Conception orientée objet

Application du cours 7 : Le musée

Cet énoncé vient en appui des diapositives du Cours.

I. Classe et interface déjà implémentées

```
public interface GestionTrophee {
        String tousLesTrophees();
        String lesTrophees(Gaulois proprietaire);
        void ajouterTrophee(Gaulois proprietaire, Equipement trophee);
        String donnerGauloisDonnateur();
        String donnerTrophees();
        String donnerInventaire();
}

public class Musee {
   private String nom;
   private int tarif;
   private GestionTrophee gestionnaireTrophee;
   public Musee(String nom, GestionTrophee gestionnaireTrophee) {
      this.nom = nom;
      this.gestionnaireTrophee = gestionnaireTrophee;
   }
   public String getNom() {
      return nom;
   }
   public int getTarif() {
      return tarif;
   }
   public void setTarif(int tarif) {
      this.tarif = tarif;
   }
   public void ajouterTrophee(Gaulois proprietaire, Equipement trophee) {
      gestionnaireTrophee.ajouterTrophee(proprietaire, trophee);
   }
   public String tousLesTrophees() {
      return gestionnaireTrophee.tousLesTrophees();
   }
   public String lesTrophees(Gaulois proprietaire) {
      return gestionnaireTrophee.lesTrophees(proprietaire);
   }
   public String donnerGauloisDonnateur() {
      return gestionnaireTrophee.donnerGauloisDonnateur();
   }
   public String donnerTrophees() {
      return gestionnaireTrophee.donnerTrophees();
   }
```

```
    public String donnerInventaire(){
        return gestionnaireTrophee.donnerInventaire();
    }
}
```

II. Travail à effectuer

1. Dans la classe « KeskonrixGestion », écrire l'attribut trophees comme une association entre un objet de type « Gaulois » (la clé) et une liste d'objets de la classe « Equipement ».

2. Compléter les méthodes :
   - *ajouterTrophee* qui place le nouvel équipement dans la liste de trophée d'un gaulois. Si le gaulois n'est pas dans les clés de la map, alors l'ajouter.
   - *tousLesTrophees* qui retourne une chaine contenant l'ensemble des trophées du musée classé selon leur donateur. Exemple de chaine : `System.out.println(musee.tousLesTrophees());`

```
Tous les trophées du musée sont :
Les trophées de Ordralfabétix sont :
- un casque

Les trophées de Astérix sont :
- un bouclier
- un casque
- une épée

Les trophées de Obélix sont :
- une épée
```

   - *lesTrophees* qui retourne une chaine contenant l'ensemble des trophées du musée d'un donateur en particulier. Exemple de chaine : `System.out.println(musee.lesTrophees(asterix));`

```
Les trophées de Astérix sont :
- un bouclier
- un casque
- une épée
```

3. Pour la compréhension sur les vues d'une map, on ajoute 2 attributs :
   - gauloisDonateur initialisé avec une vue sur les clés de l'association trophees,
   - collectionTrophees initialisé avec une vue sur les valeurs de l'association trophees.

   Créer les méthodes :
   - *donnerGauloisDonateur* qui renvoie une chaine contenant l'ensemble des gaulois ayant fait un don au musée. Exemple de

chaine :
`System.out.println(musee.donnerGauloisDonateur());`

```
Les gaulois ayant donné au moins un trophée au musée sont :
- Ordralfabétix
- Astérix
- Obélix
```

- *donnerTrophees* qui renvoie une chaine contenant l'ensemble des trophées du musée. Exemple de chaine :
`System.out.println(musee.donnerGauloisDonateur());`

```
Les trophées du musée sont :
- un casque
- un bouclier
- un casque
- une épée
```

4. Dans la classe « KeskonrixGestion » :
   - Ecrire l'attribut inventaire qui stocke pour chaque donateur les équipements donnés au musée associés à leur nombre d'exemplaire. Les donateurs seront placés dans l'ordre alphabétique. Exemple :
   {asterix = { Equipement.CASQUE=2, Equipement.BOUCLIER=1}, obelix = {Equipement.EPEE=1}}

   - Entourer dans la classe « Personne » (classe mère de la classe « Gaulois ») ce qui est indispensable au bon fonctionnement de la TreeMap

   - Compléter la méthode *ajouterTrophee* afin de remplir la map inventaire.

   - Créer la méthode *donnerInventaire* qui retourne une chaine correspondant à l'inventaire du musée. Exemple de chaine :
   System.out.println(musee.donnerInventaire());

```
INVENTAIRE DU MUSEE

Astérix a donné :
- 1 x un bouclier
- 2 x un casque

Obélix a donné :
- 1 x une épée

Ordralfabétix a donné :
- 1 x un casque
```

III. Classe KeskonrixGestion

public class KeskonrixGestion **implements** GestionTrophee {

//Les attributs

//Les associations

private ~~Map<Gaulois, List<Equipement>>~~ trophees

= **new** ~~HashMap()~~;

private ~~Set<Gaulois> Map<Equipement>~~ NavigableMap<Gaulois, Map<Equipement, Integer>> ~~inventaire~~

Inventaire = **new** TreeMap<>();

// Les vues sur l'association

private ~~Set<Gaulois> gauloisDonateur~~ gauloisDonnateur

= ~~trophees.keySet()~~;

private ~~Collection<List<Equipement>>~~ collectionTrophees

= ~~trophees.values()~~;

//Les méthodes

public void ajouterTrophee(Gaulois proprietaire, Equipement trophee) {

~~List<Equipement> liste;~~

~~if(!trophees.containsKey(proprietaire)){~~

~~liste = new ArrayList<>();~~

~~proprietaire.put(proprietaire, liste);~~

~~}else{~~

~~liste = trophees.get(proprietaire);~~

~~}~~

~~liste.add(trophee)~~

~~}~~

//partie concernant la map inventaire

~~Map<Equipement, Integer> equipements;~~

~~int nbEquipement=1~~

~~if(inventaire.containsKey(proprietaire)){~~
~~equipements = inventaire.get(proprietaire);~~
~~if(equipement.containKey(equipement)){~~
~~nbEquipement = equipement.get(equipement);~~
~~nbEquipement++;~~
~~}else{~~
~~equipements = new HashMap<>();~~
~~inventaire.put(proprietaire, equipements);~~
~~}~~
~~equipement.put(equipement, nbEquipement);~~
~~}~~

}

```
public String lesTrophees(Gaulois proprietaire) {
    String chaine = "Les trophées de " + proprietaire.getNom() + " sont :\n";
```

~~for(int i=0; i<nbDeTrophee; i++){~~
~~Gaulois proprietaireTrophee = trophees[i].getProprietaire();~~
~~if(proprietaire.equals(proprietaireTrophee)){~~
~~Equipement typeEquipement = trophees[i].getTrophee();~~
~~chaine += "-" + typeEquipement + "\n";}~~
~~}~~

```
    return chaine;
}
```

```
List<Equipement> equi = trophees.get(proprietaire);
for(Equipement equipement: equi){
  chaine += "-" + equipement + "\n";
}
```

```
public String tousLesTrophees() {
   String chaine = "Tous les trophées du musée sont :\n";
   for(Set<Gaulois> collection = trophees.keySet();
   for(int i=0; i<nombreDeTrophee; i++){
   for(Gaulois gaulois: collection){
   Equipement typeEquipement = trophees[i].getTrophee(t);
   chaine += "  " + typeEquipement + "\n";
   }
   return chaine;
}

//Les méthodes pour la compréhension des vues
public String donnerGauloisDonateur() {
   String chaine = "";
   if ( ! gauloisDonateurs.isEmpty() ) {
      chaine = "Les gaulois ayant donné au moins un trophée au musée
                sont :\n";
      for ( Gaulois gaulois: gauloisDonateur ) {
         chaine += " - " + gaulois + "\n";
      }
   } else {
      chaine = "Aucun gaulois n'a fait don d'un trophée au musée.";
   }
   return chaine;
}

public String donnerTrophees() {
   String chaine = "";
   if ( ! collectionTrophees.isEmpty() ) {
      chaine = "Les trophées du musée sont :\n";
      for ( Trophee trophee : trophees.liste<Equipement> ) {
         for(Equipement equipement: liste) {
         chaine += " - " + equipement + "\n";
      }
   } else {
      chaine = "Il n'y a aucun trophée au musée.";
   }
   return chaine;
}
```

liste : CollectionTrophees

```
//méthode concernant la map inventaire

public String donnerInventaire(){
  String chaine = "\nINVENTAIRE DU MUSEE\n";

  for (Gaulois proprietaire : inventaire.keySet() ) {

    chaine += "\n" + proprietaire.getNom() + " a donné :\n";

    Map<Equipement, Integer> equipements = inventaire.get
    (proprietaire);

    for (Equipement equipement : equipements.keySet() ) {

      chaine += "- " + equipement.get(equipement)
      + " x " + equipement + "\n";
    }
  }
  return chaine;
 }
}
```

IV. Classe Personne

```
public class Personne implements Comparable<Personne>{
  protected String nom;

  public Personne(String nom) {
    this.nom = nom;
  }

  public String getNom() {
    return nom;
  }

  public String toString() {
    return nom;
  }

  public int compareTo(Personnage personnage) {
    return nom.compareTo(personnage.nom);
  }

  public boolean equals(Object object) {
    if(object != null && object.getClass() == getClass()){
      Personnage personnage = (Personnage) object;
      return nom.equals(personnage.nom);
    }
    return false;
  }
```

//méthode manquante pour l'utilisation comme clé dans une HashMap

public int HashCode(){

return 31* nom.HashCode();

}

V. Extrait de la JavaDoc TreeMap

Constructor and Description
TreeMap() Constructs a new, empty tree map, using the natural ordering of its keys.
TreeMap(**Comparator**<? super **K**> comparator) Constructs a new, empty tree map, ordered according to the given comparator.

Modifier and Type	Method and Description
boolean	**containsKey**(**Object** key) Returns true if this map contains a mapping for the specified key.
Boolean	**containsValue**(**Object** value) Returns true if this map maps one or more keys to the specified value.
V	**get**(**Object** key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
Set<K>	**keySet**() Returns a **Set** view of the keys contained in this map.
V	**put**(**K** key, **V** value) Associates the specified value with the specified key in this map.
Collection<V>	**values**() Returns a **Collection** view of the values contained in this map.