TimeTable Project

The goal of the project is to apply most of the notions you learnt during the course. The application is to create a first version of a software that assists the administration with the generation of a time table for the various formations.

The goal is to write a code for answering the following questions :

1. Print on the console how many different instructors participate in teaching (as lecturer for courses, or lab assistant).
2. Print on the console how many subjects there are (a course that has a lecture, a practice session and a lab session counts for only one subject).
3. Print in a file the list of all instructors that give lectures in alphabetical order.
4. Print in a file the list of all instructors in order of the number of classes they teach (i.e. if someone teaches one lecture, two exercise sessions and a lab session, she teaches 4 classes).
5. Find a time table for semester for a program and write it in a csv file. For instance, you should be able to write the timetable for the first semster of BFA2 in the file `bfa2-sem1.csv`.

In the following, we provide some details about the problem and some help about how to read and write files. If questions 1–4 are well done, you will have at least a passing grade. The quality of your answer for question 5 will make the difference between a good and a very good grade.

Details about the data we provide

You will be provided a two files. The first contains the courses of the first semester of BFA1 and BFA2. This is to test your work on an easy instance. The second file contains most of the courses from the MIDO department. This may appear as a most challenging test, though it should be as easy to solve. The files are "csv" files (for Comma-separated values). It is a simple text format that can be directly read of used by a spreadsheet (Excel or others). The input file is formatted as follows :

— each class is described on a single line
— the fields are ordered and separated with a semi column `";"`

1. type of class :

   `CM` lecture

   `TD` pratice session

   `TP` lab session
2. name of the formation (for instance `BFA1`, `BFA2`, etc.)
3. semester number (either `1` or `2`)
4. name of the course
5. teacher of the course
6. an optional group number for the practice and lab session.

As an example, the following line expresses that BFA1 has a lecture in the first semester called "Theorie de la croissance" given by Martine Carre-Tallon.
`CM;BFA1;1;Theorie de la croissance;MARTINE CARRE-TALLON`

Details about the timetable problem

We will make the following assumptions :
- the classes must be given between Monday and Friday
- the time slots are the ones used by Dauphine.
- there are 3 types of classes : lectures, pratice sessions, and lab sessions.
- we do not consider the problem of allocating a class to a classroom (we can simply assume that there are enough rooms) and that there is no problem with the capacity of the classroom (all classrooms are sufficiently large).
- we only want to make the global timetable (we do not want to generate the week to week timetable).
- of course, there are constraints that a timetable must follow :
  - a teacher can teach only one class at a given timeslot !
  - a practice session or a lab session cannot be scheduled at the same timeslot as a lecture for the same formation
  - two lab sessions, or two practice sessions, or a practice session and a lab session can be scheduled at the same time slot only if they are for different groups.
- we could also handle additional constraints (the availability of the teachers may be limited ; some time slots must be left open for the students (lunch time or one afternoon a week). But we will not consider this possibility in this project.

Later in this document, I provide you an example of a Javacode for reading a file. To output a timetable, you can also produce a file ".csv" file (it may be easier to read in a spreadsheet than on the console. You will also find an example for writing a file.

Submission

The project should be done in pairs. Many architectures are possible and you should choose one that uses well the concepts of an object oriented langage. We advise to write the code and comments in English (at least, do not use any accents in the code).
what needs to be submitted : If the name of the authors of the project are Astérix and Obélix, store in a directory named `Asterix_Obelix`
- all the source files (i.e. all files with extension `.java`). Please comment your code so that I can understand what the methods do.
- A pdf file containing a brief explanation/description of your algorithm to make the timetable (this should not exceed one page).

The directory should then be compressed in a `zip` file. Obélix and Astérix would then submit the file `Asterix_Obelix.zip`.
submission : **by email** to `stephane.airiau@dauphine.fr`
deadline : **Friday October 10th**

Example Code to read a file

You do not need to understand all the technical details to use this code (of course, you can read in the lecture notes more details about input-output and handling exceptions. Very briefly :

- The block `try...catch` is used to handle exceptions. Whenever you access a file or communicate through the network, something may go wrong (the file is not present, you cannot write on the disk as it is full, etc). To prevent the application from closing, Java provides a mechanism to handle some failures. In this code, if a failure occurs in the block `try{...}`, the execution of the application is suspended and the block `catch` is executed.
- The object of type `FileReader` handles a file stored on your drive. The object of type `BufferedReader` is just a tool to help one read the file.

```
public static void listSchoolNames(String filename){
  try{
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line = reader.readLine();
    while (line != null){
      String[] chunks = line.split(";");
      int id= Integer.parseInt(chunks[0]);
      int capacity = Integer.parseInt(chunks[1]);
      String name =chunks[3];
      String degreeName = chunks[4];
      String mention = chunks[5];
      System.out.println("["+id+"]␣" + name);
      line = reader.readLine();
    }
    reader.close();
  }
  catch(IOException e){
    e.printStackTrace();
  }
}
```

Un exemple de code est fourni dans le fichier `ExampleBFA.java`. L'exemple donne aussi la manière pour écrire dans un fichier et communique via la console.