EPAI, Jérôme Frossard (2019)

INTERFACES UTILISATEUR ET PROGRAMMES PILOTÉS PAR ÉVÉNEMENTS

Développer une interfaces utilisateur graphique

Programme interactif

- Désigne un programme qui interagit avec un utilisateur humain par l'intermédiaire d'une **interface utilisateur.**
- Par contraste, un démon ou un service s'exécute sans aucune interaction homme-machine.
- On distingue deux types de programmes interactifs :
  - **Programme interactif séquentiel** : Le programme contrôle les interactions, l'utilisateur réagit aux sollicitations du programme.
  - **Programme interactif piloté par événement** : l'utilisateur contrôle les interactions, le programme réagit aux actions de l'utilisateur.

Interface utilisateur

- Une interface utilisateur (UI) est la partie d'un programme qui coordonne les **interactions homme-machine** (IHM).
- Ces interactions mettent en œuvre :
  - un ou plusieurs dispositifs de sortie (écran, barrette braille)
  - un ou plusieurs dispositifs d'entrée (clavier, souri, écran tactile)
- On distingue essentiellement deux types d'UI :
  - **Interface en ligne de commande (CLI) :** shell de Linux, éditeur vi, interface des équipements Cisco, etc.
  - **Interface utilisateur graphique (GUI) :** bureau de Windows ou de Linux, éditeur Visual Studio Code, Word, Excel, etc.

Interface utilisateur (suite)

- Le type d'interface (CLI ou GUI) et le type de programme interactif (séquentiel ou piloté par événement) sont deux chose distinctes.
- Même s'il est généralement plus facile :
  - de réaliser un programme interactif séquentiel avec une interface en ligne de commande.
  - de réaliser un programme interactif piloté par événement avec une interface utilisateur graphique.
- Toutes les combinaisons sont possibles et **un même programme peut avoir plusieurs interfaces utilisateurs.**

Interface utilisateur graphique (GUI)

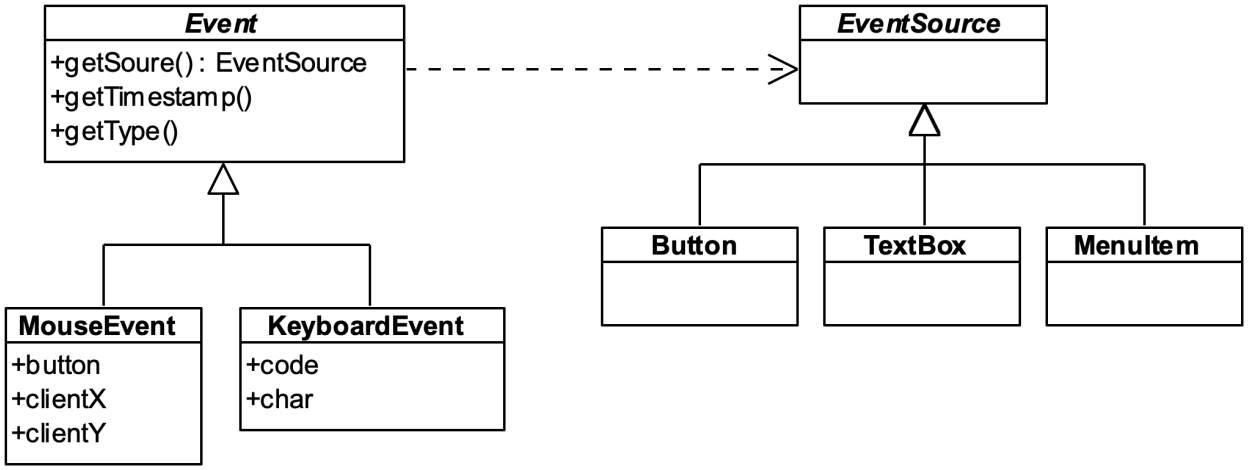
- Une interface utilisateur graphique obéit le plus souvent au paradigme WIMP (Window, Icons, Menus, Pointing device) :
  - Un programme dessine son interface dans une **fenêtre**, c.-à-d. une zone rectangulaire attribuée au programme par le système.
  - À l'intérieur d'une fenêtre, des **widgets** (menus, boutons, zones de texte, listes déroulantes, etc.) indiquent à l'utilisateur les actions qu'il est possible de faire.
  - L'utilisateur **manipule directement** l'interface du programme à l'aide d'un **dispositif de pointage** (souris, écran tactile, etc.).
- Une interfaces utilisateur graphiques est le plus souvent réalisée à l'aide d'un ensemble d'outils et de bibliothèques appelé ***widget toolkit***.

Evénement

- Un événement peut représenter :
  - une action de l'utilisateur (clic sur un bouton, caractère saisi dans une zone de texte, sélection dans une liste, etc.)
  - un événement du système (arrêt de l'ordinateur, alerte batterie faible, expiration d'un timer, etc.)
- Un événement est décrit par :
  - un timestamp
  - son type (clic, sélection, survol du curseur, etc.)
  - sa source, c.-à-d. l'élément de l'interface (bouton, menu, etc.) sur lequel s'est portée l'action de l'utilisateur
  - des informations spécifiques à son type (position de la souris, état des boutons, etc.)

Représentation d'un événement

- Dans un programme écrit avec un langage orienté objet (Java, JavaScript, etc.), les éléments de l'interface (widget) et les événements sont représentés par des objets.
- Exemple de diagramme de classes (types des objets) :



Boucle d'événements

- Un programme pilotée par événement doit réagir lorsqu'un événement survient.
- Les événements qui surviennent sont placés dans une file d'attente (***event queue***) par le processus chargé de la gestion des dispositifs d'entrée.
- Le programme est construit autour d'une boucle appelée boucle d'événement (***event loop***) qui consiste à :
  - Tant qu'il y a des événements, les traite aussi rapidement que possible dans l'ordre d'arrivée.
  - Attendre qu'un événement soit placé dans la file d'attente et recommencer.

Traitement des événements

- Pour des raisons de fiabilité, **les événements sont toujours traité l'un après l'autre,** séquentiellement. Le même thread est utilisé pour la boucle d'événement, le traitement des événement et le rafraichissement de l'interface à l'écran.
- Durant le traitement d'un événement, l'interface utilisateur ne pas être rafraîchie, le temps de traitement doit donc être aussi court que possible (quelques ms).
- Pour assurer un traitement rapide, on peut utiliser :
  - Des opérations d'**E/S asynchrone** et des **promesses.**
  - Une bibliothèque permettant l'exécution de longs calculs à l'aide de ***worker threads*** (p. ex. Web Workers API).

Gestionnaire d'événements

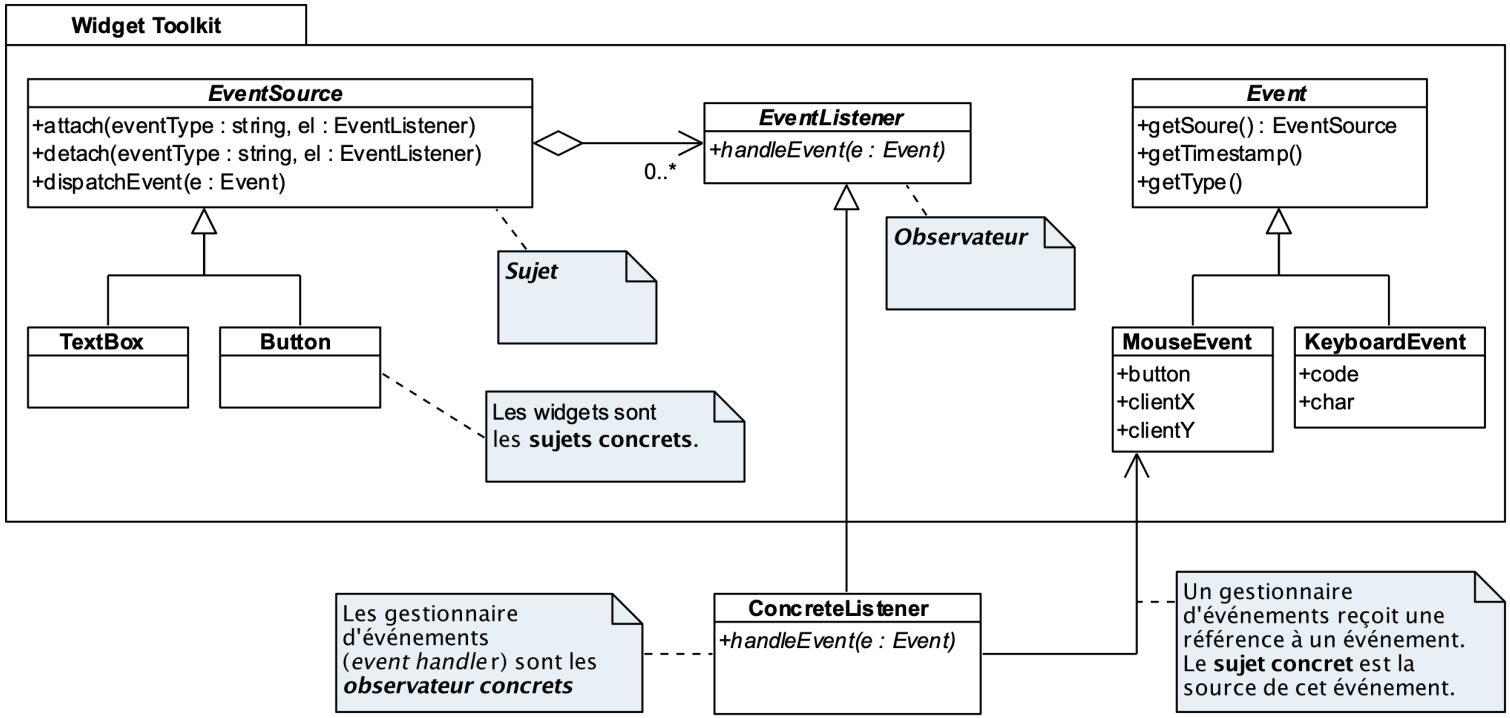
- Le traitement d'un événement est réalisé dans un sous-programme appelé gestionnaire d'événement (***event handler*** ou ***event listener***).
- Les gestionnaires d'événements sont appelé dans le corps de la boucle d'événement.
- Un gestionnaire d'événements :
  - Effectue le traitement des événements d'un ou plusieurs widgets appelé source d'événement (***event source***).
  - Reçoit une description de l'événement en paramètre.

Distribution des événements

- La distribution des événements (***event dispatching***) consiste à sélectionner le gestionnaire d'événement charger de traité un événement et à l'appeler.
- La distribution des événements constitue l'essentiel du le corps de la boucle d'événement.
- La boucle d'événement et la distribution des événements sont généralement implémentées dans le widget toolkit.

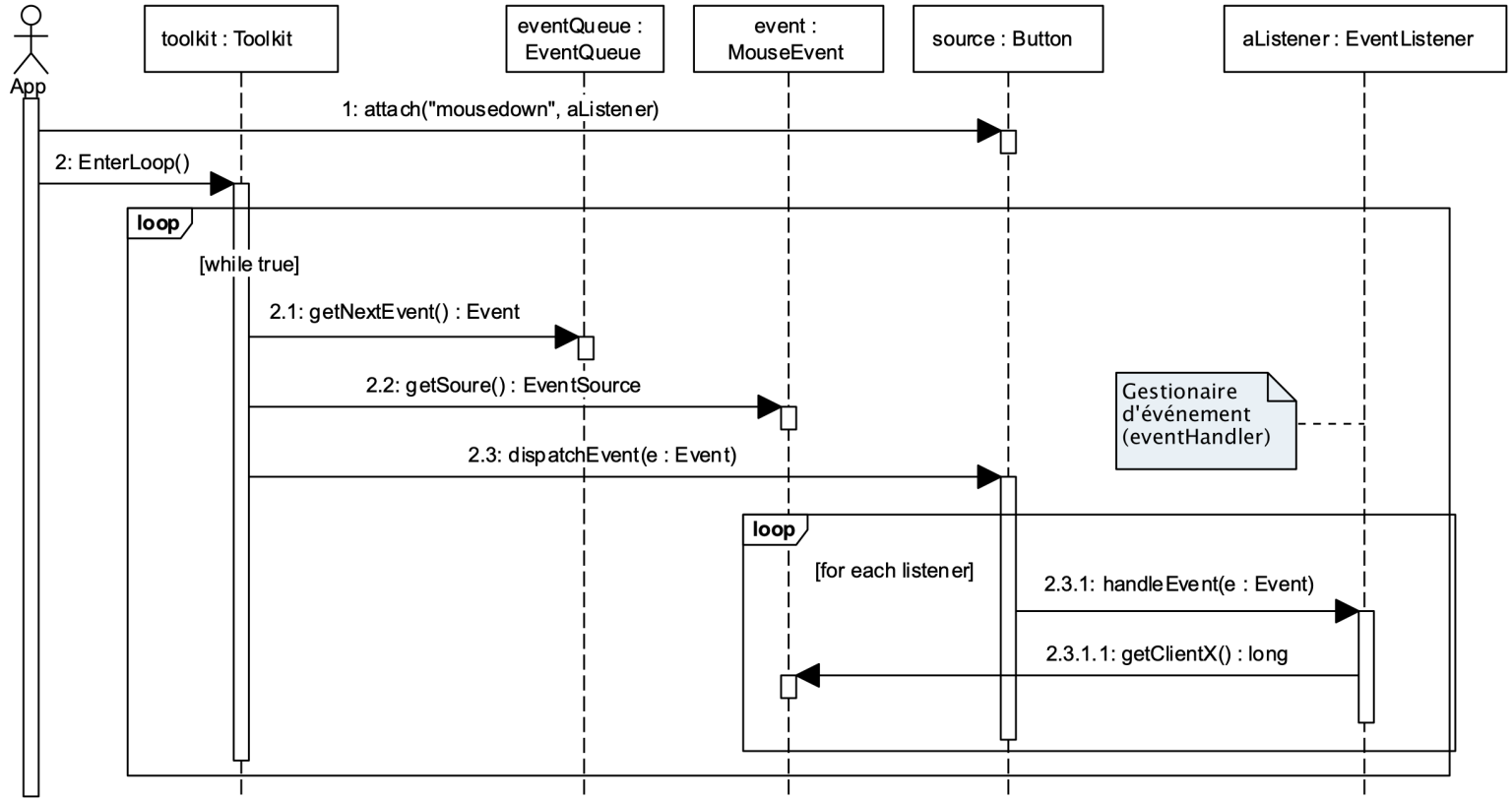
Distribution des événements (suite)

- La distribution des événement est souvent réalisée à l'aide d'une variation du patron de conception observateur (***observer pattern***).



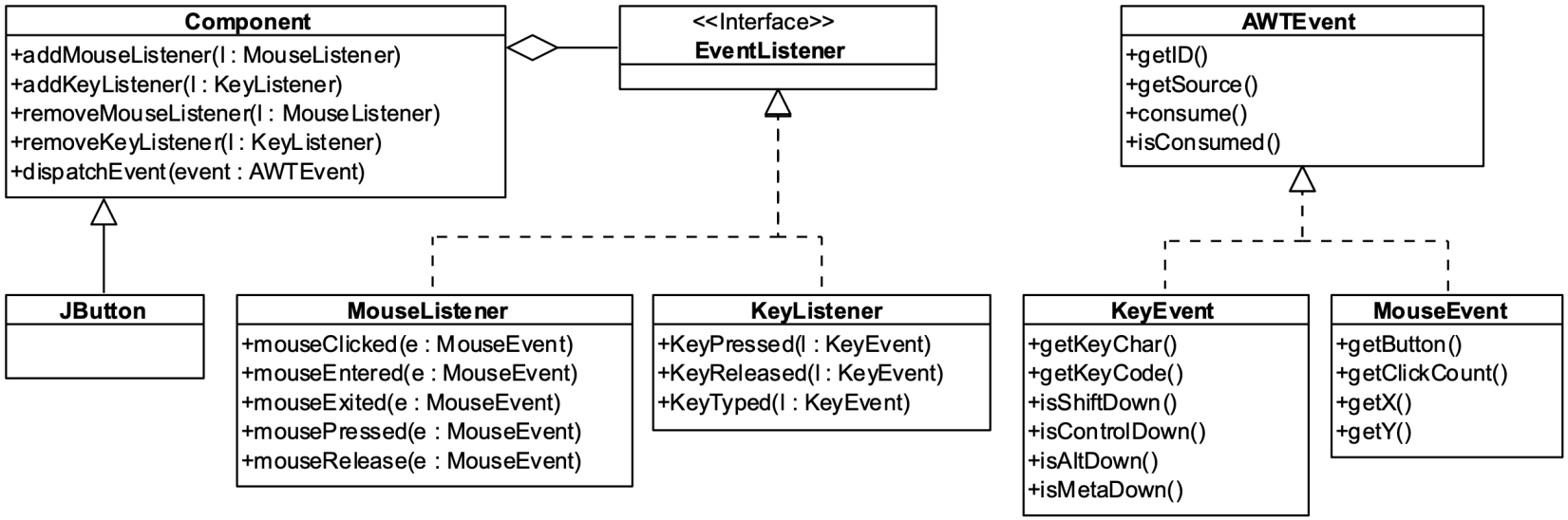
Distribution des événements (suite)

- Diagramme de séquence simplifiée de la distribution des événements :



Java Swing

- Vue simplifiée de l'implémentation du patron observateur dans le toolkit Java Swing :



HTML/JavaScript

- Vue simplifiée de l'implémentation du patron observateur dans d'un navigateur Web avec JavaScript :