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 № 4

на тему: «Разработка графического интерфейса»

по дисциплине: «Программирование на языке Python»

Вариант № 7

Выполнила: Карпикова С.П. Шифр: 170580

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверили: Захарова О.В., Раков В.И.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орел, 2019 г.

**Задание:**

Разработать и реализовать графический интерфейс для задания лабораторной работы № 3 (вариант в соответствии со списком студентов). Обязательные элементы графического интерфейса: надписи, кнопки, текстовые поля, выпадающий список, чекбоксы и/или радиокнопки, всплывающее окно с сообщением.

**Решение:**

from tkinter import \*

from tkinter.ttk import \*

from tkinter import messagebox as mb

class Medicament(object):

vendor\_code = None

name = None

cost = None

has\_prescription = False

description = None

def \_\_init\_\_(self, \*\*kwargs):

self.\_\_dict\_\_.update(kwargs)

def insert(self, table):

table.insert('', 'end', text=self.vendor\_code,

values=(

self.name,

self.cost,

'да' if self.has\_prescription else 'нет',

self.description

)

)

def without\_prescription(self, table):

if not self.has\_prescription:

self.insert(table)

class MainFrame(Frame):

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

Frame.\_\_init\_\_(self, parent)

self.parent = parent

self.medicaments = []

self.initUI()

self.bindEvents()

def initUI(self):

self.parent.title("Лекарства")

self.pack(fill=BOTH, expand=True)

self.frame\_add\_1 = Frame(self)

self.frame\_add\_1.pack(fill=X)

lbl\_vendor\_code = Label(self.frame\_add\_1, text="Артикул", width=8)

lbl\_vendor\_code.pack(side=LEFT, padx=5, pady=5)

self.entry\_vendor\_code = Entry(self.frame\_add\_1, width=15)

self.entry\_vendor\_code.pack(side=LEFT, padx=5)

lbl\_name = Label(self.frame\_add\_1, text="Название", width=10)

lbl\_name.pack(side=LEFT, anchor=N, padx=5, pady=5)

self.entry\_name = Entry(self.frame\_add\_1, width=25)

self.entry\_name.pack(side=LEFT, padx=5)

lbl\_cost = Label(self.frame\_add\_1, text="Стоимость", width=10)

lbl\_cost.pack(side=LEFT, anchor=N, padx=5, pady=5)

self.entry\_cost = Entry(self.frame\_add\_1, width=15)

self.entry\_cost.pack(side=LEFT, padx=5)

self.cvar1 = BooleanVar()

self.cvar1.set(0)

self.check\_prescription = Checkbutton(self.frame\_add\_1, text="Рецепт", variable=self.cvar1, onvalue=1, offvalue=0)

self.check\_prescription.pack(anchor=W)

self.frame\_add\_2 = Frame(self)

self.frame\_add\_2.pack(fill=X)

lbl\_desc = Label(self.frame\_add\_2, text="Описание", width=10)

lbl\_desc.pack(side=LEFT, anchor=N, padx=5, pady=5)

self.txt\_desc = Text(self.frame\_add\_2, width=25, height=3)

self.txt\_desc.pack(fill=BOTH, pady=5, padx=5)

self.btn\_add = Button(self.frame\_add\_2, text="Добавить", width=15)

self.btn\_add.pack()

self.frame\_remove = Frame(self)

self.frame\_remove.pack(fill=X)

lbl\_remove = Label(self.frame\_remove, text="Артикул", width=8)

lbl\_remove.pack(side=LEFT, padx=5, pady=5)

self.entry\_remove = Entry(self.frame\_remove, width=15)

self.entry\_remove.pack(side=LEFT, padx=5)

self.btn\_remove = Button(self.frame\_remove, text="Удалить", width=15)

self.btn\_remove.pack(side=LEFT, padx=5, pady=5)

self.frame\_table = Frame(self)

self.frame\_table.pack(fill=X)

self.table = Treeview(self.frame\_table)

self.table["columns"] = ("1", "2", "3", "4")

self.table.column("#0", width=80, minwidth=50)

self.table.column("1", width=100, minwidth=70)

self.table.column("2", width=80, minwidth=50)

self.table.column("3", width=50, minwidth=30)

self.table.column("4", width=120, minwidth=100)

self.table.heading("#0", text="Артикул", anchor=W)

self.table.heading("1", text="Название", anchor=W)

self.table.heading("2", text="Стоимость", anchor=W)

self.table.heading("3", text="Рецепт", anchor=W)

self.table.heading("4", text="Описание", anchor=W)

self.table.pack(side=TOP, fill=X)

self.main\_frame = Frame(self)

self.main\_frame.pack(fill=X)

lbl1 = Label(self.main\_frame, text="Меню", width=6)

lbl1.pack(side=LEFT, padx=5, pady=5)

self.v = StringVar()

self.v.trace('w', self.on\_combo\_change)

self.combo = Combobox(self.main\_frame, state='readonly', width=25, textvar=self.v)

self.combo['values'] = ("Добавить лекарство", "Удалить лекарство", "Вывод списка лекарств", "Лекарства без рецепта")

self.combo.current(0)

self.combo.pack(side=LEFT, padx=5, pady=5)

def bindEvents(self):

self.btn\_add.bind('<Button-1>', self.add\_new\_medicament)

self.btn\_remove.bind('<Button-1>', self.delete\_medicament)

def hide(self):

self.frame\_add\_1.pack\_forget()

self.frame\_add\_2.pack\_forget()

self.frame\_remove.pack\_forget()

self.frame\_table.pack\_forget()

def on\_combo\_change(self, index, value, op):

self.hide()

current\_state = self.combo.get()

if current\_state == "Добавить лекарство":

self.frame\_add\_1.pack(fill=X)

self.frame\_add\_2.pack(fill=X)

elif current\_state == "Удалить лекарство":

self.frame\_remove.pack(fill=X)

elif current\_state == "Вывод списка лекарств":

self.frame\_table.pack(fill=X)

self.get\_medicaments()

else:

self.frame\_table.pack(fill=X)

self.get\_without\_prescription()

def add\_new\_medicament(self, event):

vendor\_code = self.entry\_vendor\_code.get()

name = self.entry\_name.get()

cost = self.entry\_cost.get()

has\_prescription = bool(self.cvar1.get())

description = self.txt\_desc.get("1.0",END)

if len(vendor\_code) == 0 or len(name) == 0 or len(cost) == 0 or len(description) == 0:

mb.showerror("Ошибка", "Все поля должны быть заполнены!")

return

if not cost.isdigit():

mb.showerror("Ошибка", "Стоимость должна иметь численное значение!")

return

self.medicaments.append(Medicament(

vendor\_code=vendor\_code,

name=name,

cost=cost,

has\_prescription=has\_prescription,

description=description)

)

self.clear\_text()

mb.askokcancel("Успешно", "Информация добавлена!")

def get\_medicaments(self):

self.table.delete(\*self.table.get\_children())

for item in self.medicaments:

item.insert(self.table)

def get\_without\_prescription(self):

self.table.delete(\*self.table.get\_children())

for item in self.medicaments:

item.without\_prescription(self.table)

def clear\_text(self):

self.entry\_vendor\_code.delete(0, 'end')

self.entry\_name.delete(0, 'end')

self.entry\_cost.delete(0, 'end')

self.cvar1.set(0)

self.txt\_desc.delete("1.0",END)

self.entry\_remove.delete(0, 'end')

def delete\_medicament(self, event):

vendor\_code = self.entry\_remove.get()

if len(vendor\_code) == 0:

mb.showerror("Ошибка", "Заполните поле!")

return

self.medicaments = [item for item in self.medicaments if item.vendor\_code != vendor\_code]

self.clear\_text()

mb.askokcancel("Успешно", "Удалено!")

def main():

root = Tk()

root.geometry("700x300")

app = MainFrame(root)

root.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

main()