University of Victoria
Department of Computer Science
CSC 110 Fundamentals of Programming

ASSIGNMENT 6

DUE: Thursday, March 22, 2018 before 11:50 pm – By submission on ConneX only

Output: MUST Match Sample Output – Exactly

Indentation: MUST be Exactly 4 spaces or 1 tab

How to hand in your work

Submit the file (called `ClientUpdate.java`) that completely answers part II (below) through the Assignment #5 link on the CSC 110 conneX site. Please make sure you follow all the required steps for submission (including the final confirmation of your submission). If you have received an email confirming your submission, then you know it is submitted, if not, wait a few moments, then re-check what you have done and re-submit.

Learning Outcomes

When you have completed this assignment, you should understand:

- How to use a class to instantiate objects
- How to use instance methods
- How to create and use objects
- How to output to files

Problems from the Textbook:

Complete the *Chapter 8 Self-Check Problems* questions 1-3, 5-8, 10, 11, 13, 14, 16, 17, 19-21, 24, 25. They will *really* help you understand Classes and Objects.

Programming Problem Description:

This assignment requires the writing of a program that simulates the operations of a bank, from the perspective of one client who has several different types of bank accounts. An object will be created for each type of account, then transactions will be applied to the accounts and the accounts will be output in various orders to ~~both~~ the screen ~~and a file~~.

1. Examine & add comments to the Bank Account class:

The following code defines a very simple class called **BankAccount**. Observe that the class contains the data associated with only one account, and the core functionality (methods) that the account needs. More specifically, it contains 4 *data items* (or *attributes*) and a collection of *instance methods* (or *behaviours*) for changing the values of that data.

```
/* ADD COMMENTS TO THIS CLASS, DO NOT CHANGE ANY CODE! */
import java.util.*;

public class BankAccount {

  private int accountNumber;
  private String ownerName;
  private double balance;
  private String type; // Personal, Business, Charitable

  public BankAccount() {
    accountNumber    = 0;
    ownerName = "";
    balance = 0.0;
    type = "Personal";
  }

  public BankAccount(int number, String name, double initialDeposit,
                           String type) {
    accountNumber    = number;
    ownerName = name;
    balance = initialDeposit;
    this.type = type;  // Why does 'this' need to be used here??
  }

  public int getAccountNumber() {
    return accountNumber;
  }

  public void setAccountNumber(int number) {
    accountNumber = number;
  }

  public String getOwnerName() {
    return ownerName;
  }

  public void setOwnerName(String name) {
    ownerName = name;
  }

  public double getBalance() {
    return balance;
  }

  public void setBalance(double newAmount) {
    balance = newAmount;
  }

  public String getType() {
      return type;
  }

  public void deposit(double amount) {
    balance += amount;
  }

  public void withdrawl(double amount) {
    balance -= amount;
  }

  public String toString() {
    return type + ": " + accountNumber + " " + ownerName + " " + balance;
  }
}
```

Use this code but ***do not*** alter it: enhance the documentation by including:

- For each method add a precise description of its purpose, input and output.
- Explain the purpose of each member variable.

2. The Bank: A program that uses the BankAccount class:

In a different Java class, called `ClientUpdate.java`, create a bank's client processing program with the following specifications:

- The single client's file contains information on all that person's accounts (personal, business and charitable), including deposits, withdrawals, interest earned and charges. These are all stored in a text file, the name of the file should be input from the keyboard.
    - The first line of the file contains an integer, called `number`, indicating the number of accounts and the client's name.
    - The next `number` lines of the file contain, for each account, a String token indicating the type of the account and a unique integer representing the account number and a real number indicating the starting balance.

    An array of `BankAccounts` can be created to hold the client file information. Use a constructor from the `BankAccount` class to instantiate each of the accounts.

    - Following that, there are an unknown number of lines, each contains the integer identification number for the account, a String (one of "deposit", "withdrawal", "interest", "charge") indicating the type of transaction, and a real number indicating the amount of the transaction.
        - The amount of a deposit must be added to the `BankAccount`'s balance;
        - The amount of a withdrawl must be subtracted from the balance;
        - The amount of interest requires adding 'amount' % interest to the account;
        - The amount of a (service) charge must be subtracted from the account.

        Use member methods of the `BankAccount` class to alter the attributes of the appropriate `BankAccount` object.

- Once the entire file has been processed output all attributes of each account to the screen:
    - Use the instance method called `toString()` to make a single String containing all attributes of a `BankAccount` object that is being printed.

HAND IN: Submit the final code that you produced (in your `ClientUpdate.java` file) using the 'Assignments' link of the course web page. (Since multiple submissions are allowed, please submit each part as you get it working, replacing the previous submission each time.) Your final submission must occur **before 11:50 pm on Thursday, March 22, 2018**.