HW19: Bank Accounts: Linked Lists

Revise HW17 or HW18 to add the following functionality:

1. Add a **LinkedListAccounts Class**:

This class has a **private Node Class**. The Node Class has:

a) **Two private data members:**
   i) a **reference to an Account**
   ii) a **reference to the next Node in the linked list**

b) a **1-Arg Constructor: Node(Account acct)**
   where acct is a reference to the Account to store at the node

c) a **2-Arg Constructor: Node(Account acct, Node n)**
   where acct is a reference to the Account to store at the node and n is a reference to the successor node

In addition the Class has the following data members and methods:

a) **Two private data members referencing the head and tail Nodes of the linked list respectively:**

b) **public LinkedListAccounts()**
   A No-Arg Constructor that sets the head and tail of the linked list to null

c) **public boolean isEmpty()**
   Checks to see if the linked list is empty

d) **public int size()**
   Returns the number of elements in the linked list

e) **public void add(Account acct)**
   Adds acct to the end of the linked list

f) **private <E extends Comparable> void addSortedGeneric(Account acct, E e2, ArrayList<E> e)**
   A generic method that adds acct to a sorted linked list so as to continuously maintain the sorted linked list

g) **public void addSortedByAcctNum(Account acct)**
   Properly adds acct to a linked list sorted by Account Number **(must use addSortedGeneric())**

h) **public void addSortedBySSN(Account acct)**
   Properly adds acct to a linked list sorted by SSN **(must use addSortedGeneric())**

i) **public void addSortedByName(Account acct)**
   Properly adds acct to a linked list sorted by Name **(must use addSortedGeneric())**

j) **public void addSortedByBalance(Account acct)**
   Properly adds acct to a linked list sorted by Balance **(must use addSortedGeneric())**

k) **public void add(int pos, Account acct)**
   Inserts acct into the linked list at position pos

l) **public Account get(int pos)**
   Returns a **reference** to the Account value at position pos in the linked list

m) **public void set(int pos, Account acct)**
   Sets (i.e., replaces) the Account value to acct at position pos in the linked list

n) **public void setSortedByAcctNumber (Account acct)**
   Sets (i.e., replaces) the Account value to acct at the Node of the linked list with the matching account number

o) **public Account remove(int pos)**
   Remove the Node at position pos of the linked list and return its Account value

p) **public void removeSorted(Account acct)**
   Remove the Node whose account number matches that of acct in a sorted linked list

q) **public boolean remove(Account acct)**
   Remove the Node whose account number matches that of acct - return success indicator

2. A modified **Bank** class which now consists of an unsorted **Linked List of Accounts** currently active or closed.

The **Linked List should be referenced through a LinkedListAccounts object**
(e.g. LinkedListAccounts llByAcctNum)

In addition, the Bank class also maintains the linked list **sorted continuously four different ways using LinkedListAccouts objects:**

a) by Account Number (e.g., use LinkedListAccounts llByAcctNumSorted)
b) by SSN (e.g., use LinkedListAccounts llBySSNSorted)
c) by Name (e.g., use LinkedListAccounts llByNameSorted)
b) by Balance (e.g., use LinkedListAccounts llByBalanceSorted)

In addition, the Bank class has several static member variables and methods:

totalAmountInSavingsAccts - sum total of balances in all Savings accounts
totalAmountInCheckingAccts - sum total of balances in all Checking accounts
totalAmountInCDAccts - total - sum total of balances in all CD accounts
totalAmountInAllAccts - total - sum total of balances in all accounts

Make sure to **provide appropriate methods** so as to allow for the **addition to, subtraction from, and reading of**, the current values of each of these static variables.

Be sure to **print the values of all of these static variables when you print the database of accounts**.

The methods in the Bank Class should be modified as deemed necessary.

The **Bank** class does not override either the toString() or the equals() method.

3. Modify any methods of the class containing main() as deemed necessary.

Program testing should proceed as for HW17 or HW18 with these modifications:

At initialization, a neatly formatted table of the initial database of active accounts should be printed as follows:

- **a) unsorted**
- **b) sorted by account number (using the llSortedByAcctNum object)**
- **c) sorted by SSN (using the llSortedBySSN object)**
- **d) sorted by Name (using the llSortedByName object)**
- **e) sorted by Balance (using the llSortedByBalance object)**

Use method printAccts() (and additional methods if needed) as in previous assignments.

At the end, before the user quits, the program prints the contents of the final database as follows:

- **a) unsorted**
- **b) sorted by account number (using the llSortedByAcctNum object)**
- **c) sorted by SSN (using the llSortedBySSN object)**
- **d) sorted by Name (using the llSortedByName object)**
- **e) sorted by Balance (using the llSortedByBalance object)**

Submission Requirements:
Create a folder on Google Drive that will contain the following:

1. The source files (i.e., *.java files) for each of the implemented Classes:

- **pgmHW19.java**
- **LinkedListAccounts.java**
- **Bank.java; Account.java; Depositor.java, Name.java**
- **SavingsAccount.java; CheckingAccount.java; CDAccount.java**
- **Check.java; TransactionTicket.java; TransactionReceipt.java;**
- **implemented Comparator Classes**
- **implemented Exception Classes**
- **etc.**

2. The text file containing the initial database of accounts (e.g., initAccounts.txt)
3. The test cases text file (e.g., myTestCases.txt)
4. The output text file which contains all of the required program output (e.g., pgmOutput.txt)
Then, make the folder shareable and send me a link to the folder.