Programmation orientée objet en langage JAVA

Chapitre 5 : Les Interfaces Homme-Machine

Claude Duvallet

Université du Havre
UFR Sciences et Techniques
25 rue Philippe Lebon - BP 540
76058 LE HAVRE CEDEX
Claude.Duvallet@gmail.com
http://litis.univ-lehavre.fr/~duvallet/

Les Interfaces Homme-Machine

Introduction (1/3)

- La communication avec l'utilisateur se fait grâce à des composants graphiques :
  - menus,
  - boites de dialogue,
  - cases à cocher,
  - zone de saisie,
  - ...

$\Rightarrow$ le programme réagit à des évènements : programmation évènementielle.

- La gestion des interfaces graphiques est intégrée dans Java :
  - "boucle" gérée par le système d'exploitation.

Introduction (2/3)

- La portabilité des IHM en Java se fait au moyen de l'AWT :
  - éléments graphiques : boutons, cases à cocher, liste déroulantes, barres de défilement, menus déroulants,...
  - utilisables dans les applets et les applications autonomes.
  - s'adaptent au visuel des différents environnements car construit sur les fonctions natives des systèmes d'exploitation.
  - dérivables pour les adapter à ses propres besoins.
  - l'AWT fournit aussi différents protocoles d'agencement des éléments graphiques.

Introduction (3/3)

- Contient la définition conceptuel des objets :
  - leur comportement et leur spécification.
  - pas leur aspect graphique qui dépend de la plateforme d'exécution.
- Les inconvénients possibles :
  - la portabilité ne permet pas d'exploiter les fonctions graphiques avancées de certains environnements.
  - dans la version 1.0, certains composants n'avaient pas le même comportement sur différents environnements.
- L'avantage majeur :
  - l'AWT fournit un outil d'interfaçage complètement gratuit et portable.

Support des interfaces graphiques

- *Abstract Windowing Toolkit (AWT) :*
  - support pour les composants courants,
  - système de fenêtrage,
  - limité.
- *Swing :*
  - boîte à outils construite sur AWT,
  - plus robuste et plus de fonctionnalités.

Les composants AWT (1/2)

- Les principaux composants :
  - Les conteneurs (Container) : ils sont capables de contenir des composants graphiques ou d'autres conteneurs.
  - Les canevas (Canvas) : ce sont des objets graphiques très simples sur lesquels on peut dessiner et afficher des images.
  - les composants graphiques usuels : ce sont tous les objets classiques qui sont présents dans les interfaces graphiques tels que les boutons, les étiquettes, les cases à cocher, etc.
- Les conteneurs :
  - `Container` : Gestionnaire de composants graphiques.
  - `Window` : Fenêtre sans bordure.
  - `Frame` : Fenêtre avec bordure.
  - `Dialog` : Boîte de dialogue.
  - `FileDialog` : Boîte de dialogue pour la sélection des fichiers.
  - `Panel` : Conteneur pouvant contenir d'autres conteneurs.

Les composants AWT (2/2)

- Les composants graphiques classiques :
  - `Button` : Bouton.
  - `CheckBox` : Cases à cocher.
  - `Label` : Texte statique.
  - `Scrollbar` : Barre de défilement.
  - `TextArea` : Champ de texte multiligne.
  - `TextField` : Champ de texte.
  - `MenuBar` : Barre de menu.
  - `Menu` : Menu.
  - `Choice` : Menu déroulant.
  - `List` : Liste de choix.

Conteneurs

- Une interface graphique est un assemblage de conteneurs nommés *Container* :
  - il faut d'abord créer une classe représentant l'interface utilisateur graphique :
    - le *container* contient tous les autres composants.
- Les conteneurs les plus utilisés sont :

  Panel
  - représentent des panneaux.

  Frame
  - représentent des fenêtres ayant un cadre.
- Une manière de créer une application graphique consiste à faire de l'interface une sous-classe de Frame :
  - exemple :

```
public class MonInterface extends Frame{
        ...
}
```

- La méthode `setContentPane(Object)` indique le container que doit contenir la fenêtre invoquée.

Création d'une interface graphique (1/2)

- Les cadres sont invisibles lorsqu'ils sont créés :
  - pour les rendre visible :

    méthode `setVisible(true)`
  - pour les masquer :

    méthode `setVisible(false)`
- Un cadre s'affiche dans le coin supérieur gauche :
  - pour spécifier un emplacement différent :

    méthode `setBounds(int, int, int, int)`

    Les 2 premiers paramètres sont la position (x, y).

    Les 2 derniers paramètrent la largeur et la hauteur.
  - pour paramétrer le titre :

    méthode `setTitle(String)`
  - taille du cadre :

    méthode `setSize(int, int)`

Création d'une interface graphique (2/2)

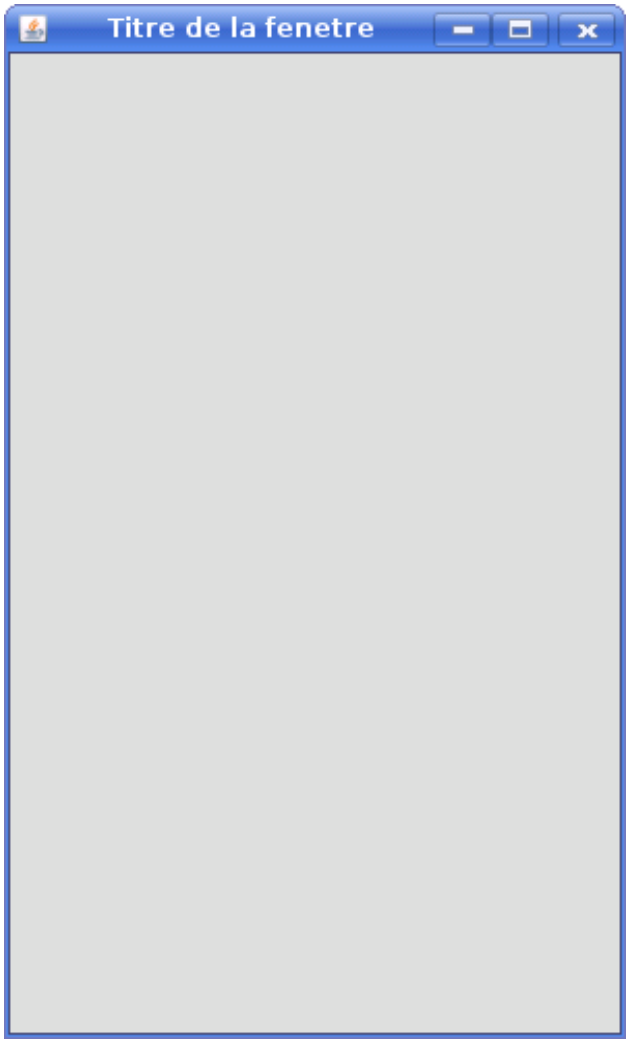
- Exemple de fenêtre :

```
import java.awt.*;

public class Fenetre extends Frame{

    public Fenetre(){
        setBounds(100, 100, 300, 500);
        setTitle("Titre de la fenetre");
    }

    public static void main(String[ ] args) {
        Fenetre fen = new Fenetre();
        fen.setVisible(true);
    }
}
```



Ajouter un composant

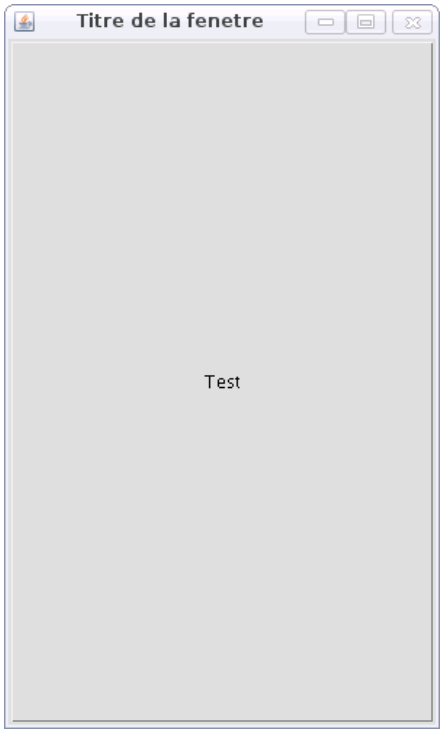
- Pour pouvoir afficher un composant, il est nécessaire de l'ajouter à un conteneur pour pouvoir être affichés.
- la méthode add permet alors d'ajouter un composant à ce conteneur :
  - référence du composant à ajouter comme argument.
- Exemple de fenêtre :

```
import java.awt.*;

public class Fenetre extends Frame{

    public Fenetre(){
        setBounds(100, 100, 300, 500);
        setTitle("Titre de la fenetre");
        add (new Button("Test"));
    }

    public static void main(String[ ] args) {
        Fenetre fen = new Fenetre();
        fen.setVisible(true);
    }
}
```



Les interfaces graphiques avec *Swing*

- *Swing* permet de créer des interfaces utilisateur graphiques plus élaborées qu'avec l'AWT.
- *Swing* repose sur l'AWT et certains éléments n'ont pas été rédéfinie tels que les gestionnaires d'agencement.
- *Swing* est employé pour concevoir des applications :
  - boutons : éléments cliquables,
  - étiquettes : fournissant des informations,
  - champs de texte : entrée clavier,
  - menu déroulant,
  - ...
- Tous les éléments de *Swing* appartiennent au package `javax.swing` :
  - `import javax.swing.*;`

Objectifs

- Introduction aux composants Swing de base, leurs méthodes et les événements qu'elles génèrent.
- Différents compoants Swing :
  - JLabel
  - JButton
  - JToggleButton
  - JCheckBox
  - JComboBox
  - JSlider
  - JTextField
  - JPasswordField
  - JTextArea

Les composants graphiques

- Les composants sont créés en utilisant des constructeurs :

```
JLabel invite = new JLabel ("Invite");
```

- Pour être utilisable, un composant doit être ajouté au contenu de la fenêtre de l'application ou à un autre composant :

```
JPanel panneauConfig = new JPanel();
scorePanel.add (invite);
```

- Les composants (sauf JLabel) peuvent générer des événements.
- Les événements sont capturés et exécutés par des "listeners" – des objets équipés pour prendre en charge un type particulier d'événement.
- Différents types d'événements sont traités par différents types de listeners.

JLabel

- Constructeurs :

```
JLabel (String text);
JLabel (ImageIcon icon);
JLabel (String text, ImageIcon icon, SwingConstants.LEFT);
        // ou CENTER, RIGHT, LEADING, TRAILING.
```

- Méthodes :

```
void setText (String text);
void setIcon (ImageIcon icon);
```

- Événements : Aucun.

JButton

- Constructeurs :

```
JButton (String text);
JButton (ImageIcon picture);
JButton (String text, ImageIcon picture);
```

- Méthodes :

```
void addActionListener (ActionListener object);
void setText (String text);
void setActionCommand (String cmd);
void setIcon (ImageIcon icon);
void requestFocus();
```

- Événements :

```
class ... implements ActionListener {
    public void actionPerformed(ActionEvent ae){
        JButton b = (JButton)ae.getSource();
        String s = ae.getActionCommand();
        ...
    }
}
```

JTextField

- Un champ de texte est une zone dans laquelle un utilisateur peut :
  - saisir,
  - modifier,
  - du texte au clavier.
- Constructeurs :

```
JTextField()              // champ de texte vide
JTextField(int)           // champ de texte de la largeur spécifiée
JTextField(String, int) // champ avec texte de largeur spécifiée
```

- Méthodes :

```
String getText()          // permet de récupérer le contenu du champs
```

JTextArea

- Les zones de texte peuvent traiter plusieurs lignes d'entrées.
- Constructeurs :

```
JTextArea(int, int)              // zone de texte avec nombre lignes et
                                 // nombre colonnes spécifiés
JTextArea(String, int, int)      // zone avec texte et nombre lignes et
                                 // nombre colonnes spécifiés
```

- Méthodes :

```
String getText()        // permet de récupérer le contenu du champs
```

JTextArea : Exemple

- Classe Fenetre

```
import javax.swing.*;

public class Fenetre extends JFrame{
    private JLabel titre = new JLabel("Formation JAVA", SwingConstants.CENTER);
    private JTextField text = new JTextField("Java");
    private JTextArea area = new JTextArea(20, 10);

    public Fenetre() {
        setBounds(250, 250, 250, 250);
        setTitle("Exemple d'interface");
        JPanel pane = new JPanel();
        pane.add(titre);
        pane.add(text);
        pane.add(area);
        setContentPane(pane);
    }

    public static void main(String[ ] args) {
        Fenetre fen = new Fenetre();
        fen.setVisible(true);
    }
}
```

JCheckBox

- Constructeurs :

```
JCheckBox (String text, boolean checked);
JCheckBox (ImageIcon icon, boolean checked);
JCheckBox (String text, ImageIcon icon, boolean checked);
```

- Méthodes :

```
void addActionListener (ActionListener object);
boolean isSelected ();
void setSelected (boolean checked);
void setText (String text);
void setIcon (ImageIcon icon);
```

- Événements :

```
class ... implements ActionListener{
    public void actionPerformed(ActionEvent ae){
        JCheckBox b = (JCheckBox)ae.getSource();
        if (b == checkBox1 && b.isSelected())
            ...
    }
}
```

Cases à cocher et boutons radio

- Les cases à cocher et boutons radio n'admettent que 2 valeurs possibles :
  - sélectionné,
  - non sélectionné.
- Les cases à cocher, JCheckBox contiennent une marque lorsqu'elles sont sélectionnées.
- Les boutons radio, JRadioButton sont des cercles contenant un point lorsqu'ils sont sélectionnés.
- Pour vérifier qu'un composant est activé, appel de sa méthode isEnabled, qui retourne une valeur booléenne.

JRadioButton (1/2)

- possède des méthodes constructeurs dont les arguments et les fonctionnalités sont les mêmes.
- Exemple :

```
import javax.swing.*;

public class Fenetre extends JFrame{
    private JRadioButton ville1 = new JRadioButton("Lyon", true);
    private JRadioButton ville2 = new JRadioButton("Paris", true);
    private JRadioButton ville3 = new JRadioButton("Lille", true);

    public Fenetre() {
        setBounds(250, 250, 250, 350);
        setTitle("Exemple d'interface");
        JPanel pane = new JPanel();
        pane.add(ville1);
        pane.add(ville2);
        pane.add(ville3);
        setContentPane(pane);
    }

    public static void main(String[] args) {
        Fenetre fen = new Fenetre();
        fen.setVisible(true);
    }
}
```

JRadioButton (2/2)

- Pour organiser en groupe, créer une classe d'objet : *ButtonGroup*.
- Exemple :

```
import javax.swing.*;
public class Fenetre extends JFrame{
    private JRadioButton ville1 = new JRadioButton("Lyon", true);
    private JRadioButton ville2 = new JRadioButton("Paris", true);
    private JRadioButton ville3 = new JRadioButton("Lille", true);
    private ButtonGroup choix = new ButtonGroup();

    public Fenetre() {
        setBounds(250, 250, 250, 350);
        setTitle("Exemple d'interface");
        JPanel pane = new JPanel();
        choix.add(ville1);
        choix.add(ville2);
        choix.add(ville3);
        pane.add(ville1);
        pane.add(ville2);
        pane.add(ville3);
        setContentPane(pane);
    }

    public static void main(String[ ] args){
        Fenetre fen = new Fenetre();
        fen.setVisible(true);
    }
}
```

Menu déroulant (1/2)

- Les listes déroulantes sont des composants qui permettent de choisir un seul composant dans une liste.
- Méthodes constructeurs :

```
JComboBox() // utilisée sans argument
```

- Méthode :

```
addItem(objet) // ajoute des éléments à la liste
```

Menu déroulant (2/2)

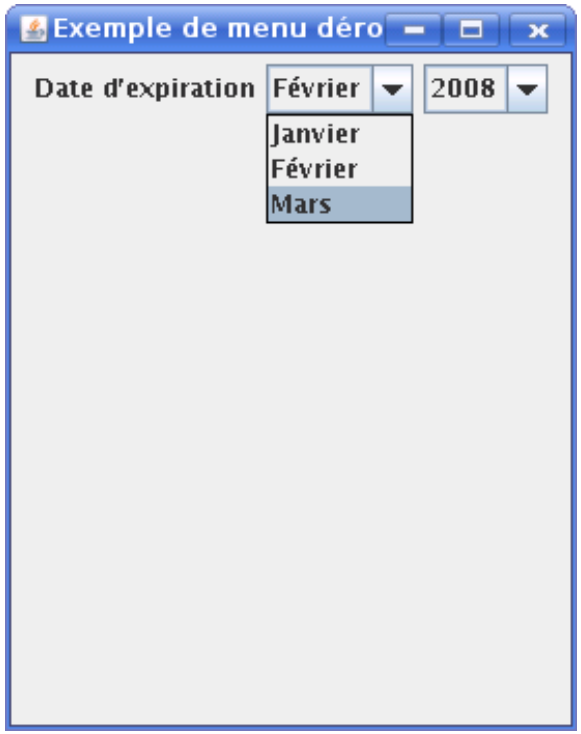
- Exemple :

```
import javax.swing.*;

public class Fenetre extends JFrame{
    private JComboBox mois = new JComboBox();
    private JComboBox annee = new JComboBox();
    private JLabel date = new JLabel("Date d'expiration");

    public Fenetre() {
        setBounds(250, 250, 250, 350);
        setTitle("Exemple de menu déroulant");
        JPanel pane = new JPanel();
        mois.addItem("Janvier");
        mois.addItem("Février");
        mois.addItem("Mars");
        annee.addItem("2008");
        annee.addItem("2009");
        pane.add(date);
        pane.add(mois);
        pane.add(annee);
        setContentPane(pane);
    }

    public static void main(String[ ] args) {
        Fenetre fen = new Fenetre();
        fen.setVisible(true);
    }
}
```



Les curseurs (1/3)

- Les curseurs permettent de paramétrer une valeur numérique en faisant coulisser un curseur au sein d'une plage de valeur délimitée par :
  - valeur minimale,
  - valeur maximale.
- Méthodes constructeurs :

```
JSlider(int, int)              // valeurs minimale et maximale
JSlider(int, int, int)         // valeurs minimale, maximale et initiale
JSlider(int, int, int, int)    // valeurs minimale, maximale, initiale et
                               // orientation (HORIZONTAL, VERTICAL)
```

Les curseurs (2/3)

- Les curseurs disposent d'un élément optionnel indiquant les graduations :
  - séparer les graduations principales selon la valeur indiquée : `setMajorTickSpacing(int)`
  - séparer les graduations secondaires selon la valeur indiquée : `setMinorTickSpacing(int)`
  - déterminer si les graduations doivent être affichées : `setPaintTick(boolean)`
  - déterminer si l'intitulé du curseur doit être affiché : `setPaintLabels(boolean)`

  $\Rightarrow$ ces méthodes doivent être appelées sur le curseur, avant que ce dernier ne soit ajouté à un conteneur.

Les curseurs (3/3)

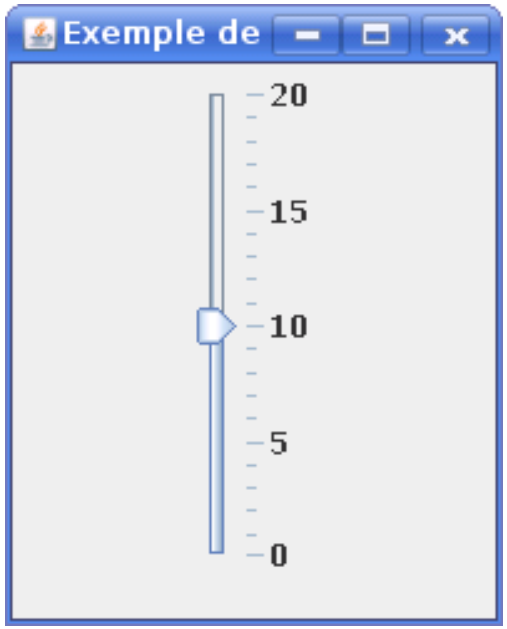
- Exemple :

```
import javax.swing.*;

public class Fenetre extends JFrame{
    private JSlider;

    public Fenetre() {
        setBounds(200, 200, 200, 250);
        setTitle("Exemple de curseur");
        JPanel pane = new JPanel();
        curseur = new JSlider(JSlider.VERTICAL, 0, 20, 10);
        curseur.setMajorTickSpacing(5);
        curseur.setMinorTickSpacing(1);
        curseur.setPaintTicks(true);
        curseur.setPaintLabels(true);
        pane.add(curseur);
        setContentPane(pane);
    }

    public static void main(String[ ] args){
        Fenetre fen = new Fenetre();
        fen.setVisible(true);
    }
}
```



Menus

- On peut ajouter un objet `JMenuBar` à `JFrame` ou `JApplet`.
- On peut ajouter des objets `JMenu` à `JMenuBar`.
- On peut ajouter `JMenus`, `JMenuItems`, `JCheckBoxMenuItems`, `JRadioButtonMenuItems`, etc. à `JMenu`.