**customerApp**

#The class 'Customer' will store customer objects having certain attributes which holds values

class Customer():

#The method below is initializing the attributes of the customer

def \_\_init\_\_(self, name, age, plan, fee):

self.name = str(name[0:20])

self.age = int(age)

self.plan = str(plan)

self.fee = float(round(fee, 2))

#The method below is a getter for reading name attribute

def getname(self):

return self.name

#The method below is a getter for reading age attribute

def getage(self):

return self.age

#The method below is a getter for reading plan attribute

def getplan(self):

return self.plan

#The method below is a getter for reading fee attribute

def getfee(self):

return self.fee

#The method below is a setter for writing(setting new value) to

name attribute

def setname(self, newname):

self.name = newname

#The method below is a setter for writing(setting new value) to

age attribute

def setage(self, newage):

self.age = newage

#The method below is a setter for writing(setting new value) to

plan attribute

def setplan(self, newplan):

self.plan = newplan

#The method below is a setter for writing(setting new value) to

fee attribute

def setfee(self, newfee):

if newfee > 0:

self.fee = newfee

#The method below calculates data amount according to the plan and

fee input of a customer object

#if and elif statements in the method below allows different

calculations depending on the type of postpaid plan

def data(self):

if self.plan == "sp" or self.plan == "SP":

amtofdata = self.fee \* 0.9

elif self.plan == "ap" or self.plan == "AP":

amtofdata = self.fee \* 1.2

elif self.plan == "pp" or self.plan == "PP":

amtofdata = self.fee \* 1.5

return round(amtofdata)

#The method below will check whether the plan of one customer is

the same as the plan of another customer and if it's true it

will return True

def \_\_eq\_\_(self, aCustomer):

if self.getplan().lower() == aCustomer.getplan().lower():

return True

else:

return False

#The method below will check whether the data of one customer is

less than the data of another customer and if it's true it will

return True

def \_\_lt\_\_(self, aCustomer):

if self.data() < aCustomer.data():

return True

else:

return False

#The method below will check whether the fees of one customer is

less than or equal to the fees of another customer and if it's true

it will return True

def \_\_le\_\_(self, aCustomer):

if self.getfee() <= aCustomer.getfee():

return True

else:

return False

#The method below will return all the information of the customer

in the form of concatenated string

def \_\_str\_\_(self):

#The string to be concatenated and printed is determined by

the type of plan in if and elif statements below

if self.plan == "sp" or self.plan == "SP":

printplan = "saver plan"

elif self.plan == "ap" or self.plan == "AP":

printplan = "advanced plan"

elif self.plan == "pp" or self.plan == "PP":

printplan = "premium plan"

#self.fee in the concatenated string being returned below

is formatted to 2 decimal places

return str(self.name) + " " + str(self.age) + " " + printplan + \

" " + "RM" + "{:.2f}".format(self.fee)

#The class 'customerGroup' will store groups of customer objects in a list along with the group name attribute

class CustomerGroup():

#The method below is initializing the attributes of the customer group

def \_\_init\_\_(self, groupname):

self.groupname = str(groupname)

self.customers = []

#The method below is a getter for reading group name attribute

def getgroupname(self):

return self.groupname

#The method below is a getter for reading the list(self.customers)

def getcustomers(self):

return self.customers

#The method below is a setter for writing(setting new value)

to group name attribute

def setgroupname(self, newgroupname):

self.groupname = newgroupname

#The method below will add customer objects to the list(self.customers)

def addcustomer(self, aCustomer):

self.getcustomers().append(aCustomer)

#The method below will count the number of customer objects

currently stored in the list(self.customers) and return the value

def noOfCustomer(self):

number = len(self.customers)

return number

#The method below will return the total fees and data of all

customers currently stored in the list(self.customers) and

return the values

def totalValue(self):

total1 = 0

total2 = 0

#the for loop will loop through all the customers stored in the

list and add all the data and fee values

for i in self.getcustomers():

total1 += i.getfee()

total2 += i.data()

return int(round(total1)),int(round(total2))

#The method below will return the details(only name, age and data)

of a customer having a particular plan(the plan is determined

by the argument this method is accepting)

def findCustomerByType(self, aPlan):

details = ''

#The for loop will loop through all the customers stored in

the list(self.customers) and check if a particular plan matches

with the plan of one or more customers in the list

#Everytime the if condition is met, the details of those

customers will be accumulated and returned(only name, age and data)

for customer in self.getcustomers():

if customer.getplan().lower() == aPlan.lower():

details += str(customer.getname()) + " " + \

str(customer.getage()) + " " + str(customer.data()) + \

"GB" +'\n'

if details == ' ':

details = "not found"

return details

#The method below will return the details of all customers

currently stored in the list(self.customers)

def displayCustomer(self):

details= ''

#The for loop will loop through all the customers stored in the list

for customer in self.getcustomers():

details += str(customer) + '\n'

return details

#The method below will return the details of the customer having

the highest data value

def highestvalue(self):

max = 0

details = ' '

#The for loop will loop through all the customers stored

in the list(self.customers)

#Everytime a particular customer's data value is greater than

the current 'max' value, the 'max' will be updated to that value

for i in self.getcustomers():

if i.data() > max:

max = i.data()

details = str(i) + " " + str(i.data()) + "GB"

return details

#The method below will delete a customer and return that

customer's details(information)

def customerWithdraw(self, aNumber):

#If the designated number is greater than 0 and less than or equal

to the total numbers of customers stored in the list, the

customer located in index which is 1 less than the designated

number will be deleted

if aNumber > 0 and aNumber <= len(self.customers):

deletedcustomer = self.customers.pop(aNumber-1)

return str(deletedcustomer)

else:

return False

#The method below will save the details(information) of each customer

in a text file line by line

def saveToFile(self, fileName):

#with statement is used to open the file to write

#for every customer in the list(self.customers), their

individual details will be joined with 'ENTER'('\n') and

written on each line of the txt file

with open(fileName, 'w') as filecontent:

filecontent.write("\n".join(str(customer) for customer \

in self.customers))

filecontent.close()

#The method below will load the details(information) from each

line of a text file

def loadFromFile(self, fileName):

#with statement is used to open the file to read/load

#for each line in the txt file, they will be separated by '\n'

and added to the list(self.customers)

with open(fileName, 'r') as filecontent:

self.customers = [line.rstrip('\n') for line in open(fileName)]

filecontent.close()

**customerDriver**

#B1901850

#13/4/2020

from customerApp import \*

def menu():

print("Telco Subscription by Private Telco")

print("--------------------------------------------------------------")

print("1 Add a customer")

print("2 Display all customers")

print("3 Display total value of all postpaid subscription")

print("4 List number of customer in a particular group")

print("5 Display customers with user-specified postpaid plan")

print("6 Display customer with the most internet data in his/her plan")

print("7 Remove customer, based on index")

print("8 Read customer information from file")

print("9 Write customer information to file")

print("0 Quit")

print(" ")

def main():

groupName= input("Enter group's name:")

newgroup = CustomerGroup(groupName)

menu()

choice = input("Your choice?")

#while loop below will continue to loop through and ask for

input of choice until the input value is 0

while choice != "0":

if choice == "1":

customerName=input("Customer name?")

customerAge=int(input("Customer age?"))

#if statement below is checking whether the customer's age

falls between 18 and 70

if customerAge < 18 or customerAge > 70:

print("The customer's age must be between 18 and 70.")

else:

customerPlan=input("Postpaid plan('SP/AP/PP')?")

options = ['sp','SP','ap','AP','pp','PP']

#The customer will be asked to input a plan again and

again in the while loop below until the value is

valid(value is in the 'options' list)

while customerPlan not in options:

print("Invalid type! Please enter again!")

customerPlan=input("Postpaid plan('SP/AP/PP')?")

customerFee=int(input("Postpaid fee?"))

#The customer will be asked to input a fee again and again

in the while loop below until the value is greater than 0

while customerFee <= 0:

print("Invalid value! Please enter again!")

customerFee=int(input("Postpaid fee?"))

newgroup.addcustomer(Customer(customerName,customerAge, \

customerPlan,customerFee))

print("... Customer has been added successfully.")

elif choice == "2":

#if statement below will check whether the list is

currently empty or not

if len(newgroup.getcustomers()) == 0:

print("No customer registered in this group")

else:

print("All customers in this group:")

print(newgroup.displayCustomer())

elif choice == "3":

feetotal, datatotal=newgroup.totalValue()

print("Total value of all postpaid subscription is " \

+ "RM"+str(feetotal)+" and total internet \

data "+str(datatotal)+"GB")

elif choice == "4":

groupName= input("Enter group name:")

numbofcustom =newgroup.noOfCustomer()

print("Total customer in group "+"'"+groupName+"'" \

+":"+" "+str(numbofcustom))

elif choice == "5":

customerPlan=input("Postpaid plan('SP/AP/PP')?")

#the if and elif statements will allow strings to be

printed depending on the plan that the user searched

if customerPlan.lower() == "sp":

print("Customer with saver plan:")

elif customerPlan.lower() == "ap":

print("Customer with advanced plan:")

elif customerPlan.lower() == "pp":

print("Customer with premium plan:")

print(newgroup.findCustomerByType(customerPlan))

elif choice == "6":

print("Customer with the highest internet data:")

print(newgroup.highestvalue())

elif choice == "7":

#if statement below will check whether the list is

currently empty or not

if len(newgroup.getcustomers()) == 0:

print("No customer registered in this group")

else:

positiontodelete=int(input("Which customer to withdraw?"))

validcheck=newgroup.customerWithdraw(positiontodelete)

#if statement below will check whether False is returned

from the customerWithdraw method

if validcheck == False:

print("Invalid index. Try again!")

else:

#The deleted customer will be displayed in a

concatenated string if the deleted value is returned

print("Customer with \

index "+str(positiontodelete)+":"+"{"+validcheck+"} \

"+" has been removed!")

elif choice == "8":

nameoffile = input("Please enter filename to load from:")

newgroup.loadFromFile(nameoffile)

print("File loaded successfully")

elif choice == "9":

nameoffile = input("Please enter filename to save to:")

newgroup.saveToFile(nameoffile)

print("File saved successfully")

print(" ")

menu()

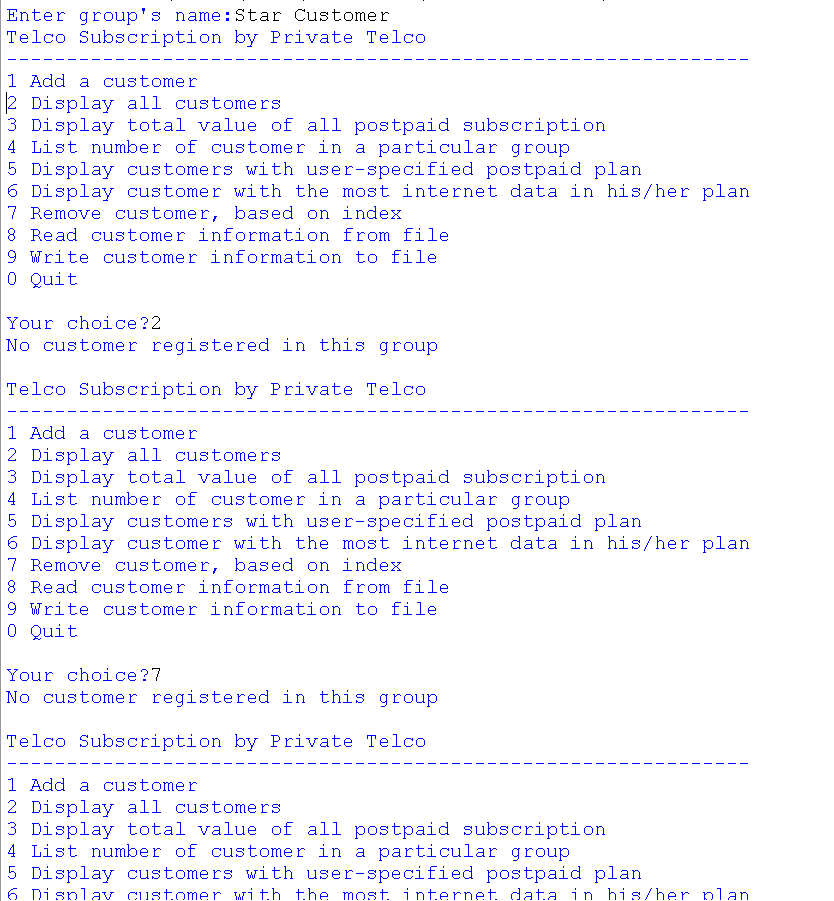
choice = input("Your choice?")

print(" ")

print("Thank you!")

main()

**Screenshots of outputs**

****





