# TP n°08

## Collections d'objets - Requêtes

**Exercice 1 Gestion de stocks**

L'entreprise Yapi est spécialisée dans l'industrie alimentaire. Elle possède plusieurs usines et points de ventes répartis sur le territoire national. Le but de cet exercice est de concevoir un programme en Java qui permet la gestion de stocks et qui o re la possibilité de recueillir plusieurs informations relatives aux produits présents dans les di érents usines et magasins que l'entreprise gère. Pour l'implémentation, on définit la classe `Produit` suivante :

```
public class Produit {
    private String nom_prod;
    private String couleur;
    private int poids;
}
```

1. Ecrire le constructeur et les di érents accesseurs pour cette classe.
2. De la même façon, définir les classes `Usine (nom_usine, ville)`, `Magasin (nom_mag, ville)` et `Provenance (produit, usine, magasin, quantite)`. La classe `Provenance` fait le lien entre les 3 autres classes et permet de désigner l'usine d'où provient un produit donné stocké dans un magasin donné.

**Rappel sur Java : classe `LinkedList`**

Le langage Java propose la classe `java.util.LinkedList<Object>` pour implémenter les listes d'objets en rangeant les objets dans une liste chainée. Pour définir une liste d'objets de classe `Produit`, on utilise le code suivant :

```
LinkedList<Produit> liste_prod=new LinkedList<Produit>();
```

L'ajout des éléments à la liste se fait avec la méthode `boolean add(Object element)` de la classe `LinkedList` et la suppression par la méthode `boolean remove(Object element)`. On peut récupérer la taille de la liste (`int size()`) et tester si elle contient un objet donné (`boolean contains(Object element)`).
Pour récupérer les éléments de la liste, on utilise un objet de type `Iterator` qu'on peut récupérer en appelant la méthode `Iterator iterator()` sur la liste. Par exemple, pour récupérer l'itérateur de l'objet `liste_prod`, on utilise le code suivant :

```
Iterator prod_iterator=liste_prod.iterator();
```

La méthode `Object next()` de la classe Iterator retourne le prochain élément dans la liste et la méthode `boolean hasNext()` retourne `true` si l'élément pointé par l'itérateur n'est pas le dernier de la liste.

Pour a cher la liste des produits, on utilise le code suivant :

```
while (prod_iterator.hastNext()) {
System.out.println(i.next());
}
```

3. Ajouter un champ `static` de type `LinkedList` à chaque classe qui contiendra tous les éléments de la classe. Modifier le constructeur en conséquence.
4. Ecrire une méthode `main` qui insère quelques éléments.

**Produits :**

| **nom_prod** | **couleur** | **poids** |
|---|---|---|
| viande | rouge | 500 gr |
| bière | jaune | 100 gr |
| lait | blanc | 1 kg |
| cola | noir | 1 kg |

**Usines :**

| **nom_usine** | **ville** |
|---|---|
| YapiLyon | Lyon |
| YapiLille1 | Lille |
| YapiLille2 | Lille |

**Magasins :**

| **nom_mag** | **ville** |
|---|---|
| Healthy Food | Paris |
| First Price | Lyon |
| YapiMag Grenoble | Grenoble |

**Provenance :**

| **nom_produit** | **nom_usine** | **nom_mag** | **quantité** |
|---|---|---|---|
| lait | YapiLyon | First Price | 700 |
| lait | YapiLille | Healthy Food | 1200 |
| viande | YapiLille | Healthy Food | 200 |
| viande | YapiLille2 | YapiMag Grenoble | 100 |
| biére | YapiLyon | Healthy Food | 3000 |
| biére | YapiLille | First Price | 2600 |
| biére | YapiLille2 | YapiMag Grenoble | 1000 |
| cola | YapiLyon | Healthy Food | 2000 |
| cola | YapiLyon | First Price | 2000 |
| cola | YapiLyon | YapiMag Grenoble | 2000 |

5. Ecrire les méthodes `toString()` de toutes les classes.

   On cherche maintenant à répondre aux requettes suivantes :

6. Ecrire une méthode void `get-usines-in-ville(String la_ville)` qui a che toutes les usines qui sont à la ville `la_ville`.
7. Ecrire une méthode qui permet d'a cher tous les magasins qui sont approvisionnés par l'usine `YapiLyon` en produit `cola`.
8. Ecrire une méthode qui a che les magasins à qui on livre de la bierre
9. Ecrire une méthode qui retourne les couples de magasins et d'usines qui sont dans la même ville. Pour répondre à cette question, on créera une nouvelle classe `triplet (magasin, usine, ville)`.