## @author Mehmet ACIKGOZ

# This program simulates a simple pop machine that allows a user to

# purchase items and receive change.

# CODE: Import the Soda class

from Soda import Soda

def main() :

# use TARGET\_FILE in place of hard-coding in sodas.txt or sodas2.txt

# in main

TARGET\_FILE = "sodas.txt"

# CODE: Call fillMachine using TARGET\_FILE and store the returned

# list into a variable named popMachine

popMachine = fillMachine(TARGET\_FILE)

# Process purchases. The beginning of the loop shell is given

purchaseMoreSodas = "y"

while (purchaseMoreSodas[0].lower() == "y") :

# CODE: Loop to display each pop machine's soda information

# You are to implicitly use the the \_\_str\_\_ method in the Soda class

for i in range( len(popMachine) ):

print( popMachine[i].\_\_str\_\_() )

# CODE: Prompt for and retrieve the name of the item to purchase

nameOfItemToPurchase = input ("Name of item? ")

# CODE: Call searchMachine to find the index of the soda in the machine

# You will need to store the returned value into a descriptively

# name variable such as index.

res = searchMachine( popMachine, nameOfItemToPurchase )

# CODE: IF the soda was not found (returned index is -1), inform

# the user

if ( res is -1 ):

print("%s %s" %(nameOfItemToPurchase, "not found in the machine."))

# CODE: ELSE IF the soda is sold out, inform the user

elif ( res > 0 and (popMachine[res].getQuantity() == 0) ):

print("%s %s" %(nameOfItemToPurchase, "is sold out."))

# CODE: ELSE proceed to get the money from the user

else:

money = float( input("Please enter money: $") )

# CODE: LOOP to prompt for and retrieve money from the

# user until enough money is entered.

while ( money < popMachine[res].getPrice() ):

money = money + float( input("Please enter money: $") )

# CODE: Reduce the quantity in stock for the soda by calling

# its purchase method. Then, tell the user to take their soda

# using the name of the soda they entered

popMachine[res].purchase()

print( "%s %s" % ("Please take your", nameOfItemToPurchase) )

# CODE: Give change IF necessary

if ( money > popMachine[res].getPrice() ):

change = money - popMachine[res].getPrice()

print("%s %.2f" %("Please take your change of ", change) )

# Prompt for and retrieve if the user is to enter more sodas

purchaseMoreSodas = input("Purchase another Item (y or n)? ")

print()

#CODE: Call storeMachine to store the pop machine data back to TARGET\_FILE

storeMachine(TARGET\_FILE, popMachine)

print("Machine data stored. Goodbye.")

## fillMachine creates and fills a pop machine list of soda elements

# @param filename The name of the file containing the soda information.

# @precondition No soda name occurs twice in the file indicated by filename.

# @precondition The file associated with filename exists in the

# correct location.

# @return A list of soda objects

# CODE: Create the signature for fillMachine (the line with def) and then

# complete the definition.

def fillMachine(filename):

# CODE: Create an empty list that will store pop machine data

result = []

# CODE: Open the file specified by filename for reading

inFile = open(filename, 'r')

# CODE: LOOP to read each line from the file and APPEND a new Soda object

# based upon information from the line into the pop machine list.

cnt = 0

for line in inFile:

line = line.strip()

parts = line.split(",")

x = Soda()

x.setName(parts[0])

x.setPrice(float(parts[1]))

x.setQuantity(int(parts[2]))

result.append(x)

cnt += 1

# CODE: Close the file

inFile.close()

# CODE: Return the pop machine list

return result

## storeMachine stores pop machine data back to file

# @param popMachine The list of sodas

# @param filename The name of the file in which to store the soda information.

# CODE: Create the signature for storeMachine (the line with def) and then

# complete the definition.

def storeMachine(filename, list):

# CODE: Open the file specified by filename for writing

outFile = open(filename, 'w')

# CODE: Write each pop machine element to a separate line in the file.

# Use the get methods. Each line in the file will be of the form:

# Soda name,price,quantity

# The price will contain 2 digits of precision, but no $

# There is no space after the commas

for i in range( len(list) ):

outFile.write("%s,%.2f,%d \n" % ( list[i].getName(), list[i].getPrice(), list[i].getQuantity() ) )

# CODE: Close the file

outFile.close()

## searchMachine performs a case-insensitive search for a given name in a list

# @param list The list (pop machine list) to search

# @param name The name to match

# @return The index of value if found or -1 if not found.

# CODE: Create the signature for searchMachine (the line with def) and then

# complete the definition.

def searchMachine(list, name):

# CODE: Loop through the list to search for a case-insensitive match

# to the given value in the name parameter. Utilize the getName() method

# and the lower method.

# See the linear search in the chapter 6 notes for a search example.

index = - 1

for i in range( len(list) ):

if ( name.lower() == list[i].getName().lower() ):

index = i

break

return index

main()

#@author Mehmet ACIKGOZ

class Soda:

def \_\_init\_\_(self, name ="", price = 0, quantity = 0):

self.setName(name)

self.setPrice(price)

self.setQuantity(quantity)

# mutators

def setName(self, name):

self.\_name = name

def setPrice(self, price):

if (price > 0 ):

self.\_price = price

else:

self.\_price = 0

def setQuantity(self, quantity):

if (quantity > 0):

self.\_quantity = quantity

else:

self.\_quantity = 0

# accesors

def getName(self):

return self.\_name

def getPrice(self):

return self.\_price

def getQuantity(self):

return self.\_quantity

def purchase(self, amount = 1):

if ( (amount <= self.\_quantity) and (amount > 0) ):

self.\_quantity = self.\_quantity - amount

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

#result = self.\_name + ", " + "$"+ '{0:.2f}'.format(self.\_price) + ", " + str(self.\_quantity)

result = self.\_name + ", " + "$"+ str("%.2f" % self.\_price) + ", " + str(self.\_quantity)

return result