CSE214

HOMEWORK 1 - SPRING 2025

HOMEWORK 1 - due Tuesday, February 11th no later than 7:00PM

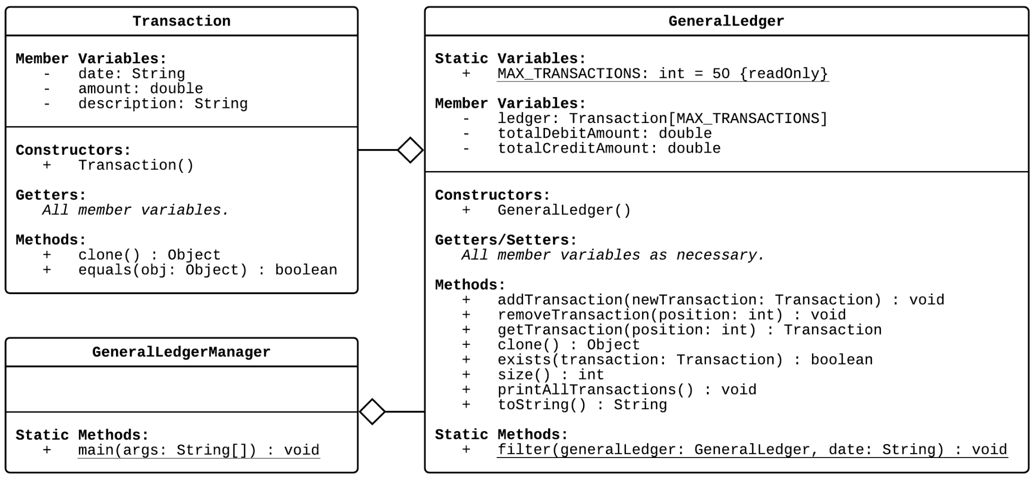
REMINDERS:

- **Submit your files only via CodeGrade on the Content page in Brightspace. Access CodeGrade by clicking "HW1 - CODEGRADE SUBMISSION LINK" for your submission.**
- **Use of a package is optional. If you wish to use it, make sure to name it "hw1" (all in lowercase). Otherwise, you will lose points.**
- **Be sure your code follows the coding style for CSE214.**
- **Make sure you read the warnings about academic dishonesty. *Remember, all work you submit for homework or exams MUST be your own work.***
- **You are not allowed to use ArrayList, LinkedList, Vector or any other Java API Data Structure classes to implement this assignment.**
- **You may use Scanner, InputStreamReader, or any other class that you wish for keyboard input.**

In this assignment, you will be the accountant for Jack's general bank account and will write a general ledger to help him keep track of his financial transactions. The general ledger is simply a list where an individual can record their transactions, or the buying and selling of a product or service. A transaction contains general information such as the date, amount (or the change in cash as a result of the transaction), and description. The general ledger should be able to record a total of 50 transactions.

UML

The UML Diagram for all the classes specified below is as follows:


Transaction
Member Variables:
- date: String
- amount: double
- description: String
Constructors:
+ Transaction()
Getters:
All member variables.
Methods:
+ clone() : Object
+ equals(obj: Object) : boolean
GeneralLedgerManager
Static Methods:
+ main(args: String[]) : void
GeneralLedger
Static Variables:
+ MAX_TRANSACTIONS: int = 50 {readOnly}
Member Variables:
- ledger: Transaction[MAX_TRANSACTIONS]
- totalDebitAmount: double
- totalCreditAmount: double
Constructors:
+ GeneralLedger()
Getters/Setters:
All member variables as necessary.
Methods:
+ addTransaction(newTransaction: Transaction) : void
+ removeTransaction(position: int) : void
+ getTransaction(position: int) : Transaction
+ clone() : Object
+ exists(transaction: Transaction) : boolean
+ size() : int
+ printAllTransactions() : void
+ toString() : String
Static Methods:
+ filter(generalLedger: GeneralLedger, date: String) : void

SPECIFICATIONS

1. Transaction

Write a fully-documented class named `Transaction` which contains the date (String), amount (double) , and description (String). You should provide accessor methods for each variable, but no mutator methods. This is because a transaction should never be modified after being created. The full list of required methods is:

- `public Transaction` - Constructor (you may include a constructor with parameters)
- Getter methods for each variable
- `public Object clone()` - The return value is a copy of this Transaction. Subsequent changes to the copy will not affect the original and vice versa. Note that the return value must be typecasted to a Transaction before it can be used. Note that the copy being returned must be a deep copy.
- `public boolean equals(Object obj)` - A return value of true indicates that obj refers to a Transaction object with the same attributes as this Transaction. Otherwise, the return value is false.

You will also need the following variables:

- `String date`
- `double amount`
- `String description`

NOTE: Normally a debit is an accounting entry that either increases an asset or expense account, or decreases a liability or equity account, and a credit is an accounting entry that either increases a liability or equity account or decreases an asset or expense account. However, for this homework assignment, we are going to assume the user has no accounting knowledge and simplify the process by making it so that a debit is simply an increase in dollar amount to the main account, and a credit is a decrease in dollar amount to the main account, regardless if the account is an asset, expense, liability, or equity account. So if the user posts a transaction with an amount value or debit of $40 to Jack's account with a current value of $0, after the transaction is posted Jack's account should now have a value of $40. Note that Jack's bank account can be negative (indicating he is in debt). So if the user posts a transaction with an amount value of -50, or credit of $50, to Jack's account with a current value of $0, after the transaction is posted Jack's account should now have a value of -$50.

NOTE: Note that the date should be inputted as a String in yyyy/mm/dd format. You must ensure however that all date values are valid. This means that the year must be between 1900 and 2050 inclusive, the month must be between 1 and 12 inclusive, and the day must be between 1 and 30 inclusive (for simplicity assume every month is 30 days long). The date should also be printed (mentioned later) in yyyy/mm/dd format.

2. GeneralLedger

Write a fully-documented class named `GeneralLedger` that stores an ordered list of transaction objects within an array. The array indices determine the order the transactions are to be posted. An accountant can insert a transaction at any position within the range of the list. The GeneralLedger can record a total of 50 transactions, so use the static final variable `int MAX_TRANSACTIONS = 50` (and refrain from using the number 50 within your code). Additionally, create variables of type double "totalDebitAmount" and "totalCreditAmount" to keep track of the running total of all debit and credit amounts for all transactions. These variables should be modified every time a transaction is added or deleted from the general ledger. For example, if Jason's general ledger is empty, "totalDebitAmount" and "totalCreditAmount" should both be initialized to 0. After a single transaction is added with amount value 50, meaning $50 is debited to his account, the value of "totalDebitAmount" should now be 50 and "totalCreditAmount" should now be 0.

NOTE: Make sure there are no gaps between Transactions in the list.

NOTE: Note that arrays in Java are 0 indexed, but your listing preference should start from 1.

The GeneralLedger class implements an abstract data type for a list of Transactions supporting some common operations on such lists. The GeneralLedger class will be based on the following ADT specifications:

`public class GeneralLedger`

- `public GeneralLedger`
    - Brief:
        - Constructs an instance of the GeneralLedger with no Transaction objects in it.
    - Postconditions:
        - This GeneralLedger has been initialized to an empty list of Transactions.

- `public void addTransaction(Transaction newTransaction)`
    - Brief:
        - Adds `newTransaction` into this `GeneralLedger` if it does not already exist. Added entries should be inserted in date order. If there are multiple entries with the same date, new entries should be placed at the end of that date's boundary.
    - Parameters:
        - `newTransaction` - the new `Transaction` to add to the ledger
    - Preconditions:
        - The `Transaction` object has been instantiated and the number of `Transaction` objects in this `GeneralLedger` is less than `MAX_TRANSACTIONS`.
    - Postconditions:
        - The new transaction is now listed in the correct order in the list. All Transactions whose date is newer than or equal to `newTransaction` are moved back one position (e.g. If there are 5 transactions in a generalLedger, positioned 1-5, and the transaction you insert has a date that's newer than the transaction in position 3 but older than the transaction in position 4, the new transaction would be placed in position 4, the transaction that was in position 4 will be moved to position 5, and the transaction that was in position 5 will be moved to position 6.) (See sample output for more information).
    - Throws:
        - `FullGeneralLedgerException` - Thrown if there is no more room in this `GeneralLedger` to record additional transactions.
        - `InvalidTransactionException` - Thrown if the transaction amount is 0 or if the date is invalid.
        - `TransactionAlreadyExistsException` - Thrown if there is already a transaction in this `GeneralLedger` which is equivalent to `newTransaction` (both have the same date, amount, and description).
    - NOTE: Remember that the format for inputting the date is yyyy/mm/dd.
    - NOTE: Inserting an item with a newer date than any of the other items in the list is effectively the same as adding the item to the end of the list.

- `public void removeTransaction(int position)`
    - Brief:
        - Removes the transaction located at `position` from this `GeneralLedger`.
    - Parameters:
        - `position` - The 1-based index of the Transaction to remove
    - Preconditions:
        - This generalLedger has been instantiated and 1 <= `position` <= items_currently_in_list.
    - Postconditions:
        - The Transaction at the desired position has been removed. All transactions that were originally greater than or equal to position are moved backward one position (e.g. If there are 5 Transactions in the generalLedger, positioned 1-5, and you remove the Transaction in position 4, the item that was in position 5 will be moved to position 4).
    - Throws:
        - `InvalidLedgerPositionException` - Thrown if `position` is not valid.
    - NOTE: `position` refers to the position in the general ledger, not the position in the array.

- `public Transaction getTransaction(int position)`
    - Brief:
        - Returns a reference to the `Transaction` object located at `position`.
    - Parameters:

        - `position` - The position in this `GeneralLedger` to retrieve.
    - Preconditions:
        - The GeneralLedger has been instantiated and 1 <= `position` <= items_currently_in_list.
    - Returns:
        - The `Transaction` at the specified position in this `GeneralLedger` object.
    - Throws:
        - `InvalidLedgerPositionException` - Indicates that `position`> is not within the valid range.
    - NOTE: `position` refers to the position in the general ledger, not the position in the array.

- `public static void filter(GeneralLedger generalLedger, String date)`
    - Brief:
        - Prints all transactions that were posted on the specified date.
    - Parameters:
        - `generalLedger` - The ledger of transactions to search in.
        - `date` - The date of the transaction(s) to search for.
    - Preconditions:
        - This `GeneralLedger` object has been instantiated.
    - Postconditions:
        - Displays a neatly formatted table of all `Transactions` that have taken place on the specified date.

- `public Object clone()`
    - Brief:
        - Creates a copy of this `GeneralLedger`. Subsequent changes to the copy will not affect the original and vice versa.
    - Preconditions:
        - This `GeneralLedger` object has been instantiated.
    - Returns:
        - A copy (backup) of this `GeneralLedger` object.
    - NOTE: Remember that this is supposed to return a deep copy of the object.

- `public boolean exists(Transaction transaction)`
    - Brief:
        - Checks whether a certain transaction is contained in the ledger.
    - Parameters:
        - `transaction` - The `Transaction` to check for.
    - Preconditions:
        - This `GeneralLedger` and `transaction` have been instantiated.
    - Returns:
        - `true` if this `GeneralLedger` contains `transaction`, `false` otherwise.
    - Throws:
        - `IllegalArgumentException` - Thrown if `transaction` is not a valid `Transaction` object.

- `public int size()`
    - Brief:
        - Returns the number of `Transactions` currently in this ledger.
    - Preconditions:
        - This `GeneralLedger` has been instantiated.
    - Returns:
        - The number of `Transactions` in this ledger.
    - NOTE: This method must run in O(1) time.

- `public void printAllTransactions()`
    - Brief:
        - Prints a neatly formatted table of each item in the list with its position number (as shown in the sample output).
    - Preconditions:

        - This `GeneralLedger` has been instantiated.
    - Postconditions:
        - All transactions contained within this `GeneralLedger` are printed in a neatly formatted table. See sample i/o for format.
    - NOTE: Remember that the format for printing the date is yyyy/mm/dd.
    - HINT: If your toString() method is implemented correctly, you will simply need to call it and print the results to standard output.

- `public String toString()`
    - Brief:
        - Returns a String representation of this `GeneralLedger` object, which is a neatly formatted table of each `Transaction` contained in this ledger on its own line with its position number (as shown in the sample output).
    - Returns:
        - A String representation of this `GeneralLedger` object.

- `public static final int MAX_TRANSACTIONS` - The maximum number of transactions a `GeneralLedger` can contain.
- `private Transaction[] ledger` - The array which will hold `Transactions` inserted into this `GeneralLedger` object.
- `private double totalDebitAmount` - The total debit amount for all transactions in this `GeneralLedger` object.
- `private double totalCreditAmount` - The total credit amount for all transactions in this `GeneralLedger` object.

3. GeneralLedgerManager

Write a fully documented class named GeneralLedgerManager which will act as a driver for the ledger that is based on the following specification:

`public class GeneralLedgerManager`

- `public static void main(String[] args)`
    - The main method runs a menu driven application which first creates an empty `GeneralLedger` object. The program prompts the user for a command to execute an operation. Once a command has been chosen, the program may ask the user for additional information if necessary and performs the operation. The operations should be defined as follows:
        - `A` - Add Transaction <date> <amount> <description>
          Adds a new transaction into the ledger.
        - `G` - Get Transaction <position>
          Displays information of a Transaction at a given position.
        - `R` - Remove Transaction <position>
          Removes the transaction at the given position.
        - `P` - Prints Transaction in General Ledger
          Displays all transaction in the General Ledger.
        - `F` - Filter by Transaction Date <date>
          Displays all transactions that were posted on the specified date.
        - `L` - Look for Transaction <date> <amount> <description>
          Determines whether the transaction with the given attributes is in the current General Ledger.
        - `S` - Size
          Determines the number of transactions in the General Ledger. (NOTE: This must complete in O(1) time.)
        - `B` - Backup
          Creates a copy of the given General Ledger. Changes to the copy will not affect the original and vice versa.
        - `PB` - Print Transactions in Backup

Displays all the transactions in the backed-up copy.

- `RB` - Revert to Backup
  Reverts the current General Ledger to the backed-up copy.
- `CB` - Compare Backup with Current
  Compares the backed-up ledger with the current ledger.
- `PF` - Print Financial Information
  Displays the sum of all debit dollar amounts for all transactions side-by-side with the sum of all credit dollar amount of all transactions. Also prints the net worth value of the account, which is the result of subtracting the total credit amount from the debit amount. (NOTE: This must complete in O(1) time.)
- `Q` - Quit
  Terminates the program.

4. You will need classes to handle the exceptions thrown (see class specifications above for exception classes you need).

NOTE: You may include additional methods, variables, or classes as necessary or as you find convenient.

INPUT FORMAT

- Each menu operation is entered on its own line and should be case insensitive (i.e. 'q' and 'Q' are the same).
- Check to make sure that the position, if required, is valid. If not, print and error message and return to the main menu.
- For the Add Transaction command, if the input information is valid, construct the object accordingly. Otherwise, print an error message and return to the main menu.
- You may assume Strings are at most 100 characters long.

OUTPUT FORMAT

- Each command should output the result (as shown in sample IO below) after each operation is performed.
- All menu operations must be accompanied by a message indicating what operation was performed and whether or not it was successful.
- All lists must be printed in a nice and tabular form as shown in the sample output. You may use C style formatting as shown in the following example. The example below shows two different ways of displaying the name and address at pre-specified positions 21, 26, 19, and 5 spaces wide. If the '-' flag is given, then it will be left-justified (padding will be on the right), else the region is right-justified. The 's' identifier is for strings, the 'd' identifier is for integers. Giving the additional '0' flag pads an integer with additional zeros in front.

```
String name = "Doe Jane";
String address = "32 Bayview Dr.";
String city = "Fishers Island, NY";
int zip = 6390;

System.out.println(String.format("%-21s%-26s%19s %05d", name, address, city, zip));
System.out.printf("%-21s%-26s%19s %05d", name, address, city, zip);
```

```
Doe Jane             32 Bayview Dr.             Fishers Island, NY 06390
Doe Jane             32 Bayview Dr.             Fishers Island, NY 06390
```

HINTS

- Remember that the position parameter to all of the methods in the GeneralLedger class refer to the position of a transaction within the collection (starting at position 1), and not the position of a transaction

within the array (which starts at position 0). There are two ways that you can approach this issue:

- **Store the first item in position 0, the next item in position 1, and so on. In each method, subtract one from the position given by the parameter to find the appropriate location in the array.**
- **Define your array such that it is the size of `MAX_TRANSACTIONS + 1`. Store the first item in position 1, the next item in position 2, and so on. Position 0 of the array will never be used.**

SAMPLE INPUT/OUTPUT

```
// Comments in green, input in red, output in black.

(A) Add Transaction
(G) Get Transaction
(R) Remove Transaction
(P) Print Transactions in General Ledger
(F) Filter by Date
(L) Look for Transaction
(S) Size
(B) Backup
(PB) Print Transactions in Backup
(RB) Revert to Backup
(CB) Compare Backup with Current
(PF) Print Financial Information
(Q) Quit

Enter a selection: P

No transactions currently in the general ledger.

// Menu not shown in sample i/o

Enter a selection: A

Enter Date: 2018/12/20
Enter Amount ($): -100.00
Enter Description: Purchased Christmas tree from the Elves.

Transaction successfully added to the general ledger.

// Menu not shown in sample i/o

Enter a selection: A

Enter Date: 2018/11/22
Enter Amount ($): -50.00
Enter Description: Purchased turkey from supermarket.

Transaction successfully added to the general ledger.

// Menu not shown in sample i/o

Enter a selection: A

Enter Date: 2018/11/23
Enter Amount ($): 25.00
Enter Description: Sold video games on Black Friday.

Transaction successfully added to the general ledger.

// Menu not shown in sample i/o

Enter a selection: S

There are 3 transactions currently in the general ledger.

// Menu not shown in sample i/o
```

```
Enter a selection: A

Enter Date: 2018/11/23
Enter Amount ($): -100.00
Enter Description: Bought T.V. on Black Friday.

Transaction successfully added to the general ledger.

// Menu not shown in sample i/o

Enter a selection: a

Enter Date: 2018/12/15
Enter Amount ($): -50.00
Enter Description: Purchased Christmas presents.

Transaction successfully added to the general ledger.

// Menu not shown in sample i/o

Enter a selection: P

No.    Date          Debit    Credit    Description
---------------------------------------------------------------------------------------------------
1      2018/11/22              50.00    Purchased turkey from supermarket.
2      2018/11/23    25.00              Sold video games on Black Friday.
3      2018/11/23             100.00    Bought T.V. on Black Friday.
4      2018/12/15              50.00    Purchased Christmas presents.
5      2018/12/20             100.00    Purchased Christmas tree from the Elves.

// Menu not shown in sample i/o

Enter a selection: F

Enter Date: 2018/11/23

No.    Date          Debit    Credit   Description
---------------------------------------------------------------------------------------------------
2      2018/11/23    25.00             Sold video games on Black Friday.
3      2018/11/23             100.00   Bought T.V. on Black Friday.

// Menu not shown in sample i/o

Enter a selection: G

Enter Position: 1

No.    Date          Debit    Credit    Description
---------------------------------------------------------------------------------------------------
1      2018/11/22              50.00    Purchased turkey from supermarket.

// Menu not shown in sample i/o

Enter a selection: G

Enter Position: 10

No such transaction.

// Menu not shown in sample i/o

Enter a selection: L

Enter Date: 2018/12/15
Enter Amount ($): -50.00
Enter Description: Purchased Christmas presents.

No.    Date          Debit    Credit    Description
```

```
-----------------------------------------------------------------------------------------------------
4      2018/12/15              50.00    Purchased Christmas presents.

// Menu not shown in sample i/o

Enter a selection: B

Created a backup of the current general ledger.

// Menu not shown in sample i/o

Enter a selection: CB

The current general ledger is the same as the backup copy.

// Menu not shown in sample i/o

Enter a selection: R

Enter position: 9

Transaction not removed: No such transaction in the general ledger.

// Menu not shown in sample i/o

Enter a selection: R

Enter position: 3

Transaction has been successfully removed from the general ledger.

// Menu not shown in sample i/o

Enter a selection: CB

The current general ledger is NOT the same as the backup copy.

// Menu not shown in sample i/o

Enter a selection: P

No.    Date          Debit    Credit   Description
-----------------------------------------------------------------------------------------------------
1      2018/11/22              50.00   Purchased turkey from supermarket.
2      2018/11/23    25.00             Sold video games on Black Friday.
3      2018/12/15              50.00   Purchased Christmas presents.
4      2018/12/20             100.00   Purchased Christmas tree from the Elves.

// Menu not shown in sample i/o

Enter a selection: PB

No.    Date          Debit    Credit   Description
-----------------------------------------------------------------------------------------------------
1      2018/11/22              50.00   Purchased turkey from supermarket.
2      2018/11/23    25.00             Sold video games on Black Friday.
3      2018/11/23             100.00   Bought T.V. on Black Friday.
4      2018/12/15              50.00   Purchased Christmas presents.
5      2018/12/20             100.00   Purchased Christmas tree from the Elves.

// Menu not shown in sample i/o

Enter a selection: RB

General ledger successfully reverted to the backup copy.

// Menu not shown in sample i/o

Enter a selection: CB
```

```
The current general ledger is the same as the backup copy.

// Menu not shown in sample i/o

Enter a selection: A

Enter Date: 2019/01/01
Enter Amount ($): 100.00
Enter Description: Received cash from new year's present.

Transaction successfully added to the general ledger.

// Menu not shown in sample i/o

Enter a selection: A

Enter Date: 2019/01/01
Enter Amount ($): 100.00
Enter Description: Received cash from new year's present.

Transaction not added: Transaction already exists in the general ledger.

// Menu not shown in sample i/o

Enter a selection: P

No.    Date          Debit    Credit   Description
---------------------------------------------------------------------------------------------------
1      2018/11/22               50.00   Purchased turkey from supermarket.
2      2018/11/23    25.00              Sold video games on Black Friday.
3      2018/11/23              100.00   Bought T.V. on Black Friday.
4      2018/12/15               50.00   Purchased Christmas presents.
5      2018/12/20              100.00   Purchased Christmas tree from the Elves.
6      2019/01/01   100.00              Received cash from New Year’s present.

// Menu not shown in sample i/o

Enter a selection: PF

Financial Data for Jack's Account
---------------------------------------------------------------------------------------------------
     Assets: $125.00
Liabilities: $300.00
  Net Worth: $-175.00

// Menu not shown in sample i/o

Enter a selection: Q

Program terminating successfully...
```