CSC 111aD - Fall 2013 - Lab 11:
Due 11/26/13,5pm

Instructions

- Submit to Sakai on or before the due date. If not complete, submit what you have.
- Program files should use the following file naming format:
  - Lastname_Firstinitial_Lab_Lab#.py
  - For example, for this assignment, the project is Lab #11, the file would be named: *Thomas_S_Lab_11.py*
- All programming assignments must follow the style guide, include helpful and appropriate comments, and have meaningful variable names.

The purpose of this lab is primarily to give you experience working with a simple class and objects of that class. The specific example will be that of bank accounts on which one can make deposits and withdrawals. In order to make this slightly more realistic the account balance and all transactions will be recorded in a text file that can be re-read at a later date so that additional transactions can be carried out.

More Details

This lab will utilize two testing programs provided for you (already completed) and part of a class definition that you will need to finish. One of the test programs will use the *bankaccount* class to create several bank accounts. Once a bank account is created, some information, including the initial balance, will be written to a text file so that a separate program can later read the balance from the text file and allow other transactions on the account. The second test program will read the information about an account back from a text file and allow money to be deposited or withdrawn. Each transaction on an account will be logged in the appropriate text file, as well as the balance after each transaction.

More, More Details

1. Download *bankaccount.py, createaccounts.py, usebankaccounts.py* from Sakai -> Resources -> Labs -> Lab 11 Files. Place them all in the same folder.
2. Open *createaccounts.py* in an editor. This is the program you will run to create *bankaccount* objects and, indirectly, bank account text files. Read the program for familiarity and leave it open for running later.
3. PART I - Open *bankaccount.py*. You will see the beginnings of a class definition for a bank account object. You should see that a bank account object keeps track of an account number, a name, and a balance. You should also see that the constructor writes some information out to a specially named text file. The text file will have the same name as the account number. At this point the file will contain three lines containing (1) the account number (2) the name of the account owner (3) the account balance in a form like : "B $10.00", where B indicates that this line contains the balance.

   For Part I of this lab, all you have to do is complete the function named *__str__(self)*. Look back in your notes to see what the purpose of this specially named function is. Complete the function. You can decide on the format you want to use. Mine returns a string like the following:

   Account #36712 Name: Susan Z Smith Balance: $1000.00

4. After you complete step 4, run *createaccounts.py*. Does the output make sense? Look in the folder where your files are located. Do you see five new files with names like 36712.txt, …,

44311.txt ? Open one of the files to see the contents. [Reminder: you will not see the files if your editor is only looking for .py files]

You can now close the file named *createaccounts.py* and open the file named *usebankaccounts.py*.

5. Part II – in this part of the lab you will need to add some code to the class definition in *bankaccount.py*. These enhancements will be in two parts that I'll outline below. These parts will work together to allow a program to create a *bankaccount* object associated with an already existing account and then make deposits and withdrawals from the account. Each deposit and withdrawal will be logged to the text file associated with the account. For example, after several transactions on the 54221 account, the 54221.txt file might contain:

```
54221
John J Doe
B $100.00
W $100.00
B $0.00
D $100.00
B $100.00
D $200.00
B $300.00
W $250.00
B $50.00
```

If you read through the file you can see that the account was opened with a $100 Balance. Then there was a Withdrawal of $100 to give a Balance of $0.00; then mom Deposited $100 so the Balance was $100; then dad Deposited $200 to yield a Balance of $300, then someone Withdrew $250 for a Balance of $50.00. Note that the lines use B for Balance, D for Deposit, and W for Withdrawal.

Here are the two things you need to do to make everything work correctly:

i) Complete the constructor function: First, read and understand the constructor function you were given. Note that the constructor uses default values for the initname and initbalance parameters. When we created new bank accounts in Part I we were providing arguments to the constructor for the account number, the user name, and the initial balance. For this part of the lab we will be retrieving information for accounts that have already been created. Hence we will not need to provide a name or balance to create a bankaccount object, only the account number. You need to complete the section of the __init__ function where initname == "" and initbalance == 0. What does the constructor need to do in this case?
   a. In simple words, it needs to open the text file containing information for the appropriate bank account and read the owner's name and current balance. Where are those items in the file? Well, the account owner's name will be on the second line of the file and the current balance will be on the LAST line of the file, regardless of the number of lines in the file. Remember that it will look like "B $XX.XX".
   b. Close the text file.
   c. That's all the constructor needs to do. Again, it needs to set the name and balance to the values read from the file.

ii) Add a function named *deposit(self, amount)*: This function will be very similar to the withdraw() function, it's just that it will add an amount to the balance. Use withdraw() as a model to understand how the function interacts with the text file to write an entry for the Deposit and to write the new Balance. Your deposit function will actually be simpler than withdraw() since you can't deposit too much money but you can try to withdraw more than you have. The function needs to close the file it appended to and should return some kind of notification that the transaction was successful. To check your program open the text file named 54221.txt and see if it matches the values shown on the previous page.

Deliverable:

- When your program works correctly upload just the file named *bankaccount.py* to the Assignment section of Sakai. Please use the file name Lastname_Initial_Lab_11.py.

SCORING RUBRIC:

5 pts – Program compiles and executes
10 pts – Constructor works correctly
10 pts – Program writes the correct information to the text file for each account
5 pts – Code written by students is well documented