COMS3101.004 - Perl - Homework 4

April 22, 2012

Due date: 04/27/2012 11.59pm EST
Downloads: none
Submission: Your submission should include three perl modules: BasicMath.pm, Bank.pm & BankAccount.pm

1 Basic math module(5)

Write a traditional module BasicMath.pm with 3 functions

- max - returns the max of the arguments passed
- min - returns the min of the arguments passed
- avg - returns the average of the arguments passed

In all three functions, the user may pass an arbitrary number of arguments to the function. (Hint: we did see the max function in lecture 2.) Your module should automatically import these functions into the program where it is 'use'ed. In other words, you should be able to call these functions as 'avg(@nums)' rather than 'BasicMath::avg(@nums). Hint: use Exporter module.

2 Let's save money (20)

Write two modules Bank.pm and BankAccount.pm to implement the functionality of a very simple bank. You can test your modules using the testbank.pl file that can be downloaded from the homework page.

2.1 BankAccount.pm (8)

The BankAccount module defines a class with three attributes: name, accountNo and balance, and methods to retrieve these attributes. It also provides methods to deposit and withdraw money as well as print all account details. The BankAccount object should be implemented using a hash reference to store the data. To be precise, the BankAccount class should contain the following features

Attributes:

- name - name of account holder
- accountNo - account number
- balance - account balance

Constructor:

- creates the hash reference to store the object data. Initialize it with 'name' of account holder and account number passed as arguments. The constructer takes arguments and should be called as BankAccount->new("Alex",999999999999);

Methods:

- getName - retrieves the name
- getAccountNo - retrieves the account number
- getBalance - retrieves the balance
- deposit - amount is passed as an argument and is added to the balance
- withdraw - withdraw (deduct) the amount (passed as argument) from the balance
- printAccountDetails - prints the name of account holder, account number and balance in separate lines.

2.2 Bank.pm (12)

The Bank module defines a class with two attributes: name and accountBase, and methods to add new account, close an account and basic banking operations such as deposit, withdrawal and getBalance, given a specific account number. It should also have a method to apply monthly interests on all accounts. Lastly, it should have a listAccounts method to list all accounts. It should also be implemented using a hash reference to store the data. To be precise, the Bank class should contain the following features

Attributes:

- name - bank's name
- accountBase - a reference to a has, which is keyed by account numbers and values are objects of BankAccount class.
- nextAccountNumber - keeps track of account number to use for the next new account added.

Constructor:

- creates the hash reference to store the object data. Initialize it with 'name' of bank passed as an argument. The accountBase should be initialized to an empty hash reference. It should also initialize attribute nextAccountNumber to a 12 digit value of your choice.

Methods:

- newAccount - creates a new account for the person whose name is passed as an argument. You will construct a new BankAccount object using the supplied name and nextAccountNumber variable, and add it to the accountBase hash with it's account number as the key. The attribute nextAccountNumber should be incremented to ensure that the next account gets a new account number. This method returns the account number of the newly created account.
- getBalance - given an account number, calls the getBalance function on the corresponding BankAccount object.
- deposit - given an account number and amount, calls the deposit function on the corresponding BankAccount object for the amount.
- withdraw - given an account number and amount, calls the withdraw function on the corresponding BankAccount object for the amount.
- applyInterests - this method should act on all BankAccounts in the bank. The rate of interest is fixed as 12% annual rate, which means, the balance should be increased by 1% every month. So, each call to this method would increase the balance of all accounts by 1%. You can use the deposit method of the account to increase the balance.
- listAccounts - prints details of each account in the Bank in the increasing order of account numbers using the printAccountDetails method of each account.