Backup System for Student Course Assignments

Create a class Node to store the B-Number (4 digit int) and an arraylist of course names (strings). The possible course names are "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K".

Write code for a tree that has the features to insert, search, and delete Nodes. You need to write code for the tree by yourself. However, you are allowed to get help from a book or online source (for example the source code for BST, AVL, 2-3, 2-3-4, etc.) If you do so, cite the URL of where you got the code from both in your source code and also README.txt. It will be considered CHEATING if you do not cite the source of any code on tree that you use/adapt.

As you need to modify the code to implement the Observer pattern, you can't just use an in-built tree in Java. Each Node of the tree should implement both the Subject and the Observer interface.

Do not use the built-in Observer classes/interfaces in Java.

Populate the tree using the data from an input file, input.txt, that has the following format:

1234:C

2345:D

1234:A

1234:D

1234:E

2345:F

3425:C

...

For class participation, send me input.txt and delete.txt so that I can post it to the course assignment page. The first 10 sample files will get class participation points.

The inputs may not be unique (some entries may repeat). So, either ignore the repeated inputs manually (if you use an ArrayList, for example), or use a key-value data structure that ensures unique entries.

Assume that the input.txt and delete.txt will be well formatted.

Your tree builder should be such that when you create a node (node_orig as the variable name), you should clone it twice to get two Nodes (let's say backup_Node_1 and backup_Node_2 as the variables holding the references). Setup backup_Node_1 and backup_Node_2 as observers of node_orig. node_orig, backup_Node_1, and backup_Node_2, should be instances of the same Node class. As node_orig is the subject, you should store the references of backup_Node_1 and backup_Node_2 in a data structure for listeners in Node_orig (array list of references, for example).

Apply the following delete operations, processing line at a time, from the file delete.txt. The file has the following format:

1234:C

2345:D

1234:A

...

Search for the node with the B_Number in the line, and then delete the corresponding course in that Node. If that course does not exist to delete, then ignore and move to the next line. If a node does not exist with that BNumber, then ignore and move to the next line. Once the changes to the node_orig are done, the changes should be updated to both the corresponding nodes (call notifyAll(...)). Note that the updates for a line in delete.txt should be sent, before the next line is processed.

From the command line accept the following args: input.txt, delete.txt, output1.txt, output2.txt, output3.txt.

Add a method to your tree, printNodes(Results r, ...), that stores in Results the list of courses for each student, sorted by the B_Number.

Your driver code should do the following:

Read command line arguments.

Build the three trees, based on input.txt and the observer pattern.

Apply updates according to delete.txt

Create a new Results instance and call printNodes(...) on each of the three trees to store the BNumber and list of courses to store in Results. So, each of the three trees will use a different instance of Results.

Call a method in Results, via the method in FileDisplayInterface, to write the data stored in Results to output1.txt, output2.txt, and output3.txt, for the three trees. You should run a "diff" on the three output files to make sure your Observer pattern worked correctly.