Лабораторная работа № 5

Заполнение товарного чека

Задача: Создать оконное приложение для ведения диалога с пользователем. Организовать программное заполнение товарных чеков данными, вводимыми с клавиатуры.

Порядок выполнения задания:

I. Открыть документ, содержащий образец *товарного чека* (см. лабораторную работу 1). Создать на его основе шаблон (рисунок 1).

Наименование организации (ФИО индивидуального предпринимателя):

ИНН: ???

Адрес: ???

Товарный чек № {{Number}} от {{Date}} года

№	Артикул	Наименование товара	Ед. изм.	Кол-во	Цена за 1 ед. (руб.)	Сумма (руб.)

Всего наименований {{Count}} на сумму {{Summa}}

Продавец {{FIO}}

подпись: (инициалы, фамилия)

Рисунок 1 – Образец шаблона товарного чека

II. Ввести реквизиты «вашей» организации (выделены красным цветом).

III. Сохранить документ под именем **Sales_receipt** в формате **.docx** в папке Шаблоны.

IV. Внутри папки с предыдущим проектом (см. лабораторные работы 2 и 3) создать проект для ввода данных с клавиатуры и заполнения документа «Товарный чек».
Примечание: расположение проекта может быть другим, но в этом случае нужно правильно определить пути доступа к файлам, нужным для работы.

V. Добавить в проект три файла с описанием классов (**Product.py**, **Packing_list.py**, **Interface.py**) и файл с основным исполняемым кодом (**Main.py**):

Код файла **Product.py**:

```
class Product:
    """
    Класс, описывающий данные о товаре.
    """

    def __init__(self, code, name, unit, count, price):
        # Это конструктор.
        # В нем характеристики товара:
        # артикул, наименование, ед.измерения, количество и цена,
        # определяют поля для экземпляров класса.
        self.code = code
        self.name = name
        self.unit = unit
        self.count = count
        self.price = price

    @property
    def sum(self):
        # Это свойство.
        # Вычисление суммы товара.
        return self.count * self.price

    def __str__(self):
        # Формат вывода данных
        # при обращении к экземпляру класса.
        return
f"{self.code}\t{self.name}\t{self.unit}\t{self.count}\t{self.price}\t{
self.sum}"
```

Код файла **Packing_list.py**:

```
import docx
import datetime
from docxtpl import DocxTemplate
# Внешний модуль необходимо добавить к интерпретатору
# (в проекте на PyCharm или в командной строке).
from number_to_string import get_string_by_number
from Product import Product

class Pack:
    """
    Класс, описывающий создание списка данных
    для заполнения товарного чека
    и создания текстового файла на основе шаблона.
    """

    def __init__(self):
        # Очистка списка данных о товаре.
        self.clear()

    def clear(self):
        # Создание полей, доступных только внутри класса.
        # Список товаров.
        self.  prod = []
```

```
        # Словарь для замены данных в файле.
        self.__context = {}

    def add(self, code, name, unit, count, price):
        # Добавление данных о товаре в список.
        self.__prod.append(Product(code, name, unit, count, price))

    def __getitem__(self, item):
        # Индексатор (итератор) для доступа
        # к элементам списка вне класса.
        return self.__prod[item]

    def __setitem__(self, key, value):
        # Индексатор для редактирования
        # элементов списка вне класса.
        self.__prod[key] = value

    def __len__(self):
        # Количество элементов в списке.
        return len(self.__prod)

    def save(self, number, fio):
        # Сохранение данных в файле.
        self.__context["Number"] = number
        self.__context["FIO"] = fio
        # Вызов инкапсулированных методов класса.
        self.__save_table()
        self.__save_data()

    def __save_table(self):
        # Открытие шаблона документа для редактирования
        # с помощью библиотеки docx
        document = docx.Document("..\\Шаблоны\\Sales_receipt.docx")
        # Доступ к таблице документа.
        table = document.tables[0]
        # Номер строки с данными о товаре в таблице.
        i = 0
        # Общая сумма.
        summa = 0
        for prod in self.__prod:
            # prod - данные о выводимом в таблицу товаре.
            # Добавление строки в таблицу.
            row = table.add_row()
            i += 1
            # Заполнение строки данными.
            row.cells[0].text = str(i)
            row.cells[1].text = prod.code
            row.cells[2].text = prod.name
            row.cells[3].text = prod.unit
            row.cells[4].text = str(prod.count)
            row.cells[5].text = str(prod.price)
            row.cells[6].text = str(prod.sum)
            summa += prod.sum
```

```
        # Сохранение документа под новым именем
        # (промежуточная версия документа).
        document.save("test1.docx")
        # Добавление данных словарь для замены переменных.
        self.__context["Count"] = str(i)
        self.__context["Summa"] = get_string_by_number(summa)

    def __save_data(self):
        # Открытие шаблона документа для редактирования
        # с помощью библиотеки docxtpl.
        doc = DocxTemplate("test1.docx")
        today = datetime.datetime.today()
        # Добавление данных словарь для замены переменных.
        self.__context["Date"] = today.strftime("%d.%m.%Y")
        doc.render(self.__context)
        # Сохранение готового документа
        doc.save("..\\Документы\\Товарный чек (" +
 self.__context["Number"] + ").docx")
```

Код файла **Interface.py**:

```
import tkinter as tk
from Packing_list import Pack

class Interface:
    """
    Класс создания интерфейса окна.
    """
    def __init__(self, form):
        self.form = form
        # Создание экземпляра класса для формирования товарного чека.
        self.prod = Pack()
        # Порядковый номер добавленного товара.
        # Начало отсчета.
        self.count = 0
        self.widgets()

    def widgets(self):
        # Добавление виджетов на форму.
        # grid - упаковщик (размещает данные в виде таблицы).
        tk.Label(self.form, text="Введите данные о товаре",
font="20").grid(row=0, column=0)
        # Создание списка текстовых полей.
        self.entry = [0] * 7
        # Создание списка подписей (меток).
        label = ["Артикул", "Наименование товара", "Ед. изм.",
"Кол-во", "Цена (руб.)"]
        for i in range(5):
            tk.Label(self.form, text=label[i], font="20").grid(row=1,
column=i, padx=5)
            self.entry[i] = tk.Entry(self.form, font="Arial 12")
            self.entry[i].grid(row=2, column=i, padx=5)
        # Метка нужна для разделения данных на форме.
        tk.Label(self.form).grid(row=3, column=1)
        # Добавление кнопок на форму.
        self.button_add = self.do_button(4, 1, "Добавить", self.add)
```

```
        self.button_save = self.do_button(4, 2, "Сохранить",
self.save)
        self.button_clear = self.do_button(4, 3, "Очистить",
self.clear)
        # Метка для разделения данных на форме.
        tk.Label(self.form).grid(row=6, column=1)
        tk.Label(self.form, text="Товарный чек",
font="20").grid(row=7, column=1)
        # Номер товарного чека.
        self.entry[5]=tk.Entry(self.form, font="Arial 12")
        self.entry[5].grid(row=7, column=2)
        # Многострочный вывод текста.
        self.text = tk.Text(self.form, height=20, width=80,
font="Arial 12", wrap=tk.WORD)
        self.text.grid(row=8, column=0, columnspan=4)
        # Данные о продавце для заполнения товарного чека.
        tk.Label(self.form, text="Продавец", font="20").grid(row=9,
column=1)
        self.entry[6]=tk.Entry(self.form, font="Arial 12")
        self.entry[6].grid(row=9, column=2)

    def do_button(self, row, column, text, func):
        # Метод для формирования кнопок.
        button = tk.Button(self.form, text=text, font="Arial 12",
command=func)
        button.grid(row=row, column=column, padx=15)
        return button

    def add(self):
        # Метод для добавления данных в товарный чек и поле text.
        self.prod.add(self.entry[0].get(), self.entry[1].get(),
                      self.entry[2].get(), int(self.entry[3].get()),
                      int(self.entry[4].get()))
        self.count += 1
        self.text.insert(0.0, f"{self.count})  {self.prod[-1]}\n")

    def clear(self):
        # Метод для очитска товарного чека и поля text.
        self.prod.clear()
        self.text.delete(0.0, tk.END)
        self.count = 0

    def save(self):
        # Метод для сохранения данных.
        self.prod.save(self.entry[5].get(), self.entry[6].get())
```

Код файла **Main.py**:

```
import tkinter as tk
from Interface import Interface

root = tk.Tk()
root.geometry("1000x600")
pack_list = Interface(root)
root.mainloop()
```

Примечание:

Для того чтобы не усложнять программный код, в нем отсутствует проверка корректности вводимых данных, в том числе номеров товарных чеков.

Дополнительные задания:

1) Добавить в класс **Packing_list.py** методы для удаления товара (последнего в списке) и редактирования данных о товаре (по номеру элемента).
2) Добавить на форму кнопки для выполнения этих операций и связать их с методами класса.