Conception orientée objet

Application du cours 6 : Parchemin

Cet énoncé vient en appui des diapositives du Cours.

I. Méthode hashCode & equals

En Gaule il existe quelques parchemins provenant de Rome que l'on souhaite stocker dans une bibliothèque.
Les gaulois souhaitent cependant ne garder qu'un seul exemplaire d'un parchemin.

Aider Keskonrix à écrire la classe « Parchemin » en ajoutant les méthodes *equals* et *hashCode*.

```
public class Parchemin {
  private String titre;
  private Personnage auteur;
  private Date date;

  public Parchemin(String titre, Personnage auteur, Date date) {
    this.titre = titre;
    this.auteur = auteur;
    this.date = date;
  }

  public String toString() {
    return titre + ", " + auteur + ", " + date;
  }

  //Méthode hashCode
```

```
public int hashCode() {
    return 31*titre.hashCode() + 31*auteur.hashCode() + 31*
        date.hashCode()
}
```

```
//Méthode equals
public boolean equals (Object objet){
    if(objet instanceof Parchemin){
    Parchemin parcheminToCompare =(Parchemin) objet ;
    return (titre.equals(parcheminToCompare.titre)
            && auteur.equals(parcheminToCompare.auteur)
            && date.equals(parcheminToCompare.date));
    }
    return false;
}
```

II. Classe HashSet

1. Objets existants

Ci-dessous les parchemins détenus par la bibliothèque.

```
Romain cesar = new Romain("Cesar");
Gaulois druide = new Gaulois("Panoramix");
Gaulois assurancetourix = new Gaulois("Assurancetourix");
Parchemin effetsPotion =
   new Parchemin("Les effets secondaires de la potion magique", druide,
                 new Date(21, 12, -70));
Parchemin musiqueBestOf =
   new Parchemin("Mes plus grands succès", assurancetourix,
                 new Date(30, 04, -45));
Parchemin guerreDesGaules =
   new Parchemin("Commentaires sur la guerre des gaules", cesar,
                 new Date(12, 07, -50));
```

2. *Travail à effectuer*

Veuillez utiliser la javadoc de la classe « HashSet » donnée à la fin du livret.

Créer l'ensemble **ensembleParchemin** contenant les trois parchemins.

```
Set<Parchemin> ensembleParchemin = new HashSet<>();
ensembleParchemin.add(effetPotion);
ensembleParchemin.add(musiqueBoatOf);
ensembleParchemin.add(guerreDesGaules);
```

Créer l'ensemble **parcheminsAPreter** contenant l'ensemble des parchemins du musée à partir de l'ensemble **ensembleParchemin**.

```
Set<Parchemin> parcheminAPreter = new HashSet<>(ensembleParchemin);
```

Créer l'ensemble vide **parcheminsPretes** qui contiendra les parchemins prêtés par le musée.

```
Set<Parchemin> parcheminsPretes = new HashSet<>();
```

```
Parchemins stockés au musée :
- Mes plus grands succès, Assurancetourix, 30/4/-45
- Commentaires sur la guerre des gaules, Cesar, 12/7/-50
- Les effets secondaires de la potion magique, Panoramix, 21/12/-70
```

Pneumatix nous apporte un nouvel exemplaire de « Commentaires sur la guerre des gaules » de Jules César.
Que se passe-t-il lors de son ajout dans l'ensemble **ensembleParchemin** ?

Le nouvel exemplaire est considéré comme doublon et n'est pas ajouté dans l'ensemble

Répondre aux questions suivantes ?

- A-t-on prêté des parchemins ?

System.out.println("A-t-on prêté des parchemins ? "
+ !parcheminsPretes.isEmpty());

- Transférer les parchemins « Commentaires sur la guerre des gaules » et « Mes plus grands succès » de l'ensemble **parcheminsAPreter** à l'ensemble **parcheminsPretes** sans utiliser la méthode *add*.
  - Supprimer les parchemins à transférer de l'ensemble **parcheminsAPreter**
  - Ajouter tous les parchemins de l'ensemble **ensembleParchemin** dans l'ensemble **parcheminsPreter**
  - Supprimer dans l'ensemble **parcheminsPretes** tous les parchemins de l'ensemble **parcheminsAPreter**

parcheminsAPreter.remove(guerreDesGaules);
parcheminsAPreter.remove(musiqueBertof);
parcheminsPretes.addAll(ensembleParchemin);
parcheminsPretes.removeAll(parcheminAPreter);

- L'ensemble **parcheminsAPretes** contient uniquement le parchemin « Les effets secondaires de la potion magique ».

  Ci-dessous les instructions permettant l'affichage du hashCode :
  - du parchemin « Les effets secondaires de la potion magique »,
  - de l'ensemble **parcheminsAPretes**.

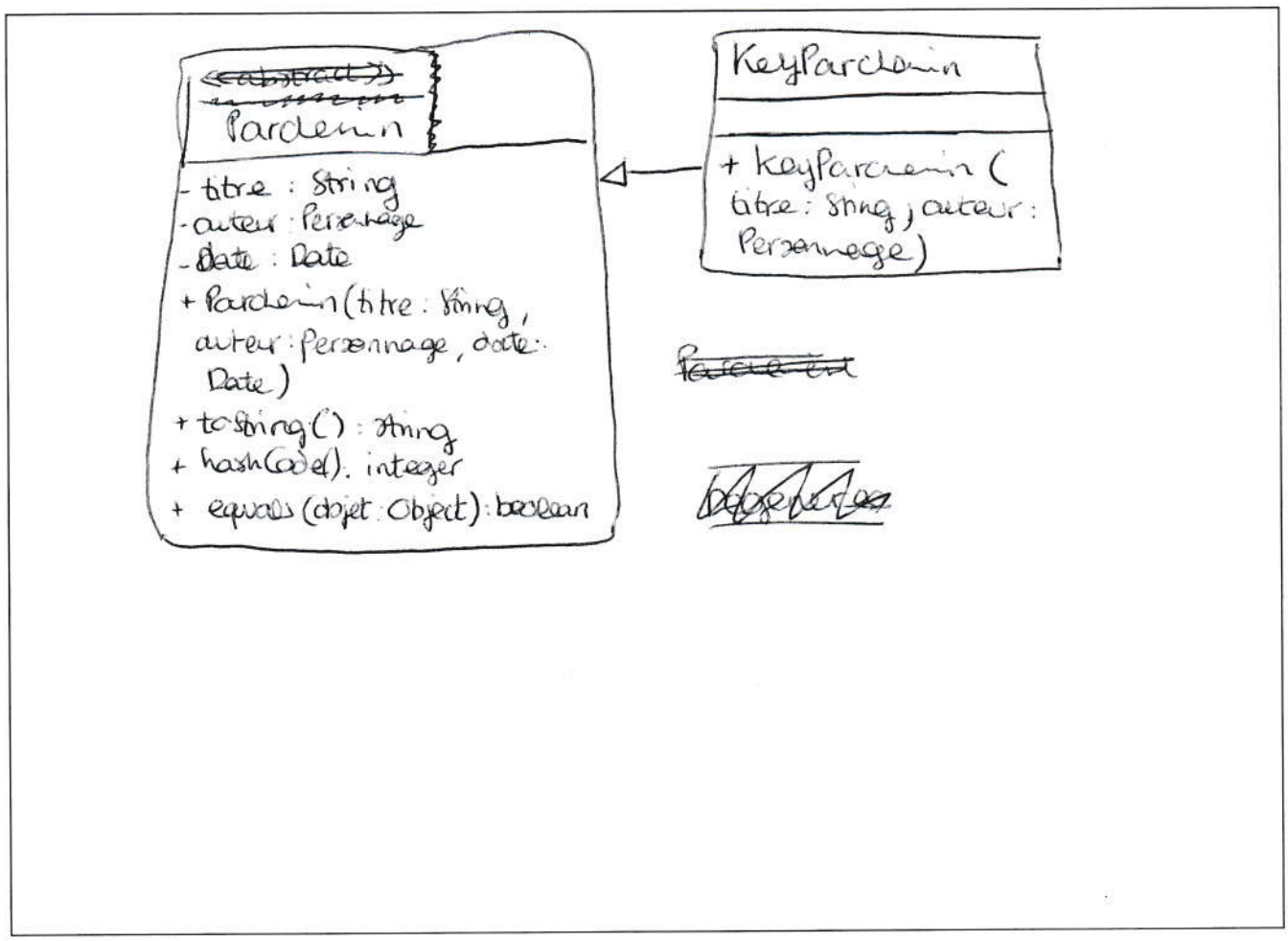
```
System.out.println("Le hashCode du parchemin effetsPotion est : "
                + effetsPotion.hashCode());
System.out.println("Le hashCode de l'ensemble parcheminsAPreter est : "
                + parcheminsAPreter.hashCode());
```

  A votre avis, le hashCode de l'ensemble **parcheminsPretes** est-il identique à celui du parchemin effetsPotion ?

```
Le hashCode du parchemin effetsPotion est : 679629913
Le hashCode de l'ensemble parcheminsAPreter est :
```

III. Les objets et les classes dégénérées

Admettons que deux parchemins sont maintenant considérés identiques s'ils ont le même titre et le même auteur, indépendamment de la date. Donner le diagramme UML comportant la classe Parchemin et sa classe dégénérée.



Donner le code correspondant à la classe dégénérée.

```
public class KeyParchemin {
    public KeyParchemin (String titre, Personnage auteur) {
        super (titre, auteur, NULL);
    }
}
```

Comment savoir si le parchemin « Les effets secondaires de la potion magique » écrit par Panoramix est dans l'ensemble parcheminsAPreter?

Nous avons deux possibilités :

- Utiliser un objet dégénéré

Parchemin parcheminDegeneres = new Parchemin <>("Les effets secondaire de la potion magique", druide, null);

System.*out*.println("A-t-on le parchemin sur la potion magique ? "
+ parcheminDegeneres.contains(parcheminDegenere));

);

- Utiliser une classe dégénérée

Parchemin cleParchemin = new keyParchemin ("Les effets secondaire", druide);

System.*out*.println("A-t-on le parchemin sur la potion magique ? "
+ parcheminAPreter.contains(cleParchemin));

);

JAVA Doc : HashSet

Constructor and Description
HashSet() Constructs a new, empty set; the backing HashMap instance has default initial capacity (16) and load factor (0.75).
HashSet(**Collection**<? extends **E**> c) Constructs a new set containing the elements in the specified collection.
HashSet(int initialCapacity) Constructs a new, empty set; the backing HashMap instance has the specified initial capacity and default load factor (0.75).
HashSet(int initialCapacity, float loadFactor) Constructs a new, empty set; the backing HashMap instance has the specified initial capacity and the specified load factor.

Method Summary

Modifier and Type	Method and Description
boolean	**add**(**E** e) Adds the specified element to this set if it is not already present.
void	**clear**() Removes all of the elements from this set.
Object	**clone**() Returns a shallow copy of this HashSet instance: the elements themselves are not cloned.
boolean	**contains**(**Object** o) Returns true if this set contains the specified element.
boolean	**isEmpty**() Returns true if this set contains no elements.
Iterator<E>	**iterator**() Returns an iterator over the elements in this set.
boolean	**remove**(**Object** o) Removes the specified element from this set if it is present.
int	**size**() Returns the number of elements in this set (its cardinality).

Methods inherited from class java.util.AbstractSet
equals, hashCode, removeAll
Methods inherited from class java.util.AbstractCollection
addAll, containsAll, retainAll, toArray, toArray, toString
Methods inherited from class java.lang.Object
finalize, getClass, notify, notifyAll, wait, wait, wait
Methods inherited from interface java.util.Set
addAll, containsAll, equals, hashCode, removeAll, retainAll, toArray, toArray