RIT | Golisano College of Computing and Information Sciences
School of Information

ISTE-120 - Computational Problem Solving for the Information Domain I
Homework Assignment 4 (HW04)

DELIVERABLE

Submit a single zipfile to the MyCourses HW4 assignment folder. Homework MUST be submitted on time to receive full credit. Homework submitted to the “late” assignment folder will receive a maximum grade of 80%. Homework not submitted to either assignment folder will receive no credit.

ASSIGNMENT

Based on earlier course material, this assignment will enhance the BankAccount class to demonstrate how abstraction and encapsulation enable evolutionary software changes.

Start with the version of BankAccount.java in the Downloads for HW04 from the myCourses HW04 content area. Begin by incorporating a transaction (service) fee for every deposit and withdrawal. Supply a `setTransFee` mutator method for setting a fee charged for every transaction and modify the deposit and withdraw methods so that the fee is levied. Test the resulting class and check that the fee is computed correctly.

Before continuing, remove the changes made to the deposit and withdraw methods but keep the attribute and the mutator.

Now make a more complex change. The bank will allow a fixed number of free transactions (deposits or withdrawals) every month, and charge for transactions exceeding the free allotment. The charge is not levied immediately but at the end of the month. Supply another mutator method called `setNumFreeTrans()` that sets the free transactions number. Declare, initialize, and increment a transaction count attribute. Increment this count for each transaction.

Supply a new method, `deductMonthlyCharge()`, to the BankAccount class which deducts the monthly charge from the account balance, but only if the number of free transactions is exceeded, and then resets the transaction count. *Hint:* Use the `max` method from the `Math` class:

```
Math.max(actual transaction count, free transaction count)
```

Think about what the simple formula should be. Do NOT use an `if` statement. Design a test program that verifies that the fees are calculated correctly over several months.

The output below is shown for several scenarios with differing initial deposits and number of transactions, but always with the transaction fee of `$2.00`. The test program should produce the output below by calling methods of the BankAccount class. The program must be called BankAccountTester and must have a `main()` method. Balances should be obtained by using the `getBalance()` method. Deposits, withdrawals and applying the monthly charge should use the appropriate methods. When the Java classes are ready, compress to a single zipfile named `YourLastName-HW04.zip` and submit to the myCourses assignment folder.

Sample run:

```
Set up new account with $1000, 5 free transactions and $2 per transaction above 5
Starting Balance: 1000.0
deposit 1000, withdraw 500, withdraw 400, deposit 200 - 4 transactions
Balance: 1300.0
Expected: 1300.0
Apply monthly charge
Ending Balance Month 1: 1300.0
Expected: 1300.0

Starting Balance: 1300.0
deposit 1000, withdraw 500, withdraw 400, deposit 200, deposit 500 - 5 transactions
Balance: 2100.0
Expected: 2100.0
Apply monthly charge
Ending Balance Month 2: 2100.0
Expected: 2100.0

Starting Balance: 2100.0
deposit 1000, withdraw 500, withdraw 400, deposit 200, deposit 500, withdraw 1000 - 6
transactions
Balance: 1900.0
Expected: 1900.0
Apply monthly charge
Ending Balance Month 3: 1898.0
Expected: 1898.0

Starting Balance: 1898.0
deposit 1000, withdraw 500, withdraw 400, deposit 200, deposit 500, withdraw 1000,
deposit 100 - 7 transactions
Balance: 1798.0
Expected: 1798.0
Apply monthly charge
Ending Balance Month 4: 1794.0
Expected: 1794.0
```

Golisano College of Computing and Information Sciences
School of Information

Name: ____

Homework 4 Grade Sheet

Improved BankAccount	Point Value	Points Earned
BankAccount	60	
• Constructors initializes the appropriate values • `setTransFee` mutator is correctly added • `setNumFreeTrans` is correctly added • Counter keeps track of number of transactions • `deductMonthlyCharge` method is correctly added • `deductMonthlyCharge` method uses `max`, not `if` • Correct reset for the next month	5 10 10 5 10 10 10	
BankAccountTester	40	
• New account set up properly • Month 1 results correct • Month 2 results correct • Month 3 results correct • Month 4 results correct • Balances obtained by calling `getBalance()` method • Withdrawals, deposits, and fee deduction uses correct methods • Output matches specification	5 5 5 5 5 5 5 5	
Total Points	100	

Note: If a program fails to compile, a grade of zero will be given for the assignment. A clean compilation means that the compiler generates no warning messages.

Additional Comments: