|  |  |  |
| --- | --- | --- |
| **CS 112, Foundations of CS**  **Lab 6: Classes**  **Submit to Canvas** | | **Computer Science** |
|  |  | |

This lab is worth 100 points. The goal for this lab is:

1. **Writing a class and using instances of the newly written class**

If you do not complete the lab in the time allotted, then please return to the lab in your spare time, and complete it by the **due date, which is specified on Canvas**.

**Preliminaries**

* For this lab, create another folder, called **lab 6.**
* Navigate to the *winpython* folder on your computer. The path to the folder is the following: **C:\bin\winpython.** If you need help getting to the folder please refer back to lab 1. Once inside the *winpython* folder double click on “Spyder” icon and it will open up the Spyder IDE.

1. **Writing your own class**

For this programing task you will write a bank account class. This class will have a total of 6 methods. It will have methods which keep track of number of deposits and withdraws from the account. You will also write methods which keep track of the total amount of withdraws and deposits. If you call the deposit method twice with passing $100 as the argument, then number of deposits will equal to 2 and total for the deposits would be $200. You would also update the overall balance by $200.

Remember that each function must have at least one parameter which is “**self**”, referring to the current instance of the class. For example if you were to write a method named “**someMethod**”, which takes in one parameter named “**nums**”. Your method definition would be the following:



1. To get started create a new file and save it as ***BankAccount.py***.
2. In the comment section on top, include your name, todays date and program name (Lab 6, BankAccount.py).
3. Create a class and call it **BankAccount**.
4. Your first method should be **“\_\_init\_\_”.** It should take in “**self**” and 2 additional parameters, “**balance**” and “**name**”. Inside of the method set the balance and name to “**self.balance**” and “**self.name**”. Also set 4 additional instance variables inside of this method. The variables and their assignment are listed below:

***num\_deposits = 0***

***deposit\_total = 0***

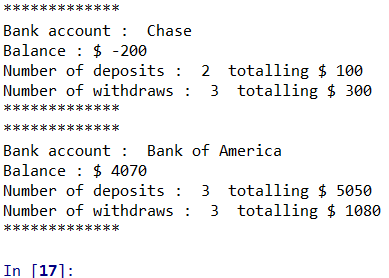
***num\_withdraws = 0***

***withdraw\_total = 0***

1. Write another method named “**numDeposits**” which just takes in 1 parameter “**self**”. Inside of this method refer to the current instance of “**num\_deposits**” and increment it by 1.
2. In a similar way (as step #5) write a method “**numWithdraws**” and increment the instance variable “**num\_withdraws**”.
3. Write a method and name it “**deposit**”, it should take in 2 parameters, “**self**” and “**amount**”. The method should update the instance variable “**balance**” by the parameter “**amount**”. It should also update the instance variable “**deposit\_total**” by “**amount**”. The last thing this method needs to do is call the method “**numDeposits**“, to update the number of total deposits.
4. In a similar way (as step #7) write a method named “**withdraw**”, which takes in 2 parameters, “**self**” and “**amount**”. This method should update the instance variables “**balance**” and “**withdraw\_total**”. It should also call the method “**numWithdraws**”, which will update the number of withdraws.
5. The last method you have to write is “**endOfMonth**” which takes in 1 parameter, “**self**”. In this method you will print out all of the information to the screen. The format for printing out the information can be seen below under sample inputs.
6. Once you have finished writing the class, create 2 instances of BankAccount. Invoke the “**deposit**” and “**withdraw**” method a few times for each instance of BankAccount, which you have just created. At the end call the method “**endOfMonth**” on each instance.
7. Sample inputs and outputs for this program can be seen below:

|  |  |
| --- | --- |
| **Transactions** | **Name = “*Chase*”**  **Starting Balance = 0** |
| Deposit | 50 |
| Deposit | 50 |
| Withdraw | 100 |
| Withdraw | 100 |
| Withdraw | 100 |

|  |  |
| --- | --- |
| **Transactions** | **Name = “*Bank of America*”**  **Starting Balance = 100** |
| Deposit | 25 |
| Deposit | 25 |
| Deposit | 5000 |
| Withdraw | 10 |
| Withdraw | 1000 |
| Withdraw | 70 |



**Rubric**

Upload your source code to Canvas. Here's what we are looking for, when grading your submission

The .py file must be thoroughly commented. If your code breaks (crashes) because you've been unable to fix a syntax error, then the comments will allow you to receive partial credit.

For this lab, make sure that the following file is uploaded to Canvas:

*BankAccount.py*

|  |  |
| --- | --- |
| **File / task** | **Points** |
| 1. **BankAccount.py** | 100 |
| Total | 100 |