Conception orientée objet

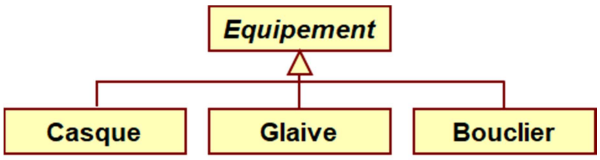
Application du cours 3 : Musée et rangement en tas

Cet énoncé vient en appui des diapositives du Cours.

I. Enoncé du TD – première partie : généricité

1. Classe RenseignementTrophee

```
public class RenseignementTrophee {
  private Gaulois proprietaire;
  private Equipement trophee;

  public RenseignementTrophee(Gaulois proprietaire, Equipement trophee) {
    this.proprietaire = proprietaire;
    this.trophee = trophee;
  }

  public Gaulois getProprietaire() {
    return proprietaire;
  }

  public Equipement getTrophee() {
    return trophee;
  }
}
```



2. Classe Paire

```
public class Paire<T, U> {
  private final T premier;
  private final U second;

  public Paire(T premier, U second) {
    this.premier = premier;
    this.second = second;
  }

  public T getPremier() {
    return premier;
  }

  public U getSecond() {
    return second;
  }
}
```

Nous souhaitons utiliser la classe générique « Paire<T, U> » afin de remplacer à terme la classe « Renseignement ».

Dans la classe « Test » nous avions des exemples de création d'objets de la classe « RenseignementTrophee ».

```
RenseignementTrophee asterixTrophee1 =
        new RenseignementTrophee(asterix, new Bouclier());
RenseignementTrophee asterixTrophee2 =
        new RenseignementTrophee(asterix, new Casque());
RenseignementTrophee obelixTrophee1 =
        new RenseignementTrophee(obelix, new Glaive());
```

Remplacer le type de ces objets par le type générique « Paire<T,U> »

asterixTrophee1 =

asterixTrophee2 =

obelixTrophee1 =

Faire afficher le trophée (le glaive) qu'a ramené Obélix.

System.out.println( );

II. Enoncé du TD – deuxième partie : itérateurs et classes itérables

1. Les Iterateurs

Obélix organise en tas les équipements récupérés sur les romains : un tas de casques, un tas de glaives, un tas de boucliers … Mais attention, il est très ordonné et ne les mélange pas !

1. Créer la classe « Tas » qui contient un tableau tasEquipement. Ce tableau est transmis par le constructeur. Un attribut nombreEquipement est initialisé à 0 et permet de connaitre le nombre d'équipements que contient le tas.
2. Ecrire la méthode *ajouterEquipement* qui ajoute un équipement au tableau tas.
3. Implémenter l'interface « *Iterator* ».

public class Tas

{

//Attributs

//Constructeur

//Méthodes

public void ajouterEquipement(equipement) {

}

//Méthodes de l'interface Iterator

Compléter le main de la classe « Test » afin qu'il affiche l'ensemble du tas.

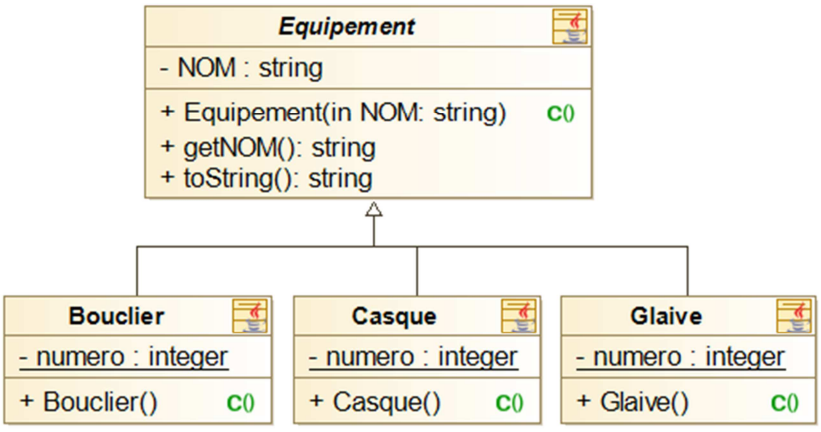
```
Glaive glaive1 = new Glaive();
Glaive glaive2 = new Glaive();
Glaive glaive3 = new Glaive();

Tas<Glaive> tasGlaive = new Tas<>(new Glaive[5]);
tasGlaive.ajouterEquipement(glaive1);
tasGlaive.ajouterEquipement(glaive2);
tasGlaive.ajouterEquipement(glaive3);
```

2. Classe Iterable

Notre classe « Tas » implémente l'interface « Iterator », mais nous pouvons le parcourir qu'une seule fois. Pour pouvoir le parcourir autant de fois que nécessaire, nous souhaitons l'améliorer afin qu'elle :

- soit itérable,
- soit générique pour avoir soit un tas de casques, soit un tas de glaives, soit un tas de boucliers …
- possède la méthode *afficherTas* qui parcourt et affiche l'ensemble d'un tas.

Spécifier, à l'aide d'UML, la nouvelle conception.



Ecrire le code Java correspondant.

public class Tas

{

//Attributs

//Constructeur

//Méthodes

public void ajouterEquipement(equipement) {

}

//Méthode de l'interface Iterable

//Classe interne

//Attributs

```
//Méthode de vérification de la concurrence entre itérateurs
                                                    
                                                    {
                                                    
                                                    
                                                    
                                                    
                                                    
}

//hasNext
                                                    
                                                    {
                                                    
                                                    
                                                    
                                                    
}

//next
                                                    
                                                    
                                                    {

                                                    
                                                    
                                                    
                                                    
                                                    
                                                    
                                                    
}
```

```
//remove
[ ]
[ ]
[ ] {
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
}
} }
```

Compléter le main de la classe « Test » afin qu'il :

- affiche l'ensemble du tas,
- supprime le premier glaive,

```
Glaive glaive1 = new Glaive(); Glaive glaive2 = new Glaive();
Glaive glaive3 = new Glaive();
Tas<Glaive> tasGlaive = new Tas<>(new Glaive[5]);
tasGlaive.ajouterEquipement(glaive1);
tasGlaive.ajouterEquipement(glaive2);
tasGlaive.ajouterEquipement(glaive3);

System.out.println("Affichage des trois glaives");
[ ]
[ ]
[ ]

System.out.println("\nSupression du premier glaive");
[ ]
[ ]
[ ]
```