TD 3

Quelques classes et interfaces de la bibliothèque des collections Java

Nous allons reprendre l'exercice 1 du TD3 et réécrire plusieurs fois la classe `Magasin` en utilisant différentes interfaces de la bibliothèque des collections Java consultables en Annexe.

Dans les trois cas décrits ci-dessous, la classe `Magasin` fournira un constructeur sans paramètre et les méthodes suivantes :

- `public boolean ajouterArticle(Article a)` qui ajoute l'article passé en paramètre au magasin.
- `public Article rechercherArticle(int reference)` qui retourne l'article correspondant à la référence passée en paramètre si elle existe dans le magasin et `null` sinon.
- `public boolean supprimerArticle (int ref )` qui supprime l'article correspondant à la référence passée en paramètre si elle existe dans le magasin et retourne `true` sinon retourne `false`.
- `public boolean promotion(int ref, int pourcentage)` qui applique la promotion à l'article correspondant à la référence passée en paramètre si elle existe dans le magasin et retourne `true` sinon retourne `false`.
- `public void liquidationTotale(int pourcentage)` qui applique une promotion à tous les articles du magasin selon le pourcentage passé en paramètre.
- `public String toString()` qui retourne les informations des articles présents dans le magasin sous forme d'une chaîne de caractères.

  Vous utiliserez les différentes méthodes permettant d'itérer sur une collection :

  - la méthode `Iterator<T> iterator()` de l'interface `Iterable<T>` et sa version `for(…:…)`
  - la méthode `default void forEach (Consumer< ? super T> action)` de l'interface `Iterable<T>`.

1. Créer une première classe `Magasin` caractérisée par un attribut `lesArticles`. Choisir le type de l'attribut parmi les interfaces de la bibliothèque des collections sachant qu'on souhaite gérer la collection des articles comme un ensemble, sans doublons. Quelles méthodes doit-on redéfinir dans la classe `Article` ?
   Ajouter à cette classe `Magasin` la méthode ci-dessous et la compléter pour que les articles de la liste retournée soient triés par ordre croissant de leur prix de vente.

```
public List<Article> triParOrdreCroissant(){
   List<Article> liste= new ArrayList<>(lesArticles);
   ...
   return liste ;
}
```

2. Créer une deuxième classe `Magasin` caractérisée par un attribut `lesArticles`. Choisir le type de l'attribut parmi les interfaces de la bibliothèque des collections sachant qu'on souhaite gérer la collection des articles comme un ensemble, sans doublons dont les

éléments sont munis d'une relation d'ordre.
Que doit-on ajouter à la classe `Article` ? Implémentez-le sachant qu'on souhaite comparer les articles par ordre croissant de leur référence.

Ajouter à cette classe `Magasin` les méthodes :

- `public Article lePremierArticle()` qui retourne l'article ayant la plus petite référence.
- `public Article leDernierArticle()` qui retourne l'article ayant la plus grande référence.

3. Quelle modification doit-on apporter à la classe `Magasin` ci-dessus si on souhaite modifier la relation d'ordre sans modifier l'ordre naturel défini dans la classe `Article`. On souhaite maintenant comparer les articles par ordre croissant de leur libellé.
4. Créer une troisième classe `Magasin` caractérisée par un attribut `lesArticles`. Choisir le type de l'attribut parmi les interfaces de la bibliothèque des collections sachant qu'on souhaite enregistrer des paires d'informations de type (référence, Article). Ajouter un article `a` de référence `r` consistera à ajouter une paire `(r, a)`.
Pour cette classe, la méthode `toString` affichera les paires sur des lignes différentes au format : Référence : r – informations sur l'article.