МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
ІКТА
Кафедра ЗІ

Звіт

до лабораторної роботи №9

з курсу: «Програмування скриптовими мовами» на тему: «Об’єктно-орієнтоване програмування»

Виконав:
студент групи КБ-305
Семенчук А.А.
Прийняв:
к.т.н., доцент
Совин Я. Р.

Львів 2024

Мета роботи – ознайомитись з реалізацією об’єктно-орієнтованого підходу в Python.

Завдання

1. Написати програму яка створює клас з атрибутами об’єкту заданими в таблиці. Для класу повинен бути перевантажені оператори порівняння за останнім атрибутом (наприклад, для автомобіля операції порівняння за швидкістю). Атрибути об’єкту повинні бути приватними з доступом через властивості класу або get- і set-методи. На базі цього класу створити клас, який зберігає список об’єктів та підтримує методи, які дозволяють через діалоговий режим виконувати такі операції:

a. Вивести весь список.

b. Додавати елементи до списку.

c. Відсортувати список за заданим атрибутом.

d. Видаляти елементи за заданим атрибутом.

e. Видаляти елемент за заданим індексом.

f. Виводити всі елементи за заданим атрибутом.

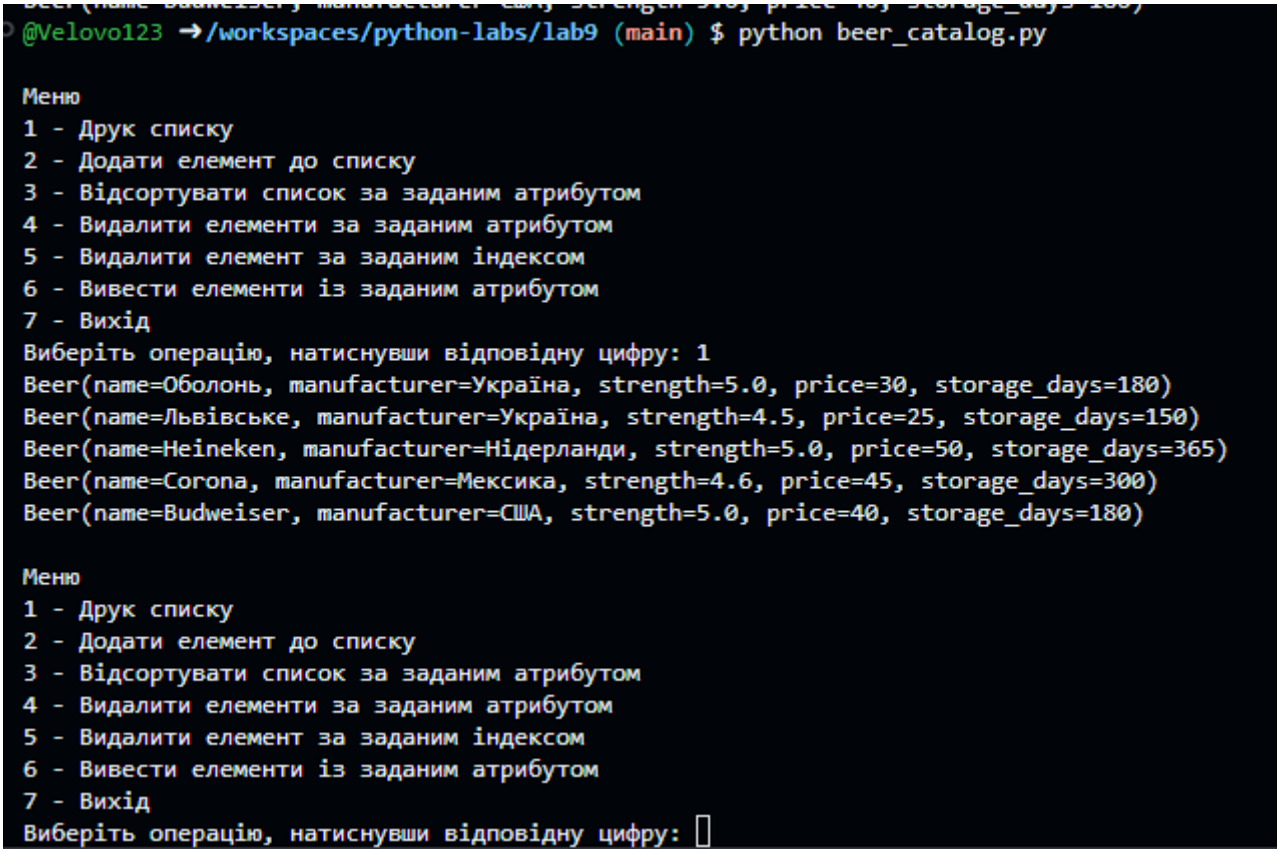
7	Пиво	Назва, виробник, міцність, ціна, термін зберігання в днях

Виконання лабораторної роботи

```
from typing import List, Any


class Beer:
    def __init__(self, name: str, manufacturer: str, strength: float, price: int,
storage_days: int):
        self.__name = name
        self.__manufacturer = manufacturer
        self.__strength = strength
        self.__price = price
        self.__storage_days = storage_days

    @property
    def name(self) -> str:
        return self.__name

    @property
    def manufacturer(self) -> str:
        return self.__manufacturer

    @property
    def strength(self) -> float:
        return self.__strength

    @property
    def price(self) -> int:
```

```
        return self.__price

    @property
    def storage_days(self) -> int:
        return self.__storage_days

    def __lt__(self, other: 'Beer') -> bool:
        return self.__storage_days < other.storage_days

    def __repr__(self) -> str:
        return (f"Beer(name={self.name}, manufacturer={self.manufacturer}, "
                f"strength={self.strength}, price={self.price},
storage_days={self.storage_days})")


class BeerCatalog:
    def __init__(self):
        self.beers = [
            Beer("Оболонь", "Україна", 5.0, 30, 180),
            Beer("Львівське", "Україна", 4.5, 25, 150),
            Beer("Heineken", "Нідерланди", 5.0, 50, 365),
            Beer("Corona", "Мексика", 4.6, 45, 300),
            Beer("Budweiser", "США", 5.0, 40, 180)
        ]

    def print_catalog(self) -> None:
        for beer in self.beers:
            print(beer)

    def add_beer(self, beer: Beer) -> None:
        self.beers.append(beer)

    def sort_by_attribute(self, attribute: str) -> None:
        self.beers.sort(key=lambda beer: getattr(beer, attribute))

    def delete_by_attribute(self, attribute: str, value: Any) -> None:
        self.beers = [beer for beer in self.beers if getattr(beer, attribute) !=
value]

    def delete_by_index(self, index: int) -> None:
        if 0 <= index < len(self.beers):
            self.beers.pop(index)

    def filter_by_attribute(self, attribute: str, value: Any) -> None:
        for beer in self.beers:
            if getattr(beer, attribute) == value:
                print(beer)


def main() -> None:
    catalog = BeerCatalog()

    while True:
```

```
        print("\nМеню")
        print("1 - Друк списку")
        print("2 - Додати елемент до списку")
        print("3 - Відсортувати список за заданим атрибутом")
        print("4 - Видалити елементи за заданим атрибутом")
        print("5 - Видалити елемент за заданим індексом")
        print("6 - Вивести елементи із заданим атрибутом")
        print("7 - Вихід")

        choice = input("Виберіть операцію, натиснувши відповідну цифру: ")
        if choice == "1":
            catalog.print_catalog()
        elif choice == "2":
            name = input("Назва: ")
            manufacturer = input("Виробник: ")
            strength = float(input("Міцність: "))
            price = int(input("Ціна: "))
            storage = int(input("Термін зберігання: "))
            catalog.add_beer(Beer(name, manufacturer, strength, price, storage))
        elif choice == "3":
            attribute = input("Атрибут для сортування (name, manufacturer,
strength, price, storage_days): ")
            catalog.sort_by_attribute(attribute)
        elif choice == "4":
            attribute = input("Атрибут для видалення (name, manufacturer, strength,
price, storage_days): ")
            value = input("Значення атрибуту для видалення: ")
            catalog.delete_by_attribute(attribute, value)
        elif choice == "5":
            index = int(input("Введіть індекс для видалення: "))
            catalog.delete_by_index(index)
        elif choice == "6":
            attribute = input("Атрибут для фільтрації (name, manufacturer,
strength, price, storage_days): ")
            value = input("Значення атрибуту для фільтрації: ")
            catalog.filter_by_attribute(attribute, value)
        elif choice == "7":
            print("Вихід з програми.")
            break
        else:
            print("Невірний вибір. Спробуйте ще раз.")


# Запуск програми
main()
```



Висновок

В ході виконання роботи було реалізовано об’єктно-орієнтований підхід у Python шляхом створення класів з приватними атрибутами, методами доступу (геттерами і сеттерами) та перевантаження оператора порівняння. Це дозволило краще зрозуміти принципи інкапсуляції, використання властивостей класу для контролю доступу до атрибутів та реалізацію функціональності через методи класів. Також була створена структура для збереження та управління списком об’єктів, що дозволило отримати практичний досвід роботи з методами класів для маніпулювання об'єктами у списку. Загалом, ця робота допомогла глибше зрозуміти принципи ООП у Python та їх застосування на практиці.