
JAVA

JavaFX

JavaFx

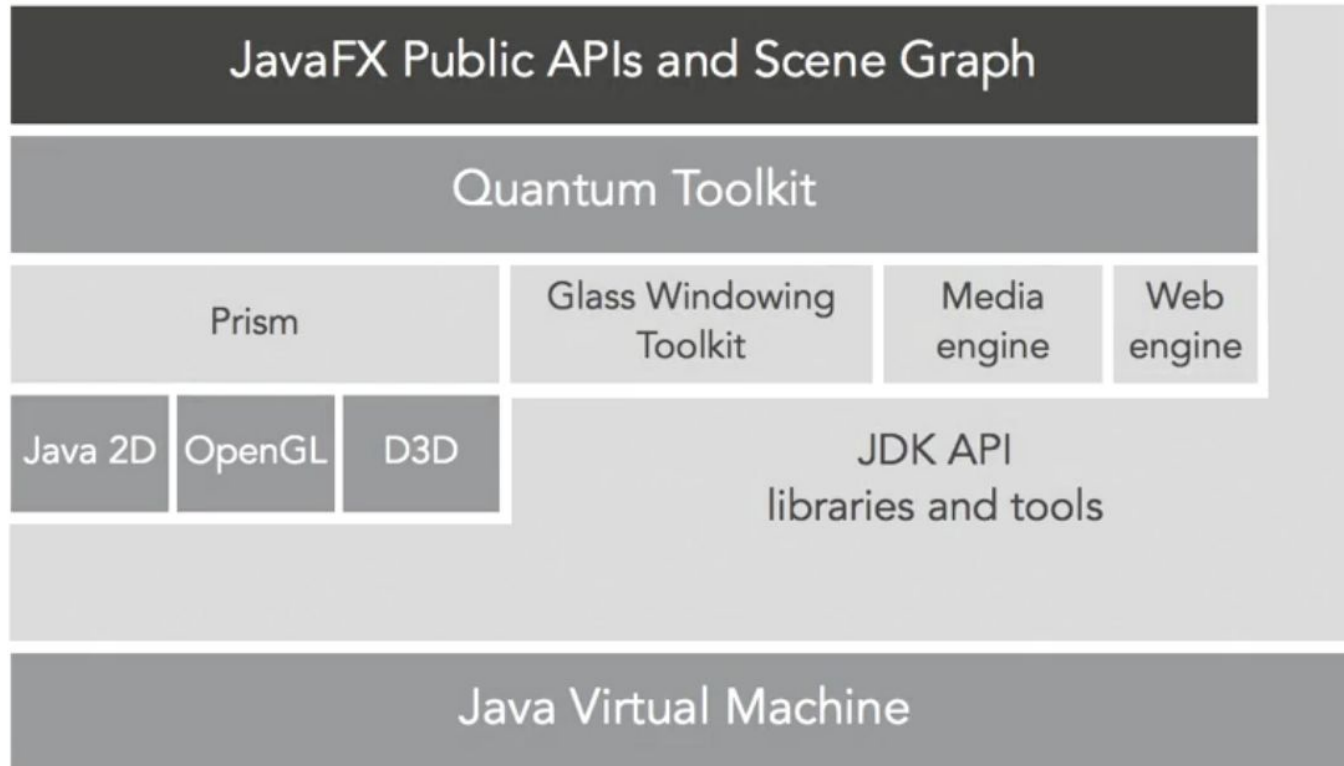
- Possède une librairie écrite comme java API
- Repaint automatique
- Utilise le MVC model
- Support l'utilisation de XML et CSS
- Comme en HTML, CSS est utilisé pour ajouter de style à une application
javaFx

JavaFx

- Separation présentation/logique en utilisant le ***FXML***
- **FXML** : XML-based Markup Language
- Simple Integration avec des app java Swing existantes
- Meilleure performance hardware
- Installé automatiquement avec la JRE

=> JavaFX est le nouveau SWING

JavaFX : Architecture



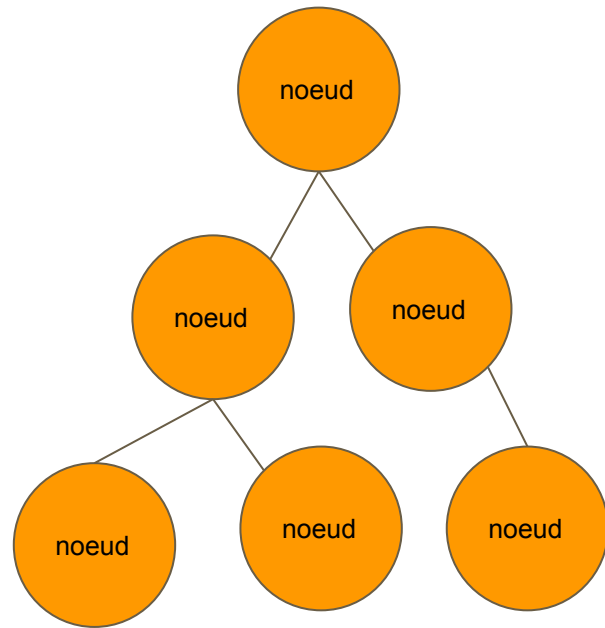
Les Threads JavaFX

- Thread de l'application JavaFx
 - Le thread principale d'une application javaFx
 - Different du thread java Swing
- Prism render Thread
 - Separation rendering/events
- Media Thread
 - Exécuté en arrière plan et assure la synchronisation des dernière Frames

JavaFx Scene Graph

Chaque noeud possède:

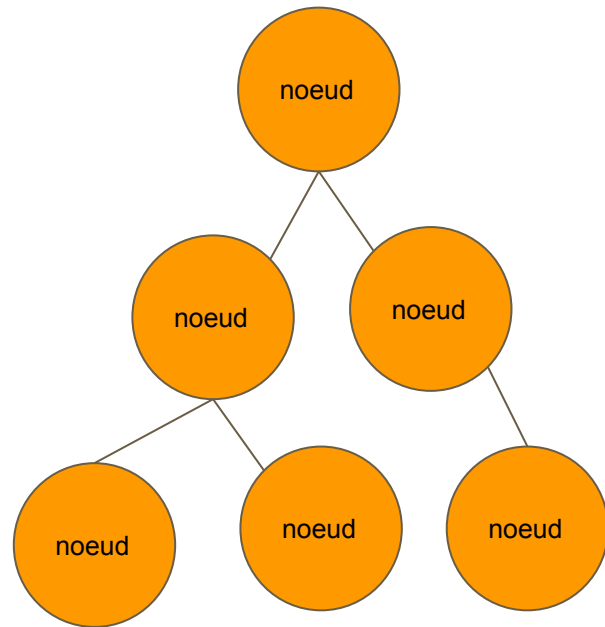
- un ID, classe de style et des dimensions
- Des Effets : blur, shadow ...
- Opacite
- Transformations
- Event handlers
- ...



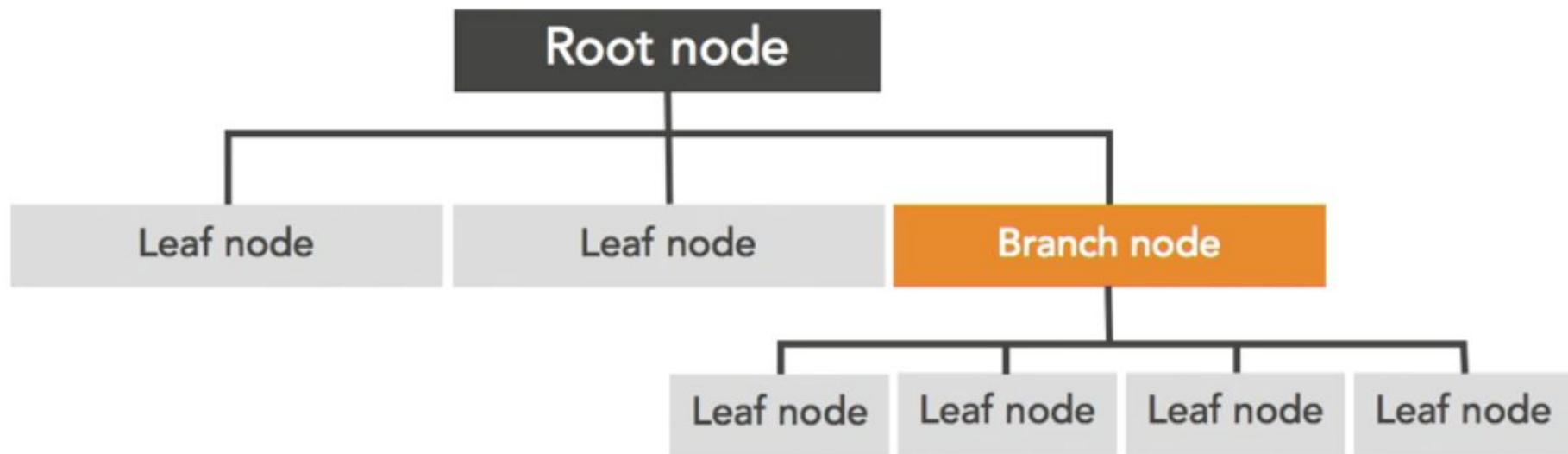
JavaFx Scene Graph

A Chaque noeud on peut ajouter:

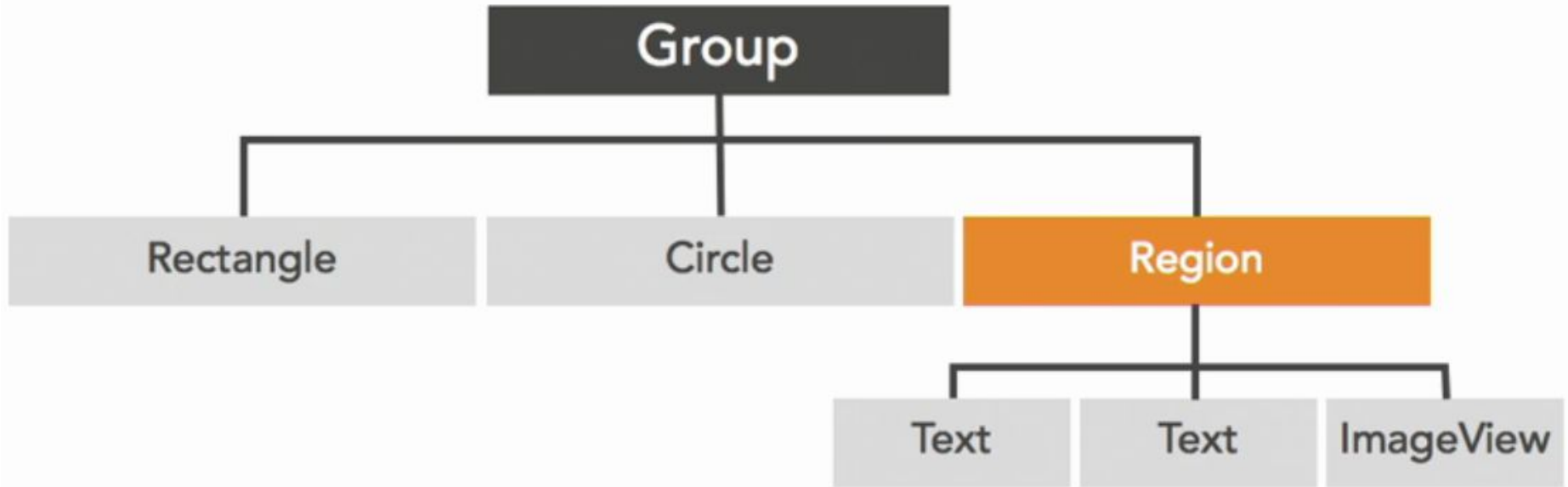
- Du texte, UI controls, un navigateur web
- Des formes 2D 3D, des image et des medias
- Des graphiques, des conteneurs, groupe
- ...



JavaFx Scene Graph



JavaFx Scene Graph:



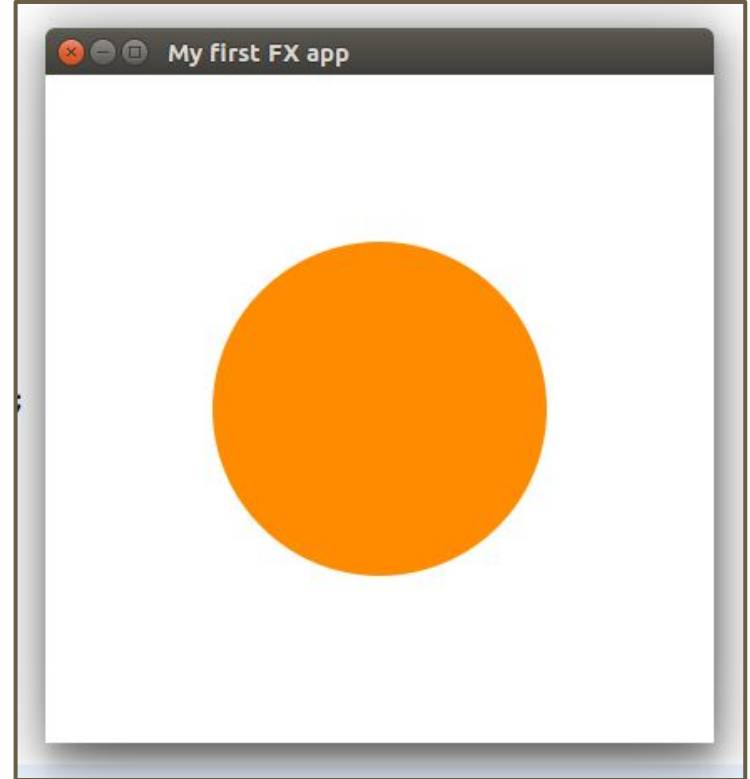
JavaFx Scene Graph: Exemple

Chaque application javaFx :

- étend la classe ***Application***
- Possède un Primary stage
- Un primary Stage contient au moins une scène
- Chaque scène contient un noeud racine (root node)

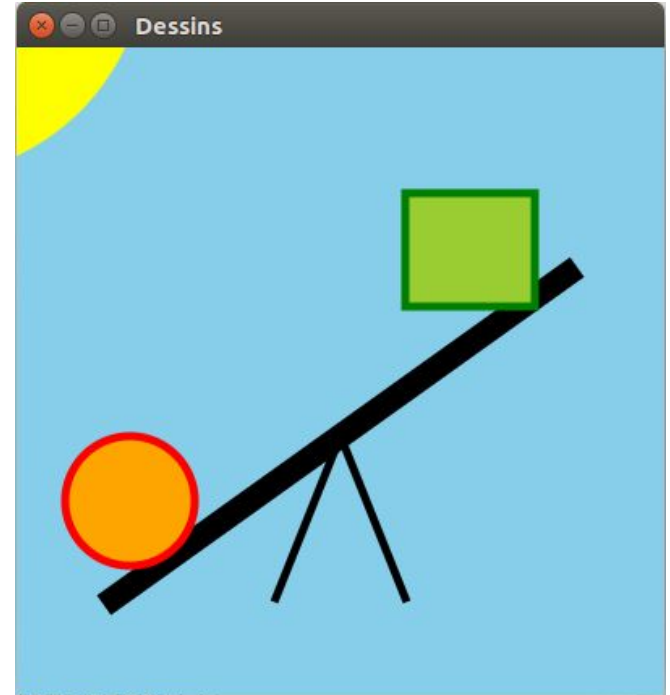
JavaFx Scene Graph: Exemple

```
3 import javafx.application.Application;
9
10 public class Main extends Application {
11     @Override
12     public void start(Stage primaryStage) {
13         try {
14             BorderPane root = new BorderPane();
15             Circle c=new Circle(200,200,100);
16             c.setFill(Color.DARKORANGE);
17             root.getChildren().add(c);
18             Scene scene = new Scene(root,400,400);
19             primaryStage.setTitle("My first FX app");
20             primaryStage.setScene(scene);
21             primaryStage.show();
22         } catch (Exception e) {
23             e.printStackTrace();
24         }
25     }
26
27     public static void main(String[] args) {
28         launch(args);
29     }
30 }
```



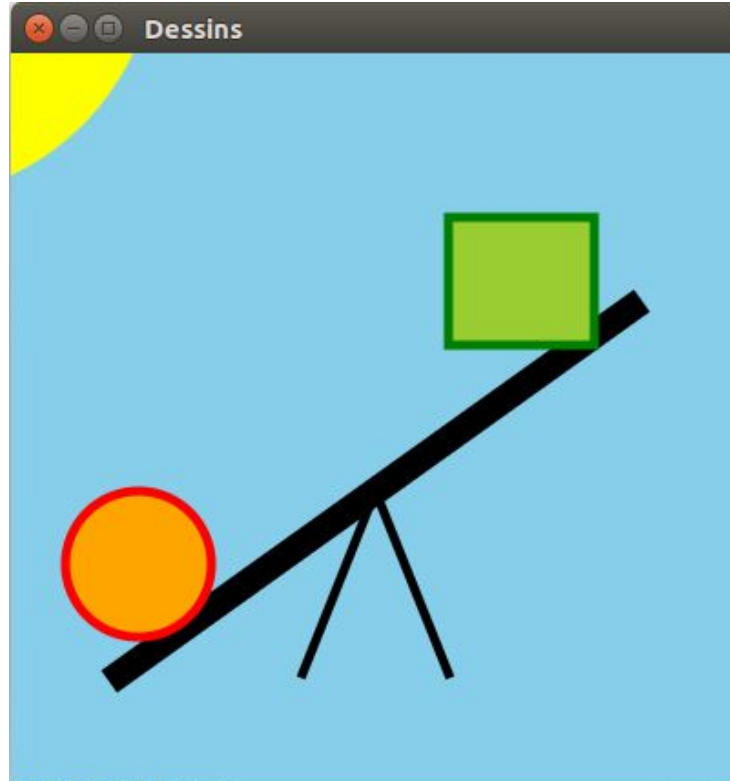
JavaFx: Formes geometrique

- Ligne
 - `Line(x1,y1,x2,y2)`
 - `setStroke(Color)`
 - `setStrokeWidth(15)`
- Circle
 - `Circle(x,y,r)`
 - `setStroke(Color)`
 - `setFill(Color)`
- Rectangle
 - `Rectangle(x,y,w,h)`
 - `setFill(Color)`
 - ...



Formes geometriques:

- exercice



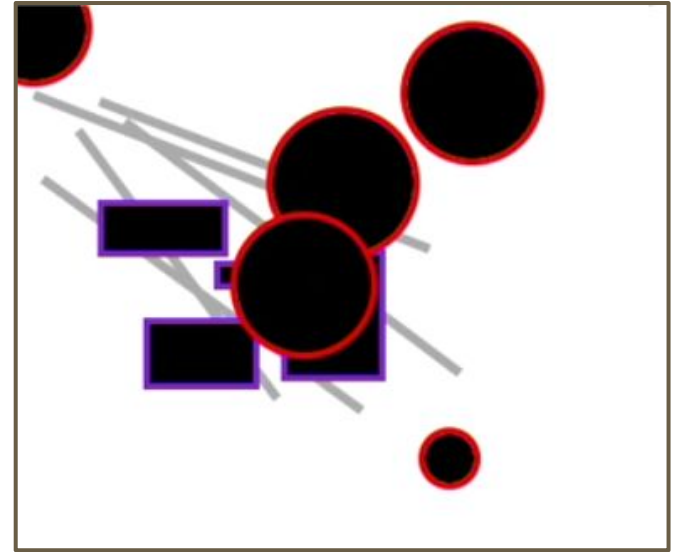
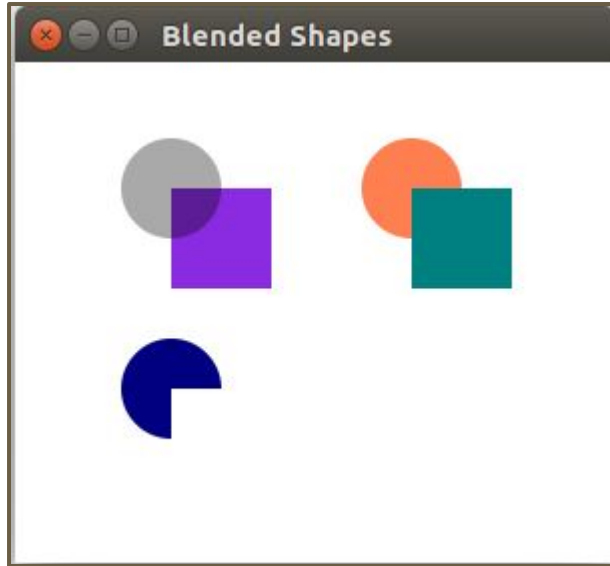
Formes geometriques: Modes de superposition

- ADD : top+bottom
- MULTIPLY
- SRC_ATOP
- SRC_OVER
- SCREEN



Formes géométriques: Modes de superposition

Exercice



La classe Text

- Herite de la classe Node
- Supporte l'application de:
 - Effet speciaux
 - Transformations
 - Animations
- Utiliser du CSS pour ajouter du style



Text

```
Text t = new Text();  
t.setCache(true);  
t.setFill(Color.FIREBRICK);  
t.setText("DropShadow ");  
t.setFont(Font.font("null", FontWeight.BOLD, 32));  
DropShadow ds=new DropShadow();  
ds.setOffsetX(3.0);  
ds.setOffsetY(3.0);  
ds.setColor(Color.GRAY);  
t.setEffect(ds);
```



DropShadow

Text

```
Text t = new Text();  
t.setCache(true);  
t.setText("Reflection ...");  
t.setFill(Color.CORNFLOWERBLUE);  
t.setFont(Font.font("null", FontWeight.BOLD, 30));  
  
Reflection r = new Reflection();  
r.setFraction(0.9);  
t.setEffect(r);  
t.translateY(50);
```



Reflection ...
REFLECTION ...

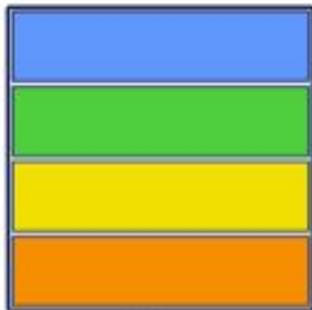
Text

```
Text t = new Text();  
t.setX(20.0f);  
t.setY(65.0f);  
t.setText("perspective");  
t.setFill(Color.BROWN);  
t.setFont(Font.font("null", FontWeight.BOLD, 36));  
t.setEffect(pt);  
t.setCache(true);
```

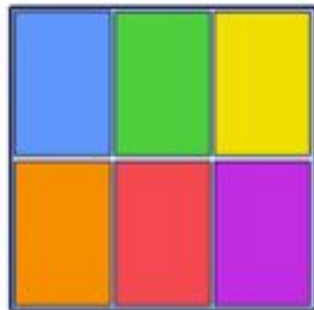


perspective

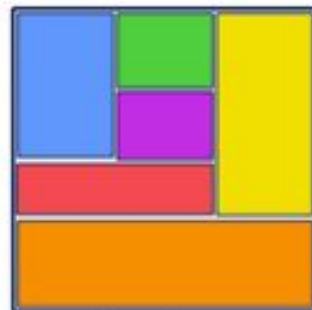
JavaFx Layout API



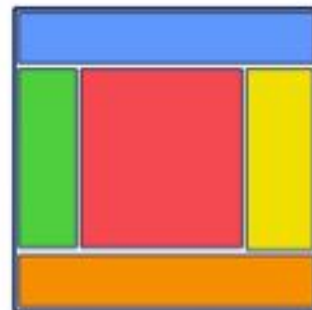
VBox



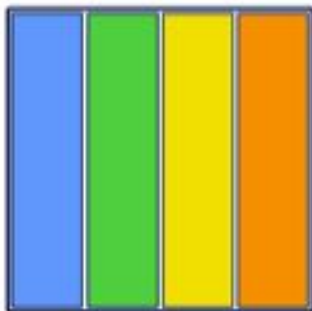
TilePane



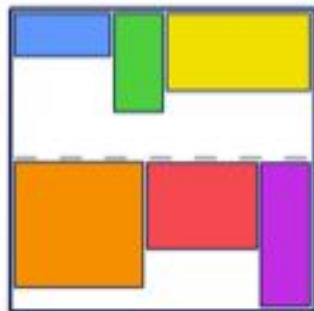
GridPane



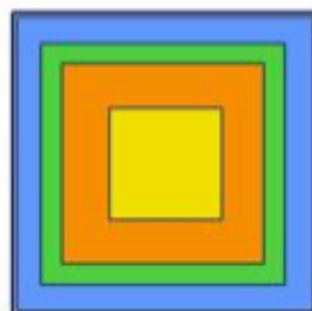
BorderPane



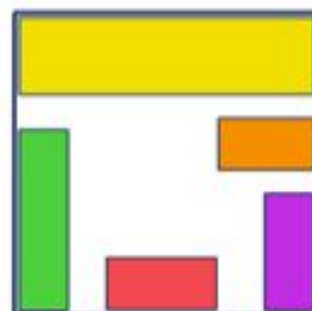
HBox



FlowPane



StackPane

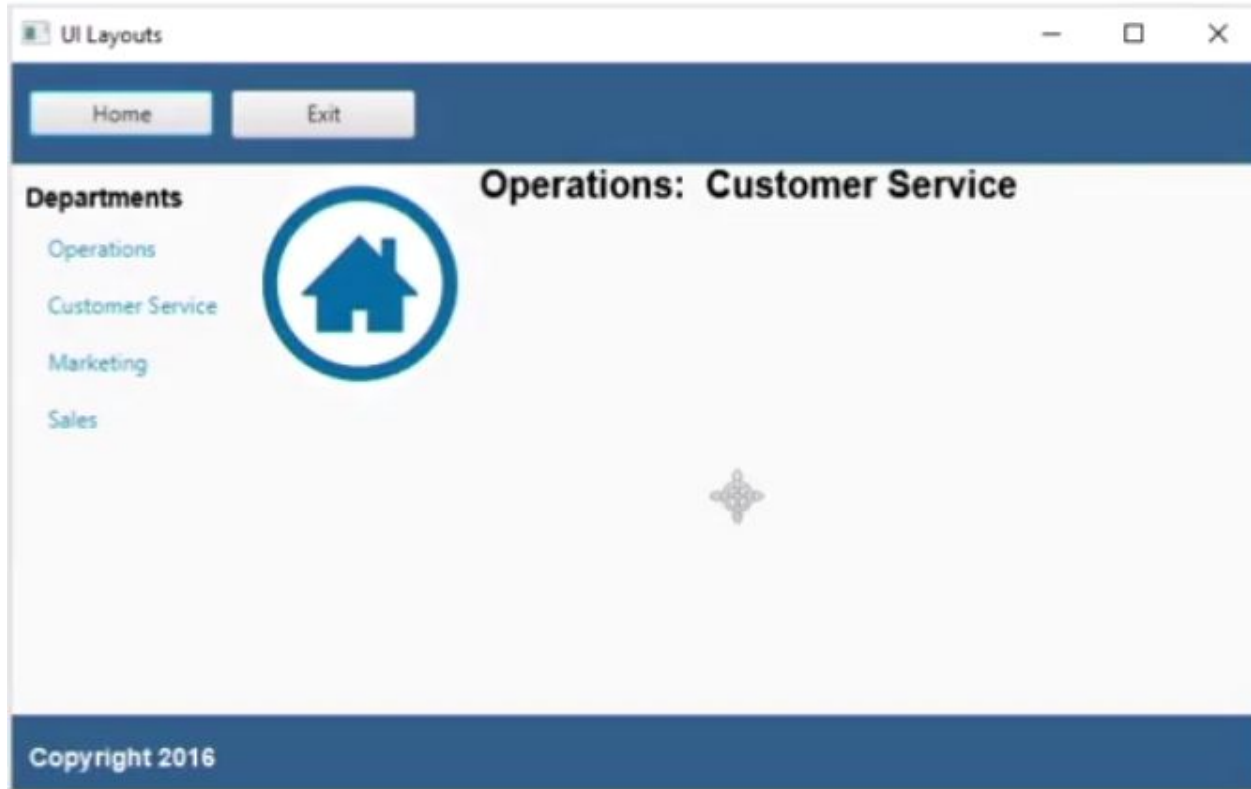


AnchorPane

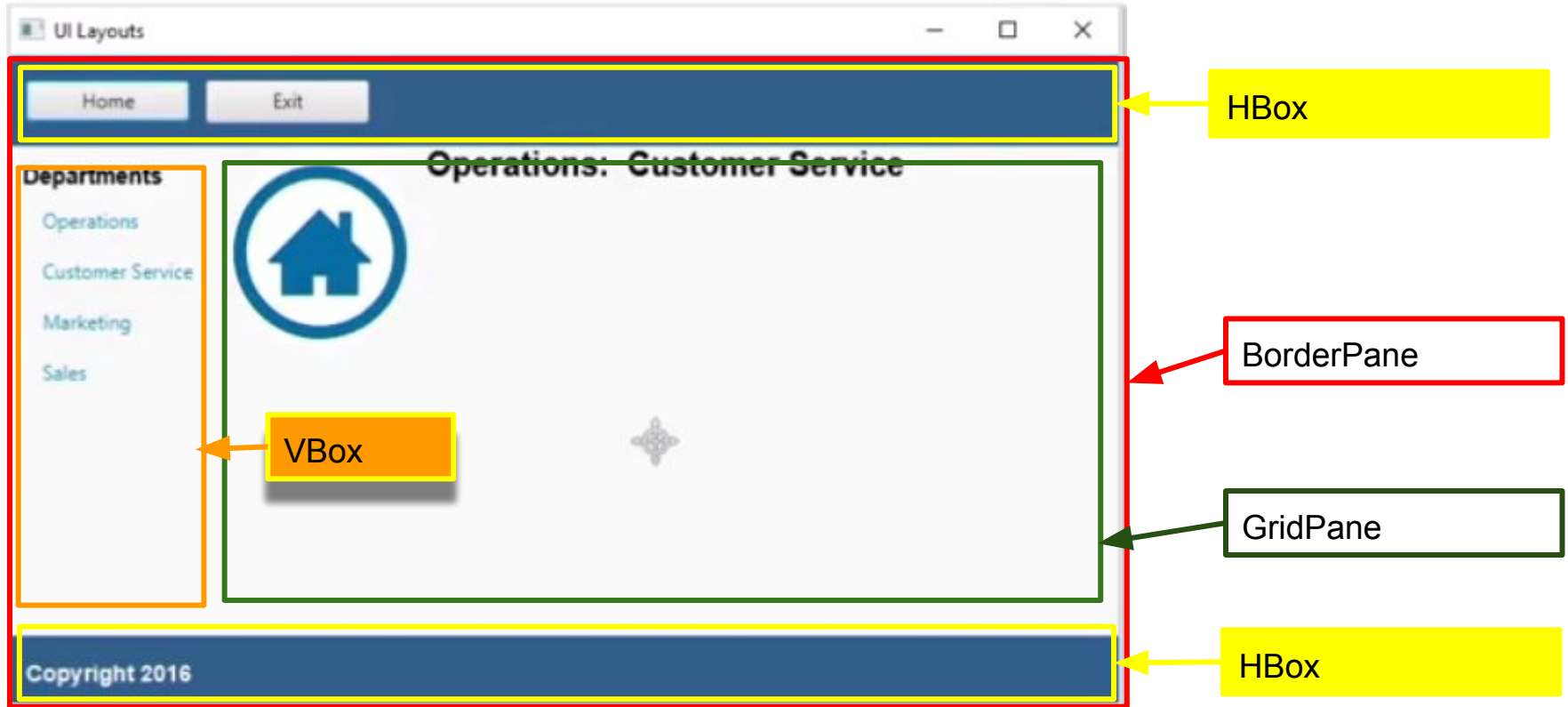
Comment choisir un Layout pane

- Où doit apparaître chaque composantes ?
- Pour quel device est destine notre app ?
- La fenêtre peut être redimensionnée ?
- A-t-on besoin d'imbriquer les Layout panes ?

Comment choisir un Layout pane



Comment choisir un Layout pane



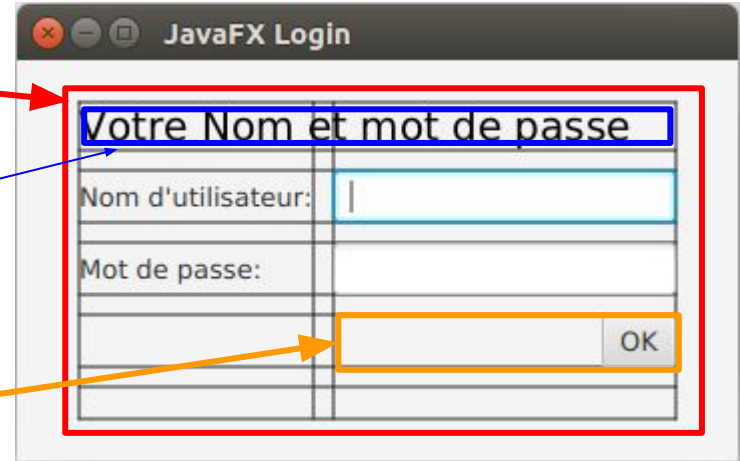
Les Layout panes

```
GridPane grid = new GridPane();
grid.setVgap(10);
grid.setHgap(10);
grid.setAlignment(Pos.CENTER);
Scene scene = new Scene(grid, 360, 200);
Text titre = new Text("Votre Nom et mot de passe ");
titre.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
Label userLbl = new Label("Nom d'utilisateur:");
TextField userTF = new TextField();
Label pwLbl = new Label("Mot de passe: ");
PasswordField pwTF = new PasswordField();
Button btn = new Button("OK");
grid.add(titre, 0, 0, 2, 1);
grid.add(userLbl, 0, 1);
grid.add(userTF, 1, 1);
grid.add(pwLbl, 0, 2);
grid.add(pwTF, 1, 2);
HBox hbBtn = new HBox();
hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
hbBtn.getChildren().add(btn);
grid.add(hbBtn, 1, 3);
final Text msgT = new Text();
grid.add(msgT, 1, 4);
btn.setOnAction(event -> {msgT.setText("Bonjour " + userTF.getText());});
primaryStage.setTitle("JavaFX Login");
primaryStage.setScene(scene);
primaryStage.show();
```



Les Layout panes

```
GridPane grid = new GridPane();
grid.setVgap(10);
grid.setHgap(10);
grid.setAlignment(Pos.CENTER);
Scene scene = new Scene(grid, 360, 200);
Text titre = new Text("Votre Nom et mot de passe ");
titre.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
Label userLbl = new Label("Nom d'utilisateur:");
TextField userTF = new TextField();
Label pwLbl = new Label("Mot de passe: ");
PasswordField pwTF = new PasswordField();
Button btn = new Button("OK");
grid.add(titre, 0, 0, 2, 1);
grid.add(userLbl, 0, 1);
grid.add(userTF, 1, 1);
grid.add(pwLbl, 0, 2);
grid.add(pwTF, 1, 2);
HBox hbBtn = new HBox();
hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
hbBtn.getChildren().add(btn);
grid.add(hbBtn, 1, 3);
final Text msgT = new Text();
grid.add(msgT, 1, 4);
btn.setOnAction(event -> {msgT.setText("Bonjour " + userTF.getText());});
primaryStage.setTitle("JavaFX Login");
primaryStage.setScene(scene);
primaryStage.show();
```



Pour afficher la grille:

`grid.setGridLinesVisible(true);`

Pour ajouter un espace entre les cellules:

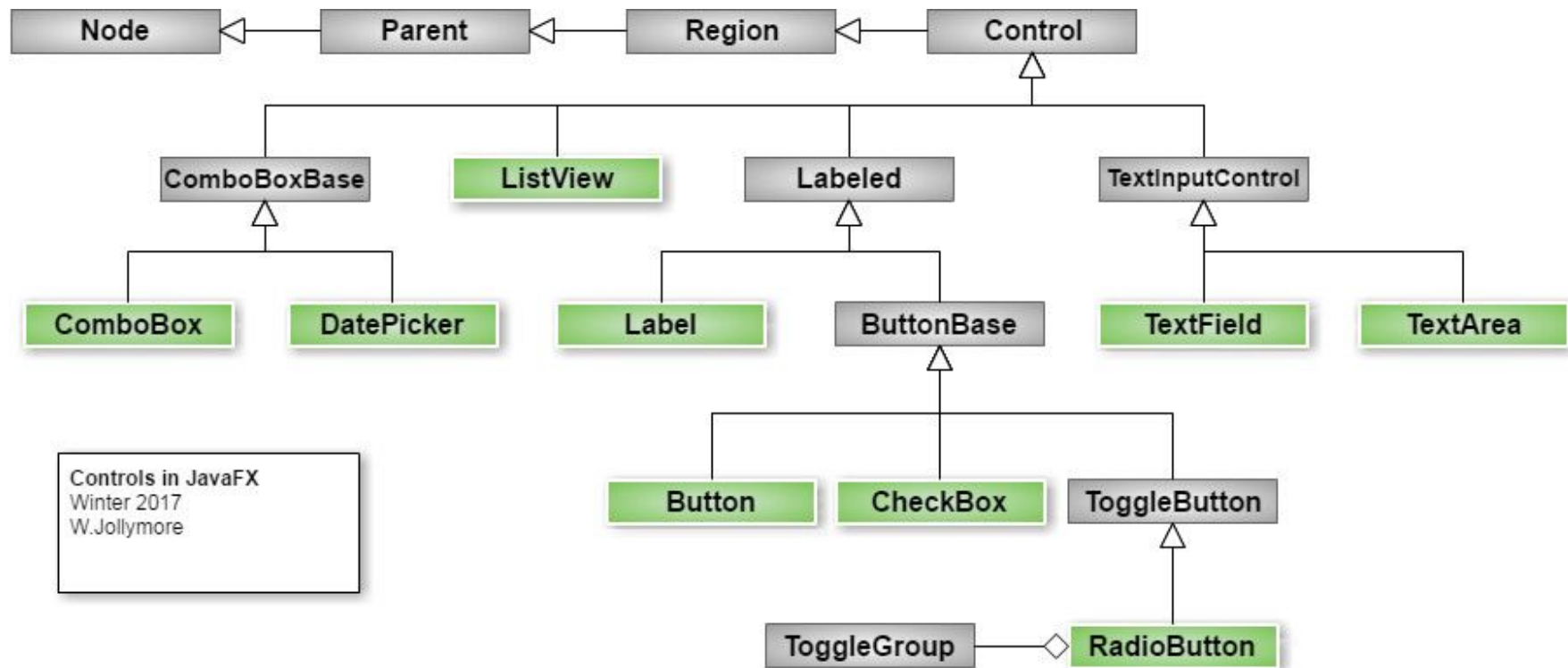
`grid.setVgap(10);`

`grid.setHgap(10);`

JavaFx UI Controls

- *javafx.scene.control*
- Apparence gerable a l aide du CSS
- Garantie l'interaction HM
- Evenements gere a l'aide des ***Events Handlers***

JavaFx UI Controls



JavaFX Event

- JavaFx support un nombre important d'événements
- ***Event*** est La classe de base du package ***javafx.event***
- Les evenement javaFX sont regroupe on:
 - InputEvent
 - ActionEvent
 - WindowEvent

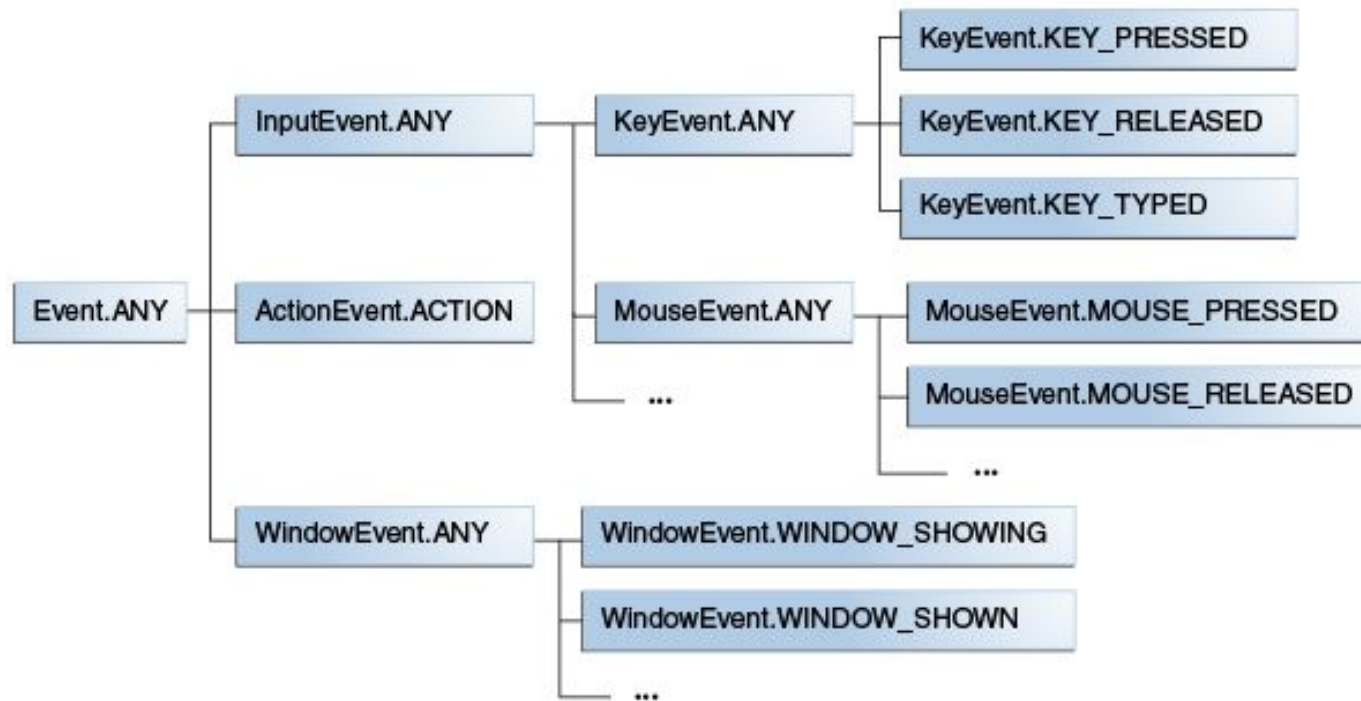
JavaFx Event Handling

JavaFX dispose d'un ensemble de Handlers pour gerer les evenements

Chaque evenement possede:

- ***target***: Le noeud qui a subit l'action
- ***Source***: La source de l'événement souris, clavier ...
- ***Type***: le type de l'événement (mouse click)

JavaFX Event Handlers



JavaFX Event Handlers

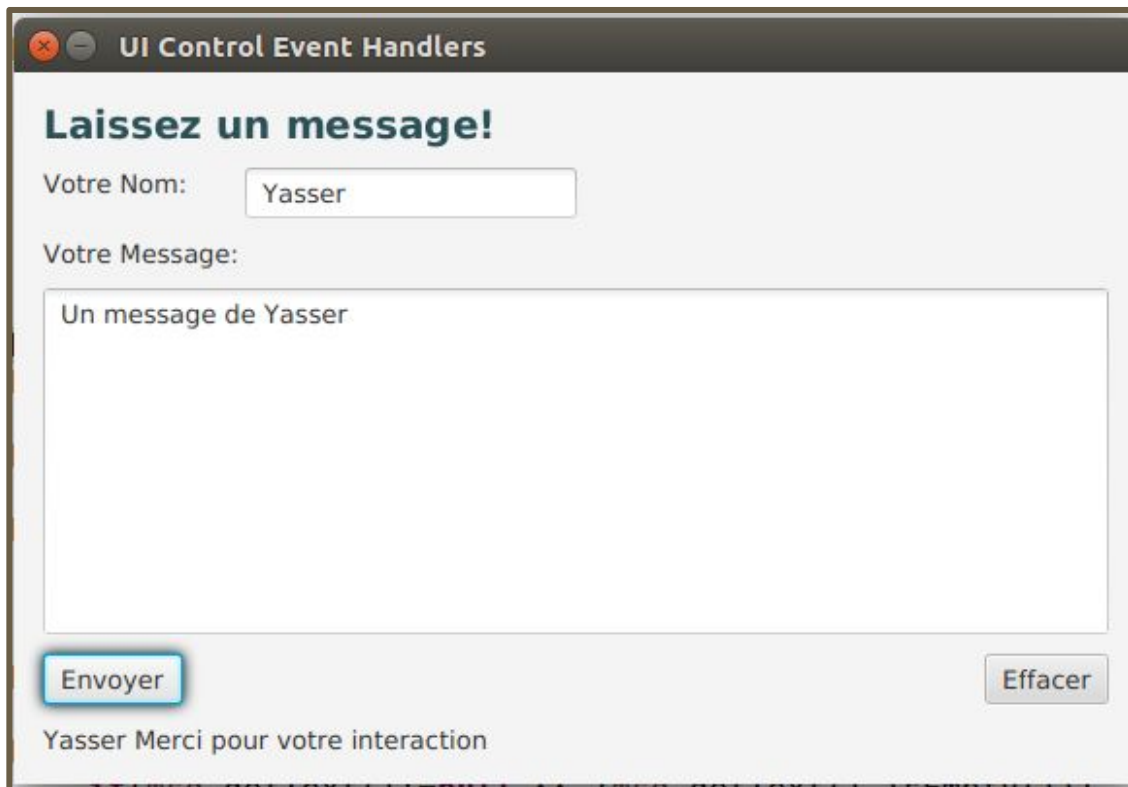
```
//Créer l'eventHandler
EventHandler<MouseEvent> eventHandler =
    new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent e) {
            System.out.println("Bonjour");
        }
    };

//Attacher l'eventHandler au noeud
noeud.addEventHandler(MouseEvent.MOUSE_CLICKED, eventHandler);
```

Ou en utilisant les fonction lambda

```
noeud.addEventHandler(MouseEvent.MOUSE_CLICKED, (MouseEvent e) -> {
    System.out.println("Bonjour");
});
```

JavaFX Event : exercice



The screenshot shows a JavaFX application window titled "UI Control Event Handlers". The window has a light gray background and a dark gray title bar. Inside the window, there is a heading "Laissez un message!" in bold dark blue text. Below the heading, there are two input fields: "Votre Nom:" with the text "Yasser" and "Votre Message:" with the text "Un message de Yasser". At the bottom left, there is a blue button labeled "Envoyer". At the bottom right, there is a gray button labeled "Effacer". Below the buttons, there is a status bar that says "Yasser Merci pour votre interaction".

UI Control Event Handlers

Laissez un message!

Votre Nom:

Votre Message:

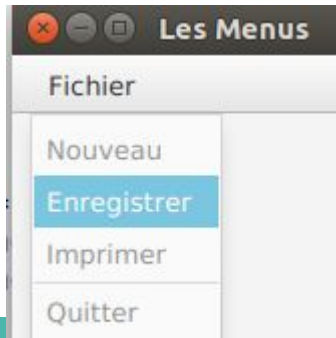
Yasser Merci pour votre interaction

Les Menus

- Identifier les éléments d'une barre de menu
- Chaque element contient des objets Menu
- Un objet Menu peut contenir des sous menu
- On peut insérer des séparateurs entre les menus
- Un Menu n'est une sous classe de classe Node
 - ➡ On ne peut pas les insérer directement dans la Scène il faut passer par un MenuBar

Les Menus

1. Créer un MenuBar
2. Créer le Menu
3. Créer les éléments du Menu
4. Ajouter ces éléments au menu
5. Associer des EventHandler aux éléments
6. Ajouter le Menu au MenuBar
7. Insérer le menuBar dans la BorderPan



```
public void start(Stage primaryStage) {  
    BorderPane root = new BorderPane();  
    Scene scene = new Scene(root, 300, 300, Color.WHITE);  
  
    MenuBar menuBar = new MenuBar(); 1  
  
    Menu menuFichier = new Menu("Fichier"); 2  
    MenuItem nvItm = new MenuItem("Nouveau");  
    MenuItem svItm = new MenuItem("Enregistrer");  
    MenuItem prtItm = new MenuItem("Imprimer");  
    MenuItem qtItm = new MenuItem("Quitter");  
    SeparatorMenuItem sp = new SeparatorMenuItem();  
    menuFichier.getItems().addAll(nvItm, svItm, prtItm, sp, qtItm); 3  
  
    qtItm.setOnAction(actionEvent -> Platform.exit()); 4  
    menuBar.getMenus().add(menuFichier); 5  
    root.setTop(menuBar); 6  
  
    primaryStage.setTitle("Les Menus");  
    primaryStage.setScene(scene);  
    primaryStage.show();  
}
```

Les Menus: Sous-menus

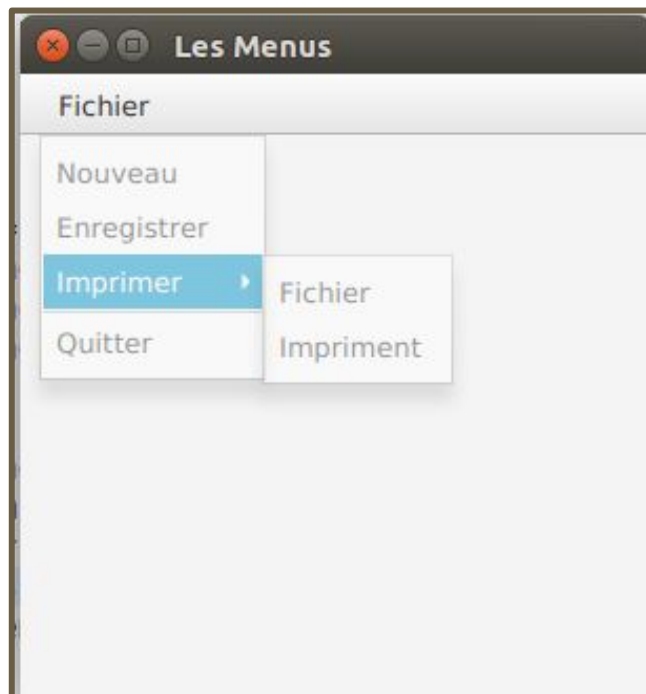
```
MenuBar menuBar = new MenuBar();

Menu menuFichier = new Menu("Fichier");
MenuItem nvItm = new MenuItem("Nouveau");
MenuItem svItm = new MenuItem("Enregistrer");
MenuItem qtItm = new MenuItem("Quitter");
SeparatorMenuItem sp=new SeparatorMenuItem();

Menu menuPrint = new Menu("Imprimer");
MenuItem toFileItm = new MenuItem("Fichier");
MenuItem toPrinterItm = new MenuItem("Imprimer");
menuPrint.getItems().addAll(toFileItm,toPrinterItm);
menuFichier.getItems().addAll(nvItm, svItm,menuPrint, sp, qtItm);

qtItm.setOnAction(actionEvent->Platform.exit());
menuBar.getMenus().add(menuFichier);
root.setTop(menuBar);

primaryStage.setTitle("Les Menus");
primaryStage.setScene(scene);
primaryStage.show();
```



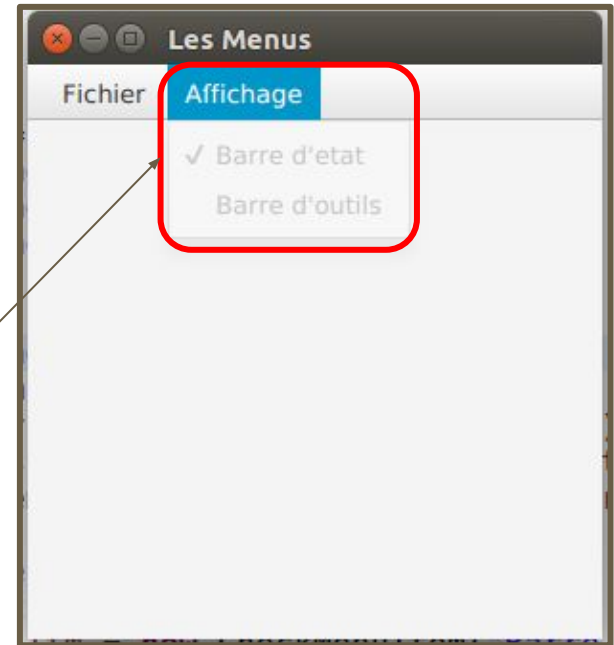
Les Menus: CheckMenuItem

```
Menu menuFichier = new Menu("Fichier");
MenuItem nvItm = new MenuItem("Nouveau");
MenuItem svItm = new MenuItem("Enregistrer");
MenuItem qtItm = new MenuItem("Quitter");
SeparatorMenuItem sp=new SeparatorMenuItem();

Menu menuPrint = new Menu("Imprimer");
MenuItem toFileItm = new MenuItem("Fichier");
MenuItem toPrinterItm = new MenuItem("Imprimer");
menuPrint.getItems().addAll(toFileItm,toPrinterItm);
menuFichier.getItems().addAll(nvItm, svItm,menuPrint, sp, qtItm);

Menu menuAffichage = new Menu("Affichage");
CheckMenuItem etatItm = new CheckMenuItem("Barre d'etat");
CheckMenuItem toolItm = new CheckMenuItem("Barre d'outils");
etatItm.setSelected(true);
menuAffichage.getItems().addAll(etatItm,toolItm);

qtItm.setOnAction(actionEvent->Platform.exit());
menuBar.getMenus().addAll(menuFichier, menuAffichage);
root.setTop(menuBar);
```



ListView

- Affiche une liste d'éléments .
- Affichage horizontal ou vertical.
- Les éléments sont représentés par un objet de la class classe ***ObservableList*** .
- Cet objet est obtenue par ***FxCollections.observableArrayList()*** .
- Attacher un ***javafx.scene.control.SelectionModel*** pour gérer la sélection des éléments.

ListView: basic

```
ObservableList<String> jours =  
    FXCollections.observableArrayList("Lundi", "Mardi",  
    "Mercredi", "Jeudi", "Vendredi", "Samedi",  
    "Dimanche");  
  
lvJours = new ListView<String>(jours);  
lvJours.setPrefSize(200,150);
```

```
root.getChildren().addAll(titre,lvJours,rep);
```



ListView: Events

```
ObservableList<String> jours =  
    FXCollections.observableArrayList("Lundi", "Mardi",  
    "Mercredi", "Jeudi", "Vendredi", "Samedi",  
    "Dimanche");  
  
lvJours = new ListView<String>(jours);  
lvJours.setPrefSize(200,150);  
  
MultipleSelectionModel<String> lvSelModel = lvJours.getSelectionModel();  
lvSelModel.selectedItemProperty().addListener(new ChangeListener<String>() {  
    public void changed(ObservableValue<? extends String> selected,  
        String oldVal, String newVal)  
    {  
        rep.setText("Vous avez choisi: " + oldVal);  
    }  
});  
  
root.getChildren().addAll(titre,lvJours,rep);
```



TableView

- Affiche un Tableau de valeurs (lignes & colonnes)
- Necessite un Data Model (les objet Beans)
- Représente chaque objet dans une ligne
- Trie automatiquement intégré



TableView: Basic

```
ObservableList<Contact> contactList = FXCollections.observableArrayList(  
    new Contact("Said", "Touzani", "06 67 67 67 67"),  
    new Contact("Badre", "Farah", "06 65 62 03 07"),  
    new Contact("Inas", "Khiam", "06 24 33 56 34"),  
    new Contact("yahya", "Douli", "06 33 56 43 67"),  
    new Contact("Kamal", "sarif", "06 45 34 00 43"),  
    new Contact("Hanane", "Rifaai", "06 22 34 28 23") );
```

```
TableView<Contact> tvContacts= new TableView<Contact>(contactList);
```

```
TableColumn<Contact, String> prnm = new TableColumn<>("Prenom");  
prnm.setCellValueFactory(new PropertyValueFactory<>("prenom"));  
tvContacts.getColumns().add(prnm);
```

```
TableColumn<Contact, String> nom = new TableColumn<>("Nom");  
nom.setCellValueFactory(new PropertyValueFactory<>("nom"));  
tvContacts.getColumns().add(nom);
```

```
TableColumn<Contact, String> num = new TableColumn<>("Telephone");  
num.setCellValueFactory(new PropertyValueFactory<>("num"));  
tvContacts.getColumns().add(num);
```

```
tvContacts.setPrefWidth(300);  
tvContacts.setPrefHeight(160);
```



TableView: Events

```
TableView(tableViewSelectionModel<Contact> tvSelContact =  
    tvContacts.getSelectionModel());  
  
tvSelContact.selectedModelProperty().addListener(new ChangeListener<Number>()  
{  
    public void changed(ObservableValue<? extends Number> selected,  
        Number oldVal, Number newVal) {  
        int index = (int)newVal;  
        rep.setText("Num sélectionné : "  
            +contactList.get(index).getNum());  
    }  
});
```

