Travaux Dirigés n°5

Java Avancé

—M1—

Generics

Varargs, generics, wildcards...

`https://classroom.github.com/a/I2BMMkk6`

▸ Exercice 1. Maximums

On veut écrire une méthode tel que le code suivant fonctionne :

```
System.out.println(myMax(42,1664)); //1664
System.out.println(myMax(2018,86,13)); //2018
```

1. Quelle doit être la signature de la méthode ? (indice)
2. Comment faire pour qu'un appel à votre méthode sans aucun argument ne compile pas ?
3. Implémenter.
4. Modifier votre méthode pour qu'elle puisse fonctionner avec n'importe quel objet `Comparable`. Par exemple :

```
System.out.println(myMax(8.6,16.64)); //16.64
System.out.println(myMax("Denis", "Cornaz")); //Denis
System.out.println(myMax(8.6, "Denis")); //ne doit pas compiler !
```

5. Tester également avec les tests suivants MaxTest

▸ Exercice 2. Be wild !

```
private static void print(List<Object> list) {
  for(Object o:list)
    System.out.println(o);
}

public static void main(String[] args) {
  List<String> list=Arrays.asList("foo", "toto");
  print(list);
}
```

1. Pourquoi le code plus haut ne compile t-il pas ?
2. Comment le modifier pour qu'il compile (sachant qu'on veut pouvoir afficher également une liste d'entiers par la même méthode) ?
3. Ajouter une méthode statique prenant en argument une liste d'objets implémentant `CharSequence` et en affichant la longueur de chaque objet.

▸ Exercice 3.

```
public static List listLength(List list) {
   ArrayList length=new ArrayList();
   for(int i=0;i<list.size();i++) {
     CharSequence seq=(CharSequence)list.get(i);
     length.add(seq.length());
   }
   return length;
}
public static void main(String[] args) {
   List l=Arrays.asList("colonel", "reyel");
   System.out.println(listLength(l)); //affiche [7, 5]
}
```

1. Rendre le code ci-dessus générique en :
   (a) Utilisant une variable de type `T`.
   (b) La notation wildcard.

▸ Exercice 4. Fusion !

1. Ecrire une méthode de fusion de deux listes renvoyant une nouvelle liste avec alternativement un élément de chaque liste. Les deux listes devront être de la même taille, et en cas de liste vide, il ne faudra pas allouer d'objet inutilement. Le code suivant doit fonctionner :

```
List<String> l1= Arrays.asList("C", "rc");
List<StringBuilder> l2= Arrays.asList(new StringBuilder("a ma"), new StringBuilder("he!"));
List<? extends CharSequence> r1=fusion(l1,l2);
List<?> r2=fusion(l1,l2);
List<Integer> l3 = Arrays.asList(1,2);
List<Integer> l4 = Arrays.asList(10,20);
List<Integer> r3 = fusion(l3,l4);
List<?> r4 = fusion(l1,l3);
```

2. Tester également avec les tests suivants FusionTest
3. Quelle est la complexité de votre solution si une des deux listes est une `LinkedList` ? Modifier votre code pour ne parcourir qu'une seule fois chaque liste.

▸ Exercice 5. Mélange !

Le but de l'exercice est la création d'une méthode mélangeant les éléments d'une liste prise en argument.

1. Ecrire une méthode prenant en arguments une liste et deux entiers, et échangeant les éléments des la liste situés aux indices correspondant aux deux entiers.
2. On souhaite que la liste prise en argument de cette méthode soit paramétré par un wildcard. Est-ce possible ?
3. Réfléchir à un algorithme permettant de mélanger les éléments d'une liste prise en argument et l'écrire. Vous pouvez utiliser la classe `Random`.
4. Tester également avec les tests suivants ShuffleTest
5. Quelle est la complexité de votre implémentation ? Et dans le cas de l'utilisation d'une `LinkedList` ? Peut-on mieux faire ? Faire mieux en s'inspirant de la méthode shuffle de `Collections` (`http://www.docjar.com/docs/api/java/util/Collections.html`).