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")

        # 배경 이미지 로드 및 크기 조절

        bg\_image = Image.open(r"C:\Users\jsm97\OneDrive\바탕 화면\algo\_team6\Restaurant.png.png")  # 천막 이미지 파일 경로에 맞게 수정

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

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

        # 배경 이미지를 표시하는 Label

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

        bg\_label.image = bg\_image

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

        # 재료 초기화

        self.ingredients = defaultdict(int)

        self.ingredients['양파'] = 7

        self.ingredients['피망'] = 7

        self.ingredients['페퍼로니'] = 7

        self.ingredients['불고기'] = 7

        self.ingredients['베이컨'] = 7

        self.ingredients['망고'] = 7

        self.ingredients['키위'] = 7

        # 세트 메뉴 관련 코드

        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()  # 처음에는 판넬을 보이지 않도록 설정

        # 메뉴 초기화

        self.menu = {

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

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

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

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

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

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

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

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

        }

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

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

        # 주문 관련 변수 초기화

        self.current\_order = []

        self.total\_orders = []

        # 현재 손님 번호

        self.current\_customer = 1

        # UI 구성

        self.create\_widgets()

        # self.result\_text 위젯 생성 후에 스크롤바 설정 추가

        scrollbar = tkinter.Scrollbar(self.master, orient="vertical")  # 수직 스크롤바 생성

        scrollbar.place(x=1020, y=300, height=480)  # 스크롤바 배치

        # 스크롤바와 self.result\_text 연결

        scrollbar.config(command=self.result\_text.yview)  # 스크롤바 설정

        self.result\_text.configure(yscrollcommand=scrollbar.set)  # self.result\_text에 스크롤바 설정

    # 두 리스트의 내용이 동일한지 확인하는 함수

    def lists\_equal(lst1, lst2):

        return Counter(lst1) == Counter(lst2)

    def complete\_current\_order(self):

        print("Debug: Inside complete\_current\_order")  # 디버깅용 출력 추가

        # 현재 주문이 비어 있는지 확인

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

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

                # 세트 메뉴 A 판넬 띄우기

                self.show\_set\_menu\_panel()

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

                # 세트 메뉴 B 판넬 띄우기

                self.show\_set\_menu\_panel()

            else:

                # 주문 내용 출력

                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]))

                # 결과 레이블에 출력

                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")

                # 현재 주문 비우기

                self.current\_order = []

                # 손님 번호 증가

                self.current\_customer += 1

                #self.update\_ingredients\_label()

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

                self.update\_ingredients\_treeview()

        else:

            # 주문 내용 출력

            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]))

            # 결과 레이블에 출력

            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)

            #스크롤 아래로 이동

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

            # 현재 주문 비우기

            self.current\_order = []

            # 손님 번호 증가

            self.current\_customer += 1

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

            self.update\_ingredients\_treeview()

            print("Debug: End of complete\_current\_order")  # 디버깅용 출력 추가

    def create\_widgets(self):

        # 메뉴 선택 프레임

        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)  # 이미지 크기 조절

            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  # 참조 유지

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

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

                button.configure(bg='#FFA7A7')  # 해당하는 메뉴의 버튼 배경색을 분홍색으로 설정합니다

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

                button.configure(bg='#BCE55C')  # 해당하는 메뉴의 버튼 배경색을 하늘색으로 설정합니다

            else:

                button.configure(bg='#FFFFFF')  # 나머지 버튼은 이전과 동일하게 연한 연두색으로 설정합니다

        # 주문내역 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)

        # 재료 상태 Treeview

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

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

        self.ingredients\_treeview.column('재료', anchor=tk.CENTER, width=130)  # 재료 컬럼 폭 조절

        self.ingredients\_treeview.column('수량', anchor=tk.CENTER, width=130)  # 수량 컬럼 폭 조절

        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()  # 처음에는 판넬을 보이지 않도록 설정

        # 주문 완료 버튼

        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)

        # 총 주문 정렬 버튼

        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)  # 테두리 선 없애기

        # 현재 재료 상태 업데이트

        self.update\_ingredients\_treeview()

    def add\_to\_order(self, pizza):

        # 주문 추가 시 재료 차감

        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()

        # 현재 주문에 추가

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

        # 현재 주문이 세트 메뉴와 일치하는지 확인하여 할인 적용

        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 changeokcancel(self):

        response = messagebox.askokcancel("확인/취소", "메뉴 변경하시겠습니까?")

        return response

    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\_set\_menu\_change(self):

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

            response = messagebox.askyesno("Set Menu Change", "Do you want to change to a set menu?")

            return response

        else:

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

            return False

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

        response = messagebox.askyesno(title, message)

        return response

    def detect\_set\_menu(self):

        # 할인 정보 초기화

        discount = 0

        # 세트 메뉴 A 또는 B 주문 확인

        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  # 할인되는 가격

                return set\_menu, discount, set\_name

        return None, 0, None

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

        # 할인을 적용할 메뉴들의 가격에서 할인

        for menu, quantity in self.current\_order:

            for i in range(quantity):

                # 재료 차감

                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.")

        # 세트 메뉴의 총 가격으로 변경

        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):

        # 현재 재료 상태 Treeview 업데이트

        # 기존 아이템 제거

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

            self.ingredients\_treeview.delete(item)

        # 새로운 아이템 추가

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

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

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

        # 현재 주문이 세트 메뉴 A 또는 B에 포함된 메뉴와 수량이 전부 일치하는지 확인

        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):

        # 현재 주문이 세트 메뉴와 완전히 일치하는지 확인

        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):

        # 총 주문이 비어 있는지 확인

        if not self.total\_orders:

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

            return

        # 총 주문된 손님들의 메뉴를 손님별로 가격의 합을 계산하여 정렬

        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)

        # 결과 레이블에 출력

        self.result\_text.delete('1.0',tk.END) #기존 텍스트 삭제

        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:

                    # 할인 적용

                    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')  # 손님 변경 시 이전 손님 정보를 마무리

                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

        # 마지막 주문자 처리

        if current\_customer is not None:

            # 할인 적용

            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')

        # 재료 예측 출력

        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):

        # 손님의 주문이 세트 메뉴 A에 해당하는지 확인

        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):

 # 손님의 주문이 세트 메뉴 B에 해당하는지 확인

        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):

        # 현재 주문된 메뉴들의 재료 소진을 예측하는 함수

        # DP 배열 초기화

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

        dp[0] = 0

        for order in self.total\_orders:

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

            for ingredient in menu\_ingredients:

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

        # 예측 결과 출력

        min\_ingredient\_count = min(dp[1:])  # 0번째는 무시하고 최소값 찾기

        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

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

    root = tk.Tk()

    app = PizzaOrderSystem(root)

    root.mainloop()