Voortgezet Programmeren (FEB23007-11)

6. Exercise

Deadline for submission: **2012-02-17** 23:59 CET

Instructions

Include in each source file (in class documentation, @author) your names and student numbers. Submit the exercise as a zip file containing _only_ the source files in root of the zip. Submit via blackboard. Note that incorrectly submitted or non-compiling exercises are automatically awarded 0 points. Remember to document your code with Javadoc-annotations.

Exercise

In this exercise we re-write an earlier exercise using Java Collections Framework and apply most of the theoretical concepts learnt during the course. Note that you should not need to write too much code for completing the exercise if you use the Collections in a proper manner.

Make a class `Student`. Students have names and student numbers. Add an accessor method for the name. Override `Student`'s `equals` and `hashCode` to define students to be equal when their student numbers are equal. Also override `toString` with a suitable implementation. Implement `Comparable<Student>` in `Student` so, that students are naturally ordered according to their student numbers (in an ascending order).

Make a class `NameComparator` implementing `Comparator<Student>`, that defines another order of the students as a legicographical order according to their names.

Make a class `Course`. It should contain information about the year, block, code, name, and students registered to the course. Add an accessor method for the name. Override `equals` to define two courses to be the same if their codes, years, and blocks are equal. Also override `toString` so, that the `String` representation of a `Course` is its information followed by registered students ordered as defined by the natural order in `Student` (preferably separated by '\n's). The course should contain a method for registering `Student`s to it. Students should be added to the course only if there are no other students already registered with the same student number. Include a method `Set<Student> getRegisteredStudents()` that gives out the students registered to the course. Implement `Comparable<Course>` to define the natural ordering of courses so, that earlier years courses come first, and within the same year the courses are ordered according to their blocks (in an ascending order), and within the same year and block in a lexicographical order of their codes.

Make a class Program. It should contain name of the program and the courses within. Include a method for registering a `Course` to the program. Courses should not be added to the program if the same course already exists. Override `toString` so, that a program's `String` representation is name of the program followed by `String` representations of it's courses (preferably separated by '\n's). Include a method `Set<Student> getStudents()` that returns unique students registered to any of the courses within the program.

Make a class Main with a main-method to run the program. Construct the following structure (programs, courses, students):

- Econometrics

  - Microeconomics (2011, block 2)
    - Tommi 111111
    - Alex 222222
    - Frederik 333333
  - Programming-2 (2010, block 1)
    - Alex 222222
    - Frederik 333333
  - Programming-3 (2010, block 1)
    - Alex 222222
    - Frederik 333333
- Business economics
  - Microeconomics (2011, block 2)
    - Tommi 111111
    - Charlie 444444
  - Accounting (2010, block 3)
    - Charlie 444444
    - Philip 555555

You can change these courses/programs/student names and numbers. Note that the Microeconomics is the same course in both programs. Now, using the functionality you implemented before, put the 2 programs in a List and print out the program information so, that for each program, the program's information gets printed out and then the contained courses in the natural order of courses (earlier first, etc). For each course, the course information should be printed out followed by the enrolled students in a natural ordering of students (lower student numbers first). In addition, for each program, print out the unique students taking courses in the program in a lexicographical order.

Hints

- Read with care the tutorial on Java Collections Framework.
- Before starting, read the standard API documentation of classes `Set`, `HashSet`, `TreeSet`, `Comparable`, `Comparator`, and `Collections`.
- The easiest way to make sure everything works as intended is to make the classes in a "test first" manner: write first the method signatures and the unit tests, and only afterwards the proper method implementations. When the tests pass, the methods probably work. In such a way you can leave writing of the Main until the end and it should be fairly trivial as you're sure that the other functionality is implemented in a working manner.