Organisation

- Introduction à Swing
- Composants et Conteneurs
- Gestion de la Mise en Page
- **Gestion des Evénements et Ecouteurs**
- Modèles
- Gestion des Dialogues
- Menus
- Outils Graphiques

Contenu

- Programmation Evenementielle
- JAVA `Event`, `ActionEvent`
- Exécution du Code IHM
- Interface `ActionListener`
- Ecouteurs : Types et Implémentation

Programmation Evenementielle

- Exécution du programme est non séquentielle
- Lors de l'accès aux composants à l'écran, génération d'événements
- Evénements sont traités, ce qui engendre la réponse du programme, *guidant* l'exécution au travers d'événements
- Pourquoi est-ce que les programmes doivent être basés sur les événements ?
  - Permettre l'interaction de l'utilisateur avec l'IHM

Traitement des Evenements Swing

- Fonctionnement des actions en Swing fondé sur événements (actions utilisateur) et écouteurs (sources des actions)
- Exemple : lors du clic sur un bouton, on aura un événement (ActionEvent dans ce cas) qui sera envoyé à tous les écouteurs du bouton. Si l'on veut savoir quand se passe l'action d'un bouton, il faut mettre un écouteur sur ce bouton et réagir en cas d'événement

Hiérarchie Java Event

```
java.lang.Object
  +--java.util.EventObject
        +--java.awt.AWTEvent
              +--java.awt.event.ActionEvent
              +--java.awt.event.TextEvent
              +--java.awt.event.ComponentEvent
                    +--java.awt.event.FocusEvent
                    +--java.awt.event.WindowEvent
                    +--java.awt.event.InputEvent
                          +--java.awt.event.KeyEvent
                          +--
  java.awt.event.MouseEvent
```

- **`import java.awt.event.*;`**

`ActionEvent`

- Type d'événement le plus commun/simple en Swing
- Représente une action se déroulant au niveau d'un composant de l'IHM
- Créée par :
  - Clic de bouton
  - Validation ou invalidation d'une check box
  - Clic de menu
  - Presser le bouton Entrée dans un champ texte
  - etc.

Exécution du code IHM : Processus et Threads

- Un programme exécuté sur un ordinateur est appelé processus
  - La majorité des SE (dits **multi-tâche**) destinés au grand public permettent à plusieurs processus de s'exécuter simultanément
  - Un programme peut engendrer plusieurs processus
    - Jean exécute Netscape – un processus
    - Malika exécute Netscape – un autre processus
- Chaque processus a son propre (principal) thread d'exécution
  - Un processus peut toutefois engendrer plusieurs threads

Exécution du code IHM : Threads Java

- Tout programme Java a un thread d'exécution principal
  - Débute avec :

    ```
    public static void main(String [] args)
    ```
- Les applications graphiques ont un thread spécifique appelé Event Dispatching Thread (EDT)
- Important de ne pas mélanger les deux
  - Actions en tâches de fond et interface toujours réactive
  - Sinon, exécution d'une action longue dans l'EDT, bloquage de celui-ci et interface figée jusqu'à la fin de cette action

L'interface Runnable

- L'interface Runnable spécifie une méthode

```
public interface Runnable {
    /**
     * Quand un objet implémentant l'interface Runnable est
     * utilisé pour créer un thread, le démarrage du thread
     * provoque l'appel de la méthode run dans ce thread
     * exécuté séparément.
     * Le contrat de la méthode run n'indique pas d'action
     * spécifique.
     *
     * @see     java.lang.Thread#run()
     */
    public abstract void run();
}
```

Comment est-ce que Runnable est utilisée ?

- Un objet qui est `Runnable` a une méthode `run`.
- Un objet qui est `Runnable` peut être exécuté dans un thread spécifique.
- La classe `javax.swing.SwingUtilities` fournit une méthode, `invokeLater`, qui prend un `Runnable` en paramètre, et appelle sa méthode `run` une fois que les autres événements dans la file de l'EDT ont été traités.
- Lien avec IHM ? Pour être "thread safe", les applications graphiques doivent exécuter leur code interface sur l'EDT.
    - Ecrire le code pour l'IHM dans un `Runnable`
    - Passer cet objet à la méthode `invokeLater`

Exemple : la classe TestJFrame

```
public class TestJFrame {
 public static void main(String[] args){
  SwingUtilities.invokeLater(new Runnable(){
   public void run(){
    // Création d'une instance de SimpleFenetre
    SimpleFenetre fenetre = new SimpleFenetre();
    fenetre.setVisible(true);//On la rend visible
   }
  });
 }
}
```

Ecoute des Evénements : Interface `ActionListener`

- Attacher un *écouteur* au composant
- La méthode appropriée de l'écouteur est appelée lorsqu'un événement se déroule (e.g. lors du clic bouton)
- Pour des actions, principe basé sur une classe implémentant `ActionListener` et écoute du composant
- Exemple du bouton
  - Ecrire une classe qui implémente l'interface `ActionListener`
  - L'ajouter comme écouteur du bouton
  - A chaque clic, la méthode **actionPerformed** va être appelée et il faudra tester le bouton qui a envoyé l'événement

Quelques Evénements de Composants et Ecouteurs Typiques

Action résultant en un événement	**Ecouteur**
Utilisateur clique sur un bouton, presse la touche *entrée* en entrant du texte dans un champ de texte, ou choisit un élément du menu	`ActionListener`
Utilisateur ferme une fenêtre	`WindowListener`
Utilisateur appuie sur un bouton de souris pendant que le curseur se trouve sur un composant	`MouseListener`
Utilisateur déplace la souris sur un composant	`MouseMotionListener`
Composant devient visible	`ComponentListener`
Composant obtient le focus du clavier	`FocusListener`
Sélection d'une liste ou d'un tableau	`ListSelectionListener`

Implémentation des Ecouteurs

- Basée sur le mécanisme de rappel : modèle **Emetteur/Souscripteur**
- Trois portions de code principales
  - 1) implémentation de l'interface
  - 2) souscription
  - 3) traitement
- Composants peuvent avoir plusieurs écouteurs
- Question : Où intégrer la classe écouteur ?
  - Exemple d'un JButton ActionListener…

Classe Ecouteur Interne

```
public class Externe {
  private class Interne implements ActionListener
  {
    public void actionPerformed(
                          ActionEvent e) {
      // réponse à l'événement
    }
  }
  public Externe() {
    …
    JButton myButton = new JButton();
    myButton.addActionListener(new Interne());
    …
  }
}
```

Classe Ecouteur Anonyme

```
public class Externe {
  public Externe() {
    …
    JButton myButton = new JButton();
    myButton.addActionListener(
      new ActionListener() {
        public void actionPerformed(ActionEvent e) {
          // réponse à l'événement
        }
      }
    );
    …
  }
}
```

`ActionListener` dans une Classe Dérivée de JFrame

```
public class Externe extends JFrame
                          implements ActionListener {
  public Externe() {
    …
    JButton myButton = new JButton();
    myButton.addActionListener(this);
    …
  }

  public void actionPerformed (ActionEvent event) {
    // réponse à l'événement
  }
}
```

Types d'Ecouteurs (1)

- Ecouteurs de composants **génériques**
  - Utilisables pour tout composant Swing
  - Types
    - ComponentListener (changement de taille, position, visibilité)
    - FocusListener (détermine le composant qui a le focus de l'interface)
    - KeyListener
      - Permet de déterminer la touche pressée par l'utilisateur pendant qu'un composant (e.g., JTextField, JLabel, JTextArea) a le focus de l'IHM
    - MouseListener (clic et mouvement à l'intérieur/extérieur de la zone occupée par le composant)
    - MouseMotionListener (changements de position au-dessus de la zone occupée par le composant)

Types d'Ecouteurs (2)

- Ecouteurs spécifiques aux composants
  - Réservés pour des actions spécifiques de composants
  - Types
    - ActionListener
    - ItemListener
    - ListSelectionListener
    - WindowListener
    - etc.
- Voir :

http://java.sun.com/docs/books/tutorial/uiswing/events/eventsandcomponents.html

Récupérer de l'Information sur les Evénements

- Classe **`EventObject`** – utilisation de sous-classes pour déterminer ce qu'il s'est déroulé
- Détermination de l'objet à la source de l'événement grâce à **`getSource();`**

Exercices

- Fenêtre avec 2 boutons ‘Ici’ et ‘Là’
  - Affichage sur la console : “Vous avez cliqué Ici” ou “Vous avez cliqué Là”

- Calculatrice à boutons
  - Affichage sur la console du réel entré