Université Paris Diderot **Programmation Orientée Objet**
L2/L3 **Année 2014-2015**

TP n°2

Interfaces graphiques

N'oubliez pas de vous inscrire sur DidEL ! Les groupes sont créés, pensez par conséquent à vous iscrire dans **votre** groupe. Si vous n'avez pas le même groupe de TD et TP, inscrivez vous dans les deux.

Le TP a pour vocation de vous faire découvrir les interfaces graphiques à l'aide de l'Environnement de Développement Intégré (IDE) Netbeans, installé sur les machines de l'université.

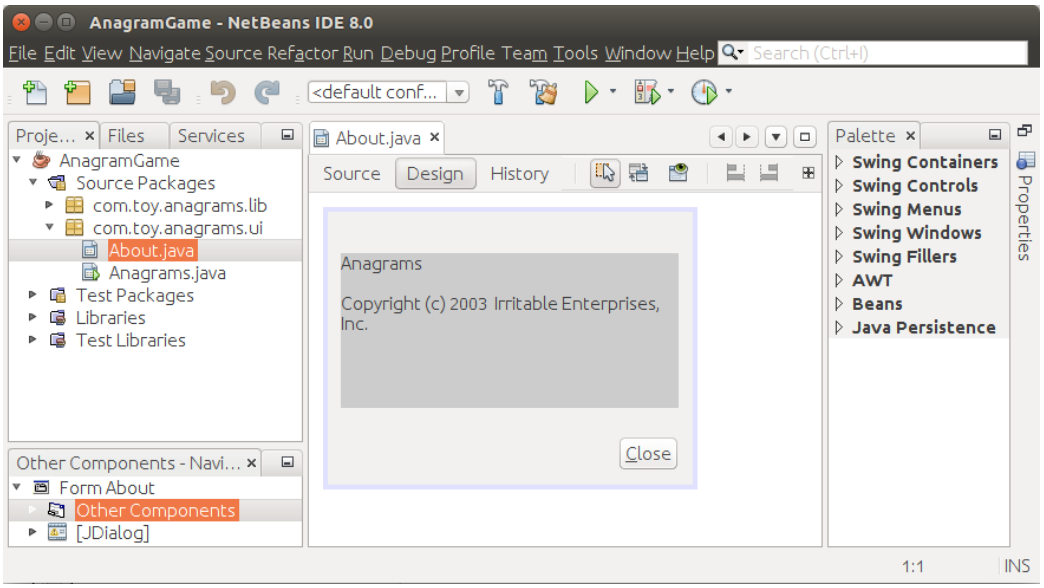
Pour travailler sur votre propre ordinateur

Exercice 1 [Installer Netbeans]
Conseil : Travaillez aujourd'hui sur les machines de l'université pour éviter de perdre du temps à télécharger puis installer le logiciel, mais installez le une fois rentrés chez vous pour pouvoir travailler à la maison !
Netbeans est téléchargeable sur `https://netbeans.org/downloads/`. Choisissez votre langue préférée (Français ou Anglais) et votre système d'exploitation (que le site devrait normalement détecter automatiquement) puis téléchargez le logiciel ; le logiciel installé sur les machines est en Anglais. Dans le doute, prenez la version complète, elle pourra toujours vous être utile pour vos propres projets et pour d'autres cours.
Ensuite, suivez pas à pas l'outil d'installation, qui est très bien conçu. Voilà, vous allez maintenant pouvoir travailler !

Un premier exemple

Exercice 2 [Créer un projet préprogrammé]
Nous allons découvrir un exemple de programme avec interface graphique, déjà présent dans Netbeans. Pour ce faire, créez un nouveau projet (cliquez **File** puis **New project**, ou encore avec le raccourci **Ctrl+Maj+N**). Ensuite, sélectionnez **Samples**, puis **Java** et enfin **Anagram Game**, puis acceptez tout ce que vous proposera l'IDE.
Ça y est ! Vous venez de créer un nouveau projet Java, dont le nom est surement **AnagramGame**, et qui contient une interface graphique ! Faites le fonctionner en exéctuant le programme **Anagrams.java** du paquet **com.toy.anagrams.ui**.

Exercice 3 [**Modifier une interface graphique en un clic**]
Les deux classes du paquet **com.toy.anagrams.ui** sont des **interfaces graphiques**, comme en témoigne leur icône inhabituelle, dans l'arborescence située à gauche de l'écran. Quand on souhaite modifier ces classes, Netbeans propose, en haut de la fenêtre de droite, plusieurs options, dont **Source et Design**. On va pour l'instant utiliser l'option **Design**, obtenant ainsi quelque chose qui ressemble à ceci :



Éditer les zones de texte (avec un clic droit, en appuyant sur **Edit Text**) pour tout traduire de l'Anglais vers le Français, et exécuter le programme ainsi obtenu.

Exercice 4 [**Modifier le code sous-jacent**]
La liste des mots "corrects" et celle de leurs anagrammes sont enregistrées directement dans la classe **StaticWordLibrary.java** du paquet **com.toy.anagrams.lib**. Modifier ces listes de manière à obtenir des mots français et leurs anagrammes, puis exécuter le programme ainsi obtenu.

Exercice 5 [**Générer des anagrammes aléatoirement**]
On va modifier la méthode **public String getScrambledWord(int idx)** : cette méthode doit désormais retourner un anagramme du mot **WORD_LIST[idx]** choisi au hasard comme dans l'exemple suivant.
Si le mot original est **objet**, alors on a une chance sur quatre de renvoyer chacun des mots **bjeto**, **jetob**, **etobj** et **tobje**. L'idée est de couper le mot en un préfixe et un suffixe non vides et de les échanger ; on n'a dont pas le droit de renoyer le mot **objet** lui-même.
Exécuter ensuite le programme obtenu : vous semble-t-il plus facile de trouver des anagrammes avec cette nouvelle version ?

Créer une calculatrice

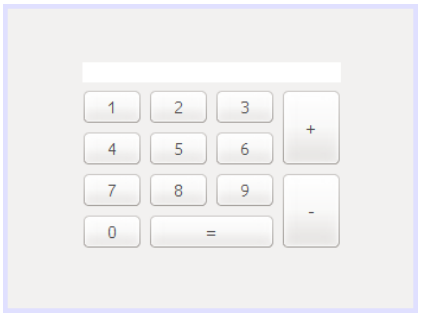
Exercice 6 [Créer sa propre interface graphique]
On va maintenant tâcher de créer notre propre interface graphique. Pour ce faire, ouvrir un nouveau projet, en choisissant **Java** puis **Java Application**, et nommez votre projet **Calculatrice**.
Normalement, vous venez de créer un nouveau projet, ainsi qu'un paquet nommé **calculatrice** et une classe Java elle aussi nommée **Calculatrice**. On ne va pas toucher à cette classe pour l'instant, mais en créer une nouvelle.
Pour ce faire, créez un nouveau projet (cliquez **File** puis **New file**, ou encore avec le raccourci **Ctrl+N**). Ensuite, sélectionnez **Swing GUI Forms**, puis **JFrameForm**, puis acceptez tout ce que vous proposera l'IDE.
On dispose maintenant d'une interface graphique vide de contenu, qui ne présente qu'un "panneau" gris, et qu'il va falloir rendre un peu plus intéressante.

Exercice 7 [Ajouter une zone de texte]
À droite de l'écran se trouve une **Palette**. Dans **Swing Controls**, sélectionnez un **Label** (**étiquette**, en Français), et positionnez-le sur le panneau gris : vous pouvez alors choisir livrement les dimensions et le positionnement de votre étiquette. C'est cette étiquette qui aura vocation a servir d'écran sur lequel s'afficheront les nombres que vous souhaitez calculer.
À l'aide d'un clic droit, et comme dans l'exercice 3, remplacez le texte affiché par cette étiquette par une chaîne vide. Puis, toujours avec un clic droit, aller dans **Properties**, et changez deux paramètres :

- dans **Background**, en première ligne, remplacez la couleur actuelle par du blanc, dont le code est [255,255,255] ;
- plus bas, cochez la case **Opaque**.

On a maintenant une étiquette opaque peinte en blanc, sur un panneau dont le fond est gris clair.

Exercice 8 [Ajouter des boutons]
Dans la rubrique **Swing Controls** de la **Palette** située à droite, on peut également ajouter un **Button** (**bouton**, en Français). De même que dans l'exercice précédent, rajoutez des boutons, dont vous choisirez la taille et modifierez le texte, de sorte d'obtenir une calculatrice qui ressemble à ceci :



Exercice 9 [Rendre les boutons actifs]
En double-cliquant sur un bouton (par exemple le bouton "1"), **NetBeans** nous amène directement dans le code Java que nous étions en train d'écrire sans nous en rendre compte. Plus précisément, il place le curseur au milieu du code d'une fonction, qui ressemble à :

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
  }
```

À quoi correspond cette fonction ? Et bien, quand on crée un programme Java usuel, il y a souvent une fonction

```
public static void main(String[] args) {
    // TODO add your handling code here:
  }
```

au sein de laquelle on va écrire le code que l'on veut voir exécuter lorsque l'on fait fonctionner notre programme Java.
Ici, c'est presque la même chose : la fonction **jButton1ActionPerformed** est la fonction que l'on va exécuter en actionnant notre bouton (par exemple en cliquant dessus). Il faut donc insérer le code que l'on veut exécuter.
À vous d'inventer les codes que vous pouvez associer à chaque bouton pour que votre calculatrice fonctionne comme une calculatrice "normale" : il faudra notamment décider ce que vous souhaitez afficher sur l'écran de la calculatrice, mais également ne pas oublier qu'il faut bien que votre calculatrice fasse effectivement des calculs : préférerez-vous que les calculs aient lieu à chaque fois que l'on appuie sur un bouton, ou bien uniquement au moment où on appuie sur le bouton "=" ?
Attention : cette question est longue et difficile. En particulier, on ne saurait trop vous conseiller de décider d'abord ce que vous attendez précisément de chaque bouton avant de commencer à coder en Java.

Exercice 10 [Pour aller plus loin...]
Si vous avez le temps, et si tout fonctionne comme prévu, n'hésitez pas à rajouter des boutons à votre calculatrice : multiplication, division, virgule, parenthèses... Tous les ajouts seront les bienvenus !