E.7 Homework Assignment E.7

In Lab Assignment C.8 you read packet data that arrived in unsorted order for a single message and reassembled packets into correct order. In this assignment, you are to assume that you are receiving packet data for an unspecified number of messages, and you will need to keep track of all the incoming packets until you have received all the packets for a particular message. When you have received all the packets, you should print the message out in sorted order and then remove the message from the active collection of messages with which you are dealing.

You do not need to change anything in the Packet class from what was used in Lab C.8.

The changes you should make from Lab C.8 to this homework are as follows. In Lab C.8 you used a TreeMap<Integer, Packet>, perhaps with variable name oneMessage, to collect and store message packets for a single message, using the packetID as the key in the TreeMap. With multiple messages coming in, and a desire to have fast access, you will need to have a TreeMap<Integer, TreeMap<Integer, Packet>> perhaps with variable name allMessages to store all the packets for all the messages. The key for the TreeMap of allMessages should be the messageID.

That is, you will have a TreeMap of TreeMaps with the “inner” TreeMap being the storage for a single message, keyed by packetID, and the “outer” TreeMap being the storage for a TreeMap of messages, keyed by messageID.

A sample Main class, which you are free to use verbatim, is on the Moodle website. This simulates a process that would be fired up at boot time and then would continue to run essentially forever. The process would read packets one at a time, store the packets in the PacketAssembler class, and then print out and remove messages when they are complete.

In order to know when you have read in all the packets for a given message, you will also need yet another TreeMap, also keyed by messageID and perhaps named packetCountRead, for the purpose of storing the total number of packets for that message that you have read so far. This can be compared against the total packet count for a message that is stored with every instance of Packet data; when the total number of packets read equals the total number of packets in the message, you have read them all, and you are done with that message.

When a packet is read, you should do a lookup in this last TreeMap. If there is no entry for that messageID, then this is the first packet you have seen for that message. You should create a new instance of oneMessage, add the new packet to that instance, add the instance of oneMessage to the TreeMap named allMessages, and store a 1 in the TreeMap named packetCountRead to indicate that you have read one packet for the message with that value of
messageID.

After having read a packet, you should check to see if your new count of packets read for this particular message is equal to the total number of packets for the message. If so, print the message and remove it from the \texttt{allMessages} structure.

\section*{Error conditions and special checks}

You may assume that the value held for the total number of packets in a message is the same in each packet with that particular message ID. Thus, checking any packet for the total packet count is ok.

You should not assume that no packet is sent twice. Before you add a packet to the \texttt{oneMessage} structure, you should do a lookup to make sure you haven’t already stored a message with that \texttt{packetID}. If you don’t do this, you are likely to count a packet twice and thus think you have read all the packets when in fact there is still one out there to be read.

You should use an \texttt{iterator} over the packets in a given message when you get to the point of actually writing out the entire message.

Note that you don’t have to do your program exactly as I did mine. I wrote both my debugging information and my output to a log file, but you could equally well use a \texttt{PrintWriter} and that possibility has been left in as an option in the \texttt{interface} for PacketAssembler.