Modalité de l'examen : Projet

1. Introduction Java
2. Introduction IHM: Swing(composants, conteneurs, layouts),[réflexion projet]
3. **Les évènements, [Cahier des charges]**
4. Boites de dialogues, [Projet]
5. MVC, [Projet]
6. Composants complexes, [Projet]
7. Présentations

Expression lambda

Les expressions lambda ne s'utilisent que dans le contexte des **interfaces fonctionnelles**.

Exemple d'interface:

```
public interface MonAction {
    void executer();
}
```

Utilisation sans lamda

```
public class ExempleSansLambda {
    public static void main(String[] args) {
        // Implémentation classique via classe anonyme
        MonAction action = new MonAction() {
            @Override
            public void executer() {
                System.out.println("Action exécutée !");
            }
        };

        action.executer();
    }
}
```

c'est une classe anonyme

version avec lamda

```
public class ExempleLambda {
    public static void main(String[] args) {
        // Utilisation d'une lambda pour implémenter la méthode executer()
        MonAction action = () -> System.out.println("Action exécutée !");

        // Appel de la méthode
        action.executer();
    }
}
```

Ce code est un raccourci lors d'une interface : action listener - mouselistener etc.

ces deux actions sont les mêmes.

Evènements

Internet

- https://docs.oracle.com/javase/tutorial/uiswing/events/handling.html

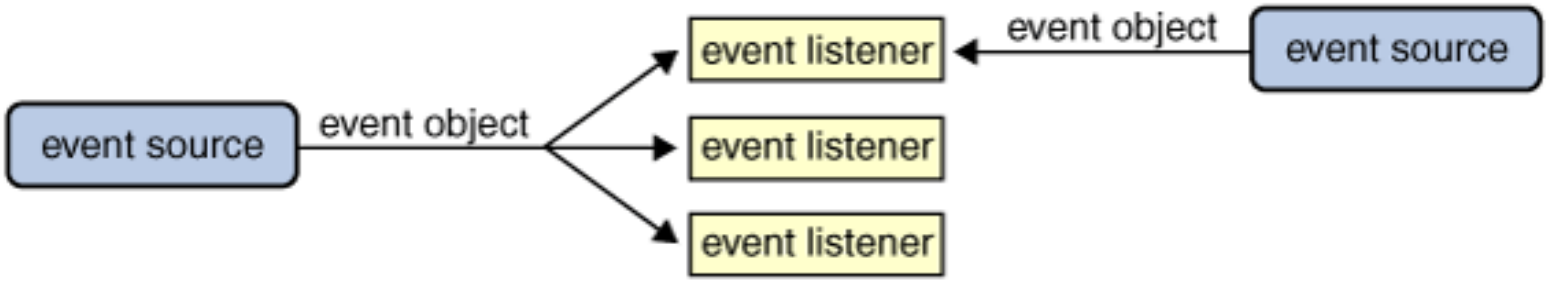
Evénement

- Les composants Swing permettent (en général) à l'utilisateur d'agir sur l'application

  – pour cela ils effectuent une série de **gestes** depuis les périphériques, lesquels sont interprétés comme **actions** logiques

    - gestes : événements bruts, bas-niveau
    - actions : événements logiques, sémantiques

Evénement



- Une sollicitation depuis un composant (*source*)
  - provoque la création d'un événement (*event*)
    - lequel est distribué aux parties fonctionnelles de traitement (*listener*)
- découplage entre interface et métier

Evénement

- Ceci nécessite qu'un Listener s'enregistre auprès d'une source
  - mécanisme de rappel (*callback*)
  - « Tiens voilà mon numéro de téléphone, comme ça tu pourras me prévenir plus tard »
- Attention : lors du rappel, soyez rapide !
  - ne faites pas traîner la conversation car le rappel est effectué dans le même Thread que celui qui gère les événements

Composant et événement

- Chaque composant Swing peut générer des événements
  - deux catégories :
    - les événements génériques aux composants Swing
    - les événements spécifiques (relatifs à la sémantique de l'objet graphique)

Evénement génériques

- Les événements génériques (principaux) :
  - ComponentEvent (taille, position, visibilité)
  - FocusEvent (*capture* le clavier ou non)
  - KeyEvent (frappe clavier)
  - MouseEvent (action souris)
  - MouseMotionEvent (action souris)

ActionEvent

- JButton, JMenuItem, JToggleButton, JRadioButton, JTextField, JPasswordField, JComboBox, JCheckBox,JFormattedTextField
  - ActionEvent
    - représente l'action logique (clic sur le bouton, choix de l'option, saisie du champ, etc)
      - attention : action logique
        - *i.e.* : le clic peut-être obtenu par un raccourci par exemple…

ActionListener

- pour qu'un objet reçoive un ActionEvent
  - il faut qu'il implémente l'interface
  - ActionListener
    - public void

      actionPerformed(ActionEvent)!
  - méthode appelée si l'objet est enregistré auprès d'une source possible
    - addActionListener(ActionListener)
  - et lorsqu'on déclenche par actions physiques l'action logique (clic sur un bouton par ex.))

si on a plusieurs possibilités de source, plusieurs bouton, alors on peu

ActionListener/ActionEvent

```
public class MaClasseDEcouteur implements ActionListener {
…
public void actionPerformed(ActionEvent e) {
      // code exécuté si l'instance est enregistrée
      // et si l'utilisateur déclenche l'Action
      }
…
}
JButton monComposant = new JButton("un bouton");
…
instanceDeMaClasseEcouteur = new MaClasseDEcouteur();
// enregistrement d'un écouteur sur un composant
// susceptible de générer une Action
monComposant.addActionListener(instanceDeMaClasseDEcouteur);
…….
```

Source/Listener

- puisqu'un Listener peut recevoir un événement provenant de différentes Sources
  - comment distinguer la source ?
    - la classe EventObject (super-classe des événements) fournit la méthode
      - Object getSource()

Source/Listener

```
public class MaClasseDEcouteur implements
                                    ActionListener {
…
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()== monComposant) { ... }
        if (e.getSource()== monAutreComposant) { ... }
    }
…
}
…
instanceDeMaClasseEcouteur = new MaClasseDEcouteur();
// enregistrement d'un écouteur sur un composant!
// susceptible de générer une Action
monComposant.addActionListener(instanceDeMaClasseDEcouteur);
monAutreComposant.addActionListener(instanceDeMaClasseDEcouteur);
…
```

Source/Listener

- utiliser une **commande** associée :
  - une chaîne de caractère portée par la Source et qui permet au Listener de réaliser des variantes de traitement
  - avantages
    - c'est la Source qui décide quelle variante elle veut obtenir
    - une Source peut facilement changer de rôle
  - String getActionCommand()
  - setActionCommand(String)

Source/Listener

```
public class MaClasseEcouteur implements ActionListener {
    private void save() { ... }
    private void quit() { ... }
    public void actionPerformed(ActionEvent e) {
    if (e.getActionCommand().equals("save and quit")) {
            save();
            quit();
        }
        if (e.getActionCommand().equals("quit")) {
             quit();
        }
    }
}
...
instanceDeMaClasseEcouteur = new MaClasseEcouteur();
JButton b = new JButton("Quitter");
b.setActionCommand("quit");
b.addActionListener(instanceDeMaClasseEcouteur);
...
```

Action

- une Action (sous-interface de ActionListener) doit répondre à différentes méthodes
  - actionPerformed évidemment
  - en particulier doit permettre d'y associer/ retrouver des valeurs associées à des clés
    - c'est une **mémoire associative**
    - les composants concernés peuvent y retrouver ce dont ils ont besoin (icône, texte, etc)

Source/Listener

```
public class Quitter extends AbstractAction {
    public Quitter() {
        putValue(Action.SHORT_DESCRIPTION,
                                        "Quitter l'application");
        putValue(Action.NAME,"Quitter");
    }
    public void actionPerformed(ActionEvent e) {
        // fait quelque chose pour quitter l'application
        System.exit(0);
    }
}
...
// instanciation d'une action pour quitter
Quitter q = new Quitter();
// un bouton qui occasionnera l'action de quitter
JButton b = new JButton(q);
......
```

utlise pour si 2 boutons font la même chose. pour éviter de réécrire le code

KeyListener

- Il peut être utile de recevoir des événements en provenance du clavier
  - interface KeyListener
- Enregistrement d'un tel écouteur sur un composant
  - addKeyListener(KeyListener)

KeyListener

- deux types d'événements :
  - la saisie d'un caractère (Unicode)
    - keyTyped(KeyEvent)
  - l'appui et le relâchement d'une touche
    - keyPressed(KeyEvent)
    - keyReleased(KeyEvent)

KeyEvent

- KeyEvent
  - char getKeyChar() pour récupérer un caractère Unicode ou CHAR_UNDEFINED
  - int getKeyCode() pour récupérer la *touche clavier*
    - renvoie un code VK_*
      - VK_A, VK_ENTER, VK_END, VK_F1,
      - VK_PAGE_DOWN…

MouseListener MouseMotionListener MouseWheelListener

- Les événements provenant de la souris sont classés en 3 catégories :
- les **clics**
  - les plus ordinaires
- les **déplacements**
  - nécessitent une attention particulière car ils peuvent être très nombreux
- la rotation de la **roue**
  - dont la gestion est particulière

MouseListener

- Les clics
  - interface MouseListener
    - addMouseListener(MouseListener) de la classe Component
  - 5 méthodes

MouseListener

- mouseClicked(MouseEvent)
- mousePressed(MouseEvent)
- mouseReleased(MouseEvent)
- mouseEntered(MouseEvent)
  - lorsque la souris entre dans l'espace du composant
- mouseExited(MouseEvent)
  - lorsque la souris sort de l'espace du composant

MouseEvent

- int getButton()
  - MouseEvent.BUTTON1
  - MouseEvent.BUTTON2
  - MouseEvent.BUTTON3
- int getClickCount()!
  - multi-clic
- int getX(), int getY()

MouseMotionListener

- les déplacements de la souris
  - ne doivent être interceptés que lorsque c'est strictement utile
    - le nombre d'événements générés peut être important
      - appels très nombreux des callbacks...
- en général on attend l'appui sur un bouton pour démarrer la réception
  - puis on arrête d'écouter en relâchant un bouton

MouseMotionListener

- les déplacements de la souris
  - interface MouseMotionListener
  - addMouseMotionListener dans les composants
  - avec bouton maintenu
    - mouseDragged(MouseEvent)
  - sans bouton maintenu
    - mouseMoved(MouseEvent)

MouseWheelListener

- int getScrollAmount()
  - nombre d'unités par clic de roue
- int getScrollType()
  - deux types : WHEEL_UNIT_SCROLL, WHEEL_BLOCK_SCROLL
- int getUnitsToScroll()
  - nombre d'unités totales à scroller (amount*rotation)
- int getWheelRotation()
  - nombre de clics de la roue

MouseWheelListener

- aucun moyen de savoir si une souris est équipée d'une roue
- interface MouseWheelListener
  - addMouseWheelListener des composants
- Méthode mouseWheelMoved(MouseWheelEvent)
  - attention, événement particulier MouseWheelEvent

Listeners courants

haute école neuchâtel berne jura arc ingénierie saint-imier le locle delémont

Listeners/Event	Methodes de Listeners	Générés par
ActionListener/ ActionEvent	actionPerformed()	AbstractButton, Button, ButtonModel, ComboBoxEditor, JComboBox, JFileChooser, JTextField, JMenuItem
FocusListener/ FocusEvent	FocusGained() FocusLost()	Component
ItemListener/ ItemEvent	itemStateChanged()	AbstractButton, ButtonModel, Checkbox, CheckboxMenuItem, Choice, ItemSelectable, JComboBox, List
KeyListener/ KeyEvent	keyPressed() keyReleased() keyTyped()	Component

Listeners courants

Listeners/ Event	Méthodes de listener	Générés par
MouseListener/ MouseEvent	mouseClicked() Component mouseEntered() mouseExited() mousePressed() mouseReleased()	Component
MouseMotionListener MouseEvent	mouseDragged() mouseMoved()	Component
TextListener TextEvent	textValueChanged()	TextComponent
WindowListener WindowEvent	windowActivated() windowClosed() windowClosing() windowDeactivated() windowDeiconified() windowIconified() windowOpened()	Window

Listeners courants

haute école neuchâtel berne jura arc ingénierie saint-imier le locle delémont

Listeners/Event	Methodes de Listeners	Générés par
ComponentListener/ ComponentEvent	componentHidden() componentShown() componentMoved() componentResized()	Component
ChangeListener ChangeEvent	stateChanged()	AbstractButton, BoundedRangeModel, ButtonModel, JProgressBar, JSlider, JTabbedPane, JViewport, MenuSelectionManager, SingleSelectionModel
ListDataListener ChangeEvent	contentsChanged() intervalAdded() intervalRemoved()	AbstractListModel, ListModel
ListSelectionListener ListSelectionEvent	valueChanged()	AbstractListModel, ListModel
CaretListener CaretEvent	caretUpdate()	JTextComponent