Travaux Dirigés de Java n°1

— Etienne.Duris@univ-mlv.fr —

Rappels importants :

- en Java, tout code est dans une classe ;
- le code source de la classe `Toto` doit être placé dans un fichier de nom `Toto.java` ;
- cette classe est compilée par la commande `javac Toto.java`, ce qui génère un fichier `Toto.class` contenant le *bytecode* de la classe `Toto` ;
- la commande `java Toto` a pour effet de lancer l'exécution du code spécifié dans la méthode *statique* `main()` de la classe `Toto` ;
- pour être trouvée lors de l'exécution, cette méthode de "test" doit être définie par :

```
public static void main(String[] args) { ... }
```

▸ **Exercice 1. (Hello Groland !)** *Écrire un petit programme Java qui affiche "`Hello Groland!`".*

▸ **Exercice 2. (Arguments de la ligne de commande)**

Écrire un programme Java qui affiche à l'écran l'argument passé sur la ligne de commande, par exemple :

```
% java Hello2 toto
% toto
```

Que se passe-t-il lorsque l'exécution est appelée sans argument ? Comment faire ?

▸ **Exercice 3. (Convertir une chaîne (`String`) en entier)**

Écrire un programme Java qui, dans la méthode `main()`, déclare et initialise un tableau `tab` de 15 entiers et affiche à l'écran la valeur de ce tableau à l'indice correspondant à un argument passé sur la ligne de commande. Par exemple :

```
% java Hello3 12
% Valeur de tab[12] :  534
```

Quels sont les moyens disponibles pour contrôler les accès au tableau (bornes).

▸ **Exercice 4. (Une classe et ses membres)**

Écrire une classe `Personne` contenant deux champs (`nom` et `age`) et des méthodes d'accès à ces champs (`getNom()`, `getAge()`, `setNom()` et `setAge()`).

Quelle est la bonne approche (orientée objet) permettant d'afficher un objet (une instance) de cette classe ?

Dans la méthode de test `main()`*, créer un nouvel objet* `Personne` *et l'afficher. Que dire des valeurs des champs de l'objet ? Comment est créé cet objet ?*

▸ Exercice 5. (Les constructeurs)

Écrire un constructeur `Personne(String nom, int age)` *qui initialise les deux champs de l'objet qu'il crée. Qu'est devenu le constructeur (par défaut) sans paramètre, utilisé dans l'exercice précédent ? Définir un deuxième constructeur sans paramètre (surcharge)* `Personne()` *qui initialise l'age à* `0` *et le nom à* `XXXX`*. Pour cela, définir une constante dans la classe* `Personne`*.*

Dans la méthode `main()`*, déclarer un tableau* `tab` *de trois personnes et initialiser les éléments* `tab[0]`*,* `tab[1]` *et* `tab[2]` *avec les références* `p1`*,* `p2` *et* `p1` *où* `p1` *est la référence à un objet construit avec un nom et un age de votre choix et* `p2` *une référence à un objet construit avec le constructeur sans paramètre. Afficher l'ensemble des personnes du tableau.*

▸ Exercice 6. (Cloner un objet)

Sur l'exemple de l'exercice précédent, que ce passe-t-il si on utilise la méthode `setAge()` *sur* `tab[2]` *pour donner un nouvel age à cette personne ?*

Définir une méthode `Object clone()` *dans la classe* `Personne()` *qui duplique les champs de l'objet courant et retourne un nouvel objet* `Personne` *"identique" à l'objet courant.*

Quelle est la différence entre cette méthode et celle de même nom définie dans la classe `Object` *? Comment utiliser cette dernière ?*

▸ Exercice 7. (La délégation, ou composition)

Écrire une classe `Quintet` *représentant un groupe de cinq personnes. L'implantation de la structure stockant les personnes (disons un tableau dans notre exemple) doit être transparente pour l'utilisateur de la classe.*

Cette classe doit disposer de deux constructeurs : un constructeur sans paramètre dont les cinq personnes sont "inconnues" et un constructeur qui accepte en paramètre cinq références de type `Personne`*. Comment faire en sorte que le code commun aux deux constructeurs n'apparaisse qu'une fois ?*

Définir une méthode `toString()` *pour la classe* `Quintet`*. La tester dans la méthode* `main()`*.*

On veut obtenir le nom de la personne la plus jeune du quintet et la moyenne d'âge du quintet. Écrire les méthodes nécessaires et les tester.

▸ Exercice 8. (L'utilisation des API : les vecteurs)

Écrire une classe `Groupe` *représentant un groupe de personnes dont le nombre n'est pas fixé a priori. Pour cela, utiliser l'API des vecteurs (*`java.util.Vector`*).*

Cette classe `Groupe` *doit disposer d'un seul constructeur créant un groupe vide. L'ajout d'une personne peut se faire par la méthode* `ajoute(Personne p)`*. Définir les même fonctionnalités dans cette classe que dans la classe* `Quintet`*.*