import tkinter as tk

import tkinter.messagebox

from tkinter import ttk,messagebox, Scrollbar

from PIL import Image, ImageTk

from collections import defaultdict, Counter

import os

from tkinter.scrolledtext import ScrolledText

class PizzaOrderSystem:

    def \_\_init\_\_(self, master):

        self.master = master

        self.master.title("피자가게 주문 프로그램")

        # Load background image

        bg\_image = Image.open("restaurant.png")

        bg\_image = bg\_image.resize((self.master.winfo\_screenwidth(), self.master.winfo\_screenheight()), Image.ANTIALIAS)

        bg\_image = ImageTk.PhotoImage(bg\_image)

        # Displaying background image

        bg\_label = tk.Label(self.master, image=bg\_image)

        bg\_label.image = bg\_image

        bg\_label.place(relwidth=1, relheight=1)

        # Ingredient Initialization

        self.ingredients = defaultdict(int)

        self.ingredients['양파'] = 19

        self.ingredients['피망'] = 16

        self.ingredients['페퍼로니'] = 13

        self.ingredients['불고기'] = 13

        self.ingredients['베이컨'] = 15

        self.ingredients['망고'] = 13

        self.ingredients['키위'] = 12

        # Change Set Menu Label Code

        self.set\_menu\_panel = tk.Frame(self.master, bg="#FFFFFF")

        self.set\_menu\_label = tk.Label(self.set\_menu\_panel, text="세트 메뉴로 변경하시겠습니까?", font=('Helvetica', 20), bg="#FFFFFF")

        self.set\_menu\_label.pack(pady=10)

        self.set\_menu\_yes\_button = tk.Button(self.set\_menu\_panel, text="예", command=self.change\_set\_menu)

        self.set\_menu\_yes\_button.pack(side=tk.LEFT, padx=10)

        self.set\_menu\_no\_button = tk.Button(self.set\_menu\_panel, text="아니오", command=self.hide\_set\_menu\_panel)

        self.set\_menu\_no\_button.pack(side=tk.RIGHT, padx=10)

        self.set\_menu\_panel.pack\_forget()

        # Menu Initialization

        self.menu = {

            '치즈피자': {'재료': ['양파'], '가격': 11000, '이미지': '치즈 피자.png'},

            '페퍼로니피자': {'재료': ['양파', '페퍼로니'], '가격': 12000, '이미지': '페퍼로니 피자.png'},

            '불고기피자': {'재료': ['불고기', '피망'], '가격': 12000, '이미지': '불고기 피자.png'},

            '베이컨피자': {'재료': ['베이컨', '피망'], '가격': 13000, '이미지': '베이컨 피자.png'},

            '크림스파게티': {'재료': ['양파', '베이컨'], '가격': 9000, '이미지': '크림 스파게티.png'},

            '불고기스파게티': {'재료': ['피망', '불고기'], '가격': 10000, '이미지': '불고기 스파게티.png'},

            '망고에이드': {'재료': ['망고'], '가격': 4000, '이미지': '망고에이드.png'},

            '키위에이드': {'재료': ['키위'], '가격': 4000, '이미지': '키위에이드.png'}

        }

        self.set\_menu\_A = [('불고기피자', 1), ('크림스파게티', 1), ('망고에이드', 1)]

        self.set\_menu\_B = [('베이컨피자', 1), ('불고기스파게티', 1), ('키위에이드', 1)]

        # Initializing Order Variables

        self.current\_order = []

        self.total\_orders = []

        # Current Customer Number

        self.current\_customer = 1

        # UI Configuration

        self.create\_widgets()

        # # Create additional scrollbar after self.result widget

        scrollbar = tkinter.Scrollbar(self.master, orient="vertical")  # Create vertical scroll bar

        scrollbar.place(x=1020, y=300, height=480)  # Scrolling bar placement

        # Link scrollbar to self.result\_text

        scrollbar.config(command=self.result\_text.yview)  # Scroll bar settings

        self.result\_text.configure(yscrollcommand=scrollbar.set)  # Set scrollbar in self.result\_text

    # Verify that the contents of both lists are the same

    def lists\_equal(lst1, lst2):

        return Counter(lst1) == Counter(lst2)

    def complete\_current\_order(self):

        # Verify that the current order is empty

        if self.current\_order\_matches\_set\_menu():

            if self.lists\_equal(self.current\_order, self.set\_menu\_A):

                # Set Menu A Pannels Floating

                self.show\_set\_menu\_panel()

            elif self.lists\_equal(self.current\_order, self.set\_menu\_B):

                # Set Menu B Pannels Floating

                self.show\_set\_menu\_panel()

            else:

                # Output Order Content

                result\_text = f"\n<{self.current\_customer}번째 손님 주문 내역>\n\n"

                total\_price = sum(order[1] for order in self.current\_order)

                for order in self.current\_order:

                    result\_text += f"{order[0]} - {order[1]}원\n"

                    self.total\_orders.append((self.current\_customer, order[0], order[1]))

                # Output to the result label

                result\_text += f"총합: {total\_price}원"

                self.result\_text.insert(tk.END,result\_text+'\n','center')

                self.result\_text.tag\_configure('center', justify='center')

                self.result\_text.pack(padx=20,pady=20, anchor="center")

                # Empty current order

                self.current\_order = []

                # Increase customer number

                self.current\_customer += 1

                #self.update\_ingredients\_label()

                self.order\_treeview.delete(\*self.order\_treeview.get\_children())

                self.update\_ingredients\_treeview()

        else:

            # Output Order Content

            result\_text = f"\n<{self.current\_customer}번째 손님 주문 내역>\n"

            total\_price = sum(order[1] for order in self.current\_order)

            for order in self.current\_order:

                result\_text += f"{order[0]} - {order[1]}원\n"

                self.total\_orders.append((self.current\_customer, order[0], order[1]))

            # Output to the result label

            result\_text += f"총합: {total\_price}원"

            self.result\_text.config(height=20, width=42)

            self.result\_text.insert(tk.END,result\_text+'\n', 'center')

            self.result\_text.tag\_configure('center', justify='center')

            self.result\_text.place(x=490,y=285)

            # Go down the scroll

            self.result\_text.see(tk.END)

            # Empty current order

            self.current\_order = []

            # Increase customer number

            self.current\_customer += 1

            self.order\_treeview.delete(\*self.order\_treeview.get\_children())

            self.update\_ingredients\_treeview()

    def create\_widgets(self):

        # Menu Selection Frame

        menu\_frame = tk.Frame(self.master, bg="#FFFFFF")

        menu\_frame.pack(pady=100)

        for pizza, data in self.menu.items():

            image\_path = data['이미지']

            image = Image.open(image\_path)

            image = image.resize((130, 130), Image.ANTIALIAS)  # Image resizing

            photo = ImageTk.PhotoImage(image)

            button = tk.Button(menu\_frame, text=pizza, image=photo, compound=tk.TOP, command=lambda p=pizza: self.add\_to\_order(p))

            button.image = photo

            button.pack(side=tk.LEFT, padx=17)

            if pizza in ['불고기피자', '크림스파게티', '망고에이드']:

                button.configure(bg='#FFA7A7')  # Background color of button on the menu is pink

            elif pizza in ['베이컨피자', '불고기스파게티', '키위에이드']:

                button.configure(bg='#BCE55C')  # Background color of button on the menu is green

            else:

                button.configure(bg='#FFFFFF')  # Set the rest of the background to white

        # Order List Treeview

        self.order\_treeview = ttk.Treeview(self.master, columns=('메뉴', '가격'), show='headings')

        self.order\_treeview.heading('메뉴', text='메뉴')

        self.order\_treeview.heading('가격', text='가격')

        self.order\_treeview.place(relx=5, rely=5, anchor=tk.CENTER)

        # Ingredient Status Treeview

        self.ingredients\_treeview = ttk.Treeview(self.master, columns=('재료', '수량'), show='headings')

        self.ingredients\_treeview.column('#0', stretch=tk.NO, width=0)

        self.ingredients\_treeview.column('재료', anchor=tk.CENTER, width=130)  # Adjust ingredient column width

        self.ingredients\_treeview.column('수량', anchor=tk.CENTER, width=130)  # Adjust quantity column width

        self.ingredients\_treeview.heading('재료', text='재료')

        self.ingredients\_treeview.heading('수량', text='수량')

        self.ingredients\_treeview.place(relx=0.84, rely=0.48, anchor=tk.CENTER)

        self.set\_menu\_panel = tk.Frame(self.master, bg="#FFFFFF")

        self.set\_menu\_label = tk.Label(self.set\_menu\_panel, text="세트 메뉴로 변경하시겠습니까?", font=('Helvetica', 25), bg="#FFFFFF")

        self.set\_menu\_label.pack(pady=10)

        self.set\_menu\_yes\_button = tk.Button(self.set\_menu\_panel, text="예", command=self.change\_set\_menu)

        self.set\_menu\_yes\_button.pack(side=tk.LEFT, padx=10)

        self.set\_menu\_no\_button = tk.Button(self.set\_menu\_panel, text="아니오", command=self.hide\_set\_menu\_panel)

        self.set\_menu\_no\_button.pack(side=tk.RIGHT, padx=10)

        self.set\_menu\_panel.pack\_forget()  # Set the panel to be invisible at first

        # Order Complete button

        order\_button = tk.Button(self.master, text="주문 완료", command=self.complete\_current\_order, width=30, height=3, bg="#FFFFFF")

        order\_button.place(relx=0.15, rely=0.93, anchor=tk.CENTER)

        # Total Order Sort button

        total\_order\_button = tk.Button(self.master, text="주문 정렬", command=self.complete\_total\_orders, width=30, height=3, bg='#FFFFFF')

        total\_order\_button.place(relx=0.83, rely=0.93, anchor=tk.CENTER)

        self.result\_text = tk.Text(self.master, bg='#FFFFFF', font=('Helvetica',17),height=80, width=40)

        self.result\_text.config(borderwidth=0, highlightthickness=0)  # Clearing Boundaries

        # Update current material status

        self.update\_ingredients\_treeview()

    def add\_to\_order(self, pizza):

        # Deduct ingredients when adding an order

        for ingredient in self.menu[pizza]['재료']:

            if self.ingredients[ingredient] > 0:

                self.ingredients[ingredient] -= 1

            else:

                self.show\_message("재료 부족", f"{ingredient}(이)가 부족합니다.")

        if self.current\_order\_matches\_set\_menu():

            self.show\_set\_menu\_panel()

        # Add to current order

        self.current\_order.append((pizza, self.menu[pizza]['가격']))

        # Discount if your current order matches the set menu

        if self.current\_order\_matches\_set\_menu():

            self.change\_set\_menu()

        self.order\_treeview.insert('', 'end', values=(pizza, self.menu[pizza]['가격']))

        self.update\_ingredients\_treeview()

    def show\_message(self, title, message):

        messagebox.showinfo(title, message)

    def show\_set\_menu\_panel(self):

        self.set\_menu\_panel.pack(side=tk.TOP, pady=50)

    def hide\_set\_menu\_panel(self):

        self.set\_menu\_panel.pack\_forget()

    def change\_set\_menu(self):

        set\_menu, discount, set\_name = self.detect\_set\_menu()

        if set\_menu:

            response = self.ask\_set\_menu\_change()

            if response:

                self.apply\_discount(set\_menu, discount, set\_name)

        else:

            self.show\_message("알림", "현재 주문이 세트 메뉴와 일치하지 않습니다.")

    def ask\_question(self, title, message):

        response = messagebox.askyesno(title, message)

        return response

    def detect\_set\_menu(self):

        # Initialize discount

        discount = 0

        # Set Menu A or B Order Confirmation

        for set\_menu, set\_discount, set\_name in [(self.set\_menu\_A, 3000, 'A'), (self.set\_menu\_B, 5000, 'B')]:

            if self.is\_set\_menu\_match(set\_menu):

                discount = set\_discount  # discount price

                return set\_menu, discount, set\_name

        return None, 0, None

    def apply\_discount(self, set\_menu, discount, set\_name):

        # Discount at the price of the menus to which the discount will be applied

        for menu, quantity in self.current\_order:

            for i in range(quantity):

                # Deduct ingredient

                for ingredient in self.menu[menu]['재료']:

                    if self.ingredients[ingredient] > 0:

                        self.ingredients[ingredient] -= 1

                    else:

                        self.show\_message("재료 부족", f"{ingredient} 재료가 부족합니다.")

        # Change total price of set menu

        total\_price = sum(self.menu[menu]['가격'] for menu, \_ in self.current\_order) - discount

        self.result\_text.insert(tk.END, f"총합: {total\_price}원 -> 세트{set\_name}\n",'center')

        self.result\_text.tag\_configure('center', justify='center')

    def update\_ingredients\_treeview(self):

        # Current ingredient status Treeview update

        # Remove existing items

        for item in self.ingredients\_treeview.get\_children():

            self.ingredients\_treeview.delete(item)

        # Add new items

        for ingredient, count in self.ingredients.items():

            self.ingredients\_treeview.insert('', 'end', values=(ingredient, count))

    def current\_order\_matches\_set\_menu(self):

        # Verify that the current order matches all menus contained in set menu A or B

        return any(set\_menu == self.current\_order for set\_menu in [self.set\_menu\_A, self.set\_menu\_B])

    def is\_set\_menu\_match(self, set\_menu):

        # Verify that the current order is completely consistent with the set menu

        set\_menu\_counts = Counter(dict(set\_menu))

        current\_order\_counts = Counter(dict(self.current\_order))

        return set\_menu\_counts == current\_order\_counts and len(self.current\_order) == len(set\_menu)

    def complete\_total\_orders(self):

        # Verify that the total order is empty

        if not self.total\_orders:

            tk.messagebox.showinfo("알림", "아직 주문된 메뉴가 없습니다.")

            return

        # # Sort total customer menus ordered by calculating the sum of prices per customer (Sorting algorithm)

        sorted\_orders = sorted(self.total\_orders, key=lambda x:

        sum(order[2] for order in self.total\_orders if order[0] == x[0]), reverse=True)

        # Print result label

        self.result\_text.delete('1.0',tk.END) #Delete existing text

        self.result\_text.insert(tk.END,"<주문 나가는 순서>\n",'center')

        self.result\_text.tag\_configure('center', justify='center')

        current\_price = float('inf')

        current\_customer = None

        for customer, pizza, price in sorted\_orders:

            if customer != current\_customer:

                if current\_customer is not None:

                    # Apply discount

                    discount = 0

                    if self.is\_set\_menu\_A(current\_customer):

                        response = messagebox.askquestion("주문 변경", "세트 메뉴로 변경하시겠습니까?")

                        if response == 'yes':

                            discount = 3000

                            current\_price -= discount

                            self.result\_text.insert(tk.END,f"세트A: {current\_price}원 (할인: {discount}원)\n\n",'center')

                            self.result\_text.tag\_configure('center', justify='center')

                        else:

                            self.result\_text.insert(tk.END, f"총합: {current\_price}원\n\n",'center')

                            self.result\_text.tag\_configure('center', justify='center')

                    elif self.is\_set\_menu\_B(current\_customer):

                        response = messagebox.askquestion("주문 변경", "세트 메뉴로 변경하시겠습니까?")

                        if response =='yes':

                            discount = 5000

                            current\_price -= discount

                            self.result\_text.insert(tk.END, f"세트B: {current\_price}원 (할인: {discount}원)\n\n",'center')

                            self.result\_text.tag\_configure('center', justify='center')

                    else:

                        self.result\_text.insert(tk.END, f"총합: {current\_price}원\n\n",'center')

                        self.result\_text.tag\_configure('center', justify='center')  # Finalize previous customer information when changing customers

                current\_customer = customer

                current\_price = 0

                self.result\_text.insert(tk.END, f"\n{customer}번째 손님:\n",'center')

                self.result\_text.tag\_configure('center', justify='center')

            self.result\_text.insert(tk.END, f"  {pizza} - {price}원\n",'center')

            self.result\_text.tag\_configure('center', justify='center')

            current\_price += price

        # Last order processing

        if current\_customer is not None:

            # Apply discount

            discount = 0

            if self.is\_set\_menu\_A(current\_customer):

                response = messagebox.askquestion("주문 변경", "세트 메뉴로 변경하시겠습니까?")

                if response == 'yes':

                    discount = 3000

                    current\_price -= discount

                    self.result\_text.insert(tk.END, f"세트A: {current\_price}원 (할인: {discount}원)\n",'center')

                    self.result\_text.tag\_configure('center', justify='center')

                else:

                    self.result\_text.insert(tk.END, f"총합: {current\_price}원\n",'center')

                    self.result\_text.tag\_configure('center', justify='center')

            elif self.is\_set\_menu\_B(current\_customer):

                discount = 5000

                current\_price -= discount

                self.result\_text.insert(tk.END, f"세트B: {current\_price}원 (할인: {discount}원)\n",'center')

                self.result\_text.tag\_configure('center', justify='center')

            else:

                self.result\_text.insert(tk.END, f"총합: {current\_price}원\n",'center')

                self.result\_text.tag\_configure('center', justify='center')

        # Refill of ingredients is recommended

        warning\_message = self.check\_depleted\_ingredient\_warning()

        print(warning\_message)

        # Predict of ingredients is printed

        prediction\_text = self.predict\_ingredient\_depletion()

        self.result\_text.insert(tk.END, "\n" + prediction\_text,'center')

        self.result\_text.tag\_configure('center', justify='center')

        self.result\_text.place(x=490,y=285)

        self.order\_treeview.delete(\*self.order\_treeview.get\_children())

        self.update\_ingredients\_treeview()

    def is\_set\_menu\_A(self, customer):

        # Verify that your order corresponds to set menu A

        customer\_orders = [order[1] for order in self.total\_orders if order[0] == customer]

        return all(menu in ['불고기피자', '크림스파게티', '망고에이드'] for menu in customer\_orders) and len(customer\_orders) == 3

    def is\_set\_menu\_B(self, customer):

        # Verify that your order corresponds to set menu B

        customer\_orders = [order[1] for order in self.total\_orders if order[0] == customer]

        return all(menu in ['베이컨피자', '불고기스파게티', '키위에이드'] for menu in customer\_orders) and len(customer\_orders) == 3

    def predict\_ingredient\_depletion(self):

        # A function of predicting the material exhaustion of currently ordered menus

        ingre = [float('inf')] \* (len(self.ingredients) + 1)

        ingre[0] = 0

        for order in self.total\_orders:

            menu\_ingredients = self.menu[order[1]]['재료']

            for ingredient in menu\_ingredients:

                ingre[len(menu\_ingredients)] = min(ingre[len(menu\_ingredients)], self.ingredients[ingredient])

        min\_ingredient\_count = min(ingre[1:])  # Find the least left ingredients

        depleted\_ingredients = [ingredient for ingredient, count in self.ingredients.items() if count == min\_ingredient\_count]

        prediction\_text = f"!!재료소진경고!!\n {', '.join(depleted\_ingredients)} 재고를 확인해주세요"

        return prediction\_text

    def check\_depleted\_ingredient\_warning(self):

        # Initialize a list that records the count of a single ingredient used in an order up to each customer

        ingredient\_counts = {ingredient: 0 for ingredient in self.ingredients}

        # Record the count of materials used in each order

        for order in self.total\_orders:

            \_, pizza, \_ = order

            for ingredient in self.menu[pizza]['재료']:

                ingredient\_counts[ingredient] += 1

        # Output the ingredient count used in the order to date

        print("현재 주문까지 사용된 재료 수:")

        for ingredient, count in ingredient\_counts.items():

            print(f"{ingredient}: {count}")

        # Selection of materials that have been used up to the maximum

        most\_depleted\_ingredient = max(ingredient\_counts, key=ingredient\_counts.get)

        count\_of\_most\_depleted\_ingredient = ingredient\_counts[most\_depleted\_ingredient]

        def fib(n, memo={}):

            if n <= 1:

                return n

            if n not in memo:

                memo[n] = fib(n - 1, memo) + fib(n - 2, memo)

            return memo[n]

        # Comparison with the number of materials most used after Fibonacci sequence calculation using dynamic programming algorithms

        fibonacci\_values = [0, 1]

        n = 2

        while True:

            value = fib(n)

            if value > count\_of\_most\_depleted\_ingredient:

                break

            fibonacci\_values.append(value)

            n += 1

        # Warnings output when the maximum number of uses of material exhausted is greater than the Fibonacci sequence value (n-1)

        warning\_message = ""

        if count\_of\_most\_depleted\_ingredient > fibonacci\_values[n - 1]:

            warning\_message = f"{most\_depleted\_ingredient}(이)가 자주 사용되어 자주 리필해놓으셔야 될 것 같습니다."

        # Outputs a message that refills are unlikely if it is less than Fibonacci sequence valuebo (n-1)

        if not warning\_message:

            warning\_message = f"재료들을 자주 리필해놓으실 필요는 없을 것 같습니다."

        return warning\_message

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

    root = tk.Tk()

    app = PizzaOrderSystem(root)

    root.mainloop()