ECS510U Algorithms and Data Structures in an Object-Oriented Framework

Exercise Sheet 10: Lists of Sets and Sets of Lists

This is an exercise to support the "Java Collections Framework" section

1) Write a class to define a `Comparator<Integer>` object which compares two integers according to their closeness to an integer given by the constructor of the `Comparator<Integer>`. For example, if the integer given by the constructor is `25`, then `22` should be greater than `30` according to the `Comparator<Integer>` object because `22` is closer to `25` (`3` difference) than `30` (`5` difference). Demonstrate using this to sort an `ArrayList<Integer>` using Java's built-in `sort` method, and to construct a `TreeSet<Integer>` ordered by a `Comparator` of this type.

2) Write two programs, each of which reads in some words all on one line, stores them in an object referred to by a variable of type `Set<String>` and then prints them out, each word on a separate line, using a for-each loop. In one program the object should be of actual type `HashSet<String>`, in the other of actual type `TreeSet<String>`.

3) Write a program that reads several lines of text consisting only of integers. The integers in each line should be stored in an object of type `List<Integer>`, and this representation of each line should be stored in an object of type `HashSet<List<Integer>>`. Then the program should print out the contents of this set.

4) Modify your answer to question 3 so that it can read any lines of text, but only stores what represents integers. For example, the line `"37 ok 150 43 end"` would lead to `37`, `150` and `43` being stored.

5) Write a modified version of your answer to question 3 or 4 where each line of integers is stored in an object of type `Set<Integer>`, and the representation of each line is stored in an object of type `List<Set<Integer>>`. Compare what is printed at the end with what is printed in the answers to question 3 and 4 when some integers are duplicated on a line and when some lines are identical to other lines.

6) Write your own static method that will take a `HashSet<List<Integer>>` object and return from it the `List<Integer>` object within it that is the longest. If the `HashSet<List<Integer>>` is empty, it should return an empty `List<Integer>` object. Use this method with the `HashSet<List<Integer>>` object created in the answer to question 3.

7) Write code for a `Comparator` object that orders `List<Integer>` objects in order of their length. Then write code for a `Comparator` object that orders `List<Integer>` objects in the order of the sum of their integers (the sum is all the integers in the list added together). Use these `Comparator` objects and the two-argument method `max` from Java's class `Collections` to find the longest list and the list which adds up to the most from the set of lists from a `HashSet<List<Integer>>` object, for example as created in question 3.

8) Modify your answer to question 3 to store the set of lists of integers as a `TreeSet<List<Integer>>` object with an ordering given by the second `Comparator` object of question 7. Then also store the set of lists of integers as a `TreeSet<List<Integer>>` object with an ordering given by the first `Comparator` object of question 7. Compare the results.

9) Modify the first `Comparator` object of question 7 so that if two lists are of the same length, the one with the greater sum is considered greater than the other one and vice versa. Modify the second `Comparator` object of question 7 so that if two lists have the same sum, the longer one is considered greater than the other one. Use these `Comparators` in a modified version of question 8.

Matthew Huntbach