TD n°9 – Comparable et Collections

Comparaison d'objets

Une classe peut spécifier **un ordre naturel** en implémentant l'interface `Comparable<T>` existante dans l'API Java, et de la forme :

```
import java.util.*;

public interface Comparable<T> {
    int compareTo(T t);
}
```

La fonction *compareTo* renvoie :

- < 0 si `this` est *inférieur* à `T`
- == 0 si `this` est *égal* à `T`
- > 0 si `this` est *supérieur* à `T`

`T` doit être la classe spécifiant l'ordre. L'ordre est défini dans la classe `T` de manière explicite.

1. Récupérer les classes fournies en annexe et les compléter.
2. Consulter la classe Collections du package java.util (https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html) et regarder les méthodes disponibles. Quelle est la différence avec la classe Arrays vue en TD9 ?
3. Afficher les résultats de la méthode compareTo entre les 3 personnes.
4. Commenter la partie « implements Comparable<Personne> » dans la ligne d'en-tête de la classe Personne ainsi que la méthode compareTo. Que se passe-t-il ? Où se trouve l'erreur ? Expliquer en se référant à la documentation.
5. On souhaite maintenant pouvoir trier une collection de personnes suivant leur âge.

L'interface Comparator<T> existante également dans l'API permet de spécifier **un ordre externe** (temporaire)

```
import java.util.*;

public interface Comparator<T> {
    int compare(T t1, T t2);
}
```

La valeur de retour de compare suit les mêmes règles que compareTo, mais l'ordre externe entre les 2 objets en paramètre est un ordre valable juste à un moment donné (rien de naturel et évident).

Cette méthode s'utilise en paramètre de la méthode Collections.sort afin de trier temporairement une collection en fonction d'un objet Comparator<T>.

a. Ajouter un attribut int age. A quel niveau de la hiérarchie se trouve-t-il ?
b. Ajouter les accesseurs correspondants
c. Modifier le main afin de trier la liste avec ce nouveau comparateur en utilisant le concept de classe anonyme.

Comparer des Pokemons et des joueurs

On souhaite pouvoir comparer des joueurs entre eux en fonction de leur niveau ou des Pokemons qu'ils possèdent dans leur collection (de même il faut pouvoir classer les Pokemons entre eux et donc les comparer).

Le but est de déterminer a priori si des joueurs sont pressentis pour gagner afin de, pourquoi pas, calculer des côtes et lancer des paris sur les combats !

Un ordre naturel sur les Joueurs est défini par l'ordre alphabétique de leur nom.

Un ordre naturel sur les Pokemons est défini par l'ordre alphabétique de leur type. Pour 2 Pokemons de même type, l'ordre dépend de leur point de vie. Un Pokemon se situe avant un autre de même type s'il a plus de points de vie.

- **Définir les ordres précédents en implémentant l'interface `Comparable<T>`.**

Boite

Le but de cet exercice est de développer la classe Boite vue en IE dans sa version générique.

1. Ecrire le code des classes `Boite` et `Objet` non générique dont l'énoncé est rappelé dans le fichier IE2018 dans SupportCours, ainsi que le main représentant les boîtes de l'image dans une classe TestBoite.
2. Modifier le code de la classe `Boite` pour la rendre **générique**, de manière à ce que le type des éléments contenus dans une boîte soit variable.
    a. Les tableaux de types génériques génèrent un warning dans Eclipse.
3. Récupérer dans SupportCours les classes `Mail` et `Balle`, qui sont d'autres « objets » que l'on peut mettre dans une boîte.
   A quoi sert la variable *static* n utilisée dans ces classes ?
4. Modifier le main pour utiliser les différents objets et la classe générique.

Extensions

Pokemons

- Modifier l'interface IAttaque pour la rendre générique : IAttaque<T>.

Boite

- Modifier la classe `Boite`. Une boîte ne contient plus un tableau de boites de taille fixe, mais une collection de boîtes de taille variable. Mettre à jour votre code.
- Enrichir la classe `Boite` des méthodes :
    a. `ajouteObjet` qui permet d'ajouter un objet dans la boîte s'il n'y en a pas déjà un.
    b. `toString` qui permet de retourner la chaine de caractères représentant une boite en fonction de son contenu.
- Enrichir la classe `Boite` de la gestion des exceptions.