TP java
gestion d'une librairie

On veut écrire un programme de gestion d'une librairie vendant des cahiers. Chaque cahier possède un nombres de pages. Un achat est constitué de plusieurs cahiers. Le prix d'un cahier est égal à son nombre de pages divisé par 2.
L'affichage d'un achat doit indiquer un cahier par ligne avec la couleur du cahier, son nombre de pages, sa quantité (1 pour l'instant), puis une ligne indiquant le prix total de l'achat.
Exemple :

```
Jaune 20  x 1
Rouge 40  x 1
prix: 30
```

Il n'y a pas de nombre maximal de cahiers pour un achat. On utilisera donc la classe `HashSet` dans une première version et ensuite java.util.ArrayList pour stocker les cahiers.

On veut avoir une méthode main permettant les appels suivants :

```
public static void main(String[] args) {
  Cahier cahier1 = new Cahier(20, "Jaune");
  Cahier cahier2 = new Cahier(40, "Rouge");

  Achat achat = new Achat();
  achat.add(cahier1);
  achat.add(cahier2);
  System.out.println(achat);
}
```

Écrire les classes Cahier et Achat ainsi que leurs méthodes nécessaires (constructeurs, geters, seters, affiche, toString, …)

Vérifier que le code suivant fonctionne correctement (le résultat doit être vrai); sinon, corriger le code :

```
public static void main(String[] args) {
  HashSet<Cahier> set = new HashSet<>();
  Cahier c = new Cahier(20, "Jaune") ;
  set.add(c);
  System.out.println(set.contains(new Cahier(20, "Jaune")));
}
```

On vend un autre article : le carnet ; un carnet possède juste une valeur de 0 à 9 indiquant un facteur de qualité. Le prix d'un carnet est égal à 3 fois son facteur de qualité.
On veut donc que le main suivant fonctionne :

```
public static void main(String[] args) {
  Cahier cahier1 = new Cahier(20, "Jaune");
  Cahier cahier2 = new Cahier(40, "Rouge");
  Carnet carnet  = new Carnet (5);

  Achat achat = new Achat();
```

```
  achat.add(cahier1);
  achat.add(cahier2);  // un cahier
  achat.add(carnet );    // un carnet
  System.out.println(achat);
 }
```

Écrire et modifier les classes et les méthodes nécessaires.

On veut permettre d'ajouter plusieurs cahiers ou plusieurs carnets à notre achat d'un coup en indiquant un paramètre supplémentaire à la méthode add, indiquant la quantité des cahiers ou des carnets que l'on veut mettre dans l'achat.
L'affichage de l' achat devra indiquer la quantité de chaque cahier et carnet :

```
 public static void main(String[] args) {
  Cahier cahier1 = new Cahier(20, "Jaune");
  Cahier cahier2 = new Cahier(40, "Rouge");
  Carnet carnet  = new Carnet (5);

  Achat achat = new Achat();
  achat.add(cahier1, 5);      // 5 cahiers
  achat.add(cahier2);
  achat.add(carnet , 7);        // 7 carnets
  System.out.println(achat);
 }
```

Modifiez le code en conséquence, sachant que pour représenter une cahier (ou un carnet) et une quantité, le plus simple est de créer une classe QuantiteArticle qui contient un cahier (ou une carnet) et une quantité (une valeur entière).

Pourquoi la classe QuantiteArticle ne doit pas être déclarée "public" ?

Enfin, au lieu de représenter les couleurs de cahier par des String, on voudrait éviter les erreurs de saisie en représentant les valeurs d'une énumération.

```
 enum CahierCouleur {
  Jaune, Rouge, Bleu;
 }
```

Modifier le code pour que le main suivant fonctionne :

```
 public static void main(String[] args) {
  Cahier cahier1 = new Cahier(20, CahierCouleur.Jaune);
  Cahier cahier2 = new Cahier(40, CahierCouleur.Rouge);
  Carnet carnet  = new Carnet (5);

  Achat achat = new Achat();
  achat.add(cahier1, 5);
  achat.add(cahier2);
  achat.add(carnet , 7);
  System.out.println(achat);  // définir la fonction toString qui retourne le montant des achats
                              // dans la classe Achat
```

```
}
```

Indications :

Pour la classe ArrayList, consulter le diaporama du cours.
La classe HashSet crée une collection d'objets qui utilise une table de hachage pour le stockage.
Une table de hachage stocke les valeurs en leurs donnant une clé unique pour les identifier.
La liste des méthodes appart ceux qui sont héritées des classes parents:

- `add(E e)` : ajouter un élément;
- `remove(Object o)` : supprimer un élément;
- `clear()` : effacer tous les éléments de HashSet;
- `contains(Object o)` : retourne true si l'objet recherché existe sinon false;
- `isEmpty()` : retourne true si la liste est vide;
- `size()` : renvoie la taille;

Exemple :

```java
import java.util.HashSet;
public class ExempleHashSet {
 public static void main(String[] args) {
 HashSet<String> hset = new HashSet<String>();
 //ajouter des éléments
 hset.add("hôtel");
 hset.add("motel");
 hset.add("fondouk");
 hset.add("ressort");
 System.out.println(hset.size());
 //supprimer l'objet motel
 hset.remove("motel");
 //tester l'existance
 System.out.println(hset.contains("motel"));
 //vérifier si set est vide
 System.out.println(hset.isEmpty());
 //parcourir HashSet
 for(String valeur:hset)
 System.out.println(valeur);
 }
}
```

Enum en java :

```
public class Test
     {    public enum Jour { lundi, mardi, mercredi, jeudi,
                             vendredi, samedi, dimanche }
          Jour j ;
          Test (Jour j) { this . j = j; }
          Jour getJour() {return j ; }

           public static void main (String args [])
          {
               Test t = new Test(Jour.mardi) ;
               if (t.getJour() == Jour.lundi)
               System.out.println("OK") ;
               else System.out.println("Non") ;

          }
     }
```