

Exercices corrigés de programmation OO Java

Préparés par :

Mlle Imene Sghaier

Année Académique : 2006-2007

Premiers Pas

I. Avant de programmer en Java

Le JDK de Sun (Java Development Kit) est l'outil essentiel pour programmer en Java. Il permet la compilation, le débogage et l'exécution d'applications et d'applets Java. Il comprend également des fonctions avancées comme les signatures numériques, la création d'archives et de documentation ou l'intégration de code natif C. La documentation de JDK est en fait l'API (Application Programming Interface) de Java (le détail de toutes les fonctions du langage). Après avoir installé le JDK, il faut définir un chemin de recherche des exécutable « *PATH* ».

Pour le faire, Lancer une session **DOS** et taper :

Set Path=C:\JDK\bin ou bien directement **Path=C:\JDK\bin**

(Donne la valeur **C:\JDK\bin** à la variable d'environnement path) Ou encore

Set Path=%Path%;C:\JDK\bin

(Ajoute le chemin **C:\JDK\bin** à la valeur de la variable d'environnement path)

Ceci ne fonctionne que si **C:\JDK** est le répertoire d'installation du JDK. Si ce n'est pas le cas, remplacer-le par votre vrai répertoire.

II. La compilation d'un code source

Pour compiler un fichier source il suffit d'invoquer la commande **javac** avec le nom du fichier source avec son extension .java :

javac NomFichier.java

Le nom du fichier doit correspondre au nom de la classe principale en respectant la classe même si le système d'exploitation n'y est pas sensible.

```
Public class HelloWorld {  
//Code}}
```

Le nom du fichier sera **HelloWorld.java**

Une fois le fichier enregistré, ouvrir une fenêtre DOS et aller au répertoire où le fichier est enregistré (**Rappel** : utiliser **cd nomrépertoire** pour monter d'un niveau et **cd..** pour descendre).

Pour le compiler taper :

Javac HelloWorld.java

Suite à la compilation, le pseudo-code Java est enregistré sous le nom HelloWorld.class, ce fichier est compilé et sera interprété par la machine virtuelle.

III. L'exécution d'un programme java

Une classe ne peut être exécutée que si elle contient une méthode main() correctement définie. Pour exécuter un fichier contenant du bytecode, toujours dans le même répertoire (dans lequel est créé le fichier .class), il suffit d'invoquer la commande java avec le nom du fichier source avec ou sans son extension .class

java NomFichier et dans notre exemple **java HelloWorld**

Exercice 1 :

On se propose de faire fonctionner un programme Java dont le rôle est d'afficher « Hello World ! »

Exercice 2 :

On se propose de faire fonctionner un programme Java dont le rôle est d'afficher le premier mot qu'on lui passe comme paramètre d'exécution.

Exercice 3:

Créer un fichier HelloWorldWithMethod.java contenant la classe HelloWorldWithMethod

2- Ajouter une méthode void Hello() qui fait l'affichage d'une chaîne de caractères exemple :

« You're Welcome ! »

3- Ajouter la méthode main dans laquelle vous faites instancier la classe HelloWorldWithMethod

et référence l'instance ainsi créée à une variable s qu'on déclare comme suit :

```
HelloWorldWithMethod s=new HelloWorldWithMethod() ;
```

4- Dans la méthode main faites un appel à la méthode Hello() de l'objet s comme suit :

```
s.Hello() ;
```

Exercice 4:

1-Créer un fichier HelloWorldWithAttribut.java contenant la classe HelloWorldWithAttribut

2- Ajouter un attribut String maChaine à la classe

3- Ajouter une méthode void Hello() qui fait l'affichage de la chaîne de caractères attribut de la classe comme suit System.out.println(this.maChaine);

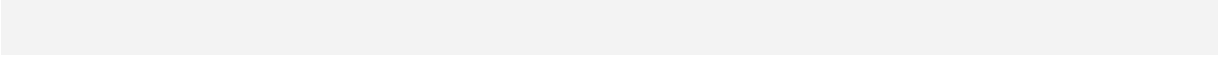
4- Ajouter la méthode main dans laquelle vous faites instancier la classe

HelloWorldWithMethodAndArg et référence l'instance ainsi créée à une variable s qu'on déclare comme suit :

HelloWorldWithMethodAndArg s=new HelloWorldWithMethodAndArg () ;

5- Dans la méthode main faite affecter l'attribut de l'objet s comme suit s.maChaine="Salut "

6- Dans la méthode main faite un appel à la méthode Hello() de l'objet s comme suit
s.Hello() ;



Généralités

Exercice 1

Ecrire un programme qui calcule la somme des 100 premiers entiers et indique à l'écran le résultat.

Exercice 2

Il s'agit de modéliser un segment de droite dont les valeurs des deux extrémités sont entières. Si on échange les deux extrémités, on considère qu'il s'agit encore du même segment. Les opérations que l'on souhaite faire sur ce segment sont :

- calculer sa longueur
- savoir si un entier donné se trouve sur le segment (c'est-à-dire s'il est compris entre la plus petite et la plus valeurs des extrémités du segment).

Ecrire un programme qui contient :

- une classe `Segment` comportant
 - deux attributs de type `int`, `extr1` et `extr2`, représentant les coordonnées (entières) des extrémités d'un segment sur un axe
 - un constructeur de ce segment recevant en argument les deux valeurs entières des extrémités du segment que l'on veut construire
 - une méthode nommée `ordonne` échangeant éventuellement les valeurs des extrémités du segment de telle sorte que la valeur de `extr1` soit au plus égale à la valeur de `extr2`.
 - une méthode retournant la longueur du segment
 - une méthode dont le prototype est :

`boolean appartient(int x);` indiquant si le point de coordonnée `x` appartient ou non au segment

- une méthode redéfinissant la méthode :

```
public String toString()
```

Celle-ci décrira une instance de `Segment` sous la forme d'une chaîne de caractères (par exemple, le segment d'extrémités -35 et 44 pourra être transformé en la chaîne de caractères : "segment [-35, 44]") (la plus petite extrémité est toujours indiquée à gauche). La méthode "retournera" (`return...`) cette chaîne.

- une classe `EssaiSegment` testant la classe `Segment` et comportant une méthode `main` à laquelle on devra fournir trois paramètres entiers par la ligne de commande : origine et extrémité d'un segment et coordonnée d'un point dont on voudra savoir s'il appartient ou non au segment. On utilisera nécessairement la méthode `toString` lorsqu'on voudra écrire le segment sur la sortie standard (l'écran).

Exercice 3

Il s'agit d'écrire un programme qui calcule la factorielle des n premiers entiers et indique à l'écran le résultat. Le nombre n doit être lu sur la ligne de commande.

Exercice 4

Il s'agit d'écrire un programme qui, étant donnée une chaîne de caractères (une instance de la classe `String`)

- calcule la chaîne inverse
- indique s'il s'agit ou non d'un palindrome

Exercice 5

Il s'agit de modéliser un vecteur de Z^2 dont l'origine est en $(0, 0)$ (un tel vecteur est donc caractérisé par deux nombres entiers relatifs). Les opérations que l'on souhaite faire sur ce segment sont :

- calculer sa longueur, par une méthode d'instance sans paramètre, nommée `longueur`, et qui retourne cette longueur sous forme d'un `double`
- savoir si le vecteur concerné est ou non plus petit qu'un autre un autre vecteur donné ; on écrira pour cela une méthode d'instance nommée `plusPetitQue` qui recevra en paramètre l'autre vecteur et qui retournera une variable de type `boolean`
- additionner au vecteur concerné un autre vecteur ; on écrira pour cela une méthode d'instance nommée `addition` qui recevra en paramètre l'autre vecteur et qui ne retournera rien

- additionner deux vecteurs donnés ; on écrira pour cela une méthode **statique** nommée aussi `addition` (en utilisant ainsi la possibilité de la surcharge) qui recevra en paramètres les deux vecteurs à additionner et qui retournera le résultat sous forme d'un objet de type `Vecteur`
- une méthode redéfinissant la méthode :

```
public String toString()
```

Celle-ci décrira une instance de `Vecteur` sous la forme d'une chaîne de caractères (par exemple, le vecteur de composantes 1 et 2 pourra être décrit par la chaîne de caractères : `"vecteur (1, 2)"`). La méthode retournera cette chaîne.

Exercice 6

Il s'agit de définir une classe modélisant une pile d'entiers `+`. Nécessairement, tous les attributs de cette classe auront le modificateur `private`, ce qui signifie qu'on ne peut pas les utiliser directement de l'extérieur de la classe (on dit encore qu'ils ne sont visibles que de leur propre classe).

Cette classe possèdera les trois méthodes suivantes (il faudra reprendre exactement les entêtes indiquées) :

- `void empiler(int n)`

Cette méthode empile la valeur `n` reçue en paramètre.

- `int depiler() throws ExceptionPileVide`

Si la pile est vide, cette méthode lance une exception, ce que vous ferez par l'instruction :

```
throw new ExceptionPileVide();
```

sinon, elle dépile un entier dont elle retourne la valeur,

```
boolean estVide()
```

Cette méthode retourne `true` si la pile est vide et `false` dans le cas contraire.

Vous implémenterez en fait une pile de deux façons différentes, en définissant une classe pour chaque façon.

- La première classe, nommée `Pile1`, utilisera un tableau d'entiers. Au départ, ce tableau aura une certaine longueur, petite (par exemple 3); il faudra veiller à agrandir

ce tableau lorsqu'il est plein et que l'on veut encore empiler un entier ; on pourra alors agrandir le tableau d'une quantité déterminée (par exemple 2).

- La seconde classe, nommée `Pile2`, utilisera une liste chaînée. Il faudra définir une classe supplémentaire pour modéliser un maillon de cette liste.
- Dans ce fichier, il faudra juste changer `Pile1` en `Pile2` si on veut tester la classe `Pile2`.