# Programmation Objet en Java - Feuille de TP 8

**Exercice 1.**

Un marchand de journaux souhaite pouvoir gérer efficacement son stock de magazines et gardant à jour un catalogue de tous ceux qu'il a en magasin et vous demande d'écrire un programme pour cela.

Un magazine peut figurer ou non dans le catalogue et s'il y figure, il peut être en plusieurs exemplaires. Le marchand de journaux veut pouvoir ajouter et enlever un ou plusieurs exemplaires d'un magazine dans le catalogue au fur et à mesure qu'il en reçoit ou qu'il en vend et aussi consulter le nombre d'exemplaires d'un magazine à un moment donné.

**Pour ce TP, vous écrirez toutes vos classes dans le package `fr.but.press`.**

a) Définir un objet `Magazine` permettant d'indiquer un nom ainsi qu'un mois (entier entre 1 et 12) et une année de parution.

- Attention à ne pas autoriser la création d'un magazine non valide.
- Ajouter de quoi afficher un magazine avec le mois en toutes lettres (pour cela, vous utiliserez un tableau contenant les noms des mois).

b) Écrire une interface `Catalog` avec :

1. une méthode `add` qui permet d'ajouter un exemplaire d'un magazine donné en paramètre
2. une méthode `add` qui permet d'ajouter plusieurs exemplaires d'un même magazine
3. une méthode `remove` qui permet de retirer un exemplaire d'un magazine donné en paramètre. Cette méthode devra renvoyer `false` si le magazine n'apparaît pas dans le catalogue (et `true` sinon)
4. une méthode `getNumber` qui permet de connaître le nombre d'exemplaires d'un magazine dans le catalogue

c) Écrire une classe `ImplementationCatalog` implémentant l'interface `Catalog`. Le catalogue sera représenté par une `Map` qui associe à chaque magazine son nombre d'exemplaires.

Vous pourrez tester votre implémentation en utilisant la classe de test fournie (bien sûr, cela ne vous empêche pas de faire vos propres tests).

1. Écrire un constructeur dans `ImplementationCatalog`.
2. Écrire les deux méthodes `add` pour ajouter des magazines en essayant de faire en sorte de ne pas dupliquer de code (tout en restant efficace).
3. Ajouter de quoi afficher un catalogue (simplement en affichant la `Map`) pour pouvoir tester l'ajout de magazines. Si cela ne fonctionne pas bien, vérifier le code de `Magazine`.
4. Écrire la méthode `getNumber` qui renvoie le nombre d'exemplaires d'un magazine. On renverra 0 si le magazine n'est pas présent dans le catalogue.
5. Écrire la méthode `remove` qui permet de retirer un exemplaire d'un magazine. Attention, un magazine ne doit pas apparaître dans le catalogue avec un nombre d'exemplaires négatif ou nul.

d) Quelles méthodes de `Magazine` sont utilisées par les méthodes `add` et `remove` du catalogue ? Pourquoi est-ce important que `Magazine` soit non mutable ? Vérifiez que c'est bien le cas dans votre code.

e) Ajouter une méthode `size` renvoyant la taille d'un catalogue qui est le nombre total d'exemplaires présents dans le stock. Dans quelle(s) classe(s) faut-il la mettre ?

f) L'affichage par défaut ne convient pas au marchand de journaux. Il voudrait un affichage ligne par ligne, avec le nombre d'exemplaires en début de ligne. Il voudrait aussi un affichage en ordre alphabétique des magazines (et lorsque qu'il y a différents numéros d'un même magazine, dans l'ordre des dates de parution). Par exemple :

```
1 x Chaussure magazine, février 2017
14 x La petite gazette des films, janvier 2017
16 x La petite gazette des films, mars 2017
3 x Leon, janvier 2017
8 x Leon, février 2017
12 x Leon, mars 2017
```

Modifier le code pour obtenir un tel affichage. Si cela ne fonctionne pas bien, vérifier le code de la classe `Magazine`.

g) Le marchand de journaux est ravi et le catalogue fonctionne tellement bien qu'il aimerait également pouvoir connaître l'ensemble de tous les numéros des magazines du même nom qu'il possède.

1. Est-ce possible ?
2. Si oui, comment ? Et, est-ce que c'est efficace ?
3. Si vous pouviez changer l'architecture du programme pour prendre en compte sa demande, comment feriez vous ?
4. (optionnel) Ajouter la méthode correspondante dans l'interface `Catalog` et écrire une deuxième classe `OtherImplementationCatalog` pour implémenter `Catalog` et qui utilise cette idée.

**Exercice 2.**

a) Écrire une classe `ArrayStack` qui implémente l'interface `Stack` donnée au TP précédent. La classe `ArrayStack` contiendra un tableau `table` d'entiers dont la taille maximale est fixée à la création. Un champ `height` désigne la hauteur de la pile, c'est-à-dire le nombre d'éléments empilés.

```
public class ArrayStack ... {
  private final int[] table;
  ...
  /**
  * Create a stack with a given capacity
  * @param capacity the stack capacity
  * @throws IllegalArgumentException is capacity is negative or null */
  public ArrayStack(int capacity){ ... }
```

b) Vérifier que la code qui teste la pile fonctionne également avec cette implantation.