МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5**

по дисциплине «Программирование на языке Python»

на тему: «Работа с файлами»

Студент: Жозеф Веншенесс \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Шифр 190362

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Группа 92ПГ

Проверил: Захарова О.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отметка о зачете: «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_» Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Орел 2021

В программу лабораторной работы № 4 добавить главное меню с пунктами

«Файл», «Справка».

Пункт главного меню «Файл» должен включать подпункты «Создать» (очистка формы для ввода новых данных), «Открыть» (считывание из файла (по указанному пути) информации об объектах и отображение её на форме), «Сохранить» (обновление текущего файла; если файл не создан, то действия пункта «Сохранить как…»), «Сохранить как…»

(сохранение информации об объектах в файл по указанному пути), «Выход»

(дружественный выход из программы).

Пункт главного меню «Справка» должен показывать информацию о разработчике программы.

**Код программы**

import distutils.util

from tkinter import \*

from tkinter import messagebox

from tkinter.ttk import Radiobutton

from tkinter.ttk import Combobox

from tkinter import ttk

from collections import namedtuple

from tkinter import filedialog

import csv

class Medicament:

code = 1

name = 'paracetamol'

cost = 1000

receipt = True

description = 'braise, amoxiciline'

def \_\_init\_\_(self, code1, name1, cost1, receipt1, description1):

self.code = code1

self.name = name1

self.cost = cost1

self.receipt = receipt1

self.description = description1

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

return str(self.code) + ' ' + self.name + ' ' + str(self.cost) + ' ' + str(

self.receipt) + ' ' + self.description

def show(self):

print(self.code, self.name, self.cost, self.receipt, self.description)

def Get\_code(self):

return self.code

def Get\_name(self):

return self.name

def Get\_cost(self):

return self.cost

def Get\_description(self):

return self.description

def Get\_receipt(self):

return self.receipt

def click():

prix = 0

cod = int(comb1.get())

for i in range(len(medList)):

if cod == medList[i].Get\_code():

messagebox.showerror('Error input', 'This code already exist! Choose another one')

cod = 0

nom = ten2.get()

try:

prix = int(ten3.get())

except ValueError:

try:

prix = float(ten3.get())

except ValueError:

messagebox.showerror('Error input', 'The price can be an integer or a float')

descr = ten4.get()

recp = selected.get()

if not cod or not nom or not prix or not descr:

messagebox.showerror('Title', 'Fill all the blank space')

else:

med = Medicament(cod, nom, prix, recp, descr)

medList.append(med)

def DeList():

verif = False

try:

delCod = int(ten5.get())

except ValueError:

messagebox.showerror('Error input', 'The code should be an integer')

delCod = 0

if not delCod:

messagebox.showerror('Error', 'Fill the blank place')

else:

for i in range(len(medList)):

if delCod == medList[i].code:

del medList[i]

verif = True

break

if not verif:

messagebox.showerror('Error input', 'This code doesnt exit')

else:

messagebox.showinfo('Success', 'The code is successfully delete! Press Show again to see the change')

def show():

for i in tree.get\_children():

tree.delete(i)

for i in range(len(medList)):

tree.insert(parent='', index=0, values=(

medList[i].code, medList[i].name, medList[i].cost, medList[i].receipt, medList[i].description))

def showSpec():

for i in tree.get\_children():

tree.delete(i)

for i in range(len(medList)):

value=distutils.util.strtobool(medList[i].receipt)

verif=bool(value)

print(verif)

if not verif:

tree.insert(parent='', index=0, values=(

medList[i].code, medList[i].name, medList[i].cost, medList[i].receipt, medList[i].description))

def saveAs():

global file\_name

file\_name = filedialog.asksaveasfilename(title='Save File', filetypes=(("CSV files", "\*.csv"), ("All Files", "\*.\*")))

if file\_name:

if file\_name.endswith(".csv"):

pass

else:

file\_name = f'{file\_name}.csv'

print(file\_name)

myFile = open(file\_name, "w+", newline="")

writeIn = csv.writer(myFile, delimiter=',')

for item in medList:

writeIn.writerow([item.code, item.name, item.cost, item.receipt, item.description])

myFile.close()

def saveFile():

global file\_name

if not file\_name:

file\_name = filedialog.asksaveasfilename(title='Save File', filetypes=(("CSV files", "\*.csv"), ("All Files", "\*.\*")))

if file\_name:

if file\_name.endswith(".csv"):

pass

else:

file\_name = f'{file\_name}.csv'

print(file\_name)

myFile = open(file\_name, "w", newline="")

writeIn = csv.writer(myFile, delimiter=',')

for item in medList:

writeIn.writerow([item.code, item.name, item.cost, item.receipt, item.description])

else:

myFile = open(file\_name, "w", newline="")

writeIn = csv.writer(myFile, delimiter=',')

for item in medList:

writeIn.writerow([item.code, item.name, item.cost, item.receipt, item.description])

myFile.close()

def readFile():

file\_name = filedialog.askopenfilename(title='Open File', filetypes=(("CSV files", "\*.csv"), ("All Files", "\*.\*")))

medList.clear()

output\_file = open(file\_name, "r")

readIn = csv.reader(output\_file)

for item in readIn:

medList.append(Medicament(int(item[0]), item[1], item[2], item[3], item[4]))

for i in range(len(medList)):

tree.insert(parent='', index=0, values=(

medList[i].code, medList[i].name, medList[i].cost, medList[i].receipt, medList[i].description))

def newFile():

medList.clear()

for i in tree.get\_children():

tree.delete(i)

def exitProg():

window.destroy()

def info():

messagebox.showinfo('Success', 'This amazing program was written by WENCHENESSE JOSEPH\nThis program execute a lot of function related to '

'widget and file management')

window = Tk()

window.geometry('1000x700')

selected = BooleanVar()

selected.set(False)

medList = []

file\_name=""

menu = Menu(window)

newFunc = Menu(menu)

newFunc.add\_command(label='New', command=newFile)

newFunc.add\_separator()

newFunc.add\_command(label='Open', command=readFile)

newFunc.add\_separator()

newFunc.add\_command(label='Save', command=saveFile)

newFunc.add\_separator()

newFunc.add\_command(label='Save as', command=saveAs)

newFunc.add\_separator()

newFunc.add\_command(label='Exit', command=exitProg)

menu.add\_cascade(label='FILE', menu=newFunc)

newFunc1 = Menu(menu)

newFunc1.add\_command(label='About', command=info)

menu.add\_cascade(label='About the program', menu=newFunc1)

window.config(menu=menu)

col = ('code', 'name', 'price', 'receipt', 'description')

tree = ttk.Treeview(window, columns=col, show='headings')

tree.grid(row=14, column=0, sticky='nsew')

# define headings

tree.heading('code', text='Code')

tree.heading('name', text='Name')

tree.heading('price', text='price')

tree.heading('receipt', text='receipt')

tree.heading('description', text='description')

tree.column("code", anchor=W, width=80)

tree.column("name", anchor=W, width=100)

tree.column("price", anchor=W, width=80)

tree.column("receipt", anchor=W, width=80)

tree.column("description", anchor=W, width=100)

lb = Label(window, text="Information's card ", fg='red', font=('Times New Roman Bold', 18))

lb.grid(column=0, row=0)

l1 = Label(window, text="Code of the medicine ", font=('Times New Roman', 14))

l2 = Label(window, text="Name of the medicine ", font=('Times New Roman', 14))

l3 = Label(window, text="Price of the medicine ", font=('Times New Roman', 14), )

l4 = Label(window, text="Description ", font=('Times New Roman', 14))

l5 = Label(window, text=" ", font=('Times New Roman', 14))

l6 = Label(window, text="Enter the code of the medicine to delete", font=('Times New Roman', 14))

l7 = Label(window, text="See the medicine without receipt", font=('Times New Roman', 14))

l1.grid(column=0, row=1)

l2.grid(column=0, row=3)

l3.grid(column=0, row=5)

l4.grid(column=0, row=7)

l5.grid(column=0, row=9)

l6.grid(column=1, row=3)

l7.grid(column=1, row=12)

comb1 = Combobox(window)

comb1['values'] = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)

comb1.current(0)

ten2 = Entry(window, width=20, bd=3, justify=LEFT)

ten3 = Entry(window, width=20, bd=3, justify=LEFT)

ten4 = Entry(window, width=20, bd=3, justify=LEFT)

ten5 = Entry(window, width=20, bd=3, justify=LEFT)

rad1 = Radiobutton(window, text='Need receipt', value=True, variable=selected)

rad2 = Radiobutton(window, text="Don't need receipt", value=False, variable=selected)

comb1.grid(column=0, row=2)

ten2.grid(column=0, row=4)

ten3.grid(column=0, row=6)

ten4.grid(column=0, row=8)

ten5.grid(column=1, row=4)

rad1.grid(column=0, row=10)

rad2.grid(column=1, row=10)

btn = Button(window, text='Add new medicine', bg='#2186C1', fg='white', command=click)

btn1 = Button(window, text='Show all', bg='#2186C1', fg='white', command=show)

btn2 = Button(window, text='Delete', bg='#2186C1', fg='white', command=DeList)

btn3 = Button(window, text='No receipt', bg='#2186C1', fg='white', command=showSpec)

btn.grid(column=0, row=11)

btn1.grid(column=1, row=14)

btn2.grid(column=1, row=5)

btn3.grid(column=1, row=13)

window.mainloop()