C. Recanati
Interface Graphique
INFO2, 2013

TP n° 1

1. Pour vérifier que votre environnement java est cohérent, tapez :
```
% which java
```
puis
```
% which javac
```
Vérifier ainsi que la commande java coïncide bien avec celle du compilateur (commande javac).
Pour connaître la version utilisée, tapez :
```
% java -version
```
Rem : Il est possible que certains d'entre vous n'accèdent qu'aux commandes de /usr/bin ou /usr/local/bin qui ne fournissent pas une version récente.

Pour utiliser les versions récentes (et eclipse), il faut modifier votre fichier de configuration `.bashrc` en décochant le commentaire # adéquat qui précède une version configurée de java. Normalement cela suffit. Pour aller plus vite, vous pouvez aussi copier le fichier .bashrc d'un de vos camarades qui aurait une configuration correcte.

2. Pour accéder à la documentation en local, partez du répertoire où se trouve installée votre commande xxx/bin/java, et lancez un navigateur sur le fichier de doc suivant (en indiquant `file` au lieu de `http`) :
`file : /usr/`*`... repertoire xxx de java`*`/`**`docs/api/index.html`**
Mémoriser ensuite le signet pour retrouver cette adresse ultérieurement.

3. Rappel : on compile et on exécute un fichier java avec les commandes
```
%  javac  fichier.java        →  génère un fichier.class
%  java  fichier [arg1] ... [argN]
```

Exercice 1
1) Ecrire une sous-classe de la classe `JFrame` comportant un `main` qui crée une instance de la classe et l'affiche au centre de l'écran. Pour le calcul de la position et des dimensions du cadre, on utilisera la méthode `getScreenSize()` de la classe `Toolkit`. (Un objet de type `Toolkit` peut être récupéré par un appel à `getToolkit`() sur un cadre `JFrame`).
2) Créer une classe Panneau (sous-classe de `JPanel`) et redéfinir la méthode `paintComponent(Graphics g)` afin de dessiner à l'intérieur d'un panneau: a- une ligne de couleur rouge ; b- une chaîne de caractère utilisant une grosse fonte (taille 36) ; et c- un rectangle de couleur bleu. Placer ensuite un objet Panneau au centre du cadre précédent (on l'ajoute au panneau de contenu du cadre avec la méthode `add`) pour afficher les dessins demandés.

Exercice 2 *le but de cet exercice est de créer une application qui mémorise un ensemble de points tirés aléatoirement et qui les affiche dans un cadre selon différents modes (sous forme de petits ronds, de petites croix, ou de polygone).*

Un point sera une instance de la classe java.awt.Point. L'ensemble de points sera mémorisé dans la classe principale par un vecteur `ensPts`. Un champ `mode` (de type entier) de la classe principale indiquera le mode d'affichage des points. Il y aura trois modes d'affichage différents, définis par des constantes entières nommées :

- `ROND` : les points seront dessinés sous forme de petits ovales;
- `CROIX` : les points seront dessinés sous forme de croix ;
- `POLY` : le polygone joignant les points sera dessiné ;

1) Ecrire une interface définissant les constantes des modes d'affichage dans un fichier séparé. Se servir de cette interface dans l'application pour initialiser le champ `mode` d'une instance de l'application (à `ROND` par exemple).
2) Ecrire une méthode `initAleatoire` ayant trois paramètres: `nb`, le nombre de points que l'on souhaite stocker, et la `largeur` et la `hauteur` du cadre qui contiendra les points. `initAleatoire` crée `nb` Points dont les coordonnées se situent à l'intérieur du cadre dont la largeur et la hauteur sont spécifiées, puis les stocke dans le vecteur `ensPts` de la classe principale.
☞ Pour le tirage aléatoire des coordonnées des points, on utilisera dans la classe `Math` la méthode `random()` qui retourne un `double` compris entre 0.0 et 1.0. ainsi que la méthode `round(double a)` qui retourne l'entier `long` le plus proche de `a`.
3) Ecrire une sous-classe `Dessin` de `JFrame` qui possède les deux constructeurs suivants :

a) un constructeur avec deux arguments permettant de définir le titre de la fenêtre et le mode d'affichage des points. Vous supposerez d'abord que l'entier passé en paramètre est une valeur de mode correcte.
b) un constructeur à un argument permettant de définir le titre de la fenêtre et fixant le mode graphique à `ROND`.

4) Créez un `JPanel` pour dessiner (il sera ajouté au panneau `contentPane` du `JFrame`), et affichez les points dans ce `JPanel` en redéfinissant la méthode `paintComponent` de ce `JPanel`. Testez dans le `main` l'affichage des points en mode `ROND`.
5) Testez ensuite l'affichage dans les autres modes en définissant une boucle créant plusieurs cadres, et en faisant varier l'origine de ces cadres et leur mode d'affichage.
6) Traitez le cas où le mode passé en argument au constructeur de la fenêtre n'est pas l'un des 5 modes préconisés. (Vous gérerez cette erreur par le traitement d'une exception).

Exercice 3 *(s'il vous reste du temps)*
Pour améliorer l'affichage de messages d'erreurs ou de tests, écrire une méthode `getClassName (Object o)` qui retourne la chaîne du nom de la classe de l'objet, sans le chemin complet du *package*. Tester votre méthode avec des objets java disponibles dans les packages standards.

Indications 1. Récupérer le nom complet de la classe d'un objet avec `getClass` et `getName` puis rechercher dans la chaîne obtenue, l'index qui correspond à la dernière apparition du caractère point (cf. `lastIndexOf`). En extraire avec `substring` la sous-chaîne recherchée (elle commence à l'index trouvé + 1).
2. Pour pouvoir invoquer votre méthode sur un objet il faudra instancier votre classe principale dans le main. Essayer votre méthode avec par exemple `System.out` ou `System.in` qui sont des objets java.