December 8, 2020

Task 9

The ***Programmers.tsv*** file contains information on programmers: each line consists of the name of a programming language and the names of programmers knowing this language with the *TAB character* as a separator. For example

```
Groovy  Z   Y   X   D
Java    V   B   C   D   A   Z
C++ G   J   H
C#  P   S   Q   V   D
Scala   A   D   A
```

Create a class **ProgLang** defining

- constructor **ProgLang(String fileName)** which reads data from the file with the given name;
- method **getLangsMap()** returning a map with the names of programming languages as keys and collections of programmers (their names) knowing the given language as values;
- method **getProgsMap()** returning a map with the names of programmers as keys and collections of the names of the programming languages the given programmer knows;
- method **getLangsMapSortedByNumOfProgs()** returning a map with the names of programming languages as keys and collections of programmers knowing a given language as values. The map should be ordered according to the number of programmers knowing a given language; for equal numbers the order is alphabetical by the name of the language;
- metodę **getProgsMapSortedByNumOfLangs()** returning a map with the names of programmers as keys and collections of names of langauges they know as values. The map should be ordered in descending order according to the number of languages known by a given programmer; for equal numbers the order is alphabetical by the name of the progarmmer;
- method **getProgsMapForNumOfLangsGreaterThan(int n)** returning a map (programmer→collecton of languages) but including only those programmers who know more than `n` languages;
- method **sorted(...)** for which the first argument is an *arbitrary* map and the second a lambda expression. The method returns a sorted version of the passed map, where the lambda specifies how to compare its entries.
- method **filtered(...)** for which the first argument is an *arbitrary* map and the second a lambda expression returning a **boolean**. The method returns the passed map, but only with those entries for which a condition specified by the lambda is **true**.

Note: Methods **sorted** and **filtered** should be used in other methods of the class. They should be written in a way allowing the user to sort or filter *arbitrary* maps (with appropriate lambdas).

The following **main** function

download *SProgrammers.java*

```
// imports...

public class SProgrammers {
    public static void main(String[] args) {
          // for convenience only...
        BiConsumer<String,Collection<String>> pr =
                (k,v)->System.out.println(k + " = " + v);

        ProgLang pl = null;
        try {
            pl = new ProgLang("SProgrammers.tsv");
        } catch (Exception exc) {
            System.out.println("No input file? " + exc);
            return;
        }

        System.out.println("@1 Map of languages:");
        pl.getLangsMap().forEach(pr);

        System.out.println("@2 Map of programmers:");
        pl.getProgsMap().forEach(pr);

        System.out.println("@3 Languages sorted by " +
                           " number of programmers:");
        pl.getLangsMapSortedByNumOfProgs().forEach(pr);

        System.out.println("@4 Programmers sorted by " +
                           "number of languages:");
        pl.getProgsMapSortedByNumOfLangs().forEach(pr);

        System.out.println("@5 Original map of languages " +
                           "is not modified:");
        pl.getLangsMap().forEach(pr);

        System.out.println("@6 Original map of programmer" +
                           "s is not modified:");
        pl.getProgsMap().forEach(pr);

        System.out.println("@7 Map or programmers knowing" +
                           " more than 1 language:");
        pl.getProgsLangsMoreThan(1).forEach(pr);
```

```
            System.out.println("@8 Original map of programmer" +
                               "s is not modified:");
            pl.getProgsMap().forEach(pr);
        }
    }
```

with the data file as above, should print

```
@1 Map of languages:
Groovy = [Z, Y, X, D]
Java = [V, B, C, D, A, Z]
C++ = [G, J, H]
C# = [P, S, Q, V, D]
Scala = [A, D]
@2 Map of programmers:
Z = [Groovy, Java]
Y = [Groovy]
X = [Groovy]
D = [Groovy, Java, C#, Scala]
V = [Java, C#]
B = [Java]
C = [Java]
A = [Java, Scala]
G = [C++]
J = [C++]
H = [C++]
P = [C#]
S = [C#]
Q = [C#]
@3 Languages sorted by  number of programmers:
Java = [V, B, C, D, A, Z]
C# = [P, S, Q, V, D]
Groovy = [Z, Y, X, D]
C++ = [G, J, H]
Scala = [A, D]
@4 Programmers sorted by number of languages:
D = [Groovy, Java, C#, Scala]
A = [Java, Scala]
V = [Java, C#]
Z = [Groovy, Java]
B = [Java]
C = [Java]
G = [C++]
H = [C++]
J = [C++]
```

```
P = [C#]
Q = [C#]
S = [C#]
X = [Groovy]
Y = [Groovy]
@5 Original map of languages is not modified:
Groovy = [Z, Y, X, D]
Java = [V, B, C, D, A, Z]
C++ = [G, J, H]
C# = [P, S, Q, V, D]
Scala = [A, D]
@6 Original map of programmers is not modified:
Z = [Groovy, Java]
Y = [Groovy]
X = [Groovy]
D = [Groovy, Java, C#, Scala]
V = [Java, C#]
B = [Java]
C = [Java]
A = [Java, Scala]
G = [C++]
J = [C++]
H = [C++]
P = [C#]
S = [C#]
Q = [C#]
@7 Map or programmers knowing more than 1 language:
Z = [Groovy, Java]
D = [Groovy, Java, C#, Scala]
V = [Java, C#]
A = [Java, Scala]
@8 Original map of programmers is not modified:
Z = [Groovy, Java]
Y = [Groovy]
X = [Groovy]
D = [Groovy, Java, C#, Scala]
V = [Java, C#]
B = [Java]
C = [Java]
A = [Java, Scala]
G = [C++]
J = [C++]
H = [C++]
P = [C#]
S = [C#]
```

```
Q = [C#]
```

NOTE: Do not use raw types in the **ProgLang** class.

Deadline: Dec 25 (inclusive)
