Programmation événementielle
TP 4 – Interface graphique (partie 1)

Dans ce TP vous travaillerez dans votre projet Java avec le package **tp4** pour l'ensemble des questions

Rappels

Tous les objets java que nous allons utiliser proviennent du package : *javax.swing.* Fourni à partir du JDK 1.2, le package *javax.swing* est beaucoup plus riche que le package graphique basique *AWT*, sur lequel il se base. Ces deux packages ont beaucoup d'objets en commun mais *javax.swing* complète avantageusement *java.awt*.

Utilisation

Il faut tout d'abord importer les classes du package *javax.swing.* Ce package contient tous les objets de base pour créer une interface utilisateur conviviale: barre de défilement, boutons, boutons radio, saisie de texte, etc... Les noms des classes Swing sont précédés d'un "J", ce qui les différencie de celles du package AWT :

- La classe *Button* en AWT devient la classe *JButton* pour l'objet Swing
- La classe *TextField* en AWT devient la classe *JTextField* pour l'objet Swing.
- etc...

En revanche on utilise l'AWT pour gérer la plupart des événements et des mises en page (Layout).

Classes principales

Les objets les plus couramment utilisés avec Swing sont les classes :

Classe	Description
JFrame	Fenêtre graphique.
JPanel	Zone graphique, container
JButton	Objet bouton
JTextField	Zone de texte à saisir
JLabel	Texte non modifiable
JComboBox	Choix d'éléments dans une liste avec sélecteur
JScrollBar	Barre de défilement
JTable	Tableau
JList	Choix d'éléments dans une liste sans sélecteur
JCheckBox	Objet pouvant être coché ou décoché
JRadioButton	Choix exclusifs pour les options

La philosophie graphique Java

La base d'une interface graphique développée en Java est le panneau (classe *Panel*). Un panneau est un objet pouvant contenir les composants graphiques à afficher. Un panneau peut contenir d'autres panneaux. Il faut

imaginer ce panneau comme une base de construction sur laquelle on peut poser d'autres éléments pour former une interface graphique. Pour comprendre la logique des éléments graphiques, il faut savoir que tous les éléments appartiennent à la classe *Component*.

Les composants

Tous les objets graphiques Java sont des composants de la classe *Component*. Les objets swing sont des *JComponent*, classe fille de *Component.* Ces objets sont susceptibles d'être associés à des événements (vus plus loin) tels une sélection dans une liste déroulante ou un clic. Ils possèdent des méthodes de type *addItemListener()* qui les rendent sensibles aux événements. Les objets graphiques sont également appelés *widgets*.

Les containers

Un container (classe *Container*) est un composant possédant la faculté de contenir d'autres composants. Par exemple, un bouton (*JButton*) est un container car il peut contenir un titre (*JLabel*).

Les frames ou fenêtre

Une fenêtre (classe *JFrame)* est une fenêtre système autonome. Il est ainsi possible de développer facilement des applications multi-fenêtrées. La frame est un container qui possède un panneau racine dont la référence est donnée par la méthode *getContentPane()*. Un ajout de composants ou de containers dans ce dernier panneau permet de les afficher à l'écran. Attention! une *JFrame* ne peut contenir directement de composants graphiques, il faut utiliser son panneau racine.

Gestionnaires de placement

Chaque container spécifie sa méthode de mise en page. C'est la façon dont sont positionnés les objets les uns par rapport aux autres. Les objets gérant cette mise en page sont des layouts. Un panel peut contenir d'autres panels dont chacun possède un layout diffèrent. Les gestionnaires de placement les plus courants sont les suivants :

FlowLayout

Il permet une mise en place séquentielle des composants. Par défaut ils s'affichent de la gauche vers la droite. Lorsqu'il n'y a plus de place sur une ligne, les objets sont affichés sur la ligne d'en dessous.

GridLayout

Le layout est créé en spécifiant le nombre de lignes et de colonnes et chaque composant graphique est affiché dans une cellule du tableau. Toutes les cellules font la même taille (donc celle du plus grand élément).

BoxLayout

Le *BoxLayout* définit un 'empilage' des objets sur l'axe des X ou des Y. Il ressemble au *FlowLayout* mais il est plus complet car possède des attributs qui permettent un affichage plus facile et affiné.

Par défaut

Si on ne fixe pas de LayoutManager à notre panel, le premier élément posé dans ce dernier prend l'intégralité de l'espace disponible et masque les widgets ultérieurs. Dans ce cas, il est possible de placer les objets au pixel près par la méthode *setBounds(int X,int Y,int width,int height)* de la classe *Component*. Mais dans ce cas, il faut s'assurer que l'on n'utilise pas de layout et faire un *setLayout(null).*

Les événements

Définition des événements et écouteurs

Un événement est un objet de type *Event.* Les événements sont générés lors d'une action de l'utilisateur sur l'interface. Par exemple, le clic sur un bouton peut générer un événement de type *MouseEvent*. Mais ces événements n'ont de conséquences sur le programme que si l'objet possède un écouteur (listener) à l'écoute. Dans l'exemple d'un Canvas, si on clique sur ce dernier, le gestionnaire graphique Java génère un objet *MouseEvent* qui va être intercepté par un écouteur préalablement posé (comme un *MouseListener)*. Alors le

programme exécutera la méthode de traitement associée à l'écouteur, dans notre cas, le programme exécutera la méthode *mouseClicked(MouseEvent me)*. Notez que les objets événement sont toujours générés dans une application graphique mais dans le cas où il n'y a pas d'écouteurs pour les intercepter, il n'y aura pas de traitements associés. Les écouteurs et événements courants font partie du package AWT. Il faut donc penser à faire un import du package *java.awt.event*. Les composants graphiques possèdent des méthodes spécifiques permettant de poser un écouteur. Par exemple, *addActionListener(ActionListener al)* permet de poser un écouteur de type '*action sur le composant*' qui sait intercepter les *ActionEvent* comme le clic sur une liste défilante ou sur un bouton. Il existe des écouteurs pour chaque type d'événement. Les écouteurs sont toujours des interfaces. Ainsi la classe de développement courante devra implémenter toutes les méthodes de l'interface. La plupart du temps, il n'y en a qu'une. C'est cette méthode que le programme exécutera si l'événement se produit. Remarquons que si la classe courante implémente un *ActionListener*, elle devient elle-même un objet de type *ActionListener.* Par conséquent,les méthodes d'ajout d'écouteurs sur des composants graphiques comme *addActionListener(ActionListener al)* qui prennent en argument un *ActionListener* pourront prendre *this* en argument.

Ecouteurs courants

Listeners courants	Méthodes à implémenter
MouseListener	*MouseClicked(MouseEvent e)*
	MouseEntered(MouseEvent e)
	MouseExited(MouseEvent e)
	MousePressed(MouseEvent e)
	MouseReleased(MouseEvent e)
	MouseDragged(MouseEvent e)
ActionListener (exemple: les boutons)	
	ActionPerformed(ActionEvent e)
ItemListener (exemple: les listes comme les ComboBox)	
	ItemStateChanged(ItemEvent e)
FocusListener	
	FocusGained(FocusEvent e)
	FocusLost(FocusEvent e)

Il existe également des écouteurs pour gérer les actions sur une fenêtre (*WindowListener*), gérer la barre de défilement (*AdjustementListener)*, etc.

Exercice 1

a) Création d'une première interface graphique

Créez une interface graphique (une fenêtre) d'une taille de 320 pixels par 200 pixels qui porte le nom "*TEST*". Cette interface doit contenir :

- Un texte "*TEST GUI*" : (Utilisation d'un ***JLabel***)
- Un Bouton avec comme label "*Au revoir*" : (Utilisation d'un ***JButton***)

Ajoutez ces objets sur le panneau principal de la fenêtre, quel résultat voyez-vous ? Où est le problème ?

Indications :

- Créez une classe héritant de la classe ***JFrame*** pour créer la fenêtre et utiliser dans son constructeur les méthodes ***setSize*** pour modifier sa taille et ***setVisible(true)*** pour l'afficher à l'écran (appeler cette méthode en dernier)
- Utilisez la méthode ***add*** pour ajouter un composant sur un panneau (un panel). Il faudra pensez à permettre la fermeture propre de la fenêtre en utilisant
  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

b) Mise en place d'une mise en page

Pour corriger le problème précédent, nous allons mettre en place une mise en page. Choisissez votre mise en page de manière à optimiser le placement : Le titre doit être placé à gauche du bouton (qui doit avoir une taille raisonnable).

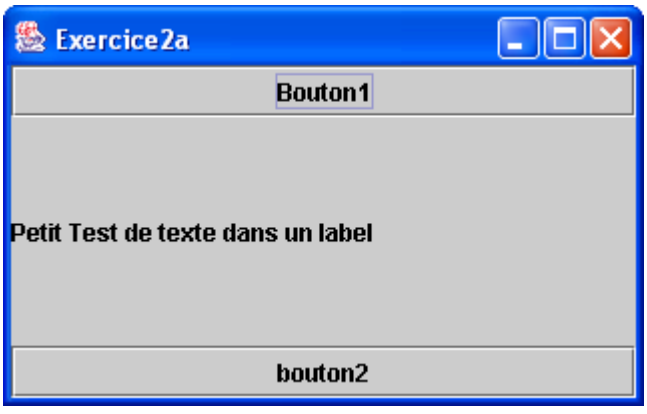
Testez la méthode ***pack*** (de la classe JFrame) en l'appelant juste avant la méthode ***setVisible(true)***. Que fait cette méthode ?

Indications : il faut penser à utiliser un conteneur de type JPanel (qui par défaut met en page les composants de gauche à droite).

Exercice 2 – Utilisation de gestionnaires de contenu

a) Utilisation d'un BorderLayout

Construisez la classe Exercice2a permettant de créer l'interface suivante. Pour le moment les interfaces restent simples, vous pouvez donc tout coder au sein de la même classe

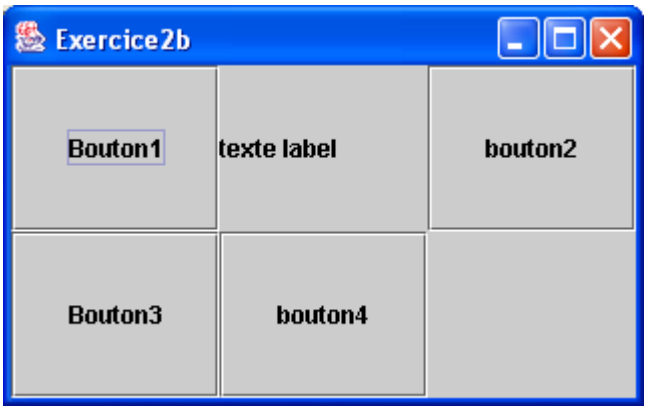


Indications :

- Utilisez la méthode ***setLayout*** pour appliquer le gestionnaire de contenu de votre choix. Exemple : panneau.setLayout(new BorderLayout()) ;
- Utilisez les constantes BorderLayout.PAGE_START, BorderLayout.PAGE_END, BorderLayout.CENTER, BorderLayout.LINE_START, BorderLayout.LINE_END pour contraindre le placement des objets. Exemple : panneau.add(bouton, BorderLayout.LINE_END) ;

b) Utilisation d'un GridLayout

Construisez la classe Exercice2b permettant de créer l'interface suivante :



Pour cela, utilisez la méthode ***setLayout*** pour appliquer le gestionnaire de contenu de votre choix. Exemple : panneau.setLayout(new GridLayout(3,2)) ;

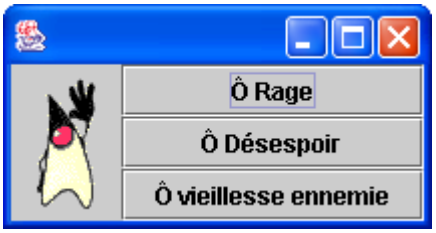
c) Utilisation de gestionnaires de placement multiples

Construire la classe Exercice2c permettant de créer l'interface suivante. :

Pour cela, utilisez plusieurs panneaux imbriqués

d) Utilisation d'une image

Construisez la classe Exercice2d permettant de créer l'interface suivante :



Pour cela, construisez un composant Image : ***new JLabel(new ImageIcon("imgjava.jpg")) ;***

Exercice 3 – Réagir aux actions avec des *listeners* (écouteurs)

Remarque : relire au besoin le cours sur la gestion des événements

Rappels : Les composants graphiques peuvent avertir d'une action effectuée sur eux-mêmes en générant un événement correspondant à cette action. Pour être averti, il faut donc écrire un objet capable d'être averti (un écouteur d'événement). Il faut pour cela d'implémenter l'interface d'écoute correspondant à cet événement.

Un ***JButton***, lorsque on le clique, génère un événement de type ***ActionEvent*** vers les objets implémentant l'interface ***ActionListener*** qui se sont enregistrés auprès du bouton avec la méthode addActionListener.

1) Ecrivez la classe Exercice3 permettant de créer l'interface suivante et affichant « clic1 » dans la console chaque fois que l'on clique sur le bouton. La classe représentant l'objet « écouteur » nécessaire sera représenté sous la forme d'une classe interne imbriquée dans la classe Exercice3.

2) Ajoutez un deuxième objet écouteur permettant d'afficher « clic2 » dans la console chaque fois que l'on clique sur le bouton. La classe représentant l'objet « écouteur » nécessaire sera représenté sous la forme d'une classe anonyme, définie au moment de la création de l'objet.

3) Ajoutez un troisième objet écouteur permettant d'afficher « clic3 » dans la console chaque fois que l'on clique sur le bouton. La classe représentant l'objet « écouteur » nécessaire sera représenté sous la forme d'une classe externe.
4) Modifiez le programme de manière à ce que le label affiche le nombre d'appuis sur le bouton. (on peut modifier le contenu d'un label avec la méthode ***label.setText(" … ")*** ;)