Practical sessions

Important information

Practical sessions (called TP below) are carried out in pairs, and in Java 1.7 [1]

During the TP1, you should first study and understand the TPforALL program, which:

1. reads a (directed or not, weighted or not) graph from a given file following the rules on the next page, and
2. stores it according to a given representation (as a matrix or as an adjacency list).
3. computes (as an example) the appropriate degrees in all the variants of the graph (see next page).

This program is a mandatory common working basis for all of you during the practical sessions. You may modify this program, but:

- only in order to add new functionalities
- you have to keep all the classes and their attributes, but you may add attributes
- you have to add to each class the methods you are asked to write and which are specific to the class
- you have to modify main() so that it directly calls the methods solving the exercises (only one method called for each exercise; see the example hereafter).
- you may add new classes according to the needs, but without unnecessary inflation
- you may not add new packages or change the default package name
- you should add comments, clear displays and (depending on your modifications of the program) precise compilation and execution instructions.

The grades will take into account the respect of these rules.

Example. Assume that Exercise 1 asks you to compute the number of paths of length 2 in an unweighted directed graph provided in a file MyDirectedGraph.txt and represented by adjacency lists. You need to:

1. in the main() method of the Main4A.java class, in the block that concerns the directed and unweighted graphs represented by adjacency lists, replace the existing example with: a line containing System.out.println("Response to exercise 1"), and another one containing a single call to GrapheL.Exercice1(...).
2. the Exercise1(...) method of the GraphL4A class - which you will write - must compute the answer to Exercise 1 and must provide all the necessary displays for the user who wants to know the answer to the question. This method may use other classes and methods (present in the supplied program, or written by you).

[1] Java version 1.7 prevents you from using newer features of Java, that would allow you to write code whose content and complexity you do not understand. You can use other versions of Java to write your code, but make sure you regularly check that compilation with version 1.7 works. If version 1.7 disappears from CIE computers (its removal is planned), use version 1.8 for compilation, but be careful not to use the new features introduced in version 8, in particular API Stream and lambda expressions. Write your own code.

Note 1. For the graded TP, you may be asked to write a method X that calls one or several methods Y already requested in previous practical sheets. In this case:

- the methods Y must be present (and correct) in the program you provide. They will only be *called* by the method X (and not *contained* in X). This implies that a TP sheet that has not been finished during the TP session must be finished afterwards, as a homework.
- only the X method will get a grade.

Note 2. Your program must include *only* classes, methods, comments etc. that are either from the TPforALL package, or your own work. Any similarity with other programs (from the class, or from Internet) will result in a grade of 0.

The file providing the input graph

The vertices are numbered from 1 to n, and you should not change this numbering for the user (you may change it in your own classes and methods, if you wish, but only there). We assume there are no loops, nor multiple edges/arcs.

The file has an arbitrary name, with extension .txt. It follows precise rules, that you may observe in the two examples below:

```
27 undirected weighted
2 : 3, 7, 10, 4 // 4.7, 12.5, -8.7, 5.3
10 : 12, 1, 2 // 1.3, 12, -8.7
1 : 27, 10, 4, 26, 7, 3 // -2, 12, 4.3, 18, -9.2, 100.12
...
```

```
27 undirected unweighted
2 : 3, 7, 10, 4
10 : 12, 1, 2
1 : 27, 10, 4, 26, 7, 3
...
```

More precisely:

- first line: the number n of vertices (here, 27), one blank, the word "undirected" or "directed", one blank, the word "unweighted" or "weighted"
- second line: an arbitrary vertex, blank, ":", blank, the unsorted list of its neighbors (if undirected) or of its successors (if directed) with a blank after each comma. If the option "weighted" appears, then follow: a blank, two "/" (no blank between them), a blank, and the list of the weights, in the appropriate order (first weight for the first arc and so on), with a blank after each comma.
- third line: an arbitrary vertex, blank, ":", blank, the unsorted list of its neighbors (if undirected) or of its successors (if directed) with a blank after each comma. As above, if weighted.
- fourth line: an arbitrary vertex, blank, ":", blank, the unsorted list of its neighbors (if undirected) or of its successors (if directed) with a blank after each comma. As above, if weighted.
- ... and so on.

The file is assumed to be correct: it contains all the vertices, in an arbitrary order, and the information is consistent with the definition of a weighted/unweighted directed/undirected graph, according to the

indications on the first line. However, each line may possibly contain some blanks after the last visible character.

The program TPforALL provides, as an example, a solution to the following exercise:

Exercise 1. Write two methods which:

1. (first method) in an undirected graph G, computes the degree of each vertex of G, stores all the degrees in a table *degree* and displays them on the screen.
2. (second method) in a directed graph G, computes the input and output degree of each vertex if G, stores them in two tables *indegree* and *outdegree* respectively, and displays them on the screen.