Banking

LAB # 4

By

Corey Henry

And

Natalie Morrison

***“On my honor, as a Mississippi State University student, I have neither***

***given nor received unauthorized assistance on this academic work.”***

Signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CSE-1384-06-201430 Intermediate Computer Programming

Class Section # 6

Jesse Farek

2/17/2015

**Analysis and Conclusions**

The lab was extremely easy, we used a class to build a banking system that would accept variables and carry information around using self. We then used inheritance in the second demo to call on our first class to create other classes. These classes also accepted variables.

Analysis Questions

1. using inheritance allowed us to shorten our code and not rewrite functions we had already created to do the same thing
2. yes it made it very easy to understand inheritance, some advantages would be if we had to work on a code that we had already created multiple functions for and needed to use some of them. Some disadvantages is that stale date could travel along with it, as well as errors or things we don’t want in our new code.

Source Code:

#Corey Henry & Natalie Morrison #Date Assigned: 10Feb15

# #

#Course CSE 1384 Sec 06 #Date Due: 07Oct2014

#File name: Banking.py

#

#Program description - creates a banking program for a bank account and college

#savings bond.

class Account:

def \_\_init\_\_(self,name,bank):

#set all the variabels for the class of the inported information

self.user = name

self.bank\_name = bank

#set the orginial bank ammount at 0

self.ammount = 0

#set orginial interest rate

self.interest\_rate = 2

# create a function that will take the deposit and add the ammount to it

def deposit(self,ammount):

self.ammount += ammount

# function to find out the interest ammount and return the interest.

def calculateInterest(self):

self.interest = self.ammount \* self.interest\_rate / 100

return(self.interest)

# create a string that will display the status of thier account and return it

def getStatus(self):

Status = (self.user + ' has a net deposit of ' + str(self.ammount) + ' dollars in '+ self.bank\_name +'.')

return(Status)

#create a account and a function that can withdrawl

class BankAccount(Account):

def withdraw(self, ammount):

self.ammount -= ammount

# print that they have overdrawn thier account if bank foes under 0 dollars

if self.ammount < 0:

print('You have overdrawn your Account')

#Create a class that will be for the college bond

class CollegeBond(Account):

#intialize the class and add a basic length of 8 if not given.

def \_\_init\_\_(self,name, bank, length = 8):

#set all the variables and class rewrites

self.user = name

self.bank\_name = bank

self.ammount = 0

self.length = length

self.interest\_rate = 10

#function that will return the length

def getBondLength(self):

return(self.length)