Université de Paris campus Grands Moulins
Complément Programmation orientée objet
M1 informatique **Année 2021-2022**

TP noté n° 1 du 15 novembre : Groupe INFO 2

Important : Vous devez rendre ce TP noté sur Moodle avant l'heure indiquée. Vous devez rendre une archive `tar` ou `zip` (pas de `rar`) contenant le code java, avec un ou des `main` pour pouvoir le tester.

Exercice 1

1. Écrivez dans une classe Exo1, une méthode

   ```
   public  static <K, V> Map<K, V> creeMap(List<K> l, Function<K, V> f)
   ```

   qui retourne un objet de type `Map<K, V>` tel que les clefs (de type `K`) sont les éléments de la liste `l` et la valeur associée à une clef k est l'image par la fonction `f` de k. Dit autrement, l'ensemble des couples de la map résultat sera $\{(k, f(k)) \mid k \in l\}$
2. Donnez une exemple d'utilisation de cette méthode dans un `main` en expliquant en commentaire et en français ce que vous calculez. Les types d'objets doivent être différents de ceux définis dans l'exercice 2.

   Pour pouvoir plus simplement définir une liste, vous pouvez utiliser ce genre de code :

   ```
   List<Integer> l1 = Arrays.asList(new Integer[]{1, 4, 12, 8, 9, 41});
   ```

   Il est bien sûr adaptable sur d'autres types d'objets.

Exercice 2

1. On crée une classe Intervalle qui représentera un intervalle d'entiers. Cette classe contiendra 2 attributs publics de type `int` et un constructeur public qui prend en argument deux entiers. On ne vérifiera pas que le début de l'intervalle est inférieur ou égal à celui de la fin, mais on supposera que cette propriété est toujours vérifiée.

   Ajoutez une méthode `toString()`.
2. Ajoutez une méthode

   ```
   public boolean tester(BiPredicate<Intervalle, Intervalle> p, Intervalle autre)
   ```

   qui testera si le couple d'intervalles `this, autre` obéit au prédicat `p`.

   vous testerez cette méthode dans un `Main` pour tester si un intervalle est inclus dans un autre au sens large. Par exemple, l'intervalle $[4, 8]$ est inclus dans $[1, 20]$, $[4, 8]$ est également inclus dans $[4, 20]$, mais $[4, 8]$ n'est inclus ni dans $[5, 6]$ ni dans $[5, 20]$.
3. Ajoutez une méthode

   ```
   public Intervalle  nouveau(Function<Intervalle, Intervalle> f)
   ```

   Attention : il ne faut appeler qu'une fois la méthode de `f` sinon, le test d'après ne marchera pas.

   Dans `Main`, vous créerez une classe anonyme pour la fonction et vous la mettrez dans une variable : cette fonction va à chaque appel, créer un intervalle de même longueur mais translaté de la manière suivante :
   — au premier appel, le début sera 0
   — à l'appel suivant, le début sera la fin de l'intervalle résultat précédent.
   Par exemple, si vous avez dans une liste ou un tableau les intervalles suivants :

   $$[14, 25],\ [10, 20],\ [3, 7],\ [12, 15] \text{ et } [6, 20].$$

   et que vous appelez la méthode `nouveau` avec la fonction en argument, vous aurez :

   $$[0, 11],\ [11, 21],\ [21, 25],\ [25, 28] \text{ et } [28, 42].$$

Testez sur une liste ou un tableau d'intervalles.

`Remarque:` On a ici une fonction avec effet de bord, qui appelée deux fois sur le même objet produit deux résultats différents. Il serait plus logique de faire une autre interface qui annonce la couleur, ou de programmer complètement diffèrement.