Fantaisie Travaux Pratiques n°2

« Itérateur spécial liste, Méthode equals,
ArrayList, LinkedList»

Ce TP poursuit le TP1 sur les monstres. Afin d'avoir toujours une version stable par sujet de TP :
- Ouvrir Eclipse, sélectionner le même workspace que pour le premier TP (par exemple : Z:\Licence3\Semestre1\POO2\workspace).
- Copier le projet fantaisieTP1 (en le sélectionnant puis ctrl+c),
- Coller le projet (ctrl+v), et le renommer fantaisieTP2.

Dans ce TP vous travaillerez donc uniquement dans ce nouveau projet.

I. Les Humains – *implémentation de diagrammes de classes*

Dans cette partie vous devez implémenter le diagramme de classes p5. Des indications pour chacune des classes sont données ci-dessous. Implémentez les classes dans l'ordre des paragraphes ci-dessous.

1. Paquetage Bataille

- Créer la classe « Camp » contenant deux opérations : *ajouter* et *eliminer*. Laisser ces opérations vides, nous y reviendrons plus tard.

  Pour ne pas oublier que ces opérations ne sont pas implémentées, Eclipse permet d'ajouter des « Task ». Ces tâches commencent par : //TODO *puis l'explication de la tâche*.

  Dans le corps de la méthode écrivez :
  //TODO methode non implementee
  Pour retrouver toutes les tâches à effectuer sélectionner :
  Window / Show View / Tasks
  Apparaît alors la liste des tâches dans un onglet sous votre code. Pour naviguer parmi les tâches il suffit de cliquer dessus …
  Profitez-en pour supprimer les tâches que vous auriez laissées par erreur.

- Créer la classe « Bataille ». Elle contient 2 attributs campHumains et campMonstres de type « Camp » et 4 méthodes.
  Les méthodes *ajouter* et *eliminer* utilisent les méthodes *ajouter* et *eliminer* de la classe « Camp» depuis respectivement campHumains et campMonstres.

2. Paquetage protagoniste

- Créer la classe « Homme » qui hérite de la classe « EtreVivant ».
  Lui ajouter le constructeur sachant que tous les hommes ont une force de vie de 70 points.
- Créer la classe « Heros » qui hérite de la classe « Homme ».
  Lui ajouter le constructeur sachant que tous les héros ont une force de vie de 100 points.
- Modifier la classe « EtreVivant » en lui ajoutant l'attribut bataille. Créer les méthodes *rejointBataille* dans les classes « EtreVivant », « Homme » et « Monstre ». La méthode de la classe mère affectera son paramètre d'entrée à l'attribut bataille. Les méthodes des classes filles utiliseront la bonne méthode *ajouter* via l'attribut bataille, pour gérer la bi-navigabilité de l'association.
- Modifier la classe « EtreVivant » en lui ajoutant la méthode abstraite *mourir* (effectivement tous les êtres vivants meurent). Implémenter cette méthode dans les classes « Homme » et « Monstre ». Les méthodes des classes filles utiliseront la bonne méthode *eliminer* via l'attribut bataille.

II. Gestion des camps

1. Les camps – Généricité, LinkedList, equals

Dans la classe « Camp » ajouter un attribut liste de type ***List*** initialisé comme un nouvel objet de la classe « LinkedList » du paquetage « *java.util* ».
Ces camps contiennent soit des monstres, soit des humains. Modifier la signature de la classe « Camp » et typer la liste correctement afin de prendre en compte cette exigence.

Compléter les méthodes *ajouter* et *eliminer* :
- *ajouter* ajoute un monstre ou un humain dans la liste à condition qu'il n'y soit pas déjà.
- *eliminer* supprime un monstre ou un humain de la liste.

Deux êtres vivants sont considérés comme identiques, si et seulement si, ils ont le même nom (il n'existe pas d'homonymes).

Nous allons maintenant procéder à des tests fonctionnels.
Pour cela :
- Générer la méthode *toString* dans les classes « Heros » et « Homme ».
- Ecrire la méthode *toString* dans la classe « Camp».

```
public String toString() {
  return liste.toString();
}
```

- Générer deux getters sur les camps dans la classe « Bataille ».
- Télécharger la classe « TestGestionProtagonistes » sous Moodle contenant le code suivant :

```
Monstre<Feu> dragotenebre = new Monstre<>("dragotenebre", 200,
    ZoneDeCombat.AERIEN, Domaine.FEU, new BouleDeFeu(4), new Lave(1),
```

```
    new Eclair(3));
Monstre<Tranchant> vampirien = new Monstre<>("vampirien", 10,
    ZoneDeCombat.AERIEN, Domaine.TRANCHANT, new Morsure(10));
Monstre<Glace> marinsangant = new Monstre<>("marinsangant", 150,
    ZoneDeCombat.AQUATIQUE, Domaine.GLACE, new PicsDeGlace(10),
    new Tornade(1));
Monstre<Tranchant> guillotimort = new Monstre<>("guillotimort", 80,
    ZoneDeCombat.TERRESTRE, Domaine.TRANCHANT, new LameAcier(10),
    new Griffe());
Homme thomas = new Homme("Thomas");
Homme louis = new Homme("Louis");
Heros arthur = new Heros("Arthur");
Heros archibald = new Heros("Archibald");
Homme alain = new Homme("Alain");

Bataille bataille = new Bataille();
thomas.rejointBataille(bataille);
louis.rejointBataille(bataille);
arthur.rejointBataille(bataille);
archibald.rejointBataille(bataille);
alain.rejointBataille(bataille);
dragotenebre.rejointBataille(bataille);
vampirien.rejointBataille(bataille);
marinsangant.rejointBataille(bataille);
guillotimort.rejointBataille(bataille);

Camp<Homme> campsHumain = bataille.getCampHumains();
Camp<Monstre<?>> campsMonstre = bataille.getCampMonstres();

System.out.println("**** TP2 ****");
System.out.println("\ncamp des humains :\n" + campsHumain);
System.out.println("\ncamp des monstres :\n" + campsMonstre);
```

2. *Visualisation des forces – ArrayList, LinkedList, Itérateur de liste, refactoring*

Pour cette partie n'hésitez pas à reprendre l'exemple du cours numéro 3, plus précisément l'exercice donné en TD sur la potion magique.

Rendre la classe « Camp» itérable. La méthode *iterator* retournera l'itérateur de la liste liste.

Créer le paquetage « *livre* » créer la classe « AideEcrivain ».
Cette classe possède l'attribut bataille, qui sera initialisé par le constructeur.

Créer la méthode *visualiserForcesHumaines*. Cette méthode ajoute un à un les hommes du camp humain dans une variable locale `listeTriee` de type « LinkedList » que vous ordonnerez de cette manière : d'abord les héros dans l'ordre donné dans la liste du camp puis les hommes dans l'ordre donné dans la liste du camp.

Attention si la méthode dépasse 20 lignes vous devrez faire un refactoring comme vu en cours.

Contraintes :

- Cet exercice a pour but de vous exercer sur l'itérateur de liste. Vous ne devrez donc pas utiliser d'autres collections que celles mentionnées dans l'énoncé. Nous verrons d'autres méthodes plus efficaces dans les cours et TPs suivants.

- Vous ne devez parcourir la liste `listeCamp` qu'une seule fois avec un foreach. (cours numéro 3 diapo : **Le parcours des collections**).
- Vous devez utiliser un itérateur de liste pour parcourir et modifier la liste `listeTriee` excepté quand la liste est vide ou lors d'un ajout en fin de liste.

Tester votre code en décommettant les lignes ci-dessous dans la classe « `TestGestionProtagonistes` » à la fin du code précédent.

```
AideEcrivain aideEcrivain = new AideEcrivain(bataille);
System.out.println("visualisation des forces humaines :\n"
      + aideEcrivain.visualiserForcesHumaines());
```

III. Diagramme de classes