CSE 102 Programming Assignment 3

Due:

Sunday, 22-May-2022 by 23:59

Deliverables:

The following Java file should be submitted to Google Classroom by the due date and time specified above. Submissions received after the deadline will be subject to the late policy described in the syllabus.

- Assignment03_{StudentNumber}.java
- ".git" folder of your assignment's git repository

Specifications:

Overview: You will continue your program to maintain the account balances for bank customers. **<u>Do not forget your headers with your assignment's Git folder, Name and Date information.</u>**

We want to be able to execute a series of Transactions as a batch. Unfortunately, sometimes if we execute a withdrawal prior to a deposit on the same account, we might get an InvalidAmountException when we could have avoided this by processing the deposit first. For this reason, we will create a method to process a Collection of Transaction objects that will process all deposits first, then all transfers, then all withdrawals. If two transactions are the same type, we will process them according to the account numbers:

- For Deposits – in order by account to
- For Transfers – in order by account to
- For Withdrawals – in order by account from

Requirements: Write and modify the following set of classes:

1. Transaction
    a. Attributes
        i. type: int - 1 for deposit, 2 for transfer, 3 for withdrawal
        ii. to: String - the account number to which to give funds
        iii. from: String - the account number from which to get funds
        iv. amount: double - the amount of funds
    b. Methods (required methods given below, other methods not listed may be necessary)
        i. Constructor that takes all four parameters in the order given
            1. If an invalid type is passed, this method should throw an InvalidParameterException.
        ii. Constructor that takes only one String parameter and assigns to either the from account (withdrawals) or the to account (deposits)
            1. If any other type gets passed, whether transfer or an invalid value, this method should throw an InvalidParameterException.
        iii. `getType()`, `getTo()`, `getFrom()`, and getAmount()
            1. no set methods should be defined; the Transaction object should be immutable

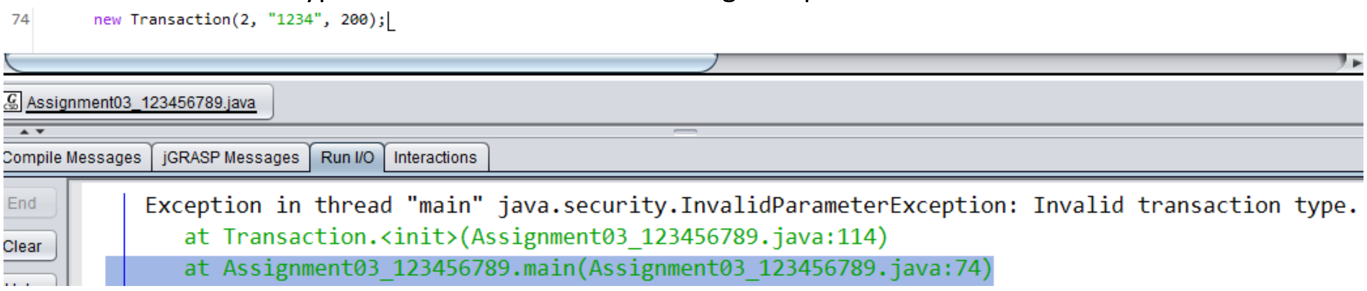
2. Bank
   a. Attributes
      i. No Change
   b. Methods – all methods from Project 02 plus the following
      i. processTransactions(transactions: Collection<Transaction>, outFile: String): Takes an unsorted List of Transaction objects and sorts them according to the description above. Then processes each Transaction by type. If an exception is encountered, writes to the file given in String an error log according to the following:
         1. "ERROR: " + Exception type + ": " + Transaction type + {tab} + Account number(s) separated by tab + {tab} + amount
3. Account – no change from Assignment 02
4. PersonalAccount – no change from Assignment 02
5. BusinessAccount – no change from Assignment 02
6. Customer – no change from Assignment 02
7. Company – no change from Assignment 02
8. Custom Exceptions – no change from Assignment 02

Design: Your program does not require a main method. You are only responsible for creating and modifying the classes and Exceptions described above.
Code: The file you submit will be named Assignment03_{StudentNumber}. You should put all java classes for this assignment inside of this file as discussed in class.
Test: You are responsible for testing your program. It is important to not rely solely on the examples presented in this Assignment description.

Grading:
MS Teams Submission: If anything is ambiguous, it is your responsibility to ask questions. It is also your responsibility to complete this assignment in a timely manner. Questions regarding this assignment will likely not be answered if received after 17:00 on the business day prior to the due date of the assignment.
Example: Using the given set of accounts from Assignment 02 (shown at right):

```
My Bank  My Bank's Address
   Company 1
      1234  0.05  1000000.0
      1235  0.03  25000.0
   Company 2
      2345  0.03  350.0
   Customer 1
      3456  150.0
      3457  250.0
   Customer 2
      4567  1000.0
```

An invalid transaction type should result in the following Exception:



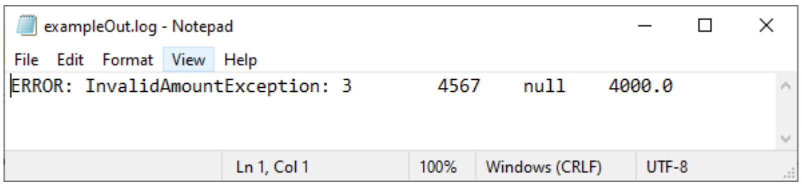
```
74      new Transaction(2, "1234", 200);
```

```
Exception in thread "main" java.security.InvalidParameterException: Invalid transaction type.
   at Transaction.<init>(Assignment03_123456789.java:114)
   at Assignment03_123456789.main(Assignment03_123456789.java:74)
```

If we run the following code without sorting the list, we get the given output and file:

```
Collection<Transaction> ts = new ArrayList<Transaction>();
ts.add(new Transaction(3, "4567", 4000));
ts.add(new Transaction(1, "4567", 3100));
b.processTransactions(ts, "exampleOut.log");
System.out.println(b);
```

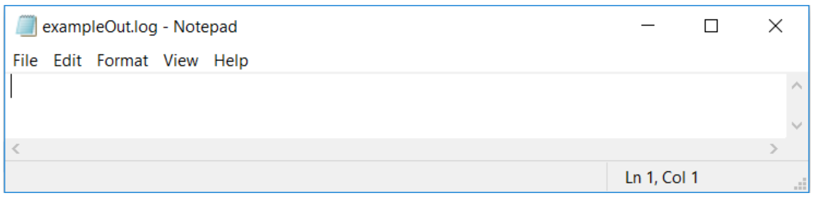
```
My Bank   My Bank's Address
   Company 1
      1234  0.05  1000000.0
      1235  0.03  25000.0
   Company 2
      2345  0.03  350.0
   Customer 1
      3456  150.0
      3457  250.0
   Customer 2
      4567  4100.0
```



```
ERROR: InvalidAmountException: 3        4567     null     4000.0
```

Your method must sort the Collection first, then we will get the following if we run the same code:

```
My Bank   My Bank's Address
   Company 1
      1234  0.05  1000000.0
      1235  0.03  25000.0
   Company 2
      2345  0.03  350.0
   Customer 1
      3456  150.0
      3457  250.0
   Customer 2
      4567  100.0
```



Note also that the below code uses a HashSet for the Collection and regardless of the order the withdrawal and deposit are added to the HashSet could result in an error.

```
Collection<Transaction> ts = new HashSet<Transaction>();
ts.add(new Transaction(1, "3456", 100));
ts.add(new Transaction(3, "3456", 300));
b.processTransactions(ts, "exampleOut.log");
System.out.println(b);
```

Therefore, your processTransactions() method must take any type of Collection and store it to a sortable Collection, then sort the Transactions according to the description above. **NOTE: Do not assume that the Collection type passed will be of a sortable type on its own. Therefore, you should import java.util.* to make sure any type of Collection that gets passed will be accepted by your method.**