

C. Recanati
Java Graphique
AIR2

Feuille de TP n° 1

Rappel: on compile et exécute un fichier java à l'aide des commandes javac et java:

```
% javac fichier.java          → génère un fichier.class
% java fichier [arg1 ... argN] → lance l'exécution
```

Pour vérifier que vos deux commandes sont cohérentes, tapez :

```
% which java
puis
% which javac
```

Vérifier ainsi que les deux commandes proviennent bien du même répertoire de binaires. [Si vous avez un problème pour trouver la commande which, c'est que votre variable PATH n'est pas correctement configurée]. Si vos commandes java ne sont pas cohérentes, il faut supprimer dans votre fichier de configuration .bashrc le caractère de commentaire en début de ligne pour garantir l'accès à une configuration cohérente de java. (Une autre solution serait de recopier le fichier .bashrc d'un de vos camarades qui aurait une configuration correcte).

Relire ensuite ce fichier dans votre shell de commandes en tapant

```
% source ~/.bashrc
```

Pour avoir la javadoc en local mettre dans votre navigateur un marque page sur l'adresse <file:///usr/share/doc/default-jdk-doc/index.html>

Exercice 1

1) Ecrire une sous-classe de la classe `JFrame` comportant un `main` qui crée un cadre instance de sa 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 objet `JFrame`.

2) Créer une classe `Panneau` (sous-classe de `JPanel`) et redéfinir sa méthode `paintComponent(Graphics g)` afin de dessiner à l'intérieur du 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 sous forme 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'affichages 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 petites croix ;
- POLY : le polygone joignant les points sera dessiné .

1) Ecrire une interface définissant les constantes pour les 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, `largeur` et `hauteur`, les dimensions du cadre qui affichera les points. La méthode `initAleatoire` créera `nb` Points dont les coordonnées toujours positives se situeront à l'intérieur du cadre, et elle les stockera 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. et la méthode `round(double a)` qui retourne l'entier long le plus proche de `a`.

3) Ecrire une sous-classe `Dessin` de `JFrame` possédant les deux constructeurs suivants :

- a) un constructeur à 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 seul argument permettant de définir le titre de la fenêtre et fixant le mode d'affichage à ROND.

4) Créez un `JPanel` pour dessiner (il sera ajouté au panneau de contenu `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 à l'aide 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, 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 (utiliser `lastIndexOf`). 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`. Tester votre méthode avec par exemple `System.out` ou `System.in` qui sont des objets java.