Assignment #5: Loops & ArrayLists

*Due: Wednesday, November 7th @ 11:59PM*

*Total Possible Points: 20*

How to Submit

* Moodle assignment (no emails or hardcopies accepted)
* Submit IDE project in zip format as *Assignment4*.zip

Goals

* To practice using different loops, and to understand when to use certain kinds of loop based on context.
* To understand how to use the ArrayList data structure as a storage mechanism for data.
* To design and develop classes that model real-world entities.

## Your Task

## **In this assignment, you will write create three classes** **that serve as the start of a small banking system**:

## BankAccount – similar to your SavingsAccount class from Assignment 2, but with different features.

1. Bank – a new class that holds a collection of bank accounts, much like a real bank.
2. BankTester – tester program to create, find, and update bank accounts.

Your program will use array lists to store multiple bank accounts in a bank. You will also make use of loop algorithms to help you complete various tasks – namely entering bank account information, finding a bank account, and computing the average size of an account's transactions.

## BankAccount Class Design

Implement these methods in your Student class. All methods are public. *See the Grading Rubric for the class's instance variables*.

|  |  |
| --- | --- |
| Lightbulb | **HINTS on the** getAverageTransactionSize() **class method**:   * Look at Java's Math class for a method that finds an absolute value * Java's ArrayList class has a method that returns the number of items it has. What is that method? |

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | What it does: | What info it needs: | What info it returns: |
| BankAccount | Constructs a BankAccount object; initializes all instance variables. | * Account number * Initial balance | N/A |
| getAccountNumber | Returns the account number | Nothing | Account Number |
| getBalance | Returns the account balance | Nothing | Balance |
| deposit | * Deposits an amount of money into the account * Updates the balance * Adds the transaction amount to the account's transaction list. | Amount | Nothing |
| withdraw | * Withdraws an amount of money from the account * Updates the balance * Adds a *negative* transaction amount to the account's transaction list | Amount | Nothing |
| getAverageTransactionSize | Returns the average of the absolute values of the amounts in the transaction list | Nothing | Average transaction size |

## Bank Class Design

Implement these methods in your Bank class. All methods are public. *See the Grading Rubric for the class's instance variable*.

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | What it does: | What info it needs: | What info it returns: |
| Bank | Constructs a Bank object; initializes its one instance variable. | Nothing | N/A |
| addAccount | Adds a new BankAccount with the given account number and initial balance to the Bank. | * Account Number * Balance | Account Number |
| findAccount | Finds and returns the BankAccount with the given account number | Account Number | BankAccount object |
| getSmallestAccount | Returns the BankAccount with the lowest balance. | Nothing | BankAccount object |
| getLargestAccount | Returns the BankAccount with the highest balance. | Nothing | BankAccount object |

## Requirements/Grading Rubric

* YES = Full credit
* Partial = Half credit
* NO = No credit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Implemented**  **Successfully?** | | |  |  |
| **Requirement** | **YES** | **Partial** | **NO** | **Max Points** | **Comments** |
| ***Program Setup (2 points)*** | | | | | |
| Has a package named bank |  |  |  | **0.5** |  |
| Has classes named Bank and BankAccount |  |  |  | **1** |  |
| Has a class named BankTester with a main() method |  |  |  | **0.5** |  |
| ***BankAccount class (6 points)*** | | | | | |
| Has three (3) private instance variables with appropriate data types:   * accountNumber: stores the account number * balance: stores the balance * transactions: stores an ArrayList<Double> of the account's transaction amounts |  |  |  | **1** |  |
| Has one (1) constructor that initializes accountNumber and balance with the given parameter values; initializes transactions with a new ArrayList object. |  |  |  | **1** |  |
| Implements the getAccountNumber and getBalance methods with correct logic and return types. |  |  |  | **1** |  |
| Implements the deposit and withdraw methods with the correct logic, parameters, and return types. |  |  |  | **2** |  |
| Implements the getAverageTransactionSize method with the correct logic and return type. |  |  |  | **1** |  |
| ***Bank class (5 points)*** | | | | | |
| Has one (1) private instance variable:   * accounts: stores an ArrayList<> of the bank's accounts. |  |  |  | **0.5** |  |
| Has one (1) constructor that initializes accounts with a new ArrayList<BankAccount> object. |  |  |  | **0.5** |  |
| Implements the addAccount and findAccount methods with the correct logic, parameters and return types. |  |  |  | **2** |  |
| Implements the getLargestAccount and getSmallestAccount methods with the correct logic and return types. |  |  |  | **2** |  |
| ***BankTester test program class (5 points)*** | | | | | |
| Constructs a Bank object |  |  |  | **0.5** |  |
| Use a while loop with a sentinel value that…   * asks the user to enter an account number and balance. If a user enters -1 instead of an account number, the loop stops * passes the entered account number and balance to the Bank's addAccount method to add a new bank account to the Bank. * asks the bank to find the just-added bank account using the account number * asks the user to make one deposit and one withdrawal to and from the bank account. |  |  |  | **2.5** |  |
| * gets the bank accounts with the smallest and largest balances; * prints their account numbers, balances and average transaction amounts; * formats the output with $ and cents to 2 decimals places |  |  |  | **2** |  |
| ***Documentation (2 points)*** | | | | | |
| Javadoc-style code comments for the Student class and all public methods, including constructors |  |  |  | **1.5** |  |
| Generates full Javadocs for the project |  |  |  | **0.5** |  |

## Test Inputs & Expected Output

**Possible input:**

101 400.0 -100.0 200.0

102 900.0 700.0 -50.0

103 200.0 -200.0 300.0

0

**Expected Output:**

Account with highest balance:

#102, current balance=$1550.00, average transaction amount=$550.00

Account with lowest balance:

#103, current balance=$300.00, average transaction amount=$233.33