HW18a: Bank Accounts: Using Collections - List Objects

Revise HW17 to add the following functionality:

1. Add the following Classes:
   a) **AccountComparatorByAcctNumber Class**
      This class **implements the Comparator Interface** and compares two Account objects by Account Numbers.
   b) **AccountComparatorBySSN Class**
      This class **implements the Comparator Interface** and compares two Account objects by their SSN.
      If the SSNs are the same, the accounts are compared by their Account Numbers.
   c) **AccountComparatorByName Class**
      This class **implements the Comparator Interface** and compares two Account objects by their Name objects.
      If the Name objects are the same, the accounts are compared by their Account Numbers.
   d) **AccountComparatorByBalance Class**
      This class **implements the Comparator Interface** and compares two Account objects by their Balance.
      If the balances are the same, the accounts are compared by their Account Numbers.
2. A modified **Bank** class which now consists of an unsorted **List of Accounts** currently active or closed.
      The **List references either an ArrayList of Accounts or a Linked List of Accounts through polymorphism**.
      (Show that it can trivially work either way)
   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 Bank Class also has one of the following sorting keys (for use with findAccts() - you can choose which one):
   a.1) **ArrayList<Integers> acctNumQuickSortKey**
   a.2) **ArrayList<Integers> acctNumBubbleSortKey**
   a.3) **ArrayList<Integers> acctNumInsertionSortKey**
   The **Bank** class does not override either the toString() or the equals() method.

2a. The **Bank** class should now also contain the following method:

```
public void sortAccountList(Comparator<Account> comp)
```

   which will sort the List of Accounts by the specified Comparator object

3. A modified **Account** class which now **implements both the Comparable Interface and the Comparator Interface**
   All methods implemented should be comparing account numbers of the two Account objects.

Program testing should proceed as for HW17 with these modifications:

At initialization, only a neatly formatted table of the initial unsorted database of active accounts should be printed. Use method printAccts() 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 an AccountComparatorByAcctNumber object)**
- **c) sorted by SSN (using an AccountComparatorBySSN object)**
- **d) sorted by Name (using an AccountComparatorByName object)**
- **e) sorted by Balance (using an AccountComparatorByBalance 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:
pgmHW18.java
Bank.java; Account.java; Depositor.java, Name.java
SavingsAccount.java; CheckingAccount.java; CDAccount.java
Check.java; TransactionTicket.java; TransactionReceipt.java;

etc. (including all of the Exception Classes (and Comparator Classes if implemented))
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.

HW18b: Bank Accounts: Using Collections - Set, HashSet, and LinkedHashSet Objects

Revise HW18a to add the following functionality:

1. The **Bank** class should now also contain the following methods:

```
public Set createHashSet()
```

which will create a polymorphic Set reference to a HashSet object of the List of accounts

```
public Set createLinkedHashSet()
```

which will create a polymorphic Set reference to a LinkedHashSet object of the List of accounts

```
public Iterator getHashSetIterator(Set<Account> acctSet)
```

which will return an Iterator of the referenced Set of Account objects

2. The **Account** class should now also contain the following methods:

```
public int hashCode()
```

which returns the account number of an Account as its Hash Code.

Program testing should proceed as for HW18a with these modifications:

At initialization, a neatly formatted table of the initial unsorted database of active accounts should be printed. Use method printAccts() as in previous assignments.

In addition, the initial database of Accounts should also be printed as follows:

a) by creating a Set reference to a HashSet object of the List of Accounts
b) by creating a Set reference to a LinkedHashSet object of the List of Accounts

Test the .contains() method of the Set class on several Account objects

If the account is found, print a message indicating that the Account object was found. Also, print its Hash Code and the account number of the Account object.
If the Account object is not found, print an appropriate message.

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

a) unsorted
b) by creating a Set reference to a HashSet object of the List of Accounts
c) by creating a Set reference to a LinkedHashSet object of the List of Accounts

HW18c: Bank Accounts: Using Collections - SortedSet, and TreeSet Objects

Revise HW18b to add the following functionality:

1. The **Bank** class should now also contain the following methods:

```
public SortedSet createTreeSet()
```

which will create a polymorphic SortedSet reference to a TreeSet object of the List of accounts sorted by Natural Order

```
public SortedSet createTreeSet(Comparator<Account> comp)
```

which will create a polymorphic SortedSet reference to a TreeSet object of the List of accounts sorted by by the specified Comparator object

Program testing should proceed as for HW18a with these modifications:

At initialization, a neatly formatted table of the initial unsorted database of active accounts should be printed. Use method printAccts() as in previous assignments.

In addition, the initial database of Accounts should also be printed as follows:

a) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Natural Order
b) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by account number (using an AccountComparatorByAcctNumber object)
c) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by SSN (using an AccountComparatorBySSN object)
d) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Name (using an AccountComparatorByName object)
e) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Balance (using an AccountComparatorByBalance object)

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

a) unsorted
b) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Natural Order
c) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by account number (using an AccountComparatorByAcctNumber object)
d) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by SSN (using an AccountComparatorBySSN object)
e) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Name (using an AccountComparatorByName object)
f) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Balance (using an AccountComparatorByBalance object)

HW18d: Bank Accounts: Using Collections

Combine parts HW18a, HW18b, and HW18c.

Program testing should proceed as for HW18a with these modifications:

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

- **a) unsorted**
- **b) by creating a Set reference to a HashSet object of the List of Accounts**
- **c) by creating a Set reference to a LinkedHashSet object of the List of Accounts**
- **d) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Natural Order**
- **e) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by account number (using an AccountComparatorByAcctNumber object)**
- **f) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by SSN (using an AccountComparatorBySSN object)**
- **g) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Name (using an AccountComparatorByName object)**
- **h) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Balance (using an AccountComparatorByBalance object)**

Test the .contains() method of the Set class on several Account objects

If the account is found, print a message indicating that the Account object was found. Also, print its Hash Code and the account number of the Account object.
If the Account object is not found, print an appropriate message.

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

- **a) unsorted**
- **b.1) by creating a Set reference to a HashSet object of the List of Accounts**
- **b.2) by creating a Set reference to a LinkedHashSet object of the List of Accounts**
- **c) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Natural Order**
- **d.1) sorted by account number using the List reference (using an AccountComparatorByAcctNumber object)**
- **d.2) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by account number (using an AccountComparatorByAcctNumber object)**
- **e.1) sorted by SSN using the List reference (using an AccountComparatorBySSN object)**
- **e.2) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by SSN (using an AccountComparatorBySSN object)**
- **f.1) sorted by Name using the List reference (using an AccountComparatorByName object)**
- **f.2) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Name (using an AccountComparatorByName object)**
- **g.1) sorted by Balance using the List reference (using an AccountComparatorByBalance object)**
- **g.2) by creating a SortedSet reference to a TreeSet object of the List of Accounts sorted by Balance (using an AccountComparatorByBalance 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:
   - **pgmHW18.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.