Message class - writing a message to the messages file Part II - availableList not empty

We keep using the following messages. Now we implement the method that writes messages and sets internal links appproately so that messages can be read from the file later.

	data (RECORD_DATA_LEN bytes)	**next** (pointer to next record: 4 byte int)
0	Dear Susan, how are you today? I am going to try to write you more frequently ab	2
1	Hi Johnny, do you want to go to the movies tonight? I have free passes. Suzie	-1
2	out the material that we had left open last time. It seems that the entries that	3
3	you had included in the invoices contain some calculation errors. For example, t	4
4	he first item has a price of $39.75 and there were 5 items ordered. The resultin	6
5	To everyone at the sales department: This is a reminder that our monthly luncheo	8
6	g price should be $39.75 x 5 = $198.75 but you have it showing as $201.25, and t	7
7	he tax is calculated based on this figure of $201.25. Could you please review th	9
8	n will take place at Favourite Pizza, 275 Pepperoni Road. See you all there.	-1
9	e entire worksheet and email me a new copy. I truly appreciate it. Sincerely Pet	10
10	er, Inventory Manager.	-1

Message class

Add an access method

public void setText(String t) that sets **text** to **t**

Implement the following method in the Message class

public int writeToMessagesFile()

that writes an entire message to the messages file. The method writes the implicit parameter to the file using first any possible available records from the availableList. If there are no records available and there is still portions of the message to be written to the file, then the records are appended and the file grows.

The method does not access the messages file directly but indirectly through the methods of the Record class.

The method takes no parameters. It writes the message as records and sets the correct next links.

The method returns the record number of the first record of the message. For example, if the message is written using records 12, 3, 6, 15, and 16 then the method returns 12.

Pseudo-code is below

```
// pseudo-code portion when the availableList is not empty

write to messages file the string s
    if availableList is empty
        first record number is set to total records
    else
        first record number is set to first number in availableList without removing
                                                    the first node from availableList

    while s is not a null string {
        if the messages file is full {
            <...code when avalableList is empty...>
        }
        else {
            get next available spot
            if the message fits in one record {
                set a record with data from message
                write the record in MODIFY mode
                set s to null string
            }
            else {
                set a record with the first RECORD_DATA_LEN chars of s and
                                                        the subsequent record number
                write the record in MODIFY mode
                remove from s the first RECORD_DATA_LEN chars
            }
        }
    }
    return the number of first record where message was written
}
```

```
public int writeToMessagesFile() {
    declare s as String s and assign text it (this will be slightly modified later)
    declare recordNumber as int and initialize to -1         // current record
    declare nextRecordNumber as int and inialize to -1       // link to next record
    declare startRecordNumber as int and initialize to -1    // first record of message


    // code continues on next page
```

```
    if availableList is empty
        set startRecordNumber to total records in messages file
    else
        set startRecordNumber to the first record number in availabeList without removing
        the element from the list: do not use getNextRecord() from AvailabeList

    declare and instantiate record as an object of the Record class

    while s is not a null string {
        if availableList is empty { < ... code if availableList is empty ...> }
        else {
            recordNumber =  get next record number from availableList (with removal)
            if s fits in one record {
                record.setData(s, Globals.END_OF_MESSAGE);
                record.writeToMessagesFile(recordNumber, Globals.MODIFY);
                s = "";  // forces out of loop
            }
            else {
                if availableList is empty
                    set nextRecordNumber to total records in messages file
                else
                    set nextRecordNumber to next record number in availableList
                                                                    (with removal)

                record.setData(first RECORD_DATA_LEN characers of s, nextRecordNumber);
                record.writeToMessagesFile(recordNumber, Globals.MODIFY);
                s = remove from s the portion that has just been written to file
            }
        }
    }
    return startRecordNumber
}
```

EmailServer

For all tests be sure to:

1) Use the file _messages.dat in the PickUp folder. This file contains the table above (page 1)
2) Use the following template in the main program

```
main() {
     declare and instantiate an object of the Record class, for example record
     declare and instantiate an object of the Message class, for example message

     open the messages file (use FileIO.openMessagesFile() method; do not open directly)

     if successful
          delete record at position 1
          delete record at position 5
          delete record at position 8

          // at this point the availableList contains three nodes

          < ...
            write code here using the different tests in the table below
          ... >

          write the message with the call message.writeToMessagesFile()
          close the messages file
     end if
}
```

3) check the expected results as outlined in the table below

Tests

These tests are for the second part of the code, where the availableList is not emtpy

test	conditions	code being tested	expected results
1	1 message that fits in a single record	availableList is not empty and message fits in one record; new message should go in record 1	messages file does not grow; no new links are set in messages file; availableList contains only two nodes
2	2 messages that will each fit on a single record	same as above but availableList will have shrunk by one node for every written record	messages file does not grow; no new links are set in messages file; availableList contains only one node; the messages are in records 1 and 5 respectively
3	1 message that requires 3 records	same as above but now a message requires more than one record	messages file does not grow; no new links are set in messages file; availableList becomes empty; the message is in records 1, 5, and 8 respectively
4	1 message that requires 5 records	the message now overflows the availableList	message file will grow by 2 records; availableList becomes empty; the message is in records 1, 5, 8, 11, 12