UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Développement des Interfaces Homme Machine (IHM)

Nourhène BEN RABAH

Docteur en Informatique

2019/2020

Adresse mail : Nourhene.Ben-Rabah@univ-paris1.fr

Événement dans JavaFx (1)

- Un **événement** (Event) dans une application à interface graphique est l'occurrence d'une interaction entre l'utilisateur et l'application;

  Exemple : un clic avec la souris, la pression avec une touche de clavier, le déplacement d'une fenêtre, un geste sur une écran tactile, etc.

- Un événement dans JavaFx est représenté par un objet de la classe Javafx.event.Event ou de l'une de ses sous-classes;

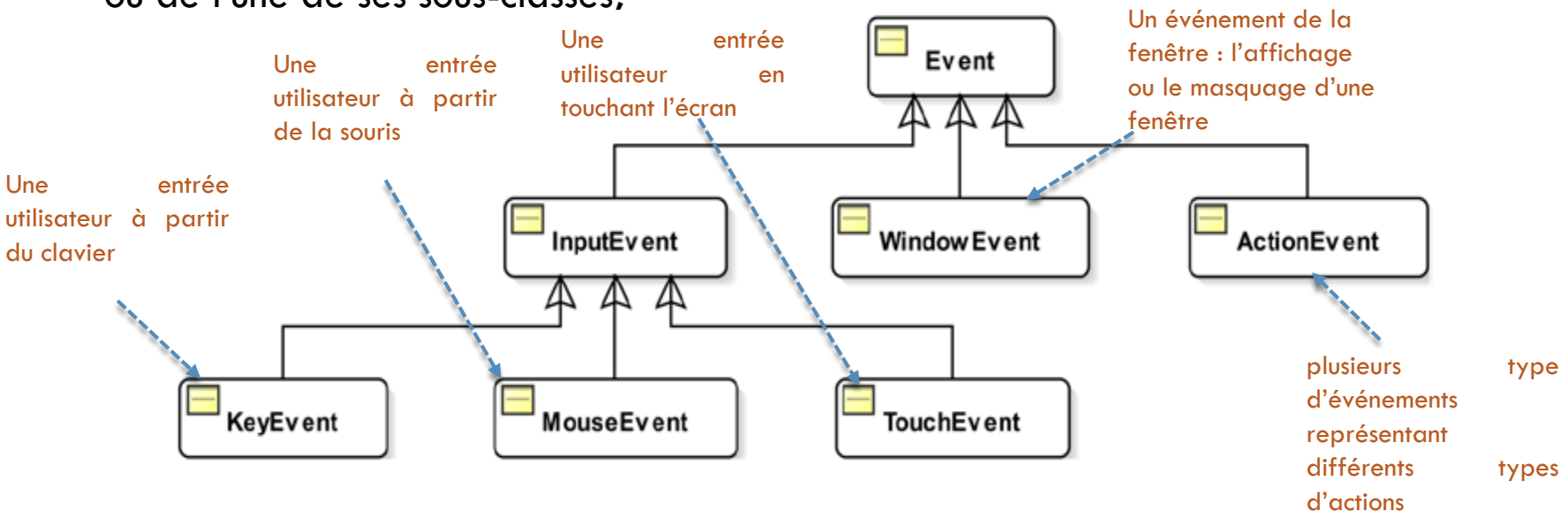


Figure 1. Un diagramme de classe partiel de la classe Javafx.event.Event

Événement dans JavaFx (2)

- Chaque événement possède trois propriétés :

a) La source de l'événement :

La source à partir de laquelle l'événement est généré.

b) La cible de l'événement :

La cible de l'événement est un élément de l'interface utilisateur qui répond à un événement. Une cible peut être une fenêtre, une scène ou un nœud.

c. Le type de l'événement

Il décrit le type de l'événement qui s'est produit. Il permet de classifier les événements à l'intérieur d'une même classe.

Exemple : la classe KeyEvent englobe KEY_PRESSED, KEY_RELEASED, KEY_TYPED

Exemple

- Supposons que nous avons une application qui a deux boutons : Play et Stop et un cercle inséré en utilisant un objet Groupe comme suit:

E1: Je clique sur le bouton Play, le cercle rouge s'affiche

- La source de l'événement : la souris
- La cible de l'événement : le bouton
- Le type de l'événement : Mouse clicked

Mouse clicked, Mouse pressed et Mouse released sont des types de MouseEvent

Événement dans JavaFx (3)

- Ces trois propriétés communes à tous les événements sont représentées par des objets de différentes classes :

Tableau 1: Les classes impliquées dans le traitement des événements

Nom	Classe/Interface	Description
Event	Classe	Une instance de cette classe représente un événement. Plusieurs sous-classes de Event existent pour représenter des événements spécifiques
EventTarget	Interface	Une instance de cette interface représente la cible de l'événement
EventType	Classe	Une instance de cette classe représente un type d'un événement.
EventHandler	Interface	Une instance de cette interface représente un gestionnaire d'événement « Event Handler » ou un « événement filtre ».

La cible de l'événement

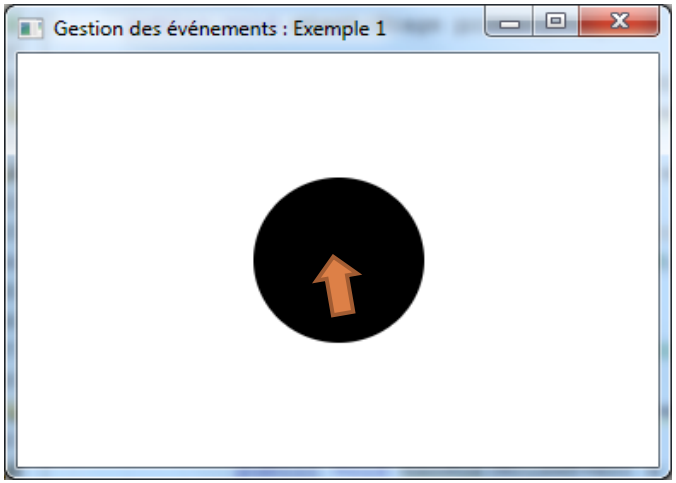
- La cible de l'événement est un élément de l'interface utilisateur qui répond à un événement;
- L'élément qui veut répondre aux événements doit implémenter l'interface «EventTarget»
- Les classes Window, Scene et Node implementent l'interface « EventTarget »

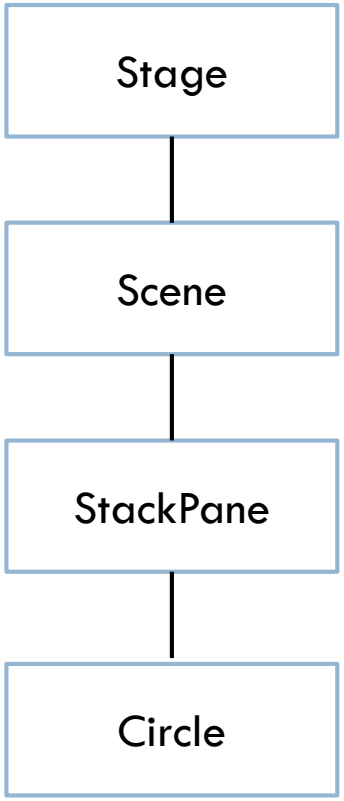
- La responsabilité principale de la cible de l'événement est de construire **une chaine de traitement de l'événement (Event Dispatch Chain) :** chemin de l'événement dans le graphe de la scène.

- Le chemin commence de la racine (Stage) jusqu'à le composant cible en parcourant tous les nœuds intermédiaires

La cible de l'événement : Event Dispatch Chain

Exemple 1 :

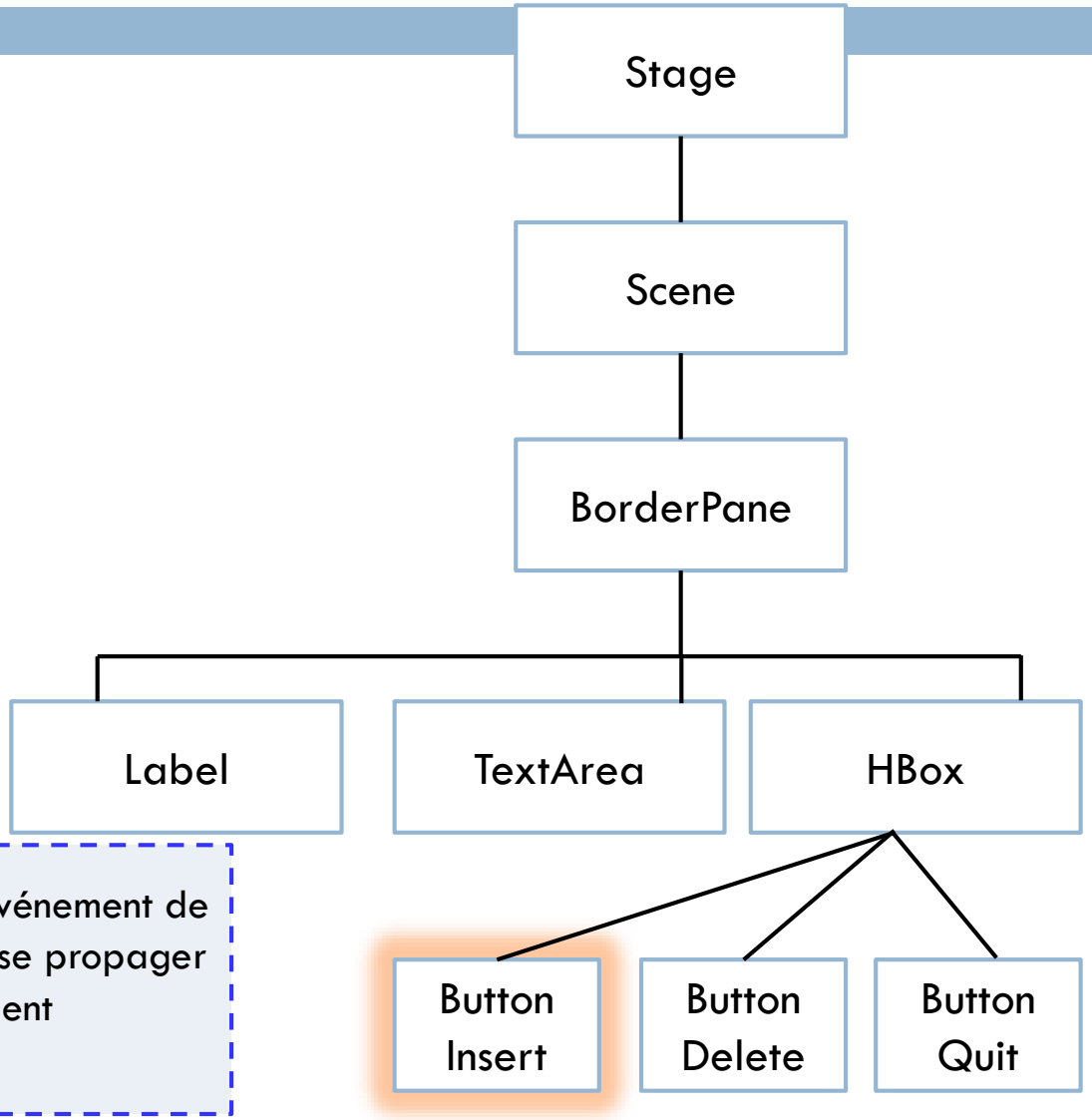




Si l'utilisateur clique sur le cercle, un événement de type MouseEvent va être déclenché et il va se propager sur le long de la chaine de traitement « Event Dispatch Chain »

La cible de l'événement : Event Dispatch Chain

Exemple 2 :



Si l'utilisateur clique sur bouton **Insert** , un événement de type ActionEvent va être déclenché et il va se propager sur le long de la chaine de traitement « Event Dispatch Chain »

Le type de l'événement

- Une instance de « EventType » définit un type d'événement
- La classe « EventType » est utilisée pour classer les événements dans une classe d'événements

  Exemple : La classe « MouseEvent » nous informe que l'utilisateur a utilisé la souris mais elle nous n'informe pas si la souris a été pressée, relâchée ou cliquée

- Un nom
- Un super type

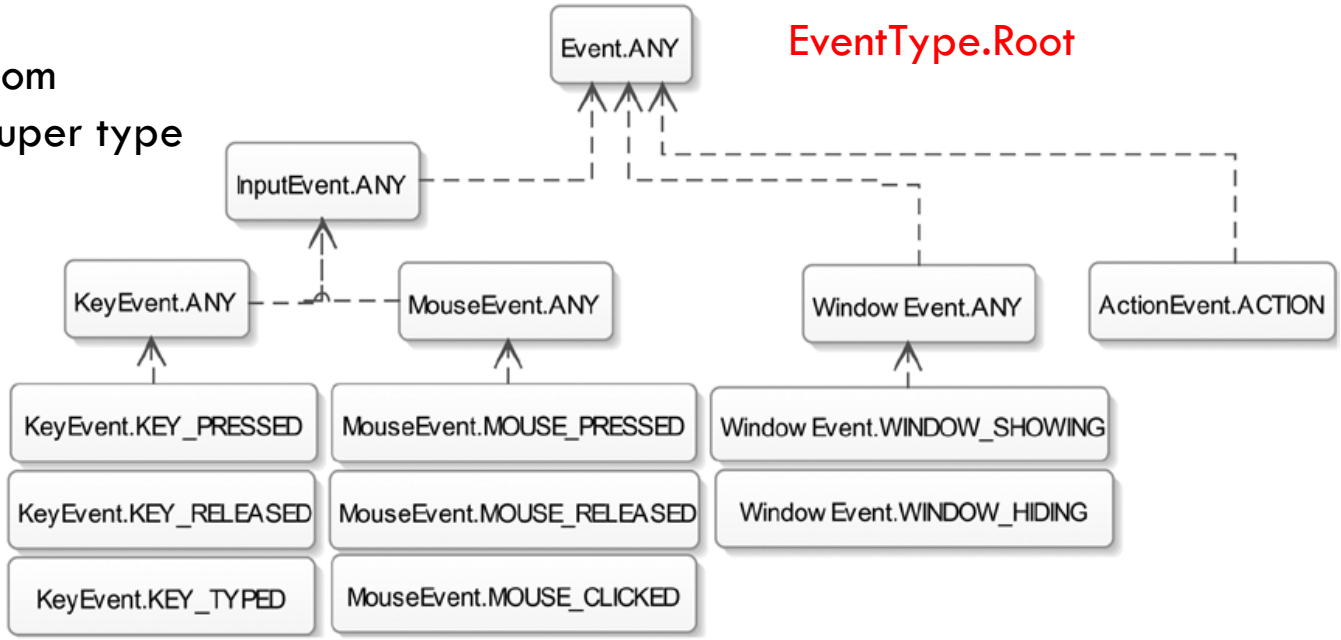


Figure 2. Une liste de certaines types d'événements

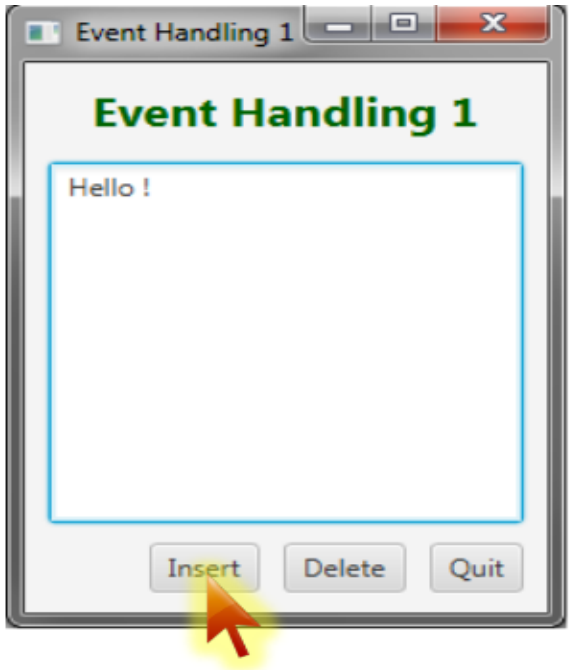
Gestion des événements (1)

- L'ors qu'un événement se produit, plusieurs étapes sont effectuées dans le cadre du traitement de l'événement :

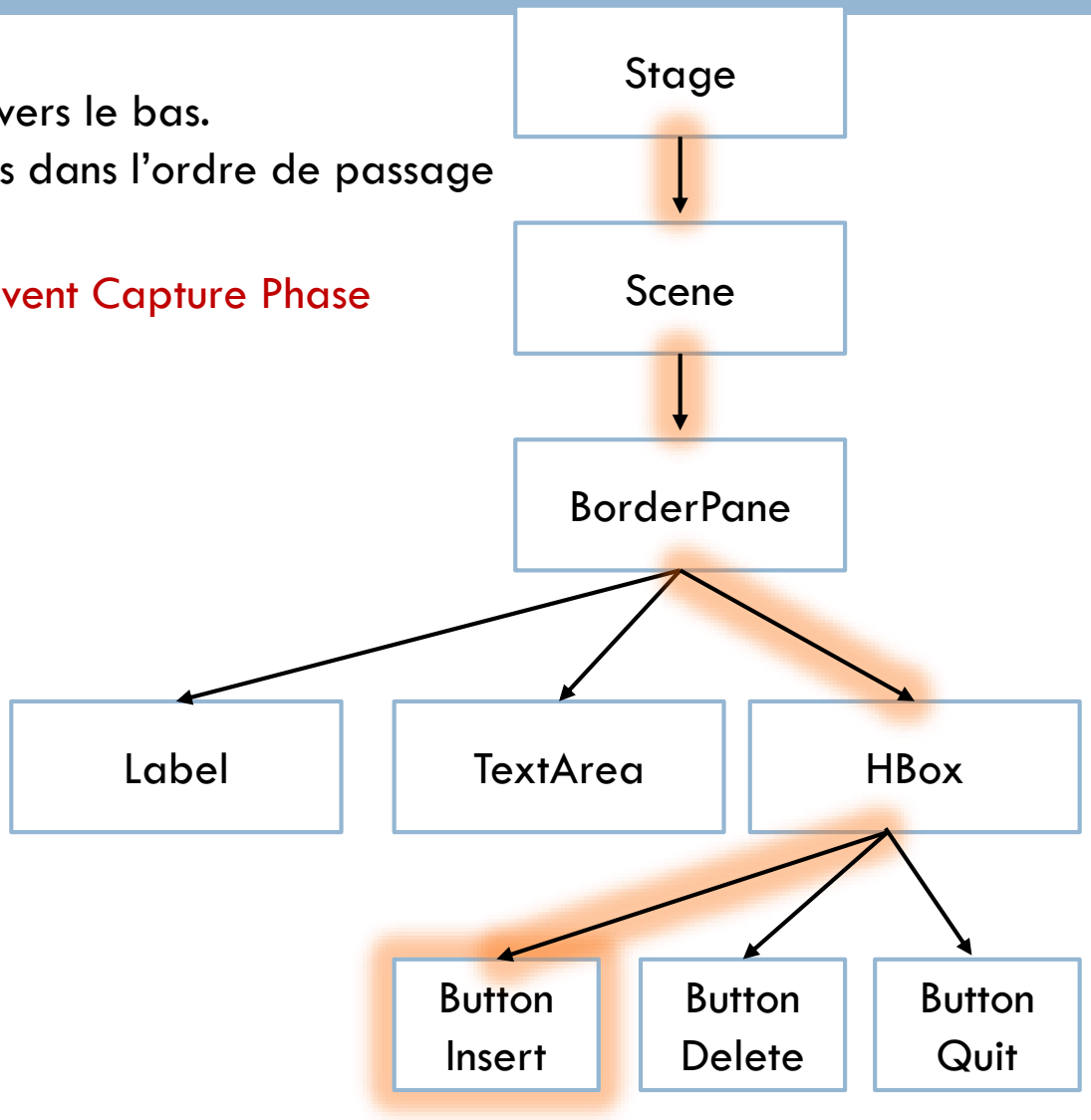
1. La sélection de la cible de l'événement
2. La détermination de la chaine de traitement « Event Dispatch Chain »
3. Le traitement des filtres d'événement (Event Filter) : Event Capture Phase
4. Le traitement des gestionnaires d'événement (Event Handler) : Event Bubbling Phase

Gestion des événements (2)

- L'événement se propage d'abord vers le bas.
- Les **filtres** enregistrés sont exécutés dans l'ordre de passage



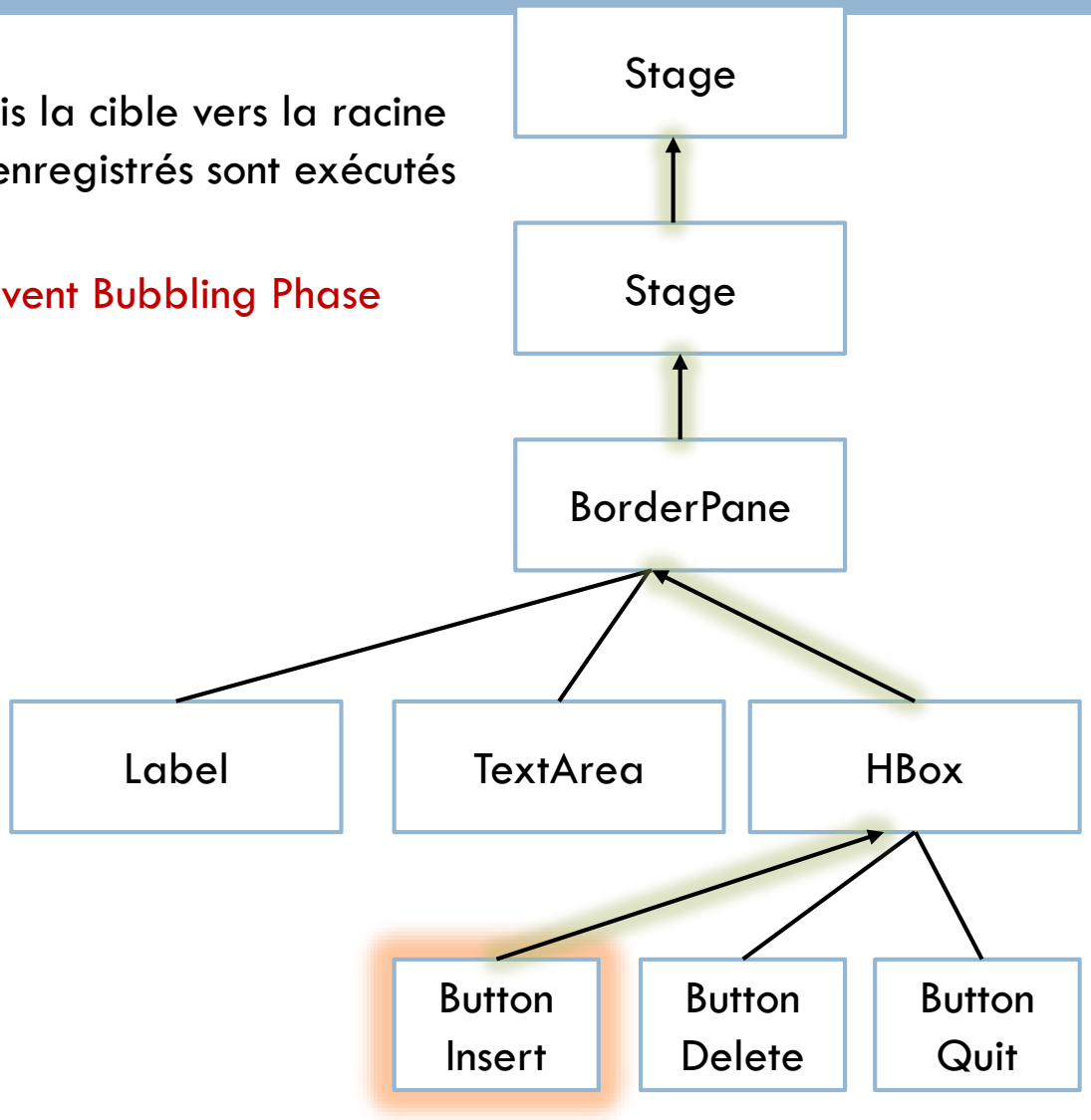
Event Capture Phase



Gestion des événements (3)

- L'événement remonte ensuite depuis la cible vers la racine
- Les **gestionnaires d'événements** enregistrés sont exécutés dans l'ordre de passage

Event Bubbling Phase



Gestion des événements (4)

- Pour gérer un événement, il faut créer un récepteur d'événement (Event Listener) et l'enregistrer sur les nœuds du graphe de scène où l'on souhaite intercepter l'événement et effectuer un traitement.

- Un récepteur d'événement peut être enregistré comme un **filtre** ou comme un **gestionnaire d'événement**. La différence principale entre les deux réside dans le moment où le code est exécuté :
  - Les **filtres** (filters) sont exécutés dans la phase descendante de la chaîne de traitement des événements (avant les gestionnaires)
  - Les **gestionnaires** (handlers) sont exécutés dans la phase montante de la chaîne de traitement des événements (après les filtres)

- Les filtres, comme les gestionnaires d'événements, sont des objets de l'interface fonctionnelle **EventHandler <T extends Event>** qui impose l'unique méthode **handle (T event)** qui se charge de traiter l'événement.

```
public interface EventHandler<T extends Event> extends EventListener
void handle(T event);
}
```

Gestion des événements (5)

Pour enregistrer un récepteur d'événement sur un nœud de graphe de scène, on peut :

- Utiliser la méthode **addEventFilter()** que possèdent tous les nœuds et qui permet d'enregistrer un filtre

- Utiliser la méthode **addEventHandler()** que possèdent tous les nœuds et qui permet d'enregistrer un gestionnaire d'événement

Utiliser une **des méthodes utilitaires** qui disposent certains composants et qui permettent d'enregistrer un gestionnaire d'événement en tant que propriété du composant.

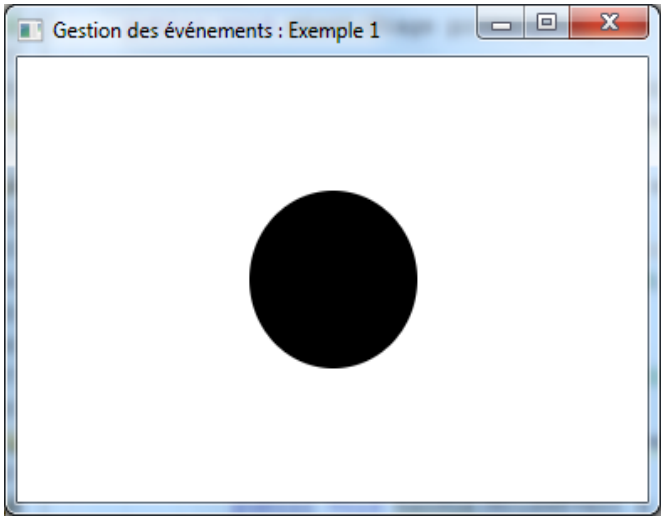
setOnEventType(EventHandler)

setOnAction (handler)
setOnKeyTyped(handler)
setOnMouseClicked (handler)

Exemple 1


```
public void start(Stage primaryStage) {

    Circle circle = new Circle (100, 100, 50);
    StackPane root = new StackPane();
    root.getChildren().add(circle);
    Scene scene = new Scene(root, 300, 250);
    primaryStage.setTitle("Gestion des événements : Exemple 1");
    primaryStage.setScene(scene);
    primaryStage.show();
}
```



Code de l'interface sans gestion de l'événement

Exemple 1


```
Circle circle = new Circle (100, 100, 50);
EventHandler<MouseEvent> a1 =new EventHandler<MouseEvent>()
{
    public void handle (MouseEvent e )
    {
        System.out.println(" Le filtre de l'événement de la souris a été crée ");
        circle.setFill(Color.YELLOW);
    }
};
circle.addEventFilter(MouseEvent.MOUSE_PRESSED, a1);

EventHandler<MouseEvent> a2= new EventHandler<MouseEvent> ()
{
    public void handle(MouseEvent e )
    {
        System.out.println(" Le gestionnaire de l'événement de la souris a été crée");
        circle.setFill(Color.RED);
    }
};
circle.addEventHandler(MouseEvent.MOUSE_PRESSED, a2);
```

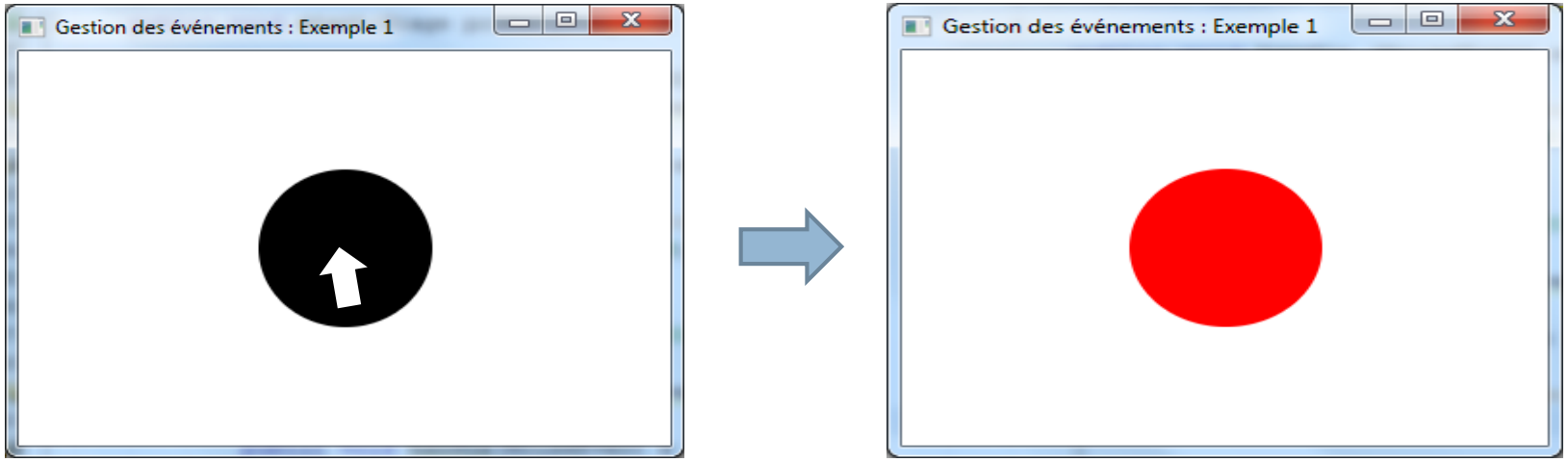
1- Création d'un filtre d'événement

2- L'enregistrement du filtre

3- Création d'un gestionnaire d'événement

4- L'enregistrement du gestionnaire

Exemple 1




```
Output - EvenementTasK0 (jfxsa-run)
compile:
Detected JavaFX Ant API version 1.3
jfx-deployment:
jar:
Copying 12 files to C:\Users\nourhene\Downloads\EvenementTasK0\dist\run1504151454
jfx-project-run:
Executing C:\Users\nourhene\Downloads\EvenementTasK0\dist\run1504151454\EvenementTasK0.jar using platform C:
 Le filtre de l'événement de la souris a été crée
 Le gestionnaire de l'événement de la souris a été crée
```

A chaque clic sur le cercle, le filtre et le gestionnaire d'événement seront exécutés, la couleur du cercle devient rouge et les deux messages sont affichés.

La syntaxe (1)

l'événement

```
EventHandler <MouseEvent> a=new EventHandler <MouseEvent>()
        {
            public void handle(MouseEvent e)
            {
                System.out.println("OK ");
            }
        };

Conposant.addEventFilter(MouseEvent.MOUSE_CLICKED, a);
```

addEventHandler

l'événement cible

Le type de l'événement

La syntaxe (2)


```
Conposant.setOnMOUSE_CLICKED (new EventHandler<MouseEvent>()
{ public void handle(MouseEvent e)

        {System.out.println("OK ");
}

});
```

Exercice 1

1. Dessinez cette interface
2. A chaque clic sur le bouton Insérer, un caractère '**A**' sera ajouté dans le composant TextArea
3. A chaque clic sur le bouton Supprimer, un caractère sera supprimé du composant TextArea
4. A chaque clic sur le bouton Quitter, l'application se termine.