Université Paris Cité

Compléments en Programmation Orientée Objet
TP noté n° 1, groupe Info 5 : Inversion de Dépendance

Exercice 1 : Utilisation d'une bibliothèque pratiquant l'inversion de dépendance

De nombreux programmes savent lancer des traitements sur des listes, ensembles, ou plus généralement des collections ou des itérables implémentant l'interface `java.util.Iterable` :

```
package java.util;

public interface Iterable<E> {
    // peu importe (déjà  implémenté par défaut)
    default void forEach(Consumer<? super T> action) { ... }

    // retourne un itérateur sur des éléments de type T
    Iterator<T> iterator();

    // peu importe (déjà  implémenté par défaut)
    default Spliterator<T> spliterator() { ... }
}
```

L'interface `Iterator<E>` ressemble à ceci :

```
package java.util;

public interface Iterator<E> {
    // peu importe (déjà  implémenté par défaut)
    default void forEachRemaining(Consumer<? super T> action) { ... }

    // retourne true si l'itération a encore des éléments
    boolean hasNext();

    // retourne l'élément suivant de l'itération
    E next();

    // peu importe (opération optionnelle déjà  implémenté par défaut)
    default void remove() { ... }
}
```

Les classes implémentant l'interface `Iterable` peuvent être itérées dans une boucle *for-each* :

```
public static void afficheIterable(Iterable<?> it) {
    for (var x: it) System.out.println(x);
}
```

Exemple d'utilisation :

```
afficheIterable(List.of(1,2,3));
```

affiche :

```
1
2
3
```

À faire : adaptez la classe `java.util.Scanner` afin de créer une implémentation d'`Iterable<String>` permettant de tester les programmes utilisant les itérables de façon intéractive. Ainsi, si, vous testez la fonction `afficheIterable` ci-dessus, ça peut donner ça :
Appel (par exemple dans votre `main` ou dans l'environnement `shell`) :

```
try (var scanner = new Scanner(System.in)) {
    afficheIterable(new Scanner2Iterable(scanner, 3));
}
```

(Les paramètres de l'adaptateur sont le scanner utilisé et le nombre d'entrées qui seront itérées.)
Affichage :

```
> Bonjour
Bonjour
> le
le
> monde !
monde !
```

(l'adapteur affiche "`> `" à chaque fois qu'une entrée clavier sur l'entrée standard est attentdue par le `Scanner`).
Aide : commencez par écrire un adaptateur de `Scanner` vers `Iterator<String>`, puis utilisez le dans `Scanner2Iterable`.