Übung 13: Java Collections

Aufgabe 1: Listen

a) Implementieren Sie in einer neuen leeren Klasse `Util` die folgende statische Methode:

```
public static <E> void reverse1(List<E> list)
```

Die Methode verwendet Generics (Typvariable `E`) und soll mit beliebigen Objekten/Elementen `E` funktionieren. Diese Methode kehrt die Reihenfolge aller Elemente in der Liste `list` um. Die übergebene Liste wird verändert und deshalb **keine neue** Liste angelegt.

b) Implementieren Sie zusätzlich die folgende statische Methode:

```
public static <E> List<E> reverse2(List<E> list)
```

Die Methode erstellt eine **neue (!)** Liste, die alle Elemente der übergegebenen Liste `list` enthält, allerdings in der umgekehrten Reihenfolge. Die zurückgegebene Liste soll **exakt den gleichen** Datentyp haben wie die übergebene Liste.

Hinweis: Sie wissen nicht, ob hinter der übergegebenen Liste z.B. eine `ArrayList` oder eine `LinkedList` steckt. Dennoch können Sie eine neue, leere Instanz der übergebenen Liste `list` erzeugen, siehe Vorlesung.

c) Testen Sie Ihre Methoden aus Aufgabe b) und c) mit der vorgegebenen JUnit-Testklasse `TestUtil`.

Aufgabe 2: Indexverstellen - Mengen, assoziative Speicher, Arrays

Gegeben ist ein Array von Listen. Jeder Eintrag des Arrays entspricht der Seite eines Buches und speichert eine Liste der Begriffe/Schlüsselwörter, die für diese Seite zu indizieren sind. Beispiel:

Buchseite 0 / Array-Index 0: → Liste: (leer)
Buchseite 1 / Array-Index 1: → Liste:(Java, Bali, Sulawesi)
Buchseite 1 / Array-Index 2: → Liste:(Bali, Sumatra, Lombok)

a) Implementieren Sie die folgende statische Methode in der vorgegebenen Klasse `Index`:

```
public static Map<String, SortedSet<Integer>> makeIndex(List<String>[] keywords)
```

Die Methode erhält ein Array von Listen mit den zu indizierenden Schlüsselwörtern für jede Buchseite und gibt einen Index zurück. Der zurückgegebene Index entspricht einer Map: Als *Key* dient das Schlüsselwort, der *Value* ist eine Menge[1] von aufsteigend sortierten Seitenzahlen auf denen das Schlüsselwort vorkommt.
Ferner soll in der Map eine alphabetische Ordnung für die Keys (also die Schlüsselwörter) verwendet werden. Welche konkrete Map-Klasse ist somit geeignet?

b) Vervollständigen Sie die Implementierung der vorgegebenen Methode `toString(…)`, die den Inhalt der Map, also alle Key-Value-Paare, als String zurückgibt.
(ggfs. https://docs.oracle.com/javase/8/docs/api).

c) Testen Sie die Klasse mit der vorgegebenen JUnit-Testklasse `TestIndex`.

[1] Keine Duplikate von Seitenzahlen.