





import tkinter as tk

from tkinter import ttk, messagebox, filedialog

class TransportApp(tk.Tk):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title("Система управления перевозками")

        self.geometry("800x600")

        self.create\_main\_menu()

        self.data = {"заводы": {}, "стройплощадки": {}, "затраты\_на\_перевозку": {}}

        self.transport\_plan = None

        self.total\_cost = None

    def create\_main\_menu(self):

        # Меню

        main\_menu = tk.Menu(self)

        self.config(menu=main\_menu)

        data\_menu = tk.Menu(main\_menu, tearoff=0)

        main\_menu.add\_cascade(label="Управление Данными", menu=data\_menu)

        data\_menu.add\_command(label="Ввод данных о заводах", command=self.open\_factory\_input)

        data\_menu.add\_command(label="Ввод данных о стройплощадках", command=self.open\_construction\_input)

        data\_menu.add\_command(label="Ввод затрат на перевозку", command=self.open\_transport\_cost\_input)

        planning\_menu = tk.Menu(main\_menu, tearoff=0)

        main\_menu.add\_cascade(label="Планирование Перевозок", menu=planning\_menu)

        planning\_menu.add\_command(label="Расчет оптимального плана перевозок", command=self.calculate\_transport\_plan)

        report\_menu = tk.Menu(main\_menu, tearoff=0)

        main\_menu.add\_cascade(label="Отчетность", menu=report\_menu)

        report\_menu.add\_command(label="Отображение плана перевозок", command=self.display\_transport\_plan)

        report\_menu.add\_command(label="Отображение общих затрат", command=self.display\_total\_cost)

        report\_menu.add\_command(label="Экспорт отчета", command=self.export\_report)

    def open\_factory\_input(self):

        # Окно для ввода данных о заводах

        factory\_window = tk.Toplevel(self)

        factory\_window.title("Ввод данных о заводах")

        tk.Label(factory\_window, text="Название завода").grid(row=0, column=0, sticky="w")

        factory\_name\_entry = tk.Entry(factory\_window)

        factory\_name\_entry.grid(row=0, column=1)

        tk.Label(factory\_window, text="Производительность").grid(row=1, column=0, sticky="w")

        factory\_prod\_entry = tk.Entry(factory\_window)

        factory\_prod\_entry.grid(row=1, column=1)

        def save\_factory():

            factory\_name = factory\_name\_entry.get()

            try:

                production\_capacity = int(factory\_prod\_entry.get())

                self.data["заводы"][factory\_name] = {"производительность": production\_capacity}

                messagebox.showinfo("Успех", f"Завод {factory\_name} добавлен.")

                factory\_window.destroy()

            except ValueError:

                messagebox.showerror("Ошибка", "Введите корректное число для производительности.")

        tk.Button(factory\_window, text="Сохранить", command=save\_factory).grid(row=2, column=0, columnspan=2, pady=10)

    def open\_construction\_input(self):

        # Окно для ввода данных о стройплощадках

        construction\_window = tk.Toplevel(self)

        construction\_window.title("Ввод данных о стройплощадках")

        tk.Label(construction\_window, text="Название стройплощадки").grid(row=0, column=0, sticky="w")

        construction\_name\_entry = tk.Entry(construction\_window)

        construction\_name\_entry.grid(row=0, column=1)

        tk.Label(construction\_window, text="Потребность").grid(row=1, column=0, sticky="w")

        construction\_demand\_entry = tk.Entry(construction\_window)

        construction\_demand\_entry.grid(row=1, column=1)

        def save\_construction():

            construction\_name = construction\_name\_entry.get()

            try:

                demand = int(construction\_demand\_entry.get())

                self.data["стройплощадки"][construction\_name] = {"потребность": demand}

                messagebox.showinfo("Успех", f"Стройплощадка {construction\_name} добавлена.")

                construction\_window.destroy()

            except ValueError:

                messagebox.showerror("Ошибка", "Введите корректное число для потребности.")

        tk.Button(construction\_window, text="Сохранить", command=save\_construction).grid(row=2, column=0, columnspan=2, pady=10)

    def open\_transport\_cost\_input(self):

         # Окно для ввода данных о затратах на перевозку

        transport\_cost\_window = tk.Toplevel(self)

        transport\_cost\_window.title("Ввод затрат на перевозку")

        tk.Label(transport\_cost\_window, text="Завод").grid(row=0, column=0, sticky="w")

        factory\_combo = ttk.Combobox(transport\_cost\_window, values=list(self.data["заводы"].keys()))

        factory\_combo.grid(row=0, column=1)

        tk.Label(transport\_cost\_window, text="Стройплощадка").grid(row=1, column=0, sticky="w")

        construction\_combo = ttk.Combobox(transport\_cost\_window, values=list(self.data["стройплощадки"].keys()))

        construction\_combo.grid(row=1, column=1)

        tk.Label(transport\_cost\_window, text="Стоимость").grid(row=2, column=0, sticky="w")

        cost\_entry = tk.Entry(transport\_cost\_window)

        cost\_entry.grid(row=2, column=1)

        def save\_cost():

           factory\_name = factory\_combo.get()

           construction\_name = construction\_combo.get()

           try:

               cost = float(cost\_entry.get())

               if factory\_name and construction\_name:

                  if factory\_name not in self.data["затраты\_на\_перевозку"]:

                      self.data["затраты\_на\_перевозку"][factory\_name] = {}

                  self.data["затраты\_на\_перевозку"][factory\_name][construction\_name] = cost

                  messagebox.showinfo("Успех", f"Стоимость перевозки от {factory\_name} к {construction\_name} добавлена.")

                  transport\_cost\_window.destroy()

               else:

                   messagebox.showerror("Ошибка", "Выберите завод и стройплощадку.")

           except ValueError:

               messagebox.showerror("Ошибка", "Введите корректное число для стоимости.")

        tk.Button(transport\_cost\_window, text="Сохранить", command=save\_cost).grid(row=3, column=0, columnspan=2, pady=10)

    def calculate\_transport\_plan(self):

        # Здесь будет реализация алгоритма расчета плана перевозок (пока заглушка)

        if not self.data["заводы"] or not self.data["стройплощадки"] or not self.data["затраты\_на\_перевозку"]:

            messagebox.showerror("Ошибка", "Необходимо ввести данные о заводах, стройплощадках и затратах на перевозку.")

            return

        self.transport\_plan = "Заглушка для плана перевозок"

        self.total\_cost = 1000  # Заглушка

        messagebox.showinfo("Успех", "План перевозок расчитан (заглушка).")

    def display\_transport\_plan(self):

         # Показывает рассчитанный план (заглушка)

        if self.transport\_plan:

             messagebox.showinfo("План перевозок", self.transport\_plan)

        else:

            messagebox.showinfo("Уведомление", "План перевозок еще не рассчитан.")

    def display\_total\_cost(self):

         # Показывает общие затраты (заглушка)

        if self.total\_cost is not None:

            messagebox.showinfo("Общие затраты", f"Общие затраты: {self.total\_cost}")

        else:

            messagebox.showinfo("Уведомление", "Общие затраты еще не рассчитаны.")

    def export\_report(self):

        # Экспорт отчета в файл

        if self.transport\_plan and self.total\_cost is not None:

           file\_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Текстовые файлы", "\*.txt")])

           if file\_path:

                try:

                    with open(file\_path, "w", encoding="utf-8") as file:

                        file.write("Отчет о плане перевозок\n\n")

                        file.write("План перевозок:\n")

                        file.write(f"{self.transport\_plan}\n")

                        file.write(f"\nОбщие затраты: {self.total\_cost}\n")

                    messagebox.showinfo("Успех", "Отчет экспортирован успешно.")

                except Exception as e:

                    messagebox.showerror("Ошибка", f"Не удалось экспортировать отчет: {e}")

        else:

           messagebox.showinfo("Уведомление", "Необходимо сперва рассчитать план и затраты.")

if \_\_name\_\_ == "\_\_main\_\_":

    app = TransportApp()

    app.mainloop()