Les interfaces utilisateur graphiques en Java

M. Berthouzoz
Cours préparé par L. Nerima
Université de Genève

Références

- Conception objet en Java avec BlueJ, David Barnes et Michael Kölling, 4e édition, Pearson Education, 2009
  Chapitre 11. Photocopies disponible sur le site Moodle "Info II", dans le dossier "Interfaces utilisateur graphiques en Java"
  https://moodle.unige.ch/mod/folder/view.php?id=968308
- Java : de l'esprit à la méthode , M. Bonjour, G. Falquet, J. Guyot, A. Le Grand, Vuibert, 2e édition, 1999
  Chapitres 15, 16, 17 et 18
  PDF en ligne: http://cui.unige.ch/isi/cours/javalettres/Java2EaM.pdf
- Tutoriel Swing (The Java Tutorials)
  https://docs.oracle.com/javase/tutorial/uiswing/index.html
- Exemples Swing (The Java Tutorials)
  http://docs.oracle.com/javase/tutorial/uiswing/examples/components/index.html
- Programmer en Java, Claude Delannoy, 11e édition, Eyrolles, 2020.

AWT et SWING

- Ces deux bibliothèques (packages) définissent les classes qui vont permettre d'instancier les composantes graphiques : champs de saisie, champs de texte, boutons, listes, boîtes à cocher, …
- Quelques classes d'AWT : *Button, Component, Frame, Label,…*
- SWING reprend pour l'essentiel les classes d'AWT ; leur nom est précédé d'un « J »: *JButton, JComponent, JFrame, JLabel,…*

- De manière générale, nous utiliserons les classes de SWING, sauf pour les conteneurs et la gestion des événements
- Attention à l'importation :

  import java.awt.;*

  import javax.swing.;*

Les conteneurs (1)

- En Java, les fenêtres contiennent des **conteneurs** : chaque fenêtre d'interface de la classe *JFrame* possède un conteneur qui lui est lié
- Un conteneur est un espace graphique qui va recevoir les composantes d'interaction (champs de saisie, champs de texte, boutons, listes, boîtes à cocher, …)
- Un conteneur peut aussi contenir d'autres conteneurs (conteneurs imbriqués)

Les conteneurs (2)

- La classe (dans le package AWT) s'appelle *Container*

- Pour instancier et initialiser le conteneur à partir de la fenêtre :
  JFrame fenetre = new JFrame("Mon IUG");
  Container conteneur = fenetre.getContentPane();
- Les principales méthodes de la classe *Container* :
  - Pour la gestion des composantes :
    add(), countComponent(), getComponent(), …
  - Pour la gestion de la mise en page :
    getLayout(), setLayout() …

Protocoles de mise en page

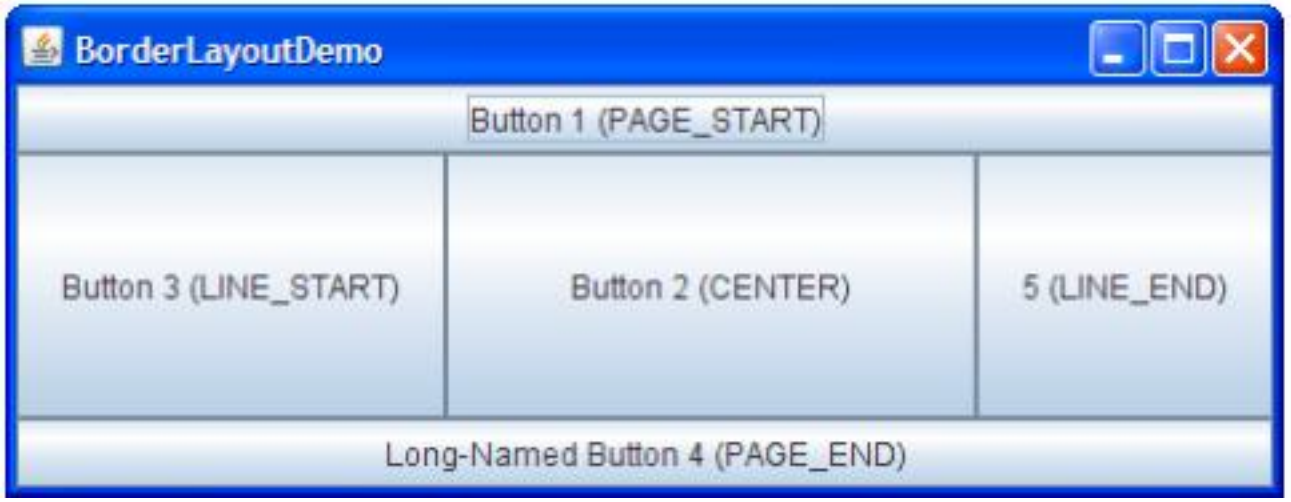
- Le protocole de mise en page va déterminer de quelle manière vont être placés les composantes graphiques dans le conteneur
- L'utilisation de protocoles va permettre d'adapter la disposition des composantes lors du redimensionnement du conteneur
- Il y a 5 protocoles
  - *FlowLayout* : mise en page « glissante »
  - *BorderLayout* : mise en page avec des marges
  - *GridLayout* : mise en page sur une « grille »
  - *CardLayout* : mise en page avec une pile de cartes
  - *GridBagLayout* : mise en page par quadrillage
  - *GridBagConstraints* : contraintes pour *GridBagLayout*
- À associer au *Container* avec la méthode *setLayout()*

FlowLayout : mise en page « glissante »

monCont.setLayout(new FlowLayout());

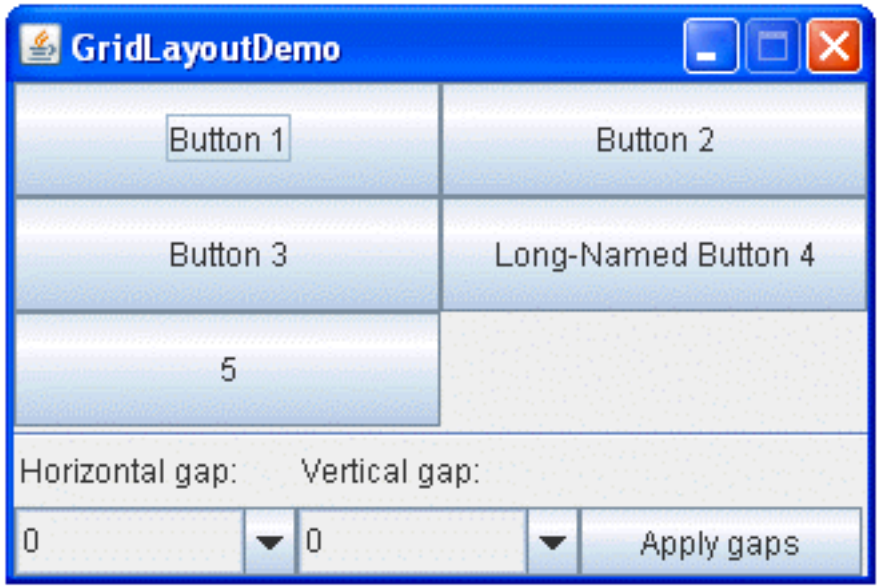
BorderLayout : mise en page avec des marges

monCont.setLayout(new BorderLayout());



GridLayout : mise en page sur une « grille »

monCont.setLayout(new GridLayout(3, 2));



CardLayout : mise en page avec une pile de cartes

monCont.setLayout(new CardLayout());

- Une "carte" est affichée à la fois
- L'utilisateur-ice accède aux cartes à l'aide d'onglets

Gestion des événements (1)

- La gestion des événements en Java se fait en *écoutant* des *actions* : lorsqu'il se passe quelque chose (p.ex. clic sur un bouton), une classe capte cette information et réagit
- La classe qui écoute l'action doit implémenter l'interface *ActionListener* :
  - Une seule méthode à implémenter – `public void actionPerformed(ActionEvent e)`
  - Cette méthode doit contenir toutes les instructions à exécuter pour réaliser l'action
- La méthode `addActionListener(ActionListener l)` permet d'associer une instance de la classe d'écoute à un composant graphique (p.ex. un bouton)

Gestion des événements (2)

- **Attention :** Si une classe implémentant *ActionListener* doit pouvoir interagir sur un composant de l'IUG (p.ex. appeler une méthode de la fenêtre), ce composant doit être passé en paramètre du constructeur de la classe implémentant *ActionListener*

- Exemple (classe *MaFenetre*)

```
GestionLire gestionLire = new GestionLire(this);
this.boutonLire.addActionListener(gestionLire);
```

- Exemple (classe *GestionLire* implémentant *ActionListener*)

```
public GestionLire(MaFenetre fen) {
    this.fenetre = fen;
}
```

Types d'événements

- Il existe de nombreux types d'événements en Java. Nous en verrons deux : *ActionEvent* et *MouseEvent*
- Un événement de type *ActionEvent* est déclenché quand on clique sur un bouton ou qu'on sélectionne un composant d'un menu
  - Interface pour écouter l'action : *ActionListener*
  - Méthode à implémenter : *actionPerformed*
- Un événement de type *MouseEvent* est déclenché quand on effectue un clic ou qu'on déplace la souris
  - Interface pour écouter l'action : *MouseListener*
  - Méthodes à implémenter : *mouseClicked*, *mousePressed*, *mouseReleased*, *mouseEntered*, *mouseExited*
  - L'objet *MouseEvent* contient (entre autres) les coordonnées du clic

IUG – ressources utiles

- Des exemples d'applications avec IUG, à la fin de la section « Séminaire Java » sur le Moodle d'Info II : Interfaces utilisateur graphiques en Java
- Le texte et les exemples du *Chapitre 11 - Construction des interfaces utilisateur* du livre BlueJ, dans le même répertoire Moodle :
  - Livre_BlueJ_chapitre_11.pdf
  - Livre_BlueJ_projets_du_chapitre11.zip