Assignment No. 2

The purpose of this assignment is to familiarize you with the Java Collections Framework, to introduce you to Java I/O programming and to provide you with an opportunity to practice your Object Oriented Programming skills.

Large documents or reference books usually have an *index* that lists the terms and topics discussed in the text document along with the pages they appear on. Here is an example for an index:

Acetone, 381
Acid rain, 95
Base-pairs, 1197
Dacron, 4, 856, 1245
Electron donors, 676
Elimination, 191, 215-217,227, 267

The index keys are sorted alphabetically and each key is followed by a list of page numbers.

Many programs for automating index preparation involve the following three steps:

1. The user marks words or phrases to be indexed in the document source.
2. The tool reads the document source and generates a *raw index file*. Each raw entry in this file consists of two arguments: an index key (the word or phrase marked in the document) and the number of the page that contains the key.
3. The tool processes the raw index file. All index keys are sorted alphabetically in ascending order. Sorting is case-sensitive, that is upper-case letters come before lower-case letters. Page numbers for the same index key are grouped into a list. More than two successive page numbers may be abbreviated as a range, for instance 259,260,261 is replaced by 259-261.

Your assignment is to write a Java program for processing a raw index file and generating an index (3rd stage). The program will get as arguments two file names. The first argument will stand for the name of an input raw index file and the second argument for the output index file.

The input raw index file will contain index entities in the following format:

Dacron ; 4
Acid rain ; 95
Elimination ; 191
Elimination ; 215
Elimination ; 216
Elimination ; 217
Elimination ; 227
Elimination ; 267
Acetone ; 381
Electron donors ; 676
Dacron ; 856
Base-pairs ; 1197
Dacron ; 1245

(in each raw a semicolon separates the index key and the page number).

The program will process the raw index file and write to the output file the generated index as described above, where page ranges are optional (so instead of “Elimination, 191, 215-217,227, 267” it can output “Elimination, 191, 215,216,217,227, 267”).

Your program needs to read and write data to files. For this purpose, you can use the `java.io` package. An explanation about this package can be found in the JDK documentation, in Sun Java tutorials or in any other Java book. For your convenient, here is an example for a simple program that reads a raw index file and prints its index entities into another file:

```
package il.ac.tau.cs.oranit.exercise2;

import java.io.*;

/**
 * This is an example to start the 2nd exercise. The program reads a raw index
 * file and prints it to another file, using a different format.
 */
public class IndexExample {
    final static char DELIMITER = ';';

    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Please supply two file names");
            return;
        }

        try {
            BufferedReader in = new BufferedReader(new FileReader(args[0]));
            PrintWriter out = new PrintWriter(new FileWriter(args[1]));

            String line;
            while ((line = in.readLine()) != null) {
                int delimiterIndex = line.indexOf(DELIMITER);
                if (delimiterIndex < 0) {
                       continue;
                }

                String argument1 = line.substring(0,delimiterIndex-1);
                String argument2 = line.substring(delimiterIndex+1, line.length());
                String key = (argument1).trim();
                Integer pageNumber = new Integer(argument2.trim());

                out.println("keyword = " + key + " ; page no. = " + pageNumber);
            }

            in.close();
            out.close();
        } catch (IOException e) {
            // We don't deal with exceptions
            // at this phase of the course
        }
    }
}
```

In your code, take care to carefully define the contract and write your program using Object Oriented Programming principles. Use a TreeMap object to represent the collection of index entries and TreeSet objects to represent the page sets.

Submission Guidelines:

Submit your program via the white-board. Make sure that a comment at the top includes your full name, your login name, and your ID number.