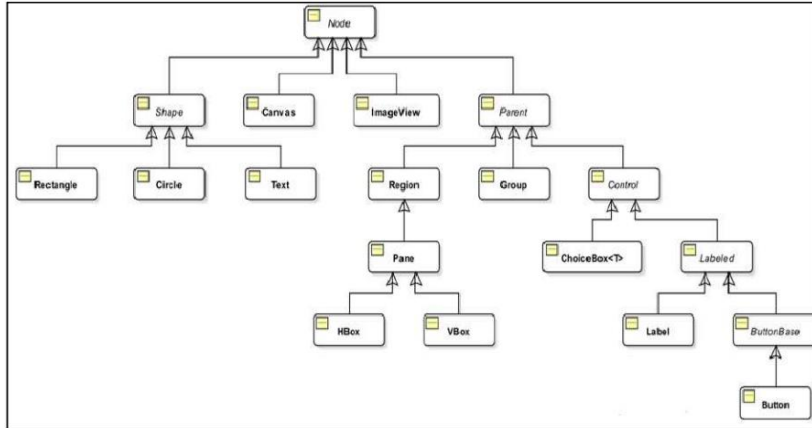
Exemple:



Safa MEJDOUB

18

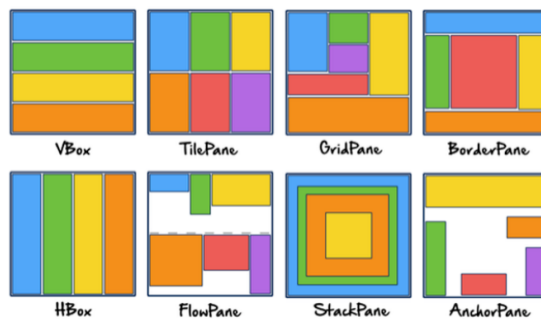
Tous les éléments contenus dans un graphe de scène sont des objets qui ont pour classe parente la classe **Node**.



Safa MEJDOUB

19

Conteneurs Layout-Panes



Safa MEJDOUB

20

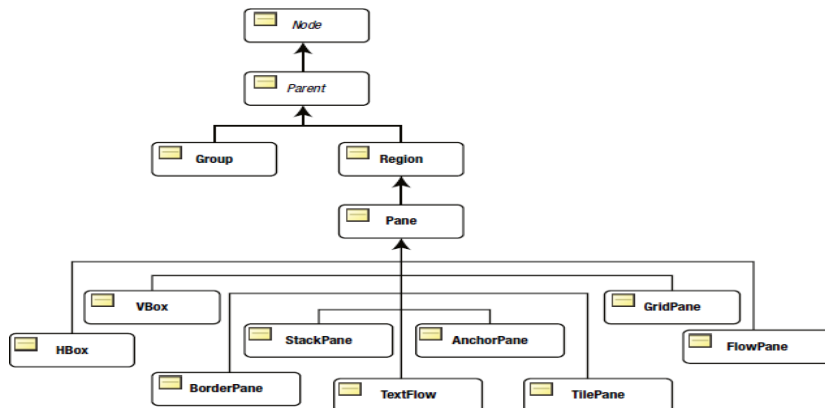
Les Conteneurs

- Dans la création des graphes de scène, les conteneurs (layout-panes) jouent un rôle important dans la structuration et la disposition des composants qui seront placés dans les interfaces.
- Les conteneurs (Layout-Pane) représentent une famille importante parmi les sous-classes de Node.

Safa MEJDOUB

21

- Ils ont pour classe parente **Pane** et **Region** qui possèdent de nombreuses propriétés et méthodes héritées par tous les conteneurs.



Safa MEJDOUB

22

Region - Structure visuelle

- C'est la classe parente des composants (Controls) et des conteneurs (Layout-Panes)
- Elle définit des propriétés qui affectent la **représentation visuelle**.
- Voici un aperçu des concepts principaux associés à la classe Region:

• **Zone de Contenu (Content Area)** : Cette zone contient les composants enfants de la région.

• **Background (Arrière-plan)** : Il s'agit de la couleur ou de l'image affichée en arrière-plan de la région.

• **Padding** : Le padding, également appelé remplissage, est une propriété utilisée pour définir l'espace entre le contenu d'un conteneur (comme un **GridPane**, un **VBox**, un **HBox**, etc.) et ses bords. *Il spécifie la distance entre le bord du conteneur et son contenu.*

• **Border (Bordure)** : La bordure est l'espace autour du padding. Elle peut être utilisée pour définir une bordure autour de la région.

Safa MEJDOUB

23

Region - Structure visuelle

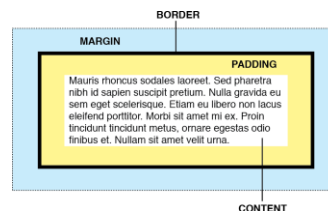
• **Margin** : Le margin est l'espace situé autour de la bordure. Il définit la distance entre les bords de la région et les autres éléments à l'extérieur de celle-ci.

• **Insets** : Les insets sont une classe JavaFX utilisée pour spécifier les marges autour d'un élément graphique. La classe `javafx.geometry.Insets` permet de *définir les marges supérieure, droite, inférieure et gauche* d'un élément graphique.

Elle prend quatre valeurs en pixels pour représenter ces marges dans cet ordre : haut, droite, bas, gauche

- Insets p=New Insets(5); crée des marges intérieures de 5 pixels sur tous les côtés de la région.

- Insets p= New Insets (5,4, 3,2); crée des marges intérieures de 5 pixels en haut, 4 pixels à droite,
- 3 pixels en bas et 2 pixels à gauche de la région.



VBox

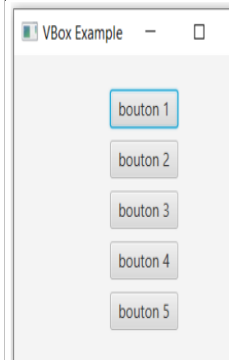
- **VBox** est un conteneur qui range ses sous composants **Verticalement** sur **une seule colonne**.
- L'ajout des composants **enfants** dans le conteneur s'effectue en invoquant d'abord la méthode générale **getChildren()** qui retourne la liste des enfants du conteneur et en y ajoutant ensuite le composant considéré (méthodes **add()** ou **addAll()**)

Safa MEJDOUB

25

```
public class VBoxExample extends Application {  
  
    public void start(Stage primaryStage) throws Exception {  
  
        //VBox place les composants verticalement (dans une colonne).  
        VBox vb=new VBox();  
        Scene scene=new Scene(vb,300,250);  
        //modifier l'espacement entre les composants  
        vb.setSpacing(10);  
        //Centrer les composants dans le VBox  
        vb.setAlignment(Pos.CENTER);  
  
        Button b1=new Button("bouton 1");  
        Button b2=new Button("bouton 2");  
        Button b3=new Button("bouton 3");  
        Button b4=new Button("bouton 4");  
        Button b5=new Button("bouton 5");  
  
        //Ajouter tous les boutons au VBox  
        vb.getChildren().addAll(b1,b2,b3,b4,b5);  
  
        primaryStage.setTitle("VBox Example");  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

Safa MEJDOUB



26

HBox

- HBox est un conteneur qui range ses sous composants **Horizontalement** sur une seule ligne.



Safa MEJDOUB

27

```
public class HBoxExample extends Application {  
  
    public void start(Stage primaryStage) throws Exception {  
  
        //HBox place les composants Horizontalement (sur une ligne).  
        HBox hb=new HBox();  
        Scene scene=new Scene(hb,600, 200);  
        hb.setSpacing(10);  
        hb.setAlignment(Pos.CENTER);  
        Button b1=new Button("bouton 1");  
        Button b2=new Button("bouton 2");  
        Button b3=new Button("bouton 3");  
        Button b4=new Button("bouton 4");  
        Button b5=new Button("bouton 5");  
  
        hb.getChildren().addAll(b1,b2,b3,b4,b5);  
        primaryStage.setTitle("HBox Example");  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
  
    public static void main(String[] args) {  
        Launch(args);  
    }  
}
```

Safa MEJDOUB

28

Group

- Un gestionnaire de mise en forme qui n'applique aucune disposition pour ses sous composants, tous les sous composants sont dans la **position(0,0)**.
- Son rôle est de **rassembler des contrôles** et d'effectuer une certaine tâche.
- Exemple, vous pouvez rassembler 2 boutons Radio mâle et femelle dans un groupe de sexe.

Safa MEJDOUB

29

```
public class GroupExample extends Application {  
    public void start(Stage primaryStage) throws Exception {  
        //Créer un nœud racine de type Group  
        Group root = new Group();  
        //Créer un nœud texte  
        Text msg = new Text("Hello JavaFx !!");  
        //Spécifier la police et la taille du texte  
        msg.setFont(Font.font("Verdana", 20));  
        //Spécifier la couleur du texte  
        msg.setFill(Color.YELLOW);  
        //spécifier l'emplacement du texte  
        msg.setX(200);  
        msg.setY(100);  
        //Créer un bouton Ok  
        Button b1 = new Button("OK");  
        //Ajouter le nœud texte au nœud racine  
        root.getChildren().add(msg);  
        //Ajouter le bouton au nœud racine  
        root.getChildren().add(b1);  
  
        Scene scene = new Scene(root, 600, 200);  
        primaryStage.setTitle("Group Example");  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
  
    public static void main(String[] args) {  
        Launch(args);  
    }  
}
```

Safa MEJDOUB



30

BorderPane

- Il permet de placer les composants enfants dans cinq zones: **Top**, **Bottom**, **Left**, **Right** et **Center**.
- Un **seul** objet Node (composant, conteneur, ...) peut être placé dans chacun de ces emplacements.



Safa MEJDOUB

31

```
public class BorderPaneExample extends Application {

    public void start(Stage primaryStage) throws Exception {

        BorderPane root=new BorderPane();
        Scene scene=new Scene(root,600, 200);

        //la marge autour du BorderPane
        root.setPadding(new Insets(15, 20, 10, 10));

        // Création bouton en Haut
        Button bt1 = new Button("Haut");
        bt1.setPadding(new Insets(10, 10, 10, 10));
        root.setTop(bt1);
        // Définir la marge pour la partie supérieure
        BorderPane.setMargin(bt1, new Insets(10, 10, 10, 10));

        // Création bouton à Gauche
        Button bt2 = new Button("Gauche");
        bt2.setPadding(new Insets(5, 5, 5, 5));
        root.setLeft(bt2);
        //Définir la marge pour la partie gauche
        BorderPane.setMargin(bt2, new Insets(10, 10, 10, 10));

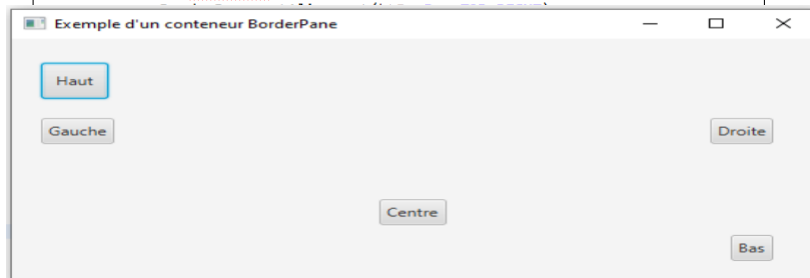
        // Création bouton au Centre
        Button bt3 = new Button("Centre");
        bt3.setPadding(new Insets(5, 5, 5, 5));
        root.setCenter(bt3);
        // Alignement du bouton dans le centre.
        BorderPane.setAlignment(bt3, Pos.BOTTOM_CENTER);
    }
}
```

Safa MEJDOUB

32


```
// Création bouton à Droite
Button bt4 = new Button("Droite");
bt4.setPadding(new Insets(5, 5, 5, 5));
root.setRight(bt4);
// Définir la marge pour la partie droite
BorderPane.setMargin(bt4, new Insets(10, 10, 10, 10));

// Création bouton en Bas
Button bt5 = new Button("Bas");
bt5.setPadding(new Insets(5, 5, 5, 5));
root.setBottom(bt5);
// Alignement du bouton dans la partie basse
```

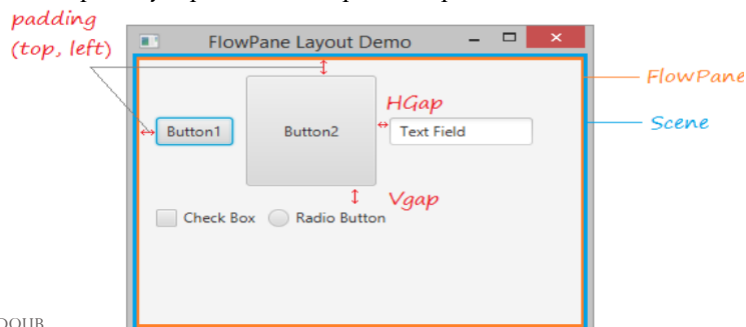


Safa MEJDOUB}}

33

FlowPane

- **FlowPane** place les composants sur une ligne horizontale ou verticale et passse à la ligne ou à la colonne suivante (wrapping) lorsqu'il n'y a plus assez de place disponible.



Safa MEJDOUB

34

```

public class FlowPaneExample extends Application {

    public void start(Stage primaryStage) throws Exception {

        //FlowPane.
        FlowPane fp=new FlowPane();
        Scene scene=new Scene(fp,300, 200);
        //appeler la méthode setHgap et setVgap pour modifier l'espace
        //horizontal et vertical entre les composants
        fp.setHgap(10);
        fp.setVgap(10);
        //Center les composants
        fp.setAlignment(Pos.CENTER);

        Button b1=new Button("bouton 1");
        Button b2=new Button("bouton 2");
        Button b3=new Button("bouton 3");
        Button b4=new Button("bouton 4");
        Button b5=new Button("bouton 5");

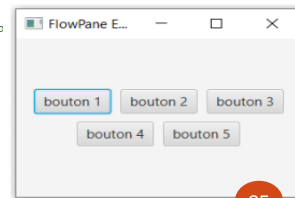
        //ajout des composants aux enfants de FlowP
        fp.getChildren().addAll(b1,b2,b3,b4,b5);

        primaryStage.setTitle("FlowPane Example");
        primaryStage.setScene(scene);
        primaryStage.show();

    }

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

```



35

GridPane

- Permet de disposer les composants enfants dans une **grille flexible** (arrangement en **lignes** et en **colonnes**), un peu à la manière d'une table HTML.
- La grille peut être **irrégulière**, la hauteur des lignes et la largeur des colonnes de la grille ne sont pas nécessairement uniformes.
- La zone occupée par un composant peut s'étendre (**span**) sur plusieurs lignes et/ou sur plusieurs colonnes.

Safa MEJDOUB

36

```

public class GridPaneExample extends Application {
    public void start(Stage primaryStage) throws Exception {
        // GridPane.
        GridPane gp = new GridPane();
        Scene scene = new Scene(gp, 300, 200);
        // appeler la méthode setHgap et setVgap pour modifier
        // l'espace
        // horizontal et vertical entre les composants
        gp.setHgap(10);
        gp.setVgap(10);

        // aligner les composants en bas à droite
        gp.setAlignment(Pos.TOP_CENTER);

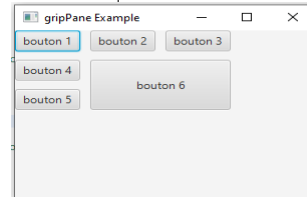
        Button b1 = new Button("bouton 1");
        Button b2 = new Button("bouton 2");
        Button b3 = new Button("bouton 3");
        Button b4 = new Button("bouton 4");
        Button b5 = new Button("bouton 5");
        Button b6 = new Button("bouton 6");

        // ajout des composants au GridPane en précisant
        (child, columnIndex, rowIndex)
        gp.add(b1, 0, 0);
        gp.add(b2, 1, 0);
        gp.add(b3, 2, 0);
        gp.add(b4, 0, 1);
        gp.add(b5, 0, 2);

        //ajout du bouton b6 en précisant (child, columnIndex, rowIndex,
        colspan, rowspan)
        gp.add(b6, 1, 1, 2, 2);
        //Agrandir b6 à la taille maximale
        b6.setMaxSize(Double.MAX_VALUE, Double.MAX_VALUE);
    }
}

```

Safa MEJDOUB



37

StackPane

- Le conteneur **StackPane** permet de disposer les composants enfants **les uns sur les autres** comme dans une pile(Stack).
- on ne peut voir que l'élément qui se situe au-dessus.



Safa MEJDOUB

38

```

public class StackPaneExample extends Application {

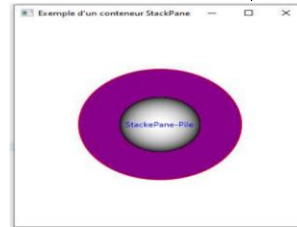
    public void start(Stage primaryStage) throws Exception {

        //créer un conteneur stackpane
        StackPane root= new StackPane();
        Scene scene=new Scene(root,400, 500);

        //dessiner un cercle
        Circle circle=new Circle(300,135,100);
        //attribuer une couleur de fond
        circle.setFill(Color.DARKMAGENTA);
        //attribuer une couleur au contour
        circle.setStroke(Color.RED);
        //dessiner un sphere
        Sphere sph= new Sphere(50);
        // mettre un texte
        Text text=new Text("StackPane-Pile");
        //colorier le texte
        text.setFill(Color.BLUE);
        //changer sa position
        text.setX(20);
        text.setY(50);

        //récupérer la liste des composants de la stackPane
        ObservableList list=root.getChildren();
        //ajouter les noeud au stackPane
        list.addAll(circle,sph,text);
    }
}

```



Safa MEJDOUB

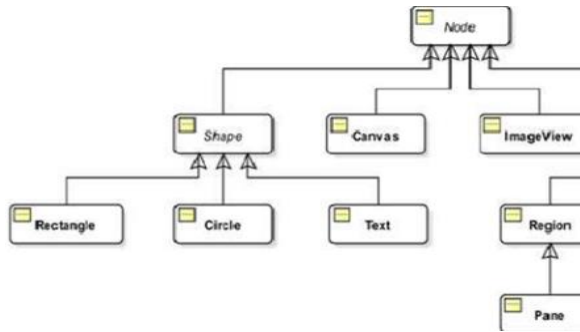
39

Les nœuds

Safa MEJDOUB

40

Nœuds géométriques



Safa MEJDOUB

41

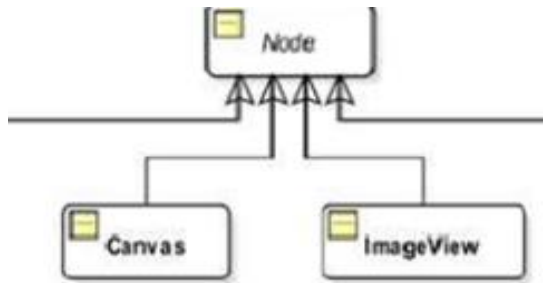
Nœuds géométriques

- Représentent des **formes géométriques** diverses ou du **texte**.
- Tous héritent d'une **classe-mère** commune nommée **Shape**.
- on cite parmi ces classes : **Arc, Circle, Ellipse, Line, Polygon, Rectangle** et **Text**.

Safa MEJDOUB

42

Nœuds graphiques



Safa MEJDOUB

43

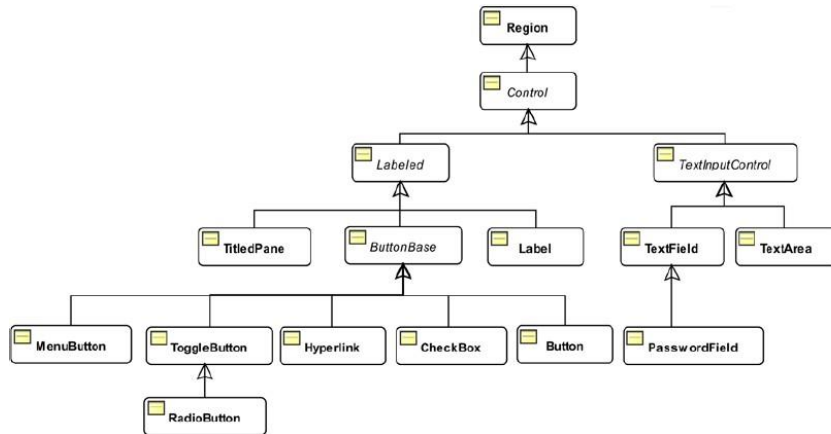
Nœuds graphiques

- Présentent un contenu graphique, qui peut être une **image** ou une **vidéo**.
- Héritent directement de la classe **Node**
- Ces nœuds sont:
 - **ImageView**: affiche une **image**,
 - **MediaView**: affiche une **vidéo**,
 - **Canvas**: affiche une image modifiable, sur laquelle il est possible de **dessiner** des primitives graphiques (lignes, rectangles, etc.).

Safa MEJDOUB

44

Nœuds de contrôle

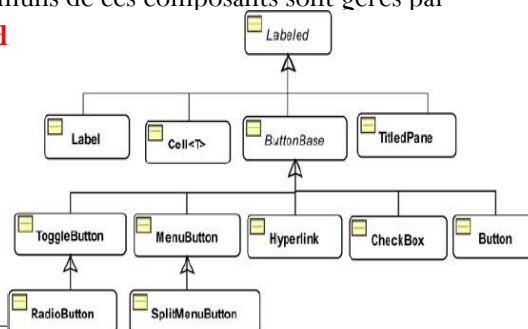


Safa MEJDOUB

45

Les contrôles étiquetés (1)

- Un contrôle étiqueté contient un **contenu textuel en lecture seule** **Étiquette**, **Button**, **CheckBox**, **RadioButton** et **Hyperlink** sont des exemples de contrôles étiquetés dans JavaFX.
- Les comportements communs de ces composants sont gérés par la classe parente **Labeled**



Safa MEJDOUB



Les contrôles étiquetés (2)

- ❑ Les textes de ces composants peuvent être accompagnés d'un autre composant, généralement un graphique, une image ou une icône.
- ❑ Quelques propriétés communes aux composants **Labeled**.

text	Texte affiché (String).
font	Police de caractères (famille, style, taille, ...), type Font .
textFill	Couleur du texte, uniforme ou avec gradient (type Paint).
underline	Indique si le texte doit être souligné (type Boolean).
alignment	Alignement général du texte (et du graphique éventuel) dans la zone (type Pos). Valable seulement si texte sur une seule ligne.
wrapText	Booléen qui définit si le texte passe à la ligne suivante lorsqu'il atteint la limite de la zone. Le caractère '\n' peut également être inséré pour forcer un retour à la ligne (inconditionnel).
textAlignment	Alignement des lignes si le texte est multiligne. Type énuméré TextAlignment (LEFT , RIGHT , CENTER , JUSTIFY).
lineSpacing	Espacement des lignes pour les textes multilignes. Type Double .

Safa MEJDOUB

47

Les contrôles étiquetés (3)

graphic	Autre composant (type Node) qui accompagne le texte. Généralement un graphique, une image ou une icône.
contentDisplay	Position du composant additionnel (<i>graphic</i>) par rapport au texte. Type énuméré ContentDisplay (LEFT , RIGHT , TOP , BOTTOM , TEXT_ONLY , GRAPHIC_ONLY).
graphicTextGap	Espacement entre le texte et le composant additionnel (<i>graphic</i>). Type Double .
mnemonicParsing	Active le <i>parsing</i> des mnémoniques dans le texte (le caractère qui suit le caractère '_'). Type Boolean .
textOverflow	Comportement si le texte est trop long pour être affiché. Type énuméré OverflowStyle (ELLIPSIS , CLIP , ...).
labelPadding *	Définit l'espace autour du texte (et du graphique éventuel). Type Insets .
EllipsisString	Chaîne de caractères utilisée lorsque le texte est tronqué (<i>ellipsis</i>). Par défaut : "..."

* property : Cette couleur est utilisée pour les propriétés en lecture seule (read-only)

Safa MEJDOUB

48

Les contrôles étiquetés (4)-Label

- Le composant Label représente un libellé (= un texte non éditable).
- Les constructeurs permettent de définir le contenu du texte et de l'éventuel composant additionnel (graphic).
 - `new Label("Hello");`
 - `new Label("Warning", warningIcon);`
- L'essentiel des fonctionnalités sont héritées de Labeled. Une seule propriété additionnelle se trouve dans Label :
 - `setLabelFor` : Permet de définir un composant auquel le libellé est associé



Safa MEJDOUB

49

Les contrôles étiquetés (5)- Button

- Les boutons sont des éléments interactifs qui permettent aux utilisateurs d'effectuer des actions lorsque ils sont cliqués.
- La classe parente **ButtonBase** rassemble les propriétés communes à différents composants qui se comportent comme des boutons : `Button`, `CheckBox`, `Hyperlink`, `MenuButton`, `ToggleButton`
- Les constructeurs permettent de définir le contenu du texte et de l'éventuel composant additionnel (graphic).
 - `new Button("Ok");`
 - `new Button("Save", saveIcon);`

Safa MEJDOUB

50

Les contrôles étiquetés (6)- Button

```
public void start(Stage primaryStage) {
    Button btn = new Button();
    VBox root = new VBox(10);
    root.setAlignment(Pos.CENTER);
    root.setPadding(new Insets(20));

    Button btOk=new Button("OK");

    Image im=new Image(this.getClass().getResourceAsStream("im2.jpg"));
    ImageView icone=new ImageView(im);
    Button btLog=new Button("Login",icone);
    btLog.setContentDisplay(ContentDisplay.TOP);
    btLog.setTextFill(Color.BLUE);
    btLog.setGraphicTextGap(10);
    btLog.setFont(Font.font(null, FontWeight.BOLD, 15));
    Button btSave=new Button("Save");
    root.getChildren().addAll(btOk,btLog,btSave);
}
```

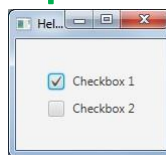


Safa MEJDOUB

51

Les contrôles étiquetés (7)

■ CheckBox



<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/CheckBox.html>

<http://example.com> — unvisited link

■ Hyperlink

<http://example.com> — link is clicked

<http://example.com> — visited link

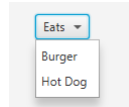
<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Hyperlink.html>

Safa MEJDOUB

52

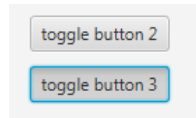
Les contrôles étiquetés (8)

- **MenuButton**



<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/MenuButton.html>

- **ToggleButton**



<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/ToggleButton.html>

Safa MEJDOUB

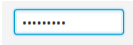
53

Saisie de textes (1)

- Les champs de texte permettent aux utilisateurs de saisir et de modifier du texte.
- La classe abstraite `TextInputControl` est la classe parente de différents composants qui permettent à l'utilisateur de saisir des textes.
- Il s'agit notamment des composants d'interface :



- ✓ `TextField`,



- ✓ `PasswordField`

- ✓ `TextArea`



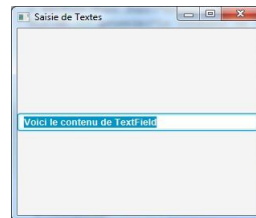
Safa MEJDOUB

54

Saisie de textes (2)

text	Le texte contenu dans le composant (<i>String</i>).
editable	Le texte peut être édité par l'utilisateur (<i>Boolean</i>).
font	Police de caractères du texte (<i>Font</i>).
length	Longueur du texte (<i>Integer</i>).
caretPosition	Position courante du curseur / point d'insertion (<i>caret</i>).
promptText	Texte affiché si aucun texte n'a été défini ou saisi par l'utilisateur (<i>String</i>). Ce texte n'est pas affiché lorsque le composant possède le focus (avec le curseur qui clignote dans le champ). Ce texte peut éventuellement remplacer un libellé ou une bulle d'aide pour le champ texte.
selectedText	Texte sélectionné (<i>String</i>).
selection	Indices (de...à) de la zone sélectionnée (<i>IndexRange</i>).
anchor	Point d'ancrage (début) de la sélection (<i>Integer</i>).

```
TextField t= new TextField();
t.setText("Voici le contenu de TextField");
t.setEditable(false);
t.setFont(Font.getFont("Arial", FontWeight.BOLD, 12));
System.out.println("La longueur de TextField est : "+t.getLength());
System.out.println("La position courante du curseur: "+t.getCaretPosition());
t.setPromptText("ToDo List");
System.out.println("Le text sélectionné: "+t.getSelectedText());
//t.getSelectedText();
```



Safa MEJDOUB

55

Saisie de textes (3)

clear()	Efface le texte (vide le champ).
copy/cut/paste()	Transfert du texte dans ou depuis le <i>clipboard</i> .
positionCaret()	Positionne le curseur à une position donnée.
forward()	Déplace d'un caractère le curseur (<i>caret</i>).
backward()	Déplace le curseur (<i>caret</i>) au début du prochain mot.
nextWord()	Déplace le curseur (<i>caret</i>) au début du prochain mot.
insertText()	Insère une chaîne de caractères dans le texte.
appendText()	Ajoute une chaîne de caractères à la fin du texte.
deleteText()	Supprime une partie du texte (de...à).
deleteNextChar()	Efface le prochain caractère.
replaceText()	Remplace une partie du texte par un autre.
selectAll()	Sélectionne l'ensemble du texte.
deselect()	Annule la sélection courante du texte.

Safa MEJDOUB

56

Mise en forme des interfaces graphiques

57

Safa MEJDOUB

Apparence Graphique

- ❑ La notion de **style**, **skin**, **thème** ou **look and feel** (L&F) caractérise l'ensemble des **aspects visuels** de l'interface graphique et de ses composants (forme, couleur, texture, ombre, police de caractères, ...).
- ❑ En JavaFX, le style des composants est défini par **des feuilles de style de type CSS**.
- ❑ Il est ainsi possible de changer globalement l'aspect de l'interface sans avoir à modifier le code de l'application.
- ❑ Si l'on veut fixer ou changer le *look and feel* des interfaces, on peut le faire au démarrage de l'application :

```
public void start(Stage primaryStage) {  
    ...  
    setUserAgentStylesheet(STYLESHEET_CASPIAN);  
    ...  
}
```

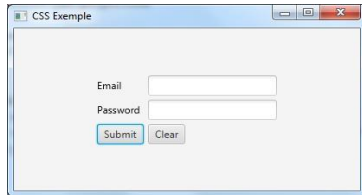
Safa MEJDOUB

58

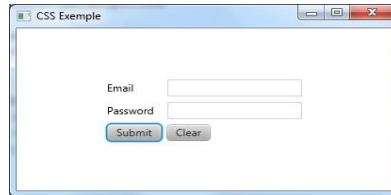
Apparence Graphique

□ Deux styles, nommés **Modena** et **Caspian**, sont prédéfinis et sont associés aux constantes :

- `STYLESET_MODENA` : Utilisé par depuis JavaFX 8
- `STYLESET_CASPIAN` : A été défini pour JavaFX 2



Le style : MODENA



Le style : CASPIAN

Safa MEJDOUB

59

Mise en Forme des GUIs

- Il y a plusieurs manières de mettre en forme une interface graphique JAVA FX :
 - Mise en forme avec du code JAVA
 - Mise en forme CSS inline
 - Mise en forme avec une feuille de style CSS externe

Safa MEJDOUB

60

Mise en forme avec du code JAVA

```
public class JavaStyle extends Application {  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
        primaryStage.setTitle("Mise en forme avec du code java ");  
        BorderPane root = new BorderPane();  
        Scene scene = new Scene(root, 600, 400);  
  
        Label l = new Label("Bienvenu dans le monde de JavaFX !!");  
        // Mise en forme du label  
        l.setPadding(new Insets(50));  
        l.setTextFill(Color.WHITE);  
        l.setFont(Font.font(STYLESHEET_CASPIAN, FontWeight.BOLD, 20));  
        l.setBorder(new Border(new  
        BorderStroke(Color.YELLOW, BorderStrokeStyle.SOLID, new CornerRadii(10), new  
        BorderWidths(5), new Insets(0))));  
  
        // Mise en forme de root  
        Background bg = new Background(  
            new BackgroundFill(Color.BLUE,  
            CornerRadii.EMPTY,  
            null));  
        root.setBackground(bg);  
  
        root.setCenter(l);  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
  
    public static void main(String[] args) {  
        Launch(args);  
    }  
}
```

61

Mise en forme : Inline Style CSS

```
public class CSSInlineStyle extends Application {  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
        primaryStage.setTitle("Inline style example ");  
        BorderPane root = new BorderPane();  
        Scene scene = new Scene(root, 600, 400);  
  
        Label l = new Label("Bienvenu dans le monde de JavaFX !!");  
        // Mise en forme du label  
        l.setStyle(  
            "-fx-text-fill: white; -fx-border-color: yellow; -fx-  
            border-width: 5px; -fx-font-weight: bold; -fx-border-radius: 10px; -fx-padding:  
            50px;");  
  
        // Mise en forme de root  
        root.setStyle("-fx-background-color: blue;");  
  
        root.setCenter(l);  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
  
    public static void main(String[] args) {  
        Launch(args);  
    }  
}
```

62

Mise en forme avec fichier CSS externe

```
public class CSSexternalStyleSheet extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("Exemple avec Feuille de style CSS ");
        BorderPane root = new BorderPane();
        Scene scene = new Scene(root, 600, 400);
        Label l = new Label("Bienvenu dans le monde de JavaFX !!");
        root.setCenter(l);

        //Pour lier la feuille de style à notre scene faire appel à
        getStylesheets().add("URL");

        scene.getStylesheets().add("ressources/css/Styles.css");

        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

S

63

Mise en forme avec fichier CSS externe

Styles.css

```
.root {
    -fx-background-color: blue;
}
.label{
    -fx-text-fill: white;
    -fx-border-color: yellow;
    -fx-border-width: 5px;
    -fx-font-weight: bold;
    -fx-border-radius: 10px;
    -fx-padding: 50px;
}
```

Safa MEJDOUB

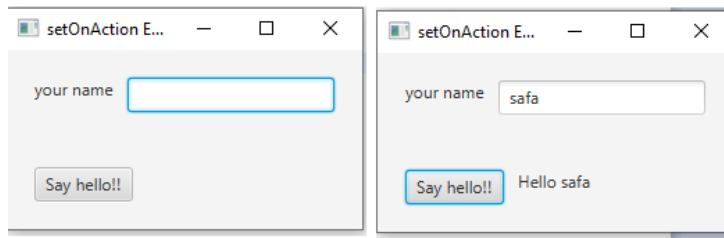
64

Gestion des évènements

65

Safa MEJDOUB

Exemple



66

Safa MEJDOUB

```

@Override
public void start(Stage primaryStage) throws Exception {
    VBox vb=new VBox();
    Scene scene=new Scene(vb);

    HBox hb1=new HBox();
    hb1.setSpacing(10);
    hb1.setPadding(new Insets(20));
    Label l1=new Label("your name");
    TextField tf=new TextField();
    hb1.getChildren().addAll(l1,tf);

    HBox hb2=new HBox();
    hb2.setSpacing(10);
    hb2.setPadding(new Insets(20));
    Button b=new Button("Say hello!!");

    Label l2=new Label();
    hb2.getChildren().addAll(b,l2);

    //Gestion des événements : action sur le bouton b
    b.setOnAction((EventHandler<ActionEvent> new
    EventHandler<ActionEvent>() {

        @Override
        public void handle(ActionEvent event) {
            l2.setText("Hello "+tf.getText());
        }
    }));

    //gestion des événements en utilisant les expressions Lambda(à
    partir de JDK8)
    //b.setOnAction(e-> { l2.setText("Hello "+tf.getText()); } );

    vb.getChildren().add(hb1);
    vb.getChildren().add(hb2);
    primaryStage.setTitle("setOnAction Example");
    primaryStage.setScene(scene);
    primaryStage.show();
}

```

Safa MEJDOUB

67

Technique déclarative FXML VS Technique procédurale

68

Safa MEJDOUB

Interfaces déclaratives vs procédurales

❏ La plateforme JavaFX offre deux techniques complémentaires pour créer les interfaces graphiques des applications :

1) Manière déclarative

- En décrivant l'interface dans un fichier FXML (syntaxe XML)
- L'utilitaire graphique **Scene Builder** facilite la création et la gestion des fichiers FXML
- L'interface peut être créée par un designer (sans connaissance Java, ou presque...)
- Séparation entre présentation et logique de l'application (**MVC**)

2) Manière procédurale

- Utilisation d'API pour construire l'interface avec du code Java
- Création et manipulation dynamique des interfaces
- Il est possible de mélanger les deux techniques au sein d'une même application

Safa MEJDOUB

69

Technique procédurale

```
public class MyJavaFxApplication extends Application{

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

    @Override
    public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("AppFx");
        BorderPane borderPaneRoot= new BorderPane();
        Scene scene=new Scene(borderPaneRoot, 800,600);
        primaryStage.setScene(scene);
        HBox hbox1= new HBox();
        hbox1.setPadding(new Insets(10,10,10,10));
        hbox1.setSpacing(10);
        Label labelNom=new Label("fruit: ");
        labelNom.setPadding(new Insets(5));
        TextField textFieldNom= new TextField();
        Button buttonAdd=new Button("Ajouter");
        hbox1.getChildren().addAll(labelNom,textFieldNom, buttonAdd );
    }
}
```

Safa MEJDOUB

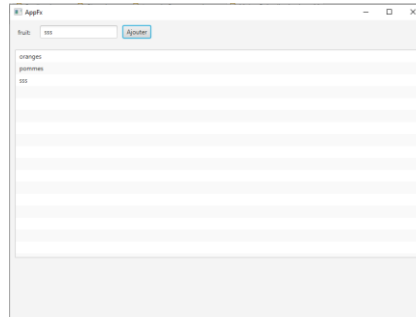
70

```
//creation de la liste
VBox vbox1= new VBox();
vbox1.setPadding(new Insets(10));

ObservableList<String> observableList= FXCollections.observableArrayList();
ListView<String> listView1= new ListView<>(observableList);
observableList.addAll("oranges", "pommes" );
//listView1.getItems().addAll("oranges", "pommes");
vbox1.getChildren().add(listView1);

BorderPaneRoot.setTop(hBox1);
BorderPaneRoot.setCenter(vBox1);
primaryStage.show();

buttonAdd.setOnAction(e-> {
    String nom=textFieldNom.getText();
    //listView1.getItems().add(nom);
    observableList.add(nom); });}
```



Safa MEJDOUB

71

Technique déclarative

```
package gui;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;

public class MyJavaFxApplication2 extends Application {

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

    @Override
    public void start(Stage primaryStage) throws Exception {
        BorderPane borderPaneRoot=
        FXMLLoader.Load(getClass().getResource("Layout.fxml"));
        Scene sene=new Scene(borderPaneRoot,600,400);
        //lier la scene au fichier css
        sene.getStylesheets().add(getClass().getResource("MyStyle.css").toString());
        primaryStage.setScene(sene);
        primaryStage.show();
    }
}
```

72

Layout.fxml

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

<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.Button?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.control.ListView?>
<?import javafx.collections.FXCollections?>
<?import java.lang.*?>

<BorderPane xmlns:fx="http://javafx.com/fxml/1" fx:controller="gui.LayoutController">
    <top>
        <HBox spacing="10">
            <padding>
                <Insets top="10" right="10" bottom="10" left="10"></Insets>
            </padding>
            <Label text="Fruit: "></Label>
            <TextField fx:id="textFieldNom"></TextField>
            <Button text="Ajouter" onAction="#addFruit"></Button>
            <Button text="Supprimer" styleClass="myButton"></Button>
        </HBox>
    </top>
    <center>
        <VBox spacing="10">
            <padding>
                <Insets top="10" right="10" bottom="10" left="10"></Insets>
            </padding>
            <ListView fx:id="listView1">
                <items>
                    <FXCollections fx:factory="observableArrayList">
                        <String fx:value="orange" />
                        <String fx:value="pomme" />
                    </FXCollections>
                </items>
            </ListView>
        </VBox>
    </center>
</BorderPane>
```

S.

73

layoutController.java

```
package gui;

import javafx.fxml.FXML;
import javafx.scene.control.ListView;
import javafx.scene.control.TextField;

public class layoutController {
    @FXML TextField textFieldNom;
    @FXML ListView<String> listView1;

    public void addFruit()
    {
        String nom=textFieldNom.getText();
        listView1.getItems().add(nom);
    }
}
```

myStyle.css

```
myStyle.css

.button{
    -fx-background-color: orange;
    -fx-border-color: blue;
    -fx-border-width: 3;
}

.myButton{
    -fx-background-color: yellow;
    -fx-border-color: red;
    -fx-border-width: 4;
}
```

Safa MEJDOUB

74



Safa MEJDOUB