**Правительство Российской Федерации**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Федеральное государственное автономное образовательное учреждение

высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ "ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Кафедра «Компьютерная безопасность»

**ОТЧЕТ**

**К ЛАБОРАТОРНОЙ РАБОТЕ №5**

**По дисциплине**

|  |
| --- |
| Работу выполнил  Студент группы СКБ 192 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.О.Ташкинов  подпись, дата  Работу проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.А. Драчев  подпись, дата  Москва 2022 |

**«Методы программирования»**

**Вариант №2**

**Паттерн: Composite**

**Постановка задачи**

**Лабораторная работа №5**

Используя паттерн Composite, реализовать иерархическую структуру с вложенными (многоуровневыми) подменю по категориям (кухня: холодные закуски, горячие закуски, салаты, супы, основные блюда; бар: безалкогольные напитки, алкогольные напитки, чай, кофе и т. д.).

1. **Алгоритм решения задачи**

Данная лабораторная работа была реализована на языке программирования Python 3.9

В файле Lab5.py реализован паттерн Composite, для это был реализован абстрактный класс IProduct являющийся интерфейсом продуктов , от которого отнаследован класс продуктов Product и класс CompoundProduct, являющийся классом композитных продуктов, от которого отнаследован класс Menu для “красивого” вывода.

**2. Выполнение задания**

2.1. IProduct (ABC):

Является базовым абстрактным методом с двумя абстрактными методами: cost и name

2.2. Product(IProduct):

Класс, наследуемый от базового абстрактного класса, в нем реализован конструктор и методы:

cost(self) -> float – задает цену продукта

name(self) -> str – задает имя продукта

2.3. CompoundProduct(IProduct)

Класс, наследуемый от базового абстрактного класса, в нем реализован конструктор и методы:

cost(self) – метод увеличивающий общую стоимость “корзины” продуктов

name(self) -> str – метод, задающий имя композита

add\_product(self, product: IProduct) – метод добавляющий продукт в композит

remove\_product(self, product: IProduct) – метод удаляющий продукт в композит

clear(self) – метод очищающий композит

2.4. Menu(CompoundProduct)

Класс, наследуемый от класса композита, продуктов, в нем переопределен метод cost, для красивого вывода содержимого композита.

**Приложение А**

А.1 исходный код Lab5.py

from abc import ABC, abstractmethod

class IProduct(ABC):

"""

Интерфейс продуктов

из меню с абстрактными

методами цена и имя

"""

@abstractmethod

def cost(self) -> float:

pass

@abstractmethod

def name(self) -> str:

pass

class Product(IProduct):

"""

Класс продукта

в нем определены

методы цена и имя

а так же определен

конструктор

"""

def \_\_init\_\_(self, name: str, cost: float):

self.\_\_cost = cost

self.\_\_name = name

def cost(self) -> float:

return self.\_\_cost

def name(self) -> str:

return self.\_\_name

class CompoundProduct(IProduct):

"""

Класс компонуемых продуктов

в нем определены

методы цена и имя

реализованы методы

удаления продукта

добавление продукта

и очищение "корзины"

а так же определен

конструктор

"""

def \_\_init\_\_(self, name: str):

self.\_\_name = name

self.products = []

def cost(self):

cost = 0

for it in self.products:

cost += it.cost()

print(f"Стоимость '{self.name()}' = {cost} $")

return cost

def name(self) -> str:

return self.\_\_name

def add\_product(self, product: IProduct):

self.products.append(product)

def remove\_product(self, product: IProduct):

self.products.remove(product)

def clear(self):

self.products = []

class Menu(CompoundProduct):

"""

Класс Заказа относледованный

от класса компануемых продуктов

переопределяет метод кост для

"красивого" вывода компануемого

объекта с уровнем вложенности

не более 2

"""

def \_\_init\_\_(self, name: str):

super(Menu, self).\_\_init\_\_(name)

def cost(self):

cost = 0

cost\_i = 0

for it in self.products:

print(it.name())

for i in it.products:

cost\_i = i.cost()

print(f"Стоимость '{i.name()}' = {cost\_i} $")

cost += cost\_i

print(f"Cost '{self.name()}' = {cost} $")

return cost

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

Menu = Menu("Меню")

Bar = CompoundProduct("Бар")

Kitchen = CompoundProduct("Кухня")

Bar.add\_product(Product("Мартини", 12))

Bar.add\_product(Product("ВодкаПиво", 10))

Kitchen.add\_product(Product("МясоПоМиэмовски", 5))

Kitchen.add\_product(Product("НормальноеМясо", 6))

Menu.add\_product(Bar)

Menu.add\_product(Kitchen)

Menu.cost()