# **Algorithm Using Pseudocode**

**Top of Form**

**You work in the IT group of a department store and the latest analytics shows there is a bug that allows customers to go over their credit limit. The company's president has asked you to develop a new algorithm to solve this problem.**

**Create your algorithm using pseudocode that determines if a department store customer has exceeded their credit limit. Be sure you gather the following inputs from the user:**

* + **Account number**
  + **Balance of the account**
  + **Total cost of all the products the customer is looking to purchase**
  + **Allowed credit limit**

**After you gather the inputs, make sure your algorithm calculates if the user can purchase the products and provides a message to the user indicating if the purchase is approved or declined. Bottom of Form**

|  |  |  |  |
| --- | --- | --- | --- |
| Classes | Methods | Variables | Imports |
| Bankaccount  Item | getBalance  getAccountNumber  getCreditLimit  totalCost  compare  main | balance  account\_number  isEqual  cost  array  account  item\_one  item\_two  item\_three | random |

Import random as ran

# imports random module to be used to generate a random bank account number.

class BankAccount:

# Bank account class to house balance and account number variables.

Account\_number = ran.getranbits(whatever acceptable rage is used)

# getranbits function of random module used to generate a number in a range established by whatever is hard coded.

def \_\_init\_\_(self reference, balance):

self.balance is equal to inputted balance #This passes the inputted data to the balance variable of the BankAccount class

def getCreditLimit(self):

return self.balance #Method returns the balance variable of the BankAccount class

def getAccountNum(self): #”Getter” method for the account number variable; can be used to

return self.accountNumber #Returns randomly generated account number through the imported random module.

def setAccountNumber( integer):

self.account\_number = integer

# Method allows someone to change the value of the account\_number variable

def compare( integer):

isEqual = True

if integer is equal to or less then self.getBalance():

# Entered integer is compared to the balance variable

isEqual is set to True

elif integer is greater then self.getBalance():

isEqual is set to false

return isEqual #internal variable isEqual is returned with the result of the comparison between numbers.

class Item: #Item class which stores the cost of an item for purchase

def \_\_init\_\_(self reference, cost): #Constructor method that takes the name and cost of an item to be purchased

self.cost = cost #Entered data into constructor sets the cost variable of Item

#Since the objective of the program is to compare a transaction of items to ones credit limit all that’s really needed is a variable to store the cost of the items being purchased.

def totalCost(self, receipt=[]):

sum = 0

# variable to hold and pass calculated cost of each item as cost is added together.

for x in receipt:

sum is equal to sum plus x #As the list gets iterated, the costs with it are added to the sum variable until the list is fully iterated.

return sum # This returns the sum of the costs in the list.

def main(self):

array = [] # creates an empty list to place objects into.

account = BankAccount(whatever balance one desires)

# Creates a BankAccount object to be used in comparison to the total cost of items in order to determine if in fact there is a high enough

# balance in the credit limit.

# Items are created from inputted costs

item\_one = Item(whatever cost is entered)

item\_two = Item(whatever cost is entered)

item\_three = Item(whatever cost is entered)

# Items are loaded into the list using the append method of list

array.append(item\_one)

array.append(item\_two)

array.append(item\_three)

account.setAccountNumber(Whatever number the user provides will serve as the argument for the method)

# Transaction and comparison to account

print(“For account number: “, account.getAccountNum())

# Account number is printed after individual enters it for the transaction.

# totalCost of items is calculated and placed into variable

costOfItems = item\_one.totalCost(array)

if account.compare(costOfItems) is True:

print(“Purchase is Approved)”

elif account.compare(costOfItems) is False:

print(“Purchase is Denied. Insufficient funds.”)

# Cost of Items variable is plugged into the compare method of the account class. If the compare method returns true, the purchase is accepted

# and the individual receives confirmation of purchase

# if the compare method returns false, the individual receives a message stating the confirmation of purchase was denied due to insufficient

# funds.

if \_\_name\_\_ is ‘\_\_main\_\_’:

main()

# if main method exists then the main method is executed.