CentraleSupélec

Object Oriented Software Engineering

Practical 04

Paolo Ballarini

Learning outcomes:

- How to sort numeric and String arrays and collections through method `sort()`
- How to sort collections of a generic type `T` w.r.t. to **natural order** induced by a comparator
- how to define **natural orders** for a user-defined class through `Comparable` and `Comparator`

Exercise 1. More practice with various Collections

Q1) Write a JAVA class called `MyArrayList<E>` that extends `ArrayList<E>` by adding a `removeDuplicate` method which, when executed, removes any duplicated element from the arraylist. For example, if the `removeDuplicate` method is executed on a list of integers like this one `[2, 2, 2, 2, 5, 5, 8, 9, 9, 2, 9, 8, 4, 1]` the resulting list will be `[2, 5, 8, 9, 4, 1]`.

Q2) Test your new class through a simple test code using the following list `[2, 2, 2, 2, 5, 5, 8, 9, 9, 2, 9, 8, 4, 1]`

Exercise 2. Reading/writing from/to files

Q1) Write a program that counts the number of occurrences of each word contained in the file `words.txt` and displays the results on screen. For this you may want to use a `Map` structure (of which kind? mapping what to what?).

Q2) Extend your program adding a method that allows you to output the words count result in a file called `wordscount.txt`. Remember that I/O on files require handling of exceptions.

Q3) Write a modified version of the above words counting program that counts the words contained in the file `words.txt` line by line. The program at first counts the word for line 1 and adds a line to the file `wordscount.txt`, then counts the words for line 2 and adds a second line to the file, etc. Each line of the file will contain the world count for the corresponding line in the `words.txt` file.

Exercise 3. Adding exceptions

In this excercise you are required add exception handling to the course marking program of Tutorial 3.

Q1) Extend the program you wrote as a solution of Exercise 2 of Tutorial 3 by including a control on the weights used for computing the final mark of a given module (starting from the mark splits). The program should control that the weights associated to a module sum up to 1. Thus if WFE (Weight for Final Exam mark), WME (Weight for Midterm Exam mark) and WPW (Weight for Practical Work mark) are the three weights of a module than the program should always check that WFW + WME + WPW = 1 and if not it should rise an exception (thus you should define your own exception).