**Практическая работа №6. Создание оконного приложения на основе класса базы данных.**

Выполнил студент группы исп21.1а Жигач Дмитрий

Задание.

1. Создать оконное приложение по своей предметной области.
2. Для отображения данных из БД использовать виджет Treeview.
3. Использовать класс, описанный в ПР№4 (из консольного приложения).
4. Предметные области см. в общей папке
5. C:\Сетевая аудитории\МДК 01.01\PyCharm TKinter

1.Модуль database.py

import sqlite3

class DB:

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

self.conn = sqlite3.connect("mybooks.db")

self.cur = self.conn.cursor()

self.cur.execute(

"CREATE TABLE IF NOT EXISTS product (id INTEGER PRIMARY KEY, company\_name TEXT, product\_name TEXT, unit\_of\_measurement TEXT, purchase\_price INTEGER, delivery\_date DATE, volume INTEGER, cost INTEGER)")

self.conn.commit()

def \_\_del\_\_(self):

self.conn.close()

def view(self):

self.cur.execute("SELECT \* FROM product")

rows = self.cur.fetchall()

return rows

def insert(self, company\_name, product\_name, unit\_of\_measurement, purchase\_price, delivery\_date, volume, cost):

self.cur.execute("INSERT INTO product VALUES (NULL,?,?,?,?,?,?,?)", (company\_name, product\_name, unit\_of\_measurement, purchase\_price, delivery\_date, volume, cost))

self.conn.commit()

def update(self, id, company\_name, product\_name, unit\_of\_measurement, purchase\_price, delivery\_date, volume, cost):

self.cur.execute("UPDATE product SET company\_name=?, product\_name=?, unit\_of\_measurement=?, purchase\_price=?, delivery\_date=?, volume=?, cost=? WHERE id=?", (company\_name, product\_name, unit\_of\_measurement, purchase\_price, delivery\_date, volume, cost, id,))

self.conn.commit()

def delete(self, id):

self.cur.execute("DELETE FROM product WHERE id=?", (id,))

self.conn.commit()

def search(self, company\_name=""):

self.cur.execute("SELECT \* FROM product WHERE company\_name=?", (company\_name,))

rows = self.cur.fetchall()

return rows

2. Модуль main.py

from tkinter import \*

from tkinter import messagebox

from tkinter import ttk # Импорт модуля ttk для Treeview

from database import DB

db = DB()

global selected\_tuple

def get\_selected\_row(event):

global selected\_tuple

index = tree.focus()

selected\_tuple = tree.item(index, 'values')

e1.delete(0, END)

e1.insert(END, selected\_tuple[0])

e2.delete(0, END)

e2.insert(END, selected\_tuple[1])

e3.delete(0, END)

e3.insert(END, selected\_tuple[2])

def view\_command():

tree.delete(\*tree.get\_children()) # Очистить существующие данные

for row in db.view():

tree.insert('', END, values=row)

def search\_command():

tree.delete(\*tree.get\_children()) # Очистить существующие данные

for row in db.search(product\_text.get()):

tree.insert('', END, values=row)

def add\_command():

db.insert(

product\_text.get(),

price\_text.get(),

comment\_text.get(),

mine\_text.get(),

date\_text.get(),

volume\_text.get(),

cost\_text.get()

)

view\_command()

def delete\_command():

if selected\_tuple:

db.delete(selected\_tuple[0])

view\_command()

else:

messagebox.showwarning("", "Выберите элемент для удаления")

def update\_command():

if selected\_tuple:

db.update(

selected\_tuple[0],

product\_text.get(),

price\_text.get(),

comment\_text.get(),

mine\_text.get(),

date\_text.get(),

volume\_text.get(),

cost\_text.get()

)

view\_command()

else:

messagebox.showwarning("", "Выберите элемент для обновления")

window = Tk()

window.title("Сельские компании")

# Устанавливаем тёмную тему

window.configure(bg='#1E1E1E')

window.option\_add('\*TButton\*background', '#1E1E1E')

window.option\_add('\*TButton\*foreground', '#FFFFFF')

window.option\_add('\*TEntry\*background', '#363636')

window.option\_add('\*TEntry\*foreground', '#FFFFFF')

window.option\_add('\*TListbox\*background', '#1E1E1E')

window.option\_add('\*TListbox\*foreground', '#FFFFFF')

def on\_closing():

if messagebox.askokcancel("", "Закрыть программу?"):

window.destroy()

window.protocol("WM\_DELETE\_WINDOW", on\_closing)

l1 = Label(window, text="Имя компании", bg='#1E1E1E', fg='#FFFFFF')

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

l2 = Label(window, text="Продукт", bg='#1E1E1E', fg='#FFFFFF')

l2.grid(row=0, column=2)

l3 = Label(window, text="Еденица измерения", bg='#1E1E1E', fg='#FFFFFF')

l3.grid(row=1, column=0)

l4 = Label(window, text="Цена продажи", bg='#1E1E1E', fg='#FFFFFF')

l4.grid(row=1, column=2)

l5 = Label(window, text="Дата доставки", bg='#1E1E1E', fg='#FFFFFF')

l5.grid(row=2, column=0)

l6 = Label(window, text="Количество", bg='#1E1E1E', fg='#FFFFFF')

l6.grid(row=2, column=2)

l7 = Label(window, text="Общая цена", bg='#1E1E1E', fg='#FFFFFF')

l7.grid(row=3, column=0)

product\_text = StringVar()

e1 = Entry(window, textvariable=product\_text, bg='#363636', fg='#FFFFFF')

e1.grid(row=0, column=1)

price\_text = StringVar()

e2 = Entry(window, textvariable=price\_text, bg='#363636', fg='#FFFFFF')

e2.grid(row=0, column=3)

comment\_text = StringVar()

e3 = Entry(window, textvariable=comment\_text, bg='#363636', fg='#FFFFFF')

e3.grid(row=1, column=1)

mine\_text = StringVar()

e4 = Entry(window, textvariable=mine\_text, bg='#363636', fg='#FFFFFF')

e4.grid(row=1, column=3)

date\_text = StringVar()

e5 = Entry(window, textvariable=date\_text, bg='#363636', fg='#FFFFFF')

e5.grid(row=2, column=1)

volume\_text = StringVar()

e6 = Entry(window, textvariable=volume\_text, bg='#363636', fg='#FFFFFF')

e6.grid(row=2, column=3)

cost\_text = StringVar()

e7 = Entry(window, textvariable=cost\_text, bg='#363636', fg='#FFFFFF')

e7.grid(row=3, column=1)

# Создание Treeview виджета

tree = ttk.Treeview(window, columns=(1, 2, 3, 4, 5, 6, 7, 8), show='headings', height=25)

tree.grid(row=4, column=0, rowspan=6, columnspan=2)

# Установка заголовков столбцов

tree.heading(1, text='id')

tree.heading(2, text='Имя компании')

tree.heading(3, text='Продукт')

tree.heading(4, text='Еденица измерения')

tree.heading(5, text='Цена продажи')

tree.heading(6, text='Дата доставки')

tree.heading(7, text='Количество')

tree.heading(8, text='Общая цена')

# Добавление вертикальной полосы прокрутки

vsb = ttk.Scrollbar(window, orient="vertical", command=tree.yview)

vsb.grid(row=4, column=2, rowspan=6, sticky='ns')

tree.configure(yscrollcommand=vsb.set)

# Привязка события для выбора строки в Treeview

tree.bind('<<TreeviewSelect>>', get\_selected\_row)

b1 = Button(window, text="Посмотреть все", width=12, command=view\_command, bg='#1E1E1E', fg='#FFFFFF')

b1.grid(row=4, column=3)

b2 = Button(window, text="Поиск", width=12, command=search\_command, bg='#1E1E1E', fg='#FFFFFF')

b2.grid(row=5, column=3)

b3 = Button(window, text="Добавить", width=12, command=add\_command, bg='#1E1E1E', fg='#FFFFFF')

b3.grid(row=6, column=3)

b4 = Button(window, text="Обновить", width=12, command=update\_command, bg='#1E1E1E', fg='#FFFFFF')

b4.grid(row=7, column=3)

b5 = Button(window, text="Удалить", width=12, command=delete\_command, bg='#1E1E1E', fg='#FFFFFF')

b5.grid(row=8, column=3)

b6 = Button(window, text="Закрыть", width=12, command=on\_closing, bg='#1E1E1E', fg='#FFFFFF')

b6.grid(row=9, column=3)

view\_command()

window.mainloop()

3. Результат

