PHP Exercise Q1 Answered

1. Create an `Account` class that a bank might use to represent customers' bank accounts. Include a data member to represent the account balance. Provide a constructor that receives an initial balance and uses it to initialize the data member. The constructor should validate the initial balance to ensure that it is greater than or equal to 0. If not, set the balance to 0. Provide three member functions:
   - `credit($amount)` should add `amount` to the current balance
   - `debit($amount)` should ensure that `amount` does not exceed the current balance and then reduce the current balance by `amount`. The balance should not change if `amount` exceeds the balance.
   - `getBalance()` should return the current balance

   i. Create a set of test data and then write a program to test the class.

```php
<?php

class Account
{
	private $balance = 0;

	public function __construct($balance){
		if($balance >= 0){
			$this->balance = $balance;
		}
	}

	public function credit($amount){
		$this->balance += $amount;
	}

	public function debit($amount){
		if($amount <= $this->balance){
			$this->balance -= $amount;
		}
	}

	public function getBalance(){
		return $this->balance;
	}
}
```

PHP Exercise Q1 Answered

Test Data

Comments	Input data	Expected Result	Status	Date
Creating account with negative balance	balance = -1	getBalance() = 0	Passed	9/5/2014
Creating account with zero balance	balance = 0	getBalance() = 0	Passed	9/5/2014
Creating account with positive balance	balance = 1	getbalance() = 1	Passed	9/5/2014
Testing credit()	balance = 1 credit = 2	getBalance() = 3	Passed	9/5/2014
Testing debit() - amount exceeds balance	balance = 1 debit = 2	getBalance() = 1	Passed	9/5/2014
Testing debit() - balance exceeds amount	balance = 2 debit = 1	getBalance() = 1	Passed	9/5/2014
Testing debit() - balance equals amount	balance = 1 debit = 1	getBalance = 0	Passed	9/5/2014

```
<? require 'Account.php' ?>
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8" />
		<title>Account Tester</title>
	</head>
	<body>
		<?php
			// Turn on assertions
			assert_options(ASSERT_ACTIVE, true);
			// Failed assertions should terminate
			assert_options(ASSERT_BAIL, true);

			echo '</br>Creating account with negative balance';
			$acc1 = new Account(-1);
			assert($acc1->getBalance() === 0);
			echo '</br>Passed</br>';

			echo '</br>Creating account with zero balance';
			$acc1 = new Account(0);
			assert($acc1->getBalance() === 0);
			echo '</br>Passed</br>';

			echo '</br>Creating account with positive balance';
			$acc1 = new Account(1);
			assert($acc1->getBalance() === 1);
			echo '</br>Passed</br>';

			echo '</br>Testing credit()';
			$acc1 = new Account(1);
			$acc1->credit(2);
			assert($acc1->getBalance() === 3);
```

```
                    echo '</br>Passed</br>';

                    echo '</br>Testing debit() - amount exceeds balance';
                    $acc1 = new Account(1);
                    $acc1->debit(2);
                    assert($acc1->getBalance() === 1);
                    echo '</br>Passed</br>';

                    echo '</br>Testing debit() - balance exceeds amount';
                    $acc1 = new Account(2);
                    $acc1->debit(1);
                    assert($acc1->getBalance() === 1);
                    echo '</br>Passed</br>';

                    echo '</br>Testing debit() - balance equals amount';
                    $acc1 = new Account(1);
                    $acc1->debit(1);
                    assert($acc1->getBalance() === 0);
                    echo '</br>Passed</br>';

                    echo '</br>All tests passed'
            ?>
        </body>
</html>
```

```
Creating account with negative balance
Passed

Creating account with zero balance
Passed

Creating account with positive balance
Passed

Testing credit()
Passed

Testing debit() - amount exceeds balance
Passed

Testing debit() - balance exceeds amount
Passed

Testing debit() - balance equals amount
Passed

All tests passed
```