TD PAC - Java n° 7

Conception d'interfaces graphiques avec Java (1)

L'objectif de ce TD est de vous familiariser avec les principaux composants graphiques des librairies SWING, afin de pouvoir concevoir des interfaces graphiques en JAVA. Etant donné la richesse des librairies, il est impossible de tout connaître par coeur... Ce travail est donc aussi l'occasion d'apprendre à se familiariser avec la documentation de java.

I) Un mot sur la documentation de java

Exercice 1 La documentation de java est disponible en ligne sur le site de `Sun Microsystems` à l'url : `http://java.sun.com/j2se/1.4.2/docs/` (penser à consulter la doc adéquate à la version du `jdk` installée sur votre machine). Pour un accès plus rapide, il est aussi possible que la doc soit aussi installée localement sur votre site. Renseignez vous auprès de votre administrateur local.

La page d'accueil de la documentation donne un certain nombre de pointeurs vers des documents utiles mais ce qui nous intéresse le plus ici est le lien appellé `Java 2 Platform API Specification`, qui donne accès à la documentation de l'ensemble des classes existantes du `jdk`. Cette documentation a été générée automatiquement à l'aide de `javadoc`. Cet outil est capable d'exploiter un format spécial de commentaires dans des sources java, pour générer automatiquement une documentation au format `html`, consultable à l'aide de n'importe quel navigateur[1].

Vous noterez que la page principale est découpée en trois zones. Dans la première (en haut à gauche) se trouvent la liste des packages, dans le seconde (en bas à gauche) une liste de classes et interfaces et dans la troisième, la page principale de description d'une classe, interface, package, ...

1. Familiarisez vous avec l'organisation de la doc. Que se passe-t-il si l'on clique sur un nom de package dans la fenêtre des packages ? Que se passe-t-il si l'on clique sur le nom du package dans la fenêtre des classes et interfaces ?
2. Retrouver les pages décrivant les classes JFrame, JPanel et JButton.
3. Pour chacune de ces classes, trouver à quel package elles appartiennent, quels sont leurs constructeurs, de quelles classes elles dérivent (i.e. leur position dans la hierarchie des classes) et les méthodes qui leurs sont applicables (notamment par héritage)

II) Conception d'une interface graphique simple

Nous nous proposons ici de réaliser une interface pour une calculette simple, dont nous enrichissons progressivement l'interface. La calculette elle même n'est pas à réaliser. Nous vous fournissons une classe `Calculette` déjà toute prête. De façon générale, lors de la mise en oeuvre d'une application informatique, il est préférable de ne pas mélanger les fonctionalités principales de l'application, avec les aspects liés à l'interface utilisateur. Il vaut mieux concevoir l'interface séparément et l'associer à l'application centrale. Ceci nécessite naturellement de bien comprendre et spécifier au préalable, comment les deux composantes communiquent.

Dans le cas présent, vous ne disposez pas du source de la classe Calculette mais seulement du fichier `Calculette.class`. Vous pouvez néanmoins vous faire une idée de ce que cette classe met à disposition à l'aide de la commande :

```
javap <NomDeClasse>
```

[1] Vous aussi, lors de l'écriture de vos classes, avez tout intérêt à prendre rapidement l'habitude d'écrire vos commentaires de façon à ce qu'ils soient exploitables par `javadoc`. Vous devrez notamment l'utiliser pour documenter votre travail de TER

Cette commande vous indiquera la liste des constantes et méthodes publiques définies dans cette classe. Utiliser cette commande pour inspecter la classe `Calculette`.

La calculette dont nous disposons ici est supposée disposer de 16 touches, correspondant aux 10 chiffres, aux quatre opérations de base, et aux touches `[=]` et `[C/CE]`. La méthode `input(n)` sert ici à répercuter sur la calculette ce qui se passe lorsque l'on appuie sur l'une des touches. La méthode `getAff()` permet de récupérer le contenu de l'écran de la calculette, qui peut être affiché à l'aide de la méthode `affiche()`.

Exercice 2 La plupart des interfaces graphiques se définissent en dérivant une nouvelle classe à partir de la classe `JFrame`.

1. Définir une classe `IGCalculette`, comme une extension de `JFrame`. Cette classe devra contenir un attribut lui permettant d'interagir avec une calculette. Cependant cette calculette devra également pouvoir transmettre les résultats de ses calculs à l'interface Graphique. Par conséquent, définir parallèlement une classe `CalculetteGraphique` comme une extension de la classe `Calculette` et comportant un attribut de la classe `IGCalculette`.
2. Il est généralement préférable de regrouper préalablement les différents composants de l'interface dans différents `JPanel` que l'on intègre au Container de la fenêtre principale (JFrame). (voir la méthode de la classe JFrame *Container getContentPane()* pour récupérer le Container d'une JFrame). Définir deux JPanels. Le premier servira pour le clavier de la calculette. Le second servira pour l'écran et éventuellement des composants permettant de configurer l'affichage sur l'ecran. Afin de pouvoir visualiser ces deux JPanels, leur donner des couleurs de fond différentes..
3. On pourra définir l'écran comme un `JTextField`, que l'on ajoutera dans le JPanel de l'écran. Cependant (au moins dans un premier temps) il ne doit pas être directement éditable.

   Dans la mesure où une `CalculetteGraphique` doit pouvoir actualiser la valeur de cet écran, on le définira comme un attribut de l'interface graphique. Définir dans la classe `IGCalculette` une methode publique d'accès permettant de modifier la valeur du texte de cet attribut puis, dans la classe `CalculetteGraphique`, surcharger la méthode `affiche()` afin qu'elle modifie dans l'interface graphique associée la valeur de l'écran.
   Tester que cela fonctionne bien en réalisant une opération simple dans une méthode `main(...)` de la classe `CalculetteGraphique`.

Exercice 3 Pour réaliser le clavier de la calculette, nous allons utiliser des composants de type JButton. Ces composants étant utilisés pour le contrôle de l'interface, il est nécessaire de les déclarer comme attributs de la classe `IGCalculette`.

1. Créer un nouvel attribut pour l'interface graphique, correspondant à un tableau de boutons pour le clavier (nb : on pourra de façon utile créer des constantes de classe pour référencer les boutons qui ne correspondent pas à des chiffres par des entiers).
2. Lors de l'initialisation de l'interface, créer les 16 boutons , chacun d'entre eux étant étiqueté par le symbole approprié, et disposer le tout de façon adéquate sur le Panel correspondant au clavier.
   Quel est le Layout manager qui vous semble le plus approprié ?

Exercice 4 Nous souhaitons illustrer l'utilisation d'autres composants SWING en rajoutant quelques outils de contrôle de l'affichage de l'écran.

1. Créer un nouveau JPanel pour recevoir ces outils de configuration et le disposer au dessus de la zone d'affichage dans le pannel correspondant à l'écran.
2. Ajouter successivement sur ce panel
   (a) un `JLabel` étiqueté par "Fonte"
   (b) un `JComboBox` auquel on associera des noms de fontes
   (c) deux `JCheckBox` auquelles on associera les chaînes "gras" et "italique"

   Ces outils seront utilisés pour contrôler le choix de la fonte utilisée pour la zone d'affichage.

III) Conception d'une l'interface réactive

Exercice 5 Pour l'instant notre interface graphique est quasiment *passive*. Nous avons pu constater que la calculette graphique était capable d'agir sur l'écran de l'interface, mais l'interface n'est pas capable d'agir sur la calculette. Pour cela, il est nécessaire d'associer à chacun des composants graphiques un gestionnaire d'événements.

1. Les JButtons sont généralement associés à des gestionnaires de type `ActionListener` pour réagir aux clics de souris. Ici chaque bouton du clavier a un effet différent. On peut imaginer créer un gestionnaire différent pour chaque bouton... mais avec 16 boutons, cela risque d'être un peu lourd !. Une autre faç on de faire est de ne créer qu'un seul gestionnaire, que l'on associe à chaque bouton, mais dont la méthode `ActionPerformed` qu'il implémente, est capable d'adapter son action en fonction du bouton qui a été cliqué. Pour connaître le bouton à l'origine de l'événement, on peut utiliser la méthode `getSource` de la classe `EventObject`.

   Implémenter une *classe interne* pour un tel gestionnaire d'événements et associer ce gestionnaire à l'ensemble des boutons du clavier.

   Tester maintenant la réactivité de votre interface graphique aux événements concernant le clavier de votre calculette graphique.
2. De manière similaire, associer un gestionnaire d'événements aux deux JCheckBoxes permettant de contrôler le style de l'affichage de l'écran (ici, c'est l'interface `ItemListener` qui doit être implémentée).
3. De manière similaire, associer un gestionnaire d'événements à la JComboBox permettant de contrôler la police de l'affichage de l'affichage. Dans la mesure un tel gestionnaire ne concerne qu'un seul composant, on pourra le définir directement comme une classe anonyme apparaissant comme paramètre de la méthode qui ajoute le gestionnaire approprié à ce type de composant.
4. Vous avez surement remarqué que la fermeture de la fenêtre correspondant à votre interface graphique de calculette ne provoque pas pour autant la terminaison du processus java. Pour cela il faut aussi associer un gestionnaire d'événements, à votre objet qui étend la classe JFrame. Associer un tel gestionnaire permettant de terminer le processus lorsque se produit la fermeture de la fenêtre (On aura intérêt à utiliser un `WindowAdapter`).