Travaux Dirigés n°4

Java Avancé

—M1—

Iterables

Lists, anonymes, iterateurs, iterable...

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

- Pour ce TD, créez un nouveau projet Maven dans votre repertoire `ja` (voir td00) avec les `groupid` et `artifactId` suivants.
  - groupId `fr.dauphine.ja.`*`nomprenom`*`.iterables`
  - artifactId `iterables`
- N'oubliez pas de commiter régulièrement et de charger les dernières versions de votre dépot avec git push !

▸ Exercice 1. De A à Z

On cherche à écrire une classe `Panel` qui représentera un intervalle de nombres entiers. Il faudra faire en sorte que `Panel` soit `Iterable`, c'est-à-dire qu'il soit possible de parcourir toutes les valeurs de l'intervalle avec une boucle for.

1. On veut que le code suivant fonctionne :

```
1 Iterator<Integer> it=panel1(1,5);
2 for(;it.hasNext();)
3    System.out.println(it.next()); // affiche 1 2 3 4 5
```

   Pour que ce code fonctionne, la fonction statique `panel1()` doit retourner un objet issu de la classe `Iterator<>` correctement implémenté. Consultez la documentation de `Iterator` et implémentez la fonction `panel1()` pour faire en sorte que le code précédent fonctionne. Attention, vous ne devez pas utiliser de liste pour stocker les éléments !

2. On souhaite simplifier la code précédent en faisant usage des *classes anonymes*. Consultez la documentation des classes anonymes ici, puis modifiez le code de la fonction `panel1()` pour retourner un itérateur implémenté à l'aide d'une classe anonyme.

3. On veut maintenant faciliter le parcours de l'intervalle en supportant la syntaxe *foreach* de Java comme dans l'exemple suivant :

```java
for(int i:panel2(1,5))
  System.out.println(i); // affiche 1 2 3 4 5
```

Quelle interface doit-on implémenter pour pouvoir itérer de la sorte ? Implémentez la fonction statique `panel2()` pour faire en sorte que cela soit possible. (On a le droit de réutiliser du code !)

4. On veut aller plus loin et faire en sorte que `Panel` puisse être traité comme n'importe quel objet du type `List`. Regardez la documentation de `AbstractList`. À quel problème répond cette classe ? Quelles méthodes (abstraites) doivent être implémentées ? Pourquoi ces méthodes ?
5. En utilisant `AbstractList` écrivez une méthode `panel()` qui retourne un objet du type `List` qui « contient » tous les éléments de l'intervalle. **Attention**, on ne veut pas **stocker réellement** tous les éléments de la liste, mais seulement les retrouver à l'aide des deux bornes données en argument).

```java
List<Integer> l = panel(3,6);
for(int i:l) {
  System.out.println(i);
  //affiche 3 4 5 6
}
System.out.println(l.get(1)); //affiche 4
```

6. Testez avec le fichier PanelTest

▸ Exercice 2. En double

On cherche ici à écrire dans une classe `Mult` quelques méthodes statiques faisant des opérations.

1. Écrire une méthode `mult` prenant en paramètres un entier et une liste d'entiers et renvoyant une nouvelle liste d'entiers où toutes les valeurs sont multipliées par le premier argument (coefficient).
2. Modifier votre implémentation afin ne pas pas avoir à allouer une nouvelle liste, mais en utilisant le principe d'une vue (voir la doc de `AbstractList`). Le faire sous forme d'une classe anonyme.
3. Quel est le problème avec le code suivant ?

```java
ArrayList<Integer> al = new ArrayList<>();
for (int i = 0; i < 1000000; i++) {
  al.add(i);
}
long t0 = System.nanoTime();
List<Integer> ret = Mult.mult(2, al);
long sum=0;
for(int val : ret) {
  sum+=val/2;
}
System.out.println((System.nanoTime() - t0));
```

```
LinkedList<Integer> ll = new LinkedList<>();
for (int i = 0; i < 1000000; i++) {
  ll.add(i);
}
t0 = System.nanoTime();
sum=0;
ret = Mult.mult(2, ll);
for(int val : ret) {
  sum+=val/2;
}
System.out.println((System.nanoTime() - t0));
```

4. Corriger le problème. (Regarder du côté de `RandomAccess`, `AbstractList`, `AbstractSequentialList`)
5. Testez avec ces tests MultTest