# TD de programmation orientée objet en Java
# Les tableaux (suite)

## Exercice 1 : Retour sur le comptage

Écrire un programme Java appelé `FullOccurrenceCount` qui permet de compter le nombre d'occurrences des éléments présents dans un tableau contenant des entiers *quelconques*. La taille du tableau est à saisir par l'utilisateur. Le programme se chargera de remplir aléatoirement le tableau (toujours en utilisant un objet de type `Random`). Le programme affichera pour chaque élément son nombre d'occurrences.

Indications : on utilisera deux tableaux, un pour stocker les valeurs présentes dans le tableau, l'autre pour compter les occurrences de chaque valeur. Par exemple, si le tableau de départ est [1, 2, -5, 3, 2, -5, 1], on aura d'une part le tableau [1, 2, -5, 3] pour les 4 valeurs distinctes et d'autre part le tableau [2, 2, 2, 1] pour les occurrences de ces 4 valeurs.

## Exercice 2 : Séquences

Écrire un programme Java appelé `SeqOne` qui demande à l'utilisateur une taille de tableau, crée un tableau de la longueur demandée contenant uniquement les chiffres 0 et 1 (le contenu doit être aléatoire), puis détermine et affiche la distribution des longueurs des séquences de 1 présentes dans un tableau contenant uniquement les entiers 0 et 1. Considérons par exemple le tableau [0, 1, 1, 1, 0, 0, 1, 1]. Il contient :

1. 5 séquences de longueur 1 ;
2. 3 séquences de longueur 2 (on compte ainsi les séquences qui se recouvrent et donc 111 compte pour 2 séquences) ;
3. 1 séquence de longueur 3.

## Exercice 3 : Séquences fréquentes

On étudie les séquences fréquentes dans un tableau contenant uniquement les nombres entiers de 0 à k (fixé par l'utilisateur). Une séquence de nombres est dite fréquente si elle apparaît au moins p fois dans le tableau (le paramètre est fixé par l'utilisateur). Comme dans l'exercice précédent, on tient compte des chevauchements et la séquence [1,1] apparaît donc 3 fois dans [1,1,1,1].

On remarque que si une séquence est fréquente, alors toutes ses sous-séquences le sont. Pour trouver les plus longues séquences fréquentes, on commence donc par chercher toutes les séquences fréquentes de longueur 1, puis on en déduit celles de longueur 2, etc.

Considérons par exemple le tableau [1,2,3,2,3,1,2,3,3] avec comme seuil de fréquence 2. On voit que les trois chiffres 1, 2 et 3 apparaissent tous au moins 2 fois et donc que les trois séquences fréquentes de longueur 1 sont : [1] (2 fois), [2] (3 fois) et [3] (4 fois). On considère tous les couples formés par ces « séquences ». Par exemple, la séquence [1, 1] est candidate, mais comme elle n'apparaît jamais dans le tableau, elle n'est pas conservée. On trouve ainsi les séquences fréquentes de longueur 2 suivantes : [1,2] (2 fois) et [2,3] (3 fois). Pour les séquences de longueur 3, on combine une séquence de longueur 1 et une séquence de longueur 2, comme par exemple [1,1,2] (qui n'apparaît jamais). On trouve ainsi [1,2,3] (2 fois). Enfin, on cherche des séquences de longueur 4, soit par combinaison de 2 séquences de longueur 2, soit d'une séquence de longueur 3 et d'une séquence de longueur 1. Aucun des séquences obtenues n'apparaît au moins 2 fois. On en déduit donc que la séquence fréquente la plus longue est [1,2,3].

On procédera de la façon suivante :

1. Commencer par écrire une méthode qui compte le nombre d'occurrences d'un tableau (la séquence) dans un autre (le tableau d'origine) selon le modèle suivant

```
public static int compte(int[] tableau, int[] sequence) {
  ...
}
```

2. Écrire ensuite une méthode `main` qui trouve toutes les séquences fréquentes de longueur 1. On demandera à l'utilisateur, les valeurs de k et de p, ainsi que la longueur du tableau de départ. On stockera les séquences dans un tableau de type `int[][]`, soit un tableau de tableaux.
3. Modifier le programme pour qu'il cherche ensuite les séquences fréquentes de longueur 2, sachant qu'une telle séquence est nécessairement constituée de deux séquences fréquentes de longueur 1.
4. Généraliser le programme pour qu'il trouve les plus longues séquences fréquentes.