1 Lab Assignment 10

You are to write a program that will read in data that looks like Internet packets and assemble them into messages. Your code will use `HashSet` and `TreeMap` to manage the packets.

Packet data exists in the file `zpackets`. Each line is a packet and has, in order

- an `int` message identifier `messageId`
- an `int` packet sequence number `packetSequence` within the message, indexed zero-up
- an `int` number `packetCount` which is the number of packets in the entire message
- a `String` data payload value

Your program will read in the data line by line. YOU DO NOT NEED TO BULLETPROOF YOUR DATA INPUT. IF THIS WERE A HOMEWORK ASSIGNMENT THEN YOU WOULD HAVE TO DO A BETTER JOB OF BULLETPROOFING, BUT SINCE THIS IS A LAB, WE WILL ASSUME THAT THE DATA IS CORRECT.

From the message identifier and the packet sequence number you will create a key for the packet that is

```
MULTIPLIER * messageId + packetSequence
```

Note that you will also have to create a method to extract `messageId` as

```
key / MULTIPLIER
```

and the sequence number as

```
key % MULTIPLIER
```

even though in this exercise you will only need the `messageId`.

You are to use a `TreeMap` to store all the packets for all the messages coming in. This is not a good idea because it means you are storing all the data in one big `TreeMap`, and in the homework exercise you will be expected to do a better job, but for the lab, this is ok.

As you read data, you are to store the packets. As you read data, you should increment a counter (in the same way you counted votes in the homework exercise) of the number of packets you have seen for this particular message. When you have read all the packets for a given message, you should assemble the message. YOU MAY ASSUME THAT ALL THE PACKETS YOU ARE READING ARE DISTINCT. THIS WILL CHANGE IN THE HOMEWORK ASSIGNMENT.

When you have read all the packets for a given message, you should dump the message to the log file. Your method to assemble packets into a message should iterate over the `TreeMap` and concatenate the packets for the given message into a `String` that is returned as the assembled message. When your method to assemble packets into a message has finished creating the assembled `String` to be returned, you should remove the packets from the `TreeMap` of packets.

THIS LAST PART IS THE TRICKIEST PART OF THIS ASSIGNMENT. IF YOU ITERATE OVER THE `TreeMap` AND REMOVE INSIDE THAT ITERATED LOOP, YOUR PROGRAM WILL PROBABLY CRASH BECAUSE YOU WILL BE CHANGING THE `keySet` OVER WHICH YOU ARE ITERATING WHILE YOU ARE DOING THE ITERATION. THIS WILL GENERATE AN ERROR DUE TO CONCURRENT UPDATES. A SIMPLER APPROACH IS TO STORE UP THE KEYS TO BE DELETED IN `HashSet` AND THEN IN A SEPARATE LOOP TO ITERATE OVER THAT `HashSet` OF KEYS AND DO THE REMOVAL.