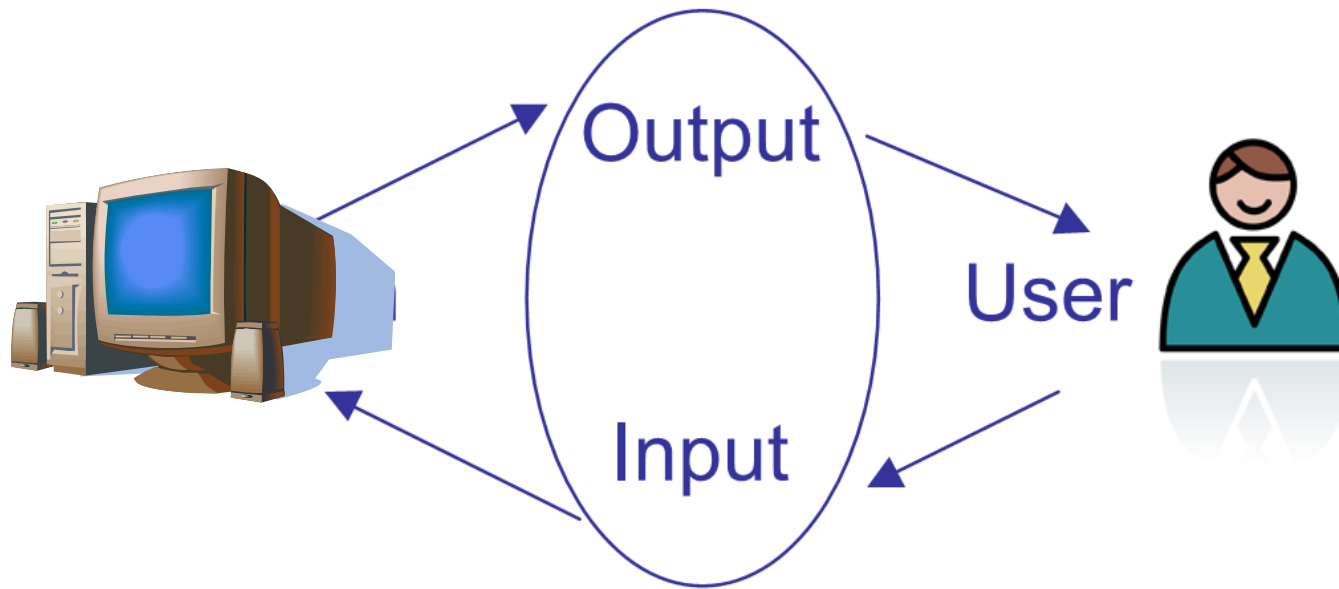
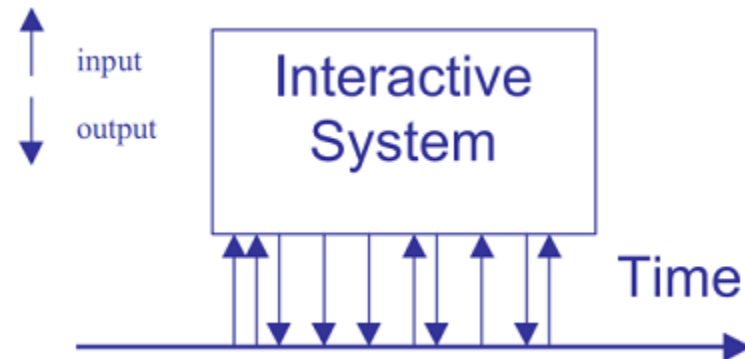


COURS – TD 5: GESTION DES EVENEMENTS

Systeme interactif - Rappel



Fonctionnement dirigé par les évènements



Entrées-Sorties

- Entrées/Input

Actions de l'utilisateur vers le système (ex: click souris)

- Évènement construit par le système contenant toutes les informations concernant cette action

- Sorties/Output

Action de l'application en réponse à un évènement utilisateur

- Rafraîchissement de l'affichage (ex: déroulement d'une liste)

Types d'évènements

- Bas niveau
 - Ex: clic de souris, appui clavier
 - Indépendant d'un composant
- Sémantique
 - EX: appui sur un JButton widget
 - Dépendant d'un composant

Agissent sur l'état de l'application

Gestion des évènements

- Stockés dans une file d'attente
- Traités les uns après les autres, 2 cas de figure
 - Aucun élément de l'application ne s'est enregistré pour être prévenu de l'occurrence de cet évènement:

Rien ne se passe

- Une partie de l'application s'est enregistrée pour être prévenue:

Elle est prévenue et agit en conséquence


Event handler

- Lorsque l'application est prévenue de l'arrivée d'un évènement qu'elle écoute
 - Elle doit le traiter
- Traitement délégué à un Event Handler
 - Précondition sur le traitement de l'évènement
 - Actions à effectuer
 - Modification de l'état de l'application
 - Rétroaction graphique

Interface Dirigée par l'utilisateur

- L'utilisateur déclenche des commandes
- Exécution non linéaire
- Ordre non prédictible
- La plupart du temps le système ne fait rien
- Les procédures de gestion d'événements

GUI program:



```
main()
{
    decl data storage;
    initialization code;

    create GUI;
    register callbacks;

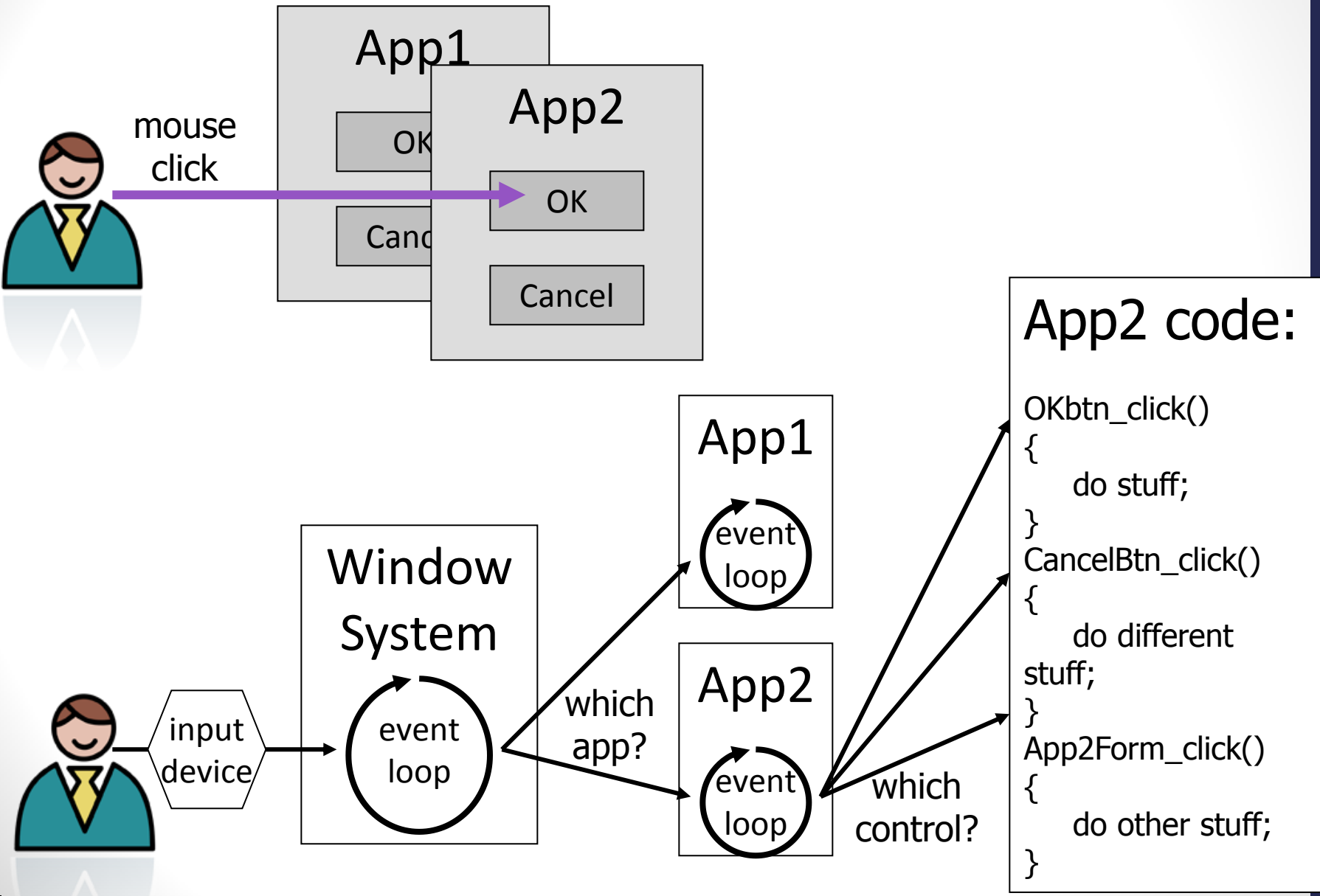
    main event loop;
}

Callback1() //button1 press
{
    code;
}

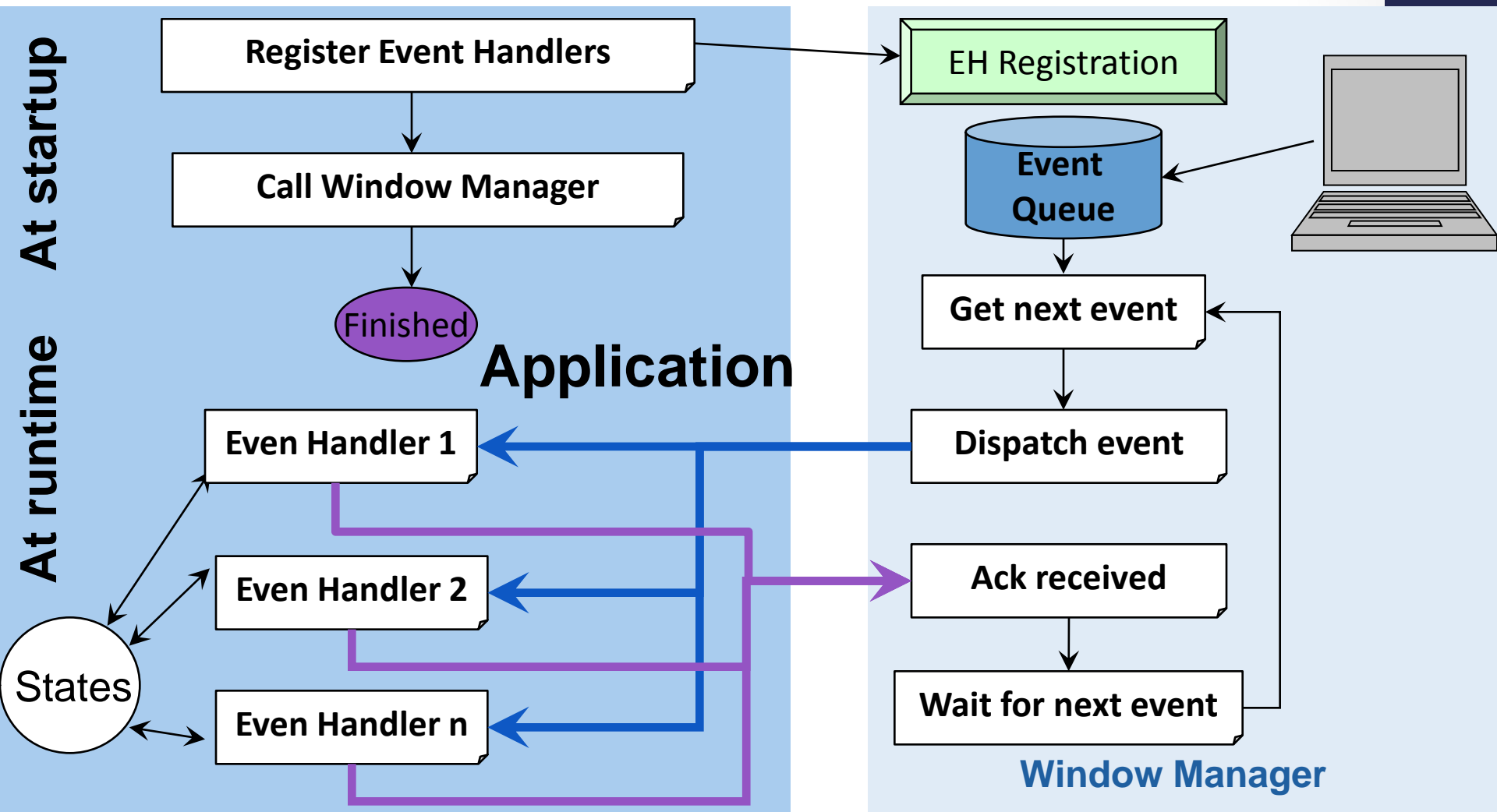
Callback2() //button2 press
{
    code;
}

...
```

GUI Events



Event-based Functioning



Gestion des évènements avec SWING

- Initial thread
- **Event dispatch thread (AWT event loop)**
- Worker threads ou background threads

1 seul thread gère les évènement de l'interface utilisateur

Protocole de programmation pour être en ligne avec la gestion des évènements SWING

Démarrage d'une GUI

```
public static void main(String args[]) {  
    SwingUtilities.invokeLater(new Runnable() {  
        public void run() {  
            createAndShowGUI();  
        }  
    });  
}
```

- Initial thread poste un évènement sur la boucle de gestion des évènements
- **Attention: pas d'appel de “new MyJFrame().setVisible(true)” seul dans le main**

Event dispatch thread

- Gère les évènements
 - Reçoit les évènements
 - Appelle les listeners appropriés
- Rafraîchit l'affichage
- **Attention aux traitements effectués dans les event handlers!!!! Perte de réactivité s'ils sont trop lourds**

(SwingWorker API pour ce cas de figure)


Développement

- Fonctionnement par événement (main loop + event handlers)
- Programmation par événement (écriture des event-handlers)
- 3 éléments principaux :
 - Composants Swing pour l'interface utilisateur
 - Listeners
 - Application

Listeners

- Pour être prévenu de l'arrivée d'un évènement, il faut s'être enregistré
- i.e. avoir déclaré qu'on écoute l'arrivée d'un évènement particulier
- i.e. avoir déclaré que l'on est un écouteur d'un évènement en particulier

Listener - fonctionnement



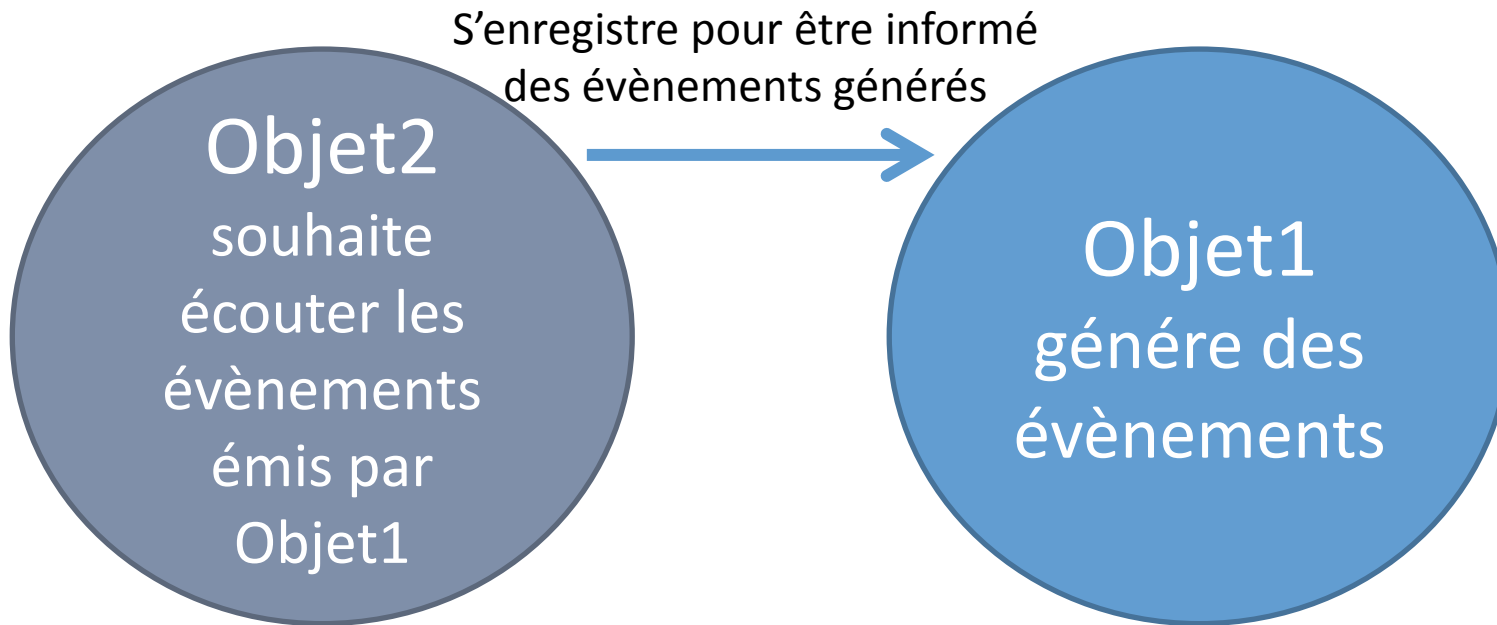
Objet1
génère des
événements

Listener - fonctionnement

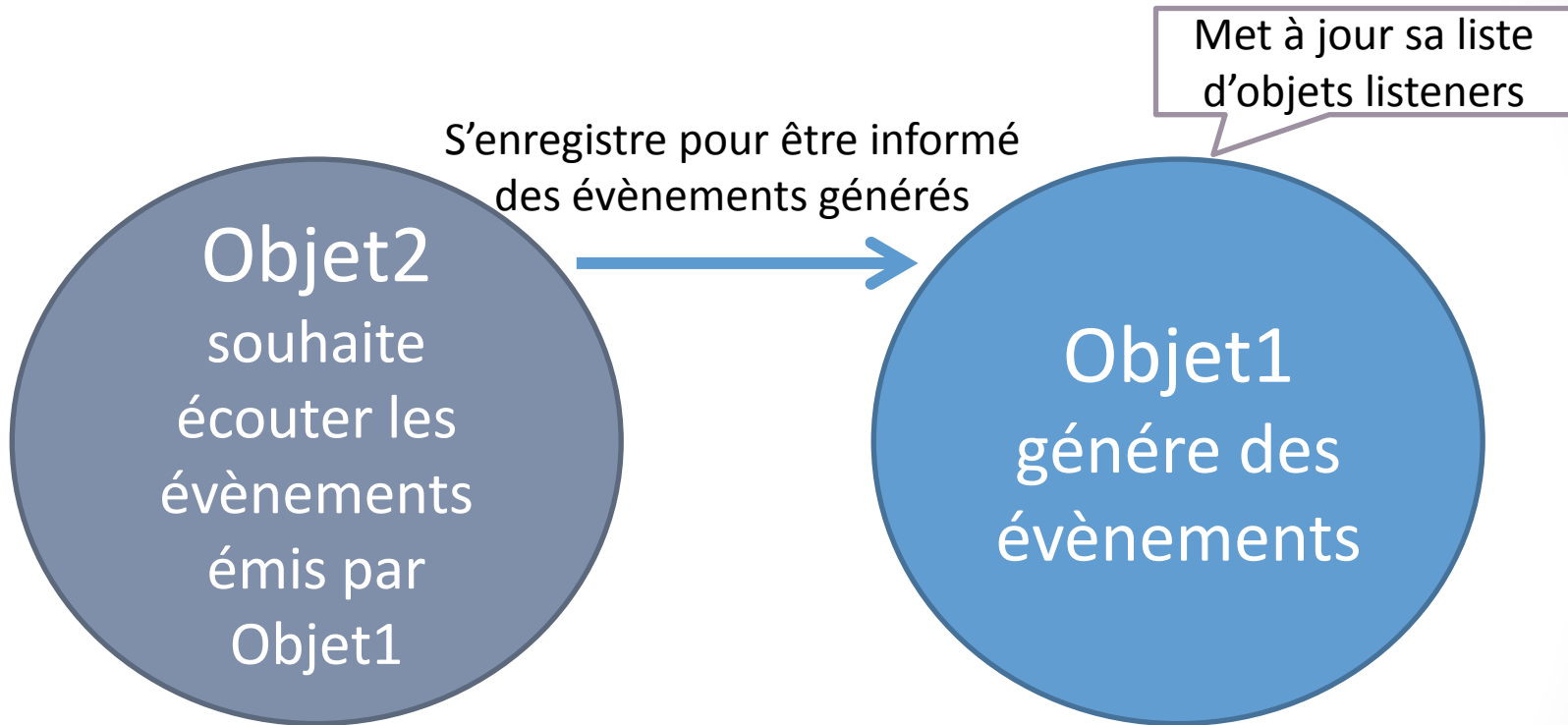
Objet2
souhaite
écouter les
événements
émis par
Objet1

Objet1
génère des
événements

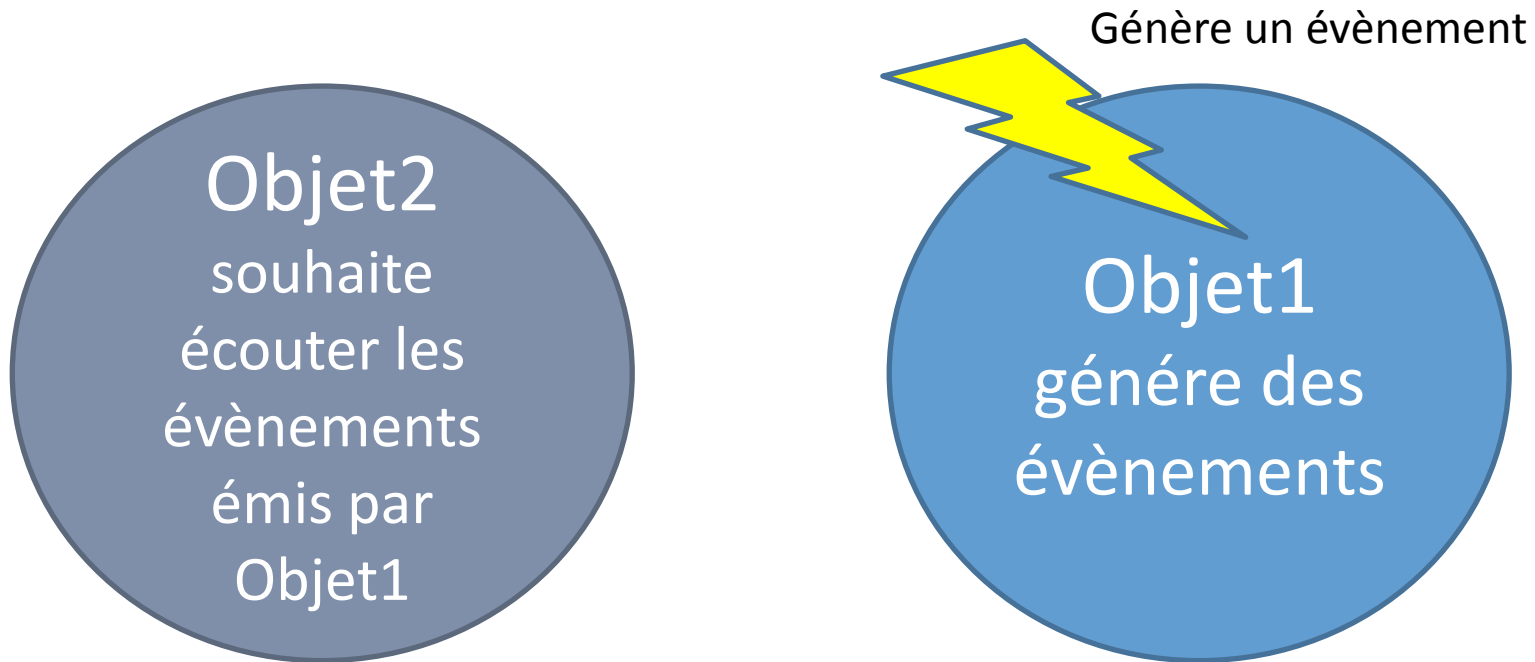
Listener - fonctionnement



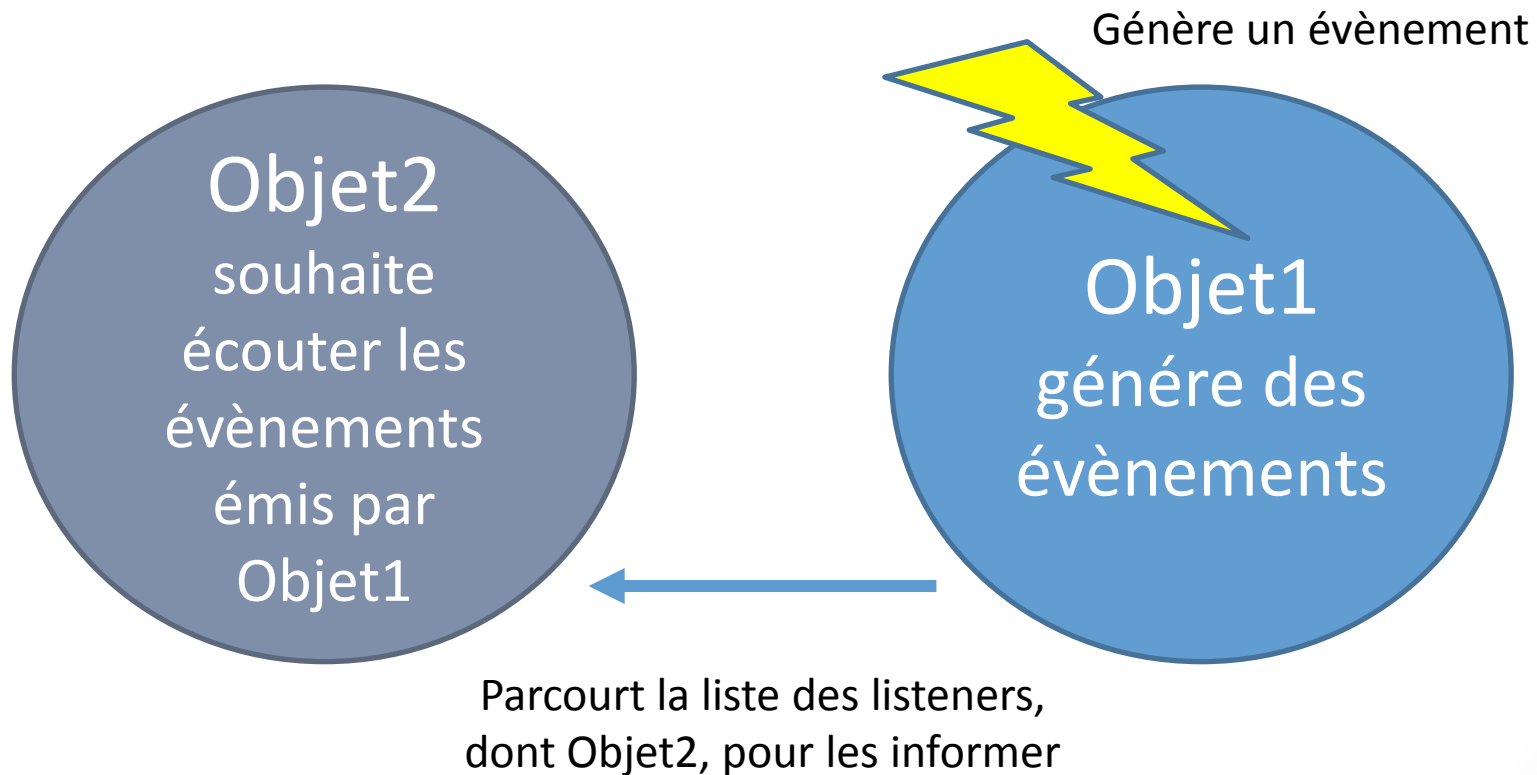
Listener - fonctionnement



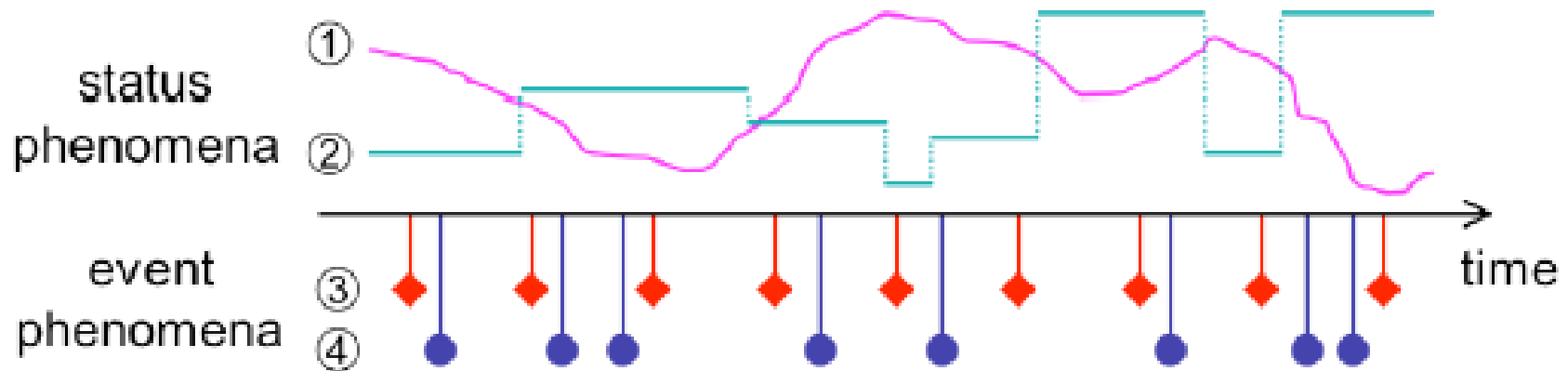
Listener - fonctionnement



Listener - fonctionnement



Etats et événements



- 1- le monde réel évolue de façon continue
- 2- les variables représentent des variations par pallier
- 3- les événements peuvent avoir une origine périodique (regarder sa montre toutes les 30s)
- 4- les événements arrivent et ont un impact sur l'état

Structure appli. par evt.

- La boucle d'événement (main event loop) : reçoit chaque événement produit par l'utilisateur
- Les gestionnaires d'événements : sont des procédures associées à chaque couple (widget, action sur un widget) et appelées par la main event loop dès que une action a été réalisée.

- Tous les event handlers ont la même structure :

EH1;

 Précondition

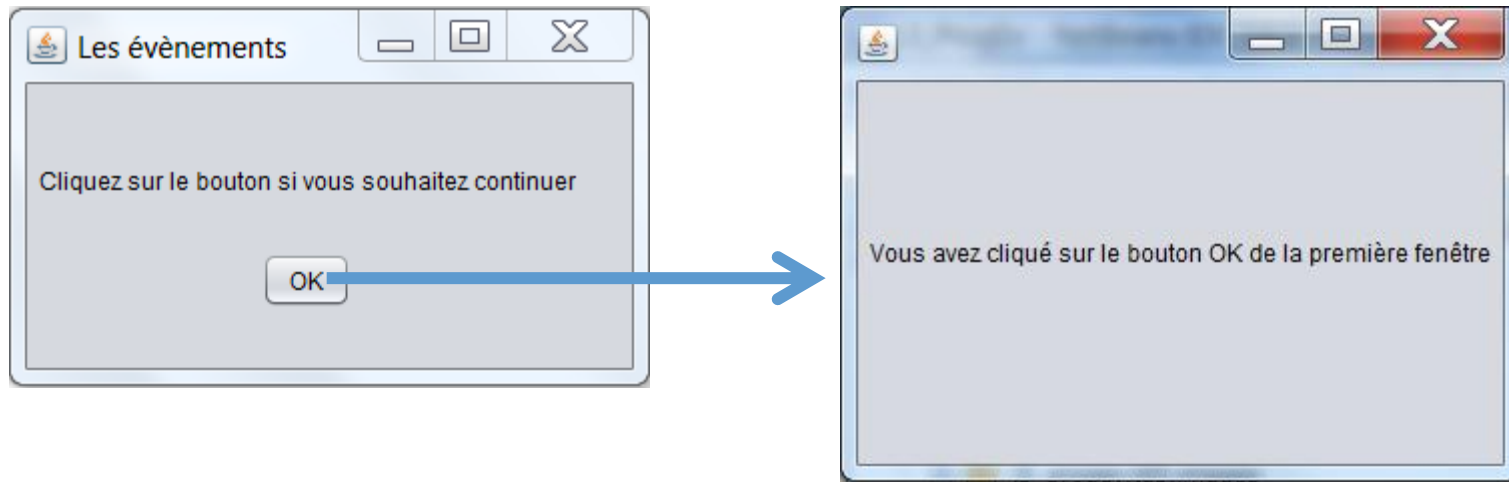
 Action;

 Modification de l'état du dialogue;

 Rétroaction graphique

Fin EH1;

Exemple



- Clic sur le bouton OK entraîne l'affichage d'une seconde fenêtre

Exemple - code

Enregistrement
du listener

```
jButton1.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton1ActionPerformed(evt);  
    }  
});
```

Sur occurrence
de l'évènement,
appel de l'event
handler

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    (new MySecondFrame()).setVisible(true);  
}
```

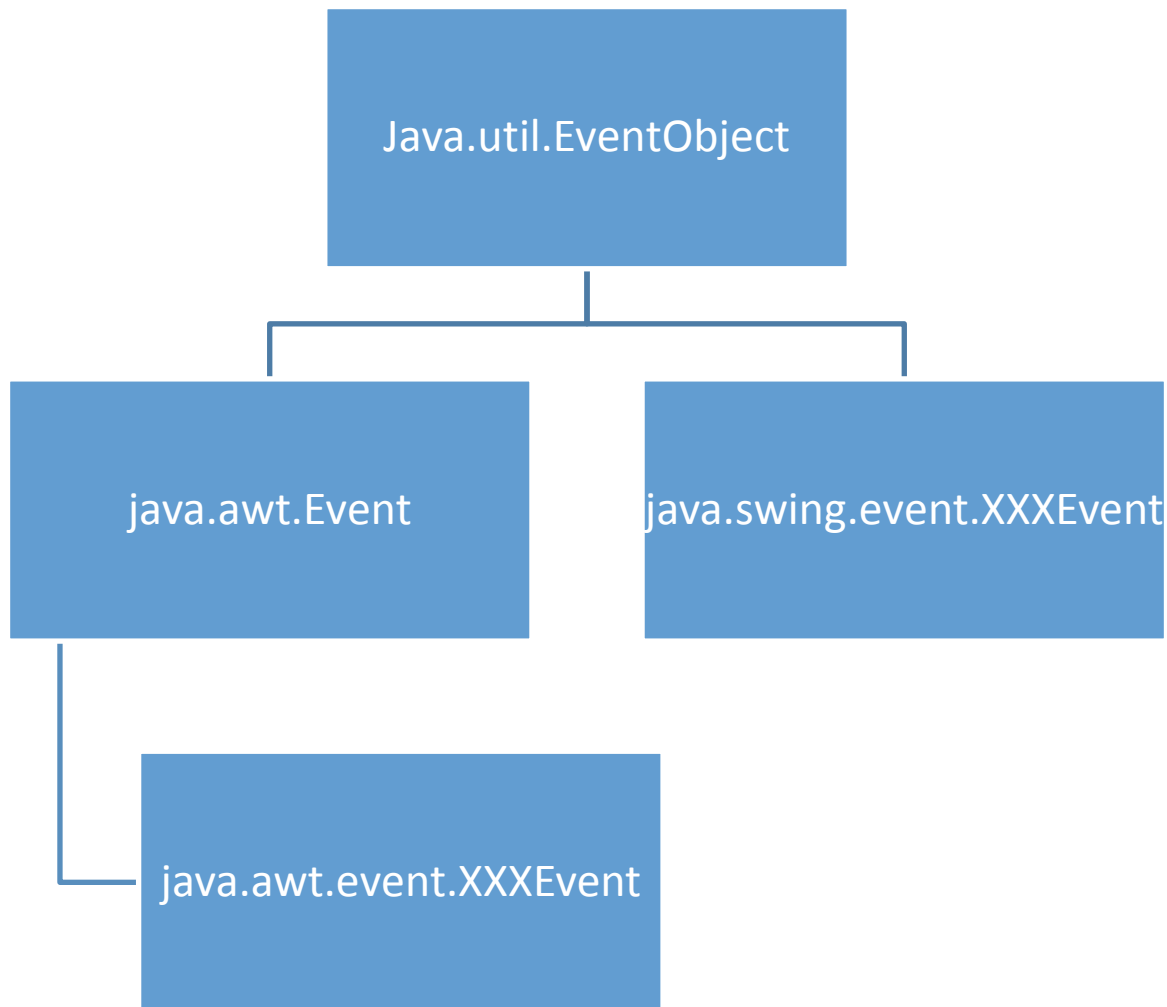
Déclaration du
traitement
effectué par
l'event handler

Association Composant-Event-Listener

Informations liées à un évènement

- Informations génériques
 - Source de l'évènement
 - Méthode getSource() permet de récupérer le pointeur vers l'objet source
 - Timestamp
- Informations particulières liées au type d'évènement

Classes d'évènements



ActionEvent

Evènement sémantique permettant d'indiquer qu'une action a été produite sur un composant

- Pointe sur l'objet source de l'évènement
- Spécifie une action particulière (cas d'un composant modal ou l'évènement envoyé dépend de son état)
- Timestamp de sa date d'émission

Exemple: appui bouton (via un clic souris ou un appui touche)

Evènements sémantiques

- Robuste
- Portable
- Permettent de s'abstraire du/des périphériques qui ont générés l'évènement

A utiliser le plus possible

Exemples d'évènements associés à des actions utilisateur

Action utilisateur	Evènement(s) émis
Clic sur un bouton	ActionEvent et MouseEvent
Appui sur un bouton avec le clavier	ActionEvent et KeyEvent
Déplacement du curseur de la souris sur un composant	MouseEvent
Iconification d'une fenêtre	WindowEvent
Déplacement du curseur dans un JTextField	CaretEvent
Sélection d'un item dans une liste	ListSelectionEvent
Passage du focus à un composant	FocusEvent

Listeners/Ecouteurs

- Ecoute les évènements auxquels il est associé
- Les objets qui génèrent des évènements possèdent des méthodes qui permettent de leur associer des listeners
- 1 objet peut avoir plusieurs listeners
- 1 listener peut écouter plusieurs objets

Correspondance Event/Listener

Type d'Event (classe)	Type de Listener (interface)
ActionEvent	ActionListener
MouseEvent	MouseListener
KeyEvent	KeyListener
WindowEvent	WindowListener
...	...

Exercice 1

- Donner le triplet (widget, event, listener) pour les actions utilisateur suivantes:
 - Clic sur un bouton simple
 - Clic sur le bouton d'iconification d'une fenêtre
 - Clic sur une case à cocher
 - Appui sur entrée dans un champ de saisie de texte
 - Clic dans un champ de saisie de texte
 - Déplacement du curseur (clic ou appui clavier) dans un champ de saisie de texte
 - Sélection d'un élément d'une liste
 - Déplacement de la souris sur un bouton

Implémentation de l'interface du listener

- Lorsque l'on déclare et instancie le listener, chaque méthode de l'interface correspondante doit être implémentée
- Lourd s'il y en a plusieurs et que l'on a seulement besoin de l'une d'elle
- Exemple: Pour l'interface `MouseListener`, 5 méthodes à implémenter
 - `mouseClicked(MouseEvent e)`, `mouseEntered(MouseEvent e)`, `mouseExited(MouseEvent e)`, `mousePressed(MouseEvent e)`, `mouseReleased(MouseEvent e)`
 - Si l'on a seulement besoin de gérer le `mouseEntered`, les autres méthodes vont être vides...
 - Solution: les **Adapters**

Adapters

- Pre-implementation d'un Listener
- Il ne reste qu'à surcharger la méthode dont on a besoin

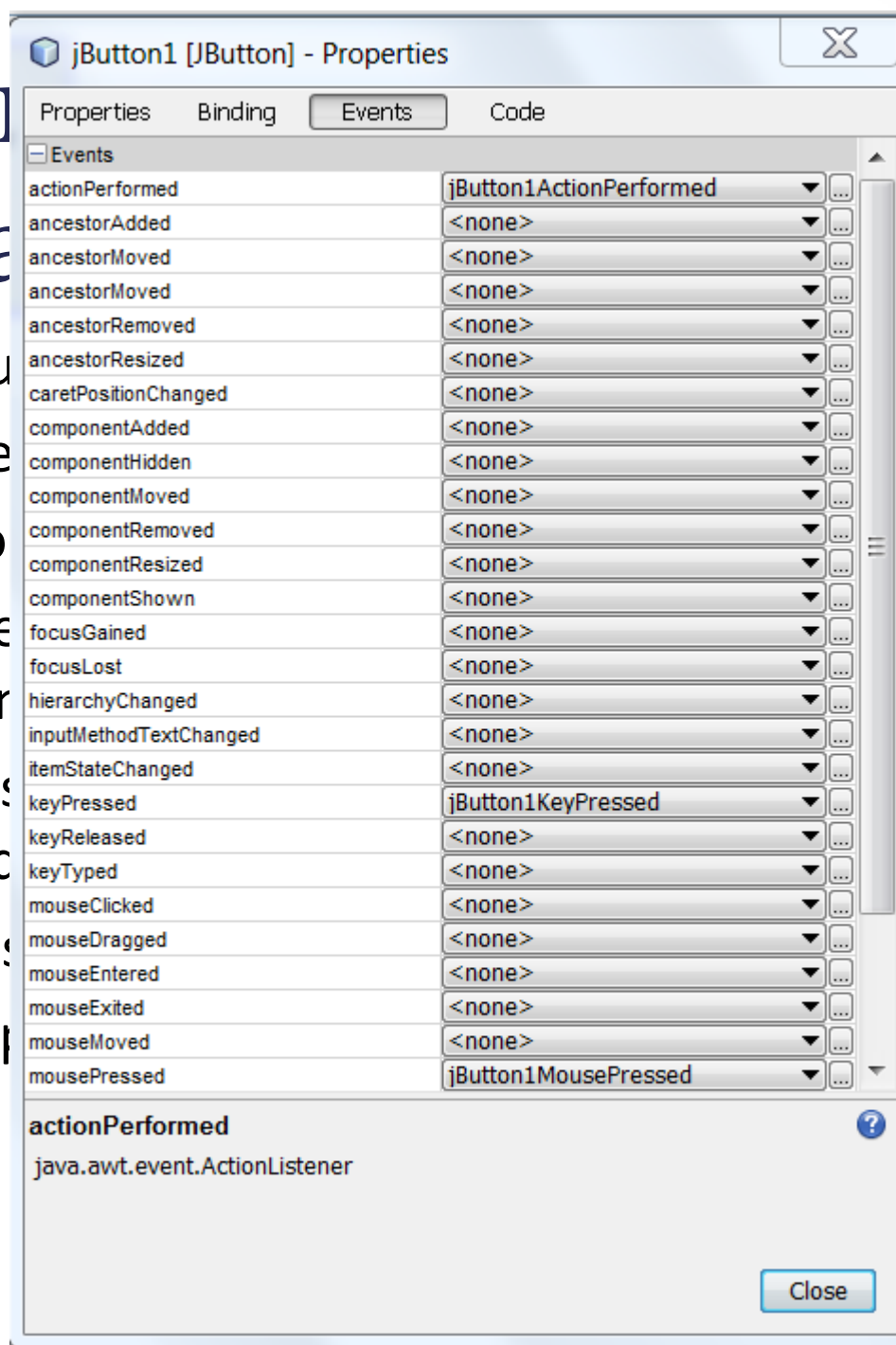
Type d'Event (classe)	Type de Listener (interface)	Type d'Adapter (classe)
ActionEvent	ActionListener	ActionAdapter
MouseEvent	MouseListener	MouseAdapter
KeyEvent	KeyListener	KeyAdapter
WindowEvent	WindowListener	WindowAdapter
...	...	

Gestion des évènements avec Netbeans et Matisse

- Dans la vue conception graphique
 - Fenêtre Propriétés, onglet Event
 - Clic droit sur le composant, menu contextuel Event
- Permet de configurer finement les Event Handlers des évènements associés au composant sélectionné
- Code d'association Listener-Event Handler généré automatiquement
 - Evite les erreurs de programmation
- Le développeur n'a plus qu'à programmer le traitement

Gestion Netbeans

- Dans la vue
 - Fenêtre
 - Clic dro
- Permet de évènement
- Code d'ass automatique
 - Evite les
- Le dévelop



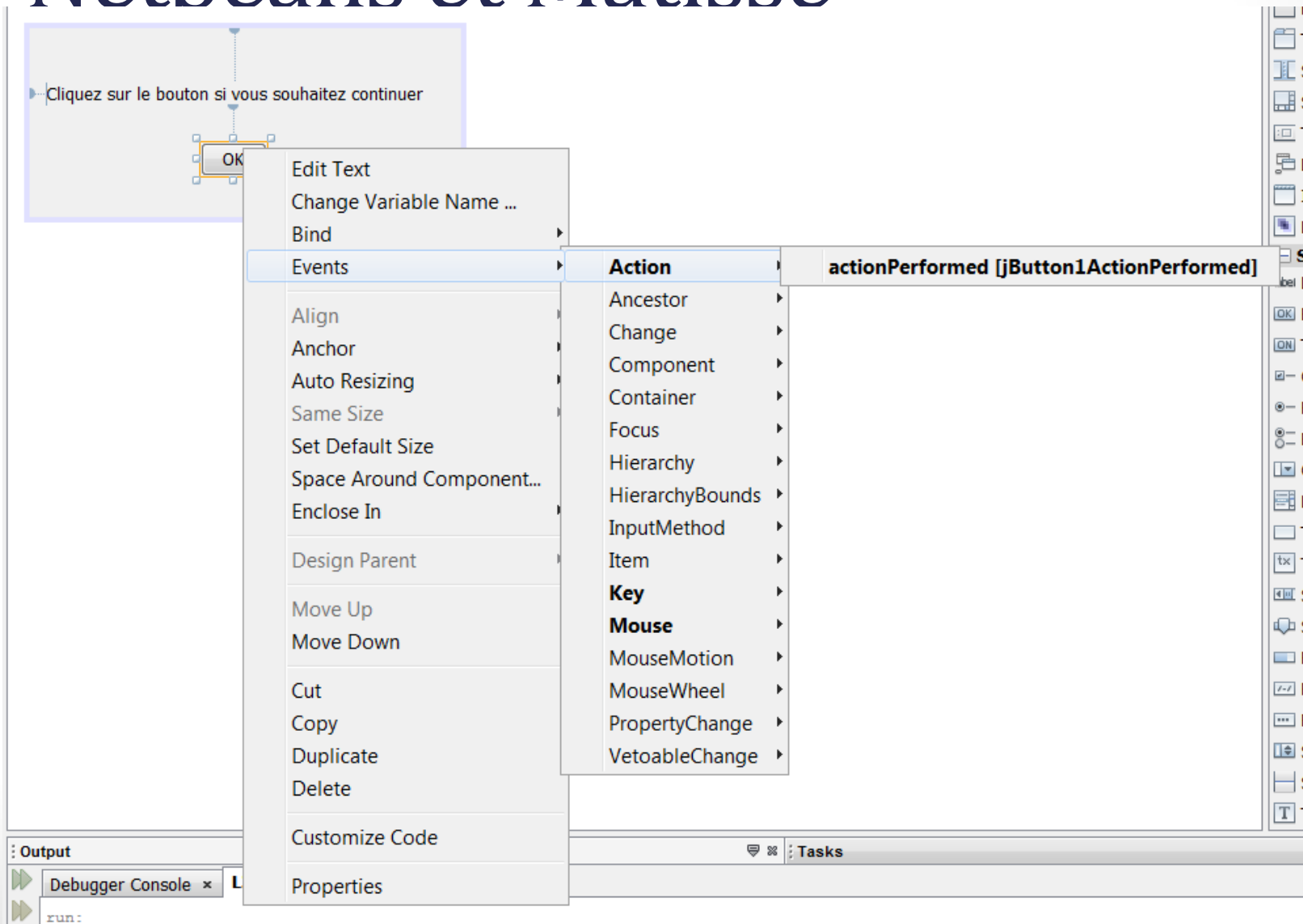
avec

l'Event

ers des
é
éré

aitement

Gestion des évènements avec Netbeans et Matisse



Code généré automatiquement pour un appui sur un JButton

```
jButton1.addActionListener(new java.awt.event.ActionListener() {  
    public void  
        actionPerformed(java.awt.event.ActionEvent evt) {  
            jButton1ActionPerformed(evt);  
        }  
});
```

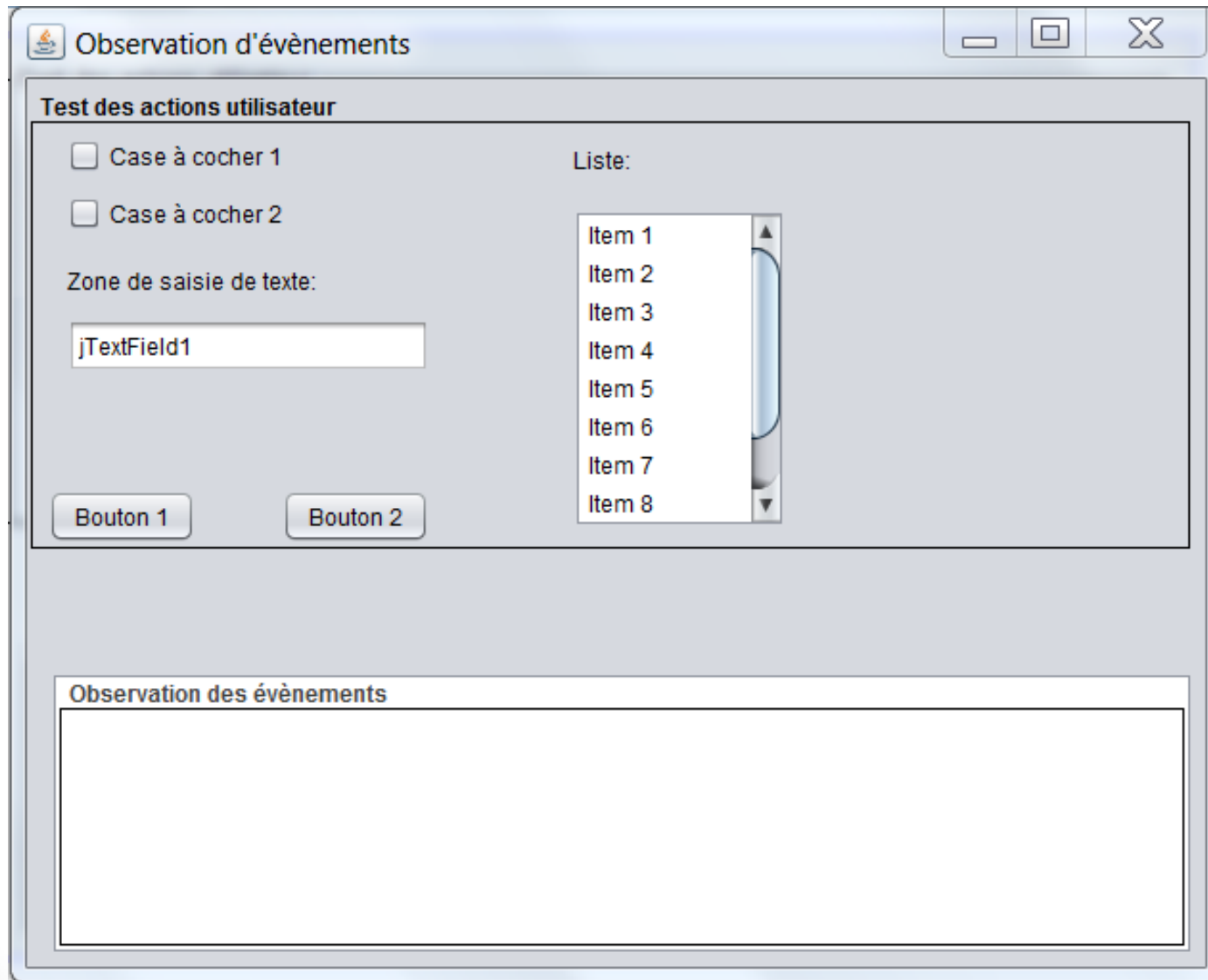
...

```
private void  
jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}
```

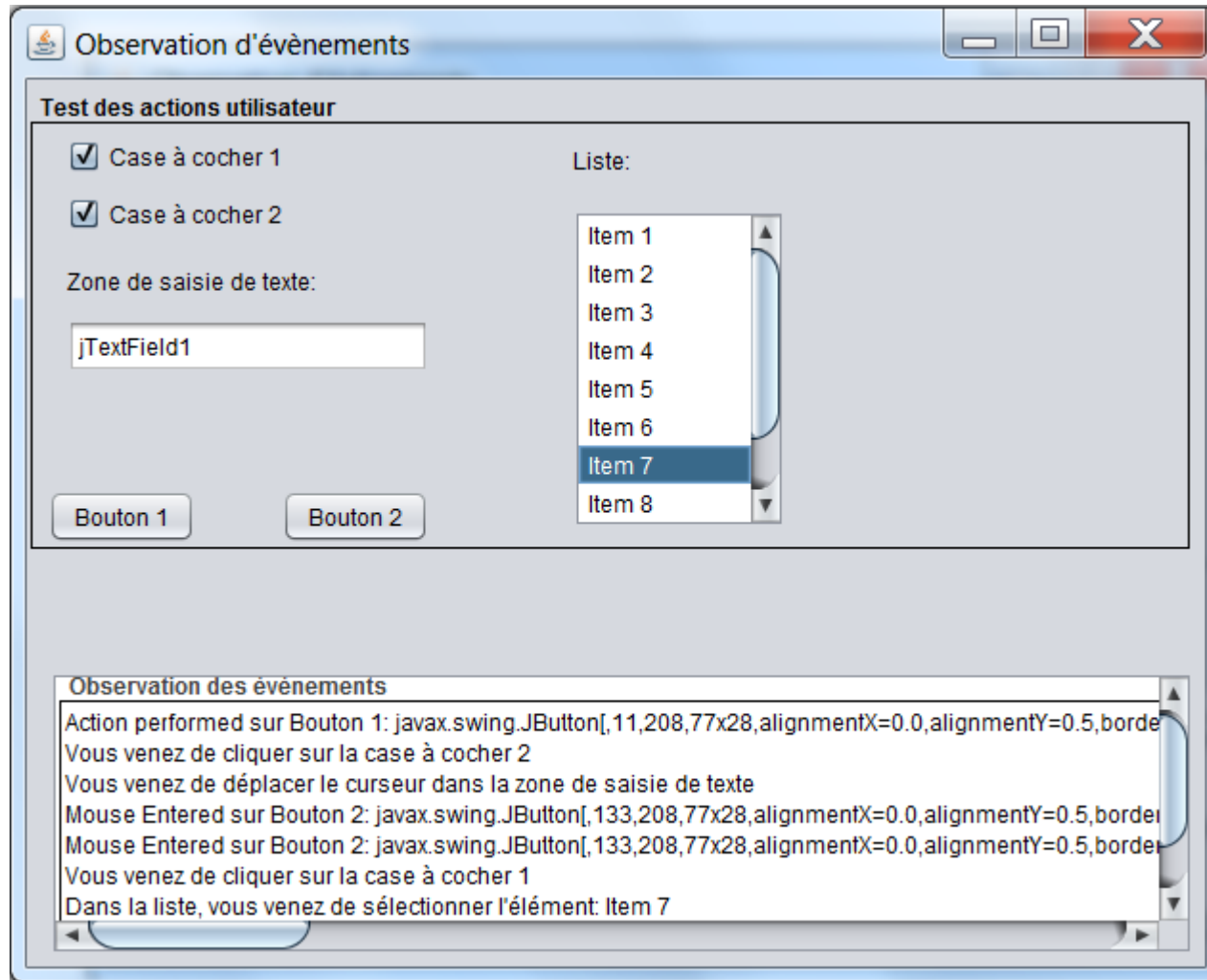
Infrastructure Netbeans

- Permet de bien séparer chaque Event Handler
 - Lisibilité
 - Modifiabilité

Exercice 2



Exercice 2



Exercice 2

- Ecrire les traitements à effectuer dans les events handlers pour permettre l'affichage des informations présentes dans la zone de texte d'observation des évènements (capture d'écran précédente)

(compléter la feuille distribuée, après les lignes

`“// TODO add your handling code here:”`)