**Wallace Final Project: Week 7**

Amanda Wallace

John E. Simon School of Business, Maryville University

SWDV 630-1W: Object Oriented Software Architecture

Rodney W. McGee, Jr.

June 25, 2023

**Wallace Final Project: Week 7**

For the final project I plan to create elements towards a Point of Sale System. I do not plan to work with anyone else, so will do the work by myself.

My plan is for the application to request if an item is to be added, receipt to be printed, print sales, and calculate total sales – not just the total for the transaction. The receipt total will be calculated based on how many of each item is added, if any discounts are included, and then tax that needs to be added as well as the total number of items.

In order to do this, I plan to re-use some code from previous projects in this class, modify some code previously done for this class, as well as write new in order to fulfill planned functionality.

**Planned Code to Re-use by Submitted Assignment Name**

*CheckingAccount.py*

#create CheckingAccount class

#include: name, address, account number, balance

#balance must be private to class using \_\_

class CheckingAccount:

def \_\_init\_\_(self, name, address, accountnumber, balance):

self.\_\_name=name

self.\_\_address=address

self.\_\_accountnumber=accountnumber

self.\_\_balance=balance

#credit process

def AccountCredit(self, amount):

self.\_\_balance=self.\_\_balance+amount

#debit process

def AccountDebit(self, amount):

if self.\_\_balance<amount:

print("Balance (${: .2f}) is insufficient. Debit for ${: .2f} unable to be completed.".format(self.\_\_balance, amount))

else:

self.\_\_balance=self.\_\_balance-amount

#show balance

def ShowBalance(self):

print("Account {} current balance: ${:.2f}".format(self.\_\_accountnumber, self.\_\_balance))

*Week.2.Driver.py*

#class import

from Submitted.CheckingAccount import CheckingAccount

#create object

account1 = CheckingAccount("Jane Doe", "1234 Sesame St", 5678, 9098.76)

#credit

account1.AccountCredit(543.21)

#excessive debit

account1.AccountDebit(10243.19)

#debit

account1.AccountDebit(10.99)

#show balance for transactions

account1.ShowBalance()

*Week3Inheritance.py*

#superclass

class Order:

def \_\_init\_\_(self, tax, discount, total, item\_qty):

self.tax = tax

self.discount = discount

self.total = total

self.item\_qty = item\_qty

#subclass 1

class Book (Order):

def \_\_init\_\_(self, title, author, bbarcode, bcost):

self.title = title

self.author = author

self.bbarcode = bbarcode

self.bookcost = bcost

#def Book\_Total(x):

book\_quantity = int(input("Enter number of books desired: "))

bcost = float(input("Enter cost of book: $"))

btotal = book\_quantity\*bcost

#subclass 2

class Journal (Order):

def \_\_init\_\_ (self, j\_type, jbarcode, jcost):

self.j\_type = j\_type

self.jbarcode = jbarcode

self.journalcost = jcost

#def Journal\_Total(y):

journal\_quantity = int(input("Enter number of journals desired: "))

jcost = float(input("Enter cost of journal: $"))

jtotal = journal\_quantity\*jcost

#subclass 3

class Pens (Order):

def \_\_init\_\_(self, p\_type, brand, packsize, pbarcode, pcost):

self.p\_type = p\_type

self.brand = brand

self.packsize = packsize

self.pbarcode = pbarcode

self.pencost = pcost

#def Pen\_Total(z):

pen\_total = int(input("Enter number of pens desired: "))

pcost = float(input("Enter cost of pens: $"))

ptotal = pen\_total\*pcost

print()

print()

print("Total for books is $", Book.btotal)

print("Total for journals is $", Journal.jtotal)

print("Total for pens is $", Pens.ptotal)

*Week6SQLAlchemy.py*

from sqlalchemy import create\_engine

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy import Column, Integer, String

from sqlalchemy.orm import sessionmaker

*#superclass*

class Order:

def \_\_init\_\_(self, tax, discount, total, item\_qty):

self.tax = tax

self.discount = discount

self.total = total

self.item\_qty = item\_qty

*#subclass 1*

class Book (Order):

def \_\_init\_\_(self, title, author, bbarcode, bcost):

self.title = title

self.author = author

self.bbarcode = bbarcode

self.bookcost = bcost

*#def Book\_Total(x):*

book\_quantity = int(input("Enter number of books desired: "))

bcost = float(input("Enter cost of book: $"))

btotal = book\_quantity\*bcost

*#subclass 2*

class Journal (Order):

def \_\_init\_\_ (self, j\_type, jbarcode, jcost):

self.j\_type = j\_type

self.jbarcode = jbarcode

self.journalcost = jcost

*#def Journal\_Total(y):*

journal\_quantity = int(input("Enter number of journals desired: "))

jcost = float(input("Enter cost of journal: $"))

jtotal = journal\_quantity\*jcost

*#subclass 3*

class Pens (Order):

def \_\_init\_\_(self, p\_type, brand, packsize, pbarcode, pcost):

self.p\_type = p\_type

self.brand = brand

self.packsize = packsize

self.pbarcode = pbarcode

self.pencost = pcost

*#def Pen\_Total(z):*

pen\_total = int(input("Enter number of pens desired: "))

pcost = float(input("Enter cost of pens: $"))

ptotal = pen\_total\*pcost

print()

print()

print("Total for books is $", Book.btotal)

print("Total for journals is $", Journal.jtotal)

print("Total for pens is $", Pens.ptotal)