Homework – Topic 7 – Bank Accounts:

You have been hired as a programmer by a major bank. Your first project is a small banking transaction system. Each account consists of a number and a balance. The user of the program (the teller) can create a new account, as well as perform deposits, withdrawals, and balance inquiries.

Initially, the account information of existing customers is to be read into a pair of parallel arrays--one for account numbers, the other for balances. The bank can handle up to MAX_NUM accounts. Use the following method to read in the data values:

```
public static int readAccts(int[] acctNum, double[] balance)
```

This method fills up the account number and balance arrays by reading from an input file until EOF is reached, and counting how many accounts are read in. It returns the actual number of accounts read in (later referred to as numAccts).

After initialization, **the main program prints the initial database of accounts and balances**. Use method printAccts() described below.

The program then allows the user to select from the following menu of transactions:

Select one of the following:

W - Withdrawal
D - Deposit
N - New account
B - Balance
Q – Quit
X – Delete Account – Extra Credit

Use the following method to produce the menu:

```
public static void menu()
```

This method only displays the menu. The **main program** then prompts the user for a selection. You should verify that the user has typed in a valid selection (otherwise print out an error message and repeat the prompt).

Once the user has entered a selection, one of the following methods should be called to perform the specific transaction. **At the end, before the user quits, the program prints the final contents of the account and balance arrays**.

```
public static int findAcct(int[] acctNum, int numAccts, int account);
```

This method returns the index of account in the acctNum array if the account exists, and -1 if it doesn't. It is called by all the remaining methods.

```
public static void withdrawal(int[] acctNum, double[] balance, int numAccts);
```

This method prompts the user for the account number. If the account does not exist, it prints an error message. Otherwise, it asks the user for the amount of the withdrawal. If the account does not contain sufficient funds, it prints an error message and does not perform the transaction.

```
pubic static void deposit(int[] acctNum, double[] balance, int num_accts);
```

This method prompts the user for the account number. If the account does not exist, it prints an error message. Otherwise, it asks the user for the amount of the deposit.

```
public static int newAcct(int[] acctNum, double[] balance, int numAccts);
```

This method prompts the user for a new account number. If the account already exists, it prints an error message. Otherwise, it adds the account to the acctNum array with an initial balance of 0. It returns the new number of accounts.

```
public static void balance(int[] acctNum, double[] balance, int numAccts);
```

This method prompts the user for an account number. If the account does not exist, it prints an error message. Otherwise, it prints the account balance.

```
public static void printAccts(int[] acctNum, double[] balance, int numAccts);
```

This method prints a **2-column table** (with column headings) of all customer information--account numbers and balances.

EXTRA CREDIT 1:

```
     public static int deleteAcct(int[] acctNum, double[] balance, int numAccts);
```

This method prompts the user for an account number. If the account does not exist, it prints an error message. Otherwise, it deletes the account. It returns the new number of accounts.

or

EXTRA CREDIT 2:

```
     public static int deleteAcct(int[] acctNum, double[] balance, int numAccts);
```

This method prompts the user for an account number. If the account does not exist it prints an error message. **If the account exists but has a non-zero balance, it prints an error message, you then make a withdrawal transaction of the balance, and then delete again - this time successfully.** The method returns the new number of accounts.

Notes:
1. All output must be file directed (you must add additional parameters to the methods as needed)
2. Only output must go to the file - not interactive prompts or menus (which go to the monitor).
3. No global variables are allowed
4. The program and all methods must be properly commented.
5. The program must be properly tested.
- **a. The initial database should consist of at least 7 accounts**
- **b. The initial database should be printed to the file**
- **c. Test at least 2 invalid menu selections**
- **d. Test at least 2 balance inquiries:**
  - **i. valid account**
  - **ii. invalid account**
- **e. Test at least 3 deposits:**
  - **i. valid account - valid deposit amount**
  - **ii. valid account - invalid deposit amount**
  - **iii. invalid account**
- **f. Test at least 4 withdrawals:**
  - **i. valid account - valid withdrawal amount**
  - **ii. valid account - invalid withdrawal amount**
  - **iii. valid account - insufficient funds**
  - **iv. invalid account**
- **g. Create at least 3 new accounts with an initial balance of 0.0**
- **h, Test the creation of at least 1 invalid new account**
- **i. Test several transactions on the new accounts (deposits, withdrawals, etc.)**
- **j. Extra Credit: Delete at least two accounts (both must be old accounts that had no transactions)**
  - **State whether you are doing Extra Credit 1 or Extra Credit 2.**
  - **Extra Credit 1 is worth +1; Extra Credit 2 1s worth +2**
- **k. quit and print the final database**

Sample Output:

```
Transaction Type: Deposit
Account Number: 9876
Current Balance: $300.50
Amount to Deposit: $123.45
New Balance: $423.95

Transaction Type: Withdrawal
Account Number: 9876
Current Balance: $423.95
Amount to Withdraw: $765.43
Error: Insufficient Funds Available
```

Required Submission:
1. The Java source code file (e.g., HW7_1.java)
2. The initial database of accounts file (initAccts.txt)
3. The test cases input file (e.g., testCases.txt)
4. The program generated output file (e.g., output.txt)