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()
```