Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення
Дисципліна: Скриптові мови програмування (Python)

Лабораторна робота №5

Тема: «ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ У МОВІ PYTHON»

Виконав: ст. гр. КН-24
Куріщенко П. В.
Перевірив: ассистент
Ткаченко О.С.

Кропивницький 2025

Варіант - 1

Мета роботи - набути навичок роботи з класами у мові Python.

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Завдання 1:

Розробити клас «магазин кави». Реалізуйте можливість вибору кавових напоїв користувачем та збереження і видалення замовлень від клієнтів.

Клас Order

Призначення:

Моделює одне конкретне замовлення клієнта на каву.

Атрибути:

- orderId – унікальний ідентифікатор замовлення (автоматично присвоюється в CoffeeShop).
- customerName – ім’я клієнта.
- coffeeType – тип кави (вибирається зі списку).

Методи:

- __init__ – конструктор, який ініціалізує всі поля.
- __str__ – повертає зручний для читання рядок із даними замовлення.
- __del__ – повідомляє, коли об'єкт видаляється (не обов’язково, але може бути корисним для відлагодження або журналювання).

Проєктні рішення:

- Розділення одного замовлення в окремий клас полегшує підтримку, розширення функціоналу (наприклад, додавання ціни або дати).
- Використання __str__ дозволяє легко виводити замовлення у зрозумілому вигляді.

Лістинг класу:

```
class Order:
    """
    Represents a coffee order in the coffee shop.
    Attributes:
        orderId (int): Unique identifier for the order.
        customerName (str): Name of the customer who placed the order.
        coffeeType (str): Type of coffee ordered.
    """
    def __init__(self, orderId, customerName, coffeeType):
        self.orderId = orderId
        self.customerName = customerName
        self.coffeeType = coffeeType


    def __str__(self):
        return f"Order ID: {self.orderId}, Customer: {self.customerName},
Coffee Type: {self.coffeeType}"

    def __del__(self):
        print(f"Order {self.orderId} for {self.customerName} has been
deleted.")
```

Клас CoffeeShop

Призначення:

Керує всіма замовленнями, надає інтерфейс для користувача: додавання, перегляд і видалення замовлень.

Атрибути:

- orders – список поточних замовлень (Order).
- order_counter – лічильник, що гарантує унікальність orderId.
- coffee_types – словник доступних типів кави (ключі — цифри, значення — назви кави).

Методи:

- add_order():

  Запитує ім’я клієнта та тип кави.

  Перевіряє коректність введення.

  Створює новий об'єкт Order і додає до списку.

- view_orders():

  Виводить усі замовлення в зрозумілому форматі.

- delete_order(order_id):

  Видаляє замовлення з вказаним orderId, якщо воно існує.

- show_menu():

  Просте меню команд користувача з match-case (вимагає Python 3.10+).

  Дозволяє керувати замовленнями з клавіатури.

Проєктні рішення:

- Застосування order_counter забезпечує автоматичне та унікальне генерування ID.
- Всі операції з замовленнями інкапсульовані в одному класі — CoffeeShop, що робить код організованим і модульним.
- Словник coffee_types дозволяє легко додати нові типи кави без зміни логіки введення.
- Керування через меню дозволяє швидко протестувати додавання, перегляд і видалення без складного інтерфейсу.

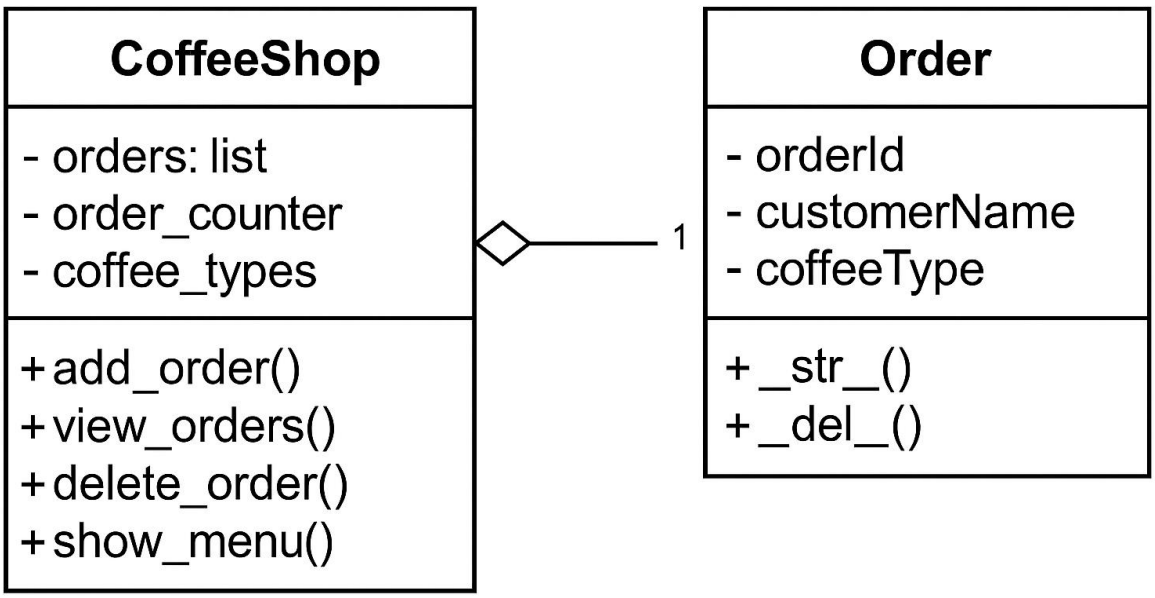
Лістинг класу:

```
class CoffeeShop:
    """
    Represents a coffee shop with order management functionality.
    """
    def __init__(self):
        self.orders = []
        self.order_counter = 0
        self.coffee_types = {
            "1": "Espresso",
            "2": "Latte",
            "3": "Cappuccino",
            "4": "Americano",
            "5": "Mocha"
        }

    def add_order(self):
        """
```

```
        Adds a new order to the coffee shop.
        Returns: None
        """
        order_id = self.order_counter
        self.order_counter += 1

        while True:
            customer_name = input("Enter customer name: ")
            if customer_name:
                break
            print("Customer name cannot be empty.")

        while True:
            print("Choose coffee type: ")
            for key, value in self.coffee_types.items():
                print(f"{key}: {value}")
            choice = input("Enter coffee type: ")
            if choice in self.coffee_types:
                coffee_type = self.coffee_types[choice]
                break
            print("Invalid coffee type. Please try again.")

        order = Order(order_id, customer_name, coffee_type)
        self.orders.append(order)
        print(f"Order {order.orderId} for {order.customerName} has been
added.")

    def view_orders(self):
        """
        Displays all current orders in the coffee shop.
        Returns: None
        """
        if not self.orders:
            print("No orders found.")
            return
        print("Current orders:")
        for order in self.orders:
            print(order)

    def delete_order(self, order_id):
        """
        Deletes an order from the coffee shop by order ID.
        Returns: None
        """
        for order in self.orders:
            if order.orderId == order_id:
                self.orders.remove(order)
```

```
                return
        print(f"Order {order_id} not found.")

    def show_menu(self):
        """
        Displays the coffee shop menu and handles user input for order
management.
        Returns: None
        """
        while True:
            print("\n1 -> Add order")
            print("2 -> View orders")
            print("3 -> Delete order")
            print("0 -> Back to main menu")

            match input("Enter your choice: "):
                case "1": self.add_order()
                case "2": self.view_orders()
                case "3":
                    order_id = int(input("Enter order ID to delete: "))
                    self.delete_order(order_id)
                case "0": return
                case _: print("Invalid choice. Please try again.")
```

Діаграма класів:



Результати роботи:

```
1 -> Add order
2 -> View orders
3 -> Delete order
0 -> Back to main menu
Enter your choice: 1
Enter customer name: Pablo
Choose coffee type:
1: Espresso
2: Latte
3: Cappuccino
4: Americano
5: Mocha
Enter coffee type: 2
Order 0 for Pablo has been added.
```

```
Enter your choice: 2
Current orders:
Order ID: 0, Customer: Pablo, Coffee Type: Latte
```

```
Enter your choice: 3
Enter order ID to delete: 0
Order 0 for Pablo has been deleted.
```

Завдання 2:

Розробити клас «домашня бібліотека». Реалізувати можливість роботи з довільним числом книг, пошуку книг за декількома параметрами (за автором, за роком видання, за жанром тощо), додавання книг у бібліотеку, видалення книг з неї, доступу до книги за номером. Написати програму, що буде демонструвати всі розроблені елементи класу.

Клас Book

Призначення:

Моделює окрему книгу з її основними характеристиками.

Атрибути:

- book_id: унікальний ідентифікатор книги.
- title: назва книги.
- author: автор книги.

- year: рік публікації.
- genre: жанр книги.

Методи:

- __init__: конструктор, ініціалізує всі властивості книги.
- __str__: повертає форматований рядок для зручного виведення інформації про книгу.
- __del__: виводить повідомлення при знищенні об'єкта (допоміжний, не обов’язковий у продакшн-коді).

Проєктні рішення:

- **Інкапсуляція**: усі дані про книгу зібрані в одному класі.
- **Унікальний book_id** дозволяє однозначно ідентифікувати кожну книгу незалежно від інших характеристик.
- Метод __str__ спрощує виведення об'єкта.

Лістинг класу:

```
class Book:
    """
    Represents a book with attributes such as ID, title, author, year, and genre.
    """
    def __init__(self, book_id, title, author, year, genre):
        self.book_id = book_id
        self.title = title
        self.author = author
        self.year = year
        self.genre = genre

    def __str__(self):
        return f"ID: {self.book_id}, Title: {self.title}, Author: {self.author}, Year: {self.year}, Genre: {self.genre}"

    def __del__(self):
        try:
            print(f"Book '{self.title}' by {self.author} has been deleted.")
        except AttributeError:
            pass  # Ignore if attributes are already gone during cleanup
```

Клас HomeLibrary

Призначення:

Реалізує логіку управління книгами в домашній бібліотеці (список, додавання, пошук, видалення, меню).

Атрибути:

- books: список об'єктів Book.
- book_counter: лічильник, що забезпечує унікальність ID кожної книги.
- genres: словник жанрів, який дозволяє вибирати жанр із заздалегідь визначеного списку.

Методи:

- add_book():
  - Запитує дані користувача про книгу.
  - Перевіряє коректність введення (непорожні поля, допустимий рік, правильний жанр).
  - Створює об’єкт Book і додає його до бібліотеки.
- delete_book(book_id):
  - Видаляє книгу за її ID, якщо вона знайдена.
- find_book():
  - Дозволяє шукати книги за:
    - ID (точний пошук)
    - автором, роком, жанром (гнучкий фільтр)
  - Виводить результати або повідомляє, що нічого не знайдено.
- show_menu():
  - Простий інтерфейс користувача через консольне меню (з використанням match-case).
  - Обробляє вибір дій: додавання, пошук, видалення або вихід.

Архітектурні рішення:

Модульність і розділення відповідальностей:

- Клас Book зберігає лише дані однієї книги.

- Клас HomeLibrary відповідає за керування колекцією книг.

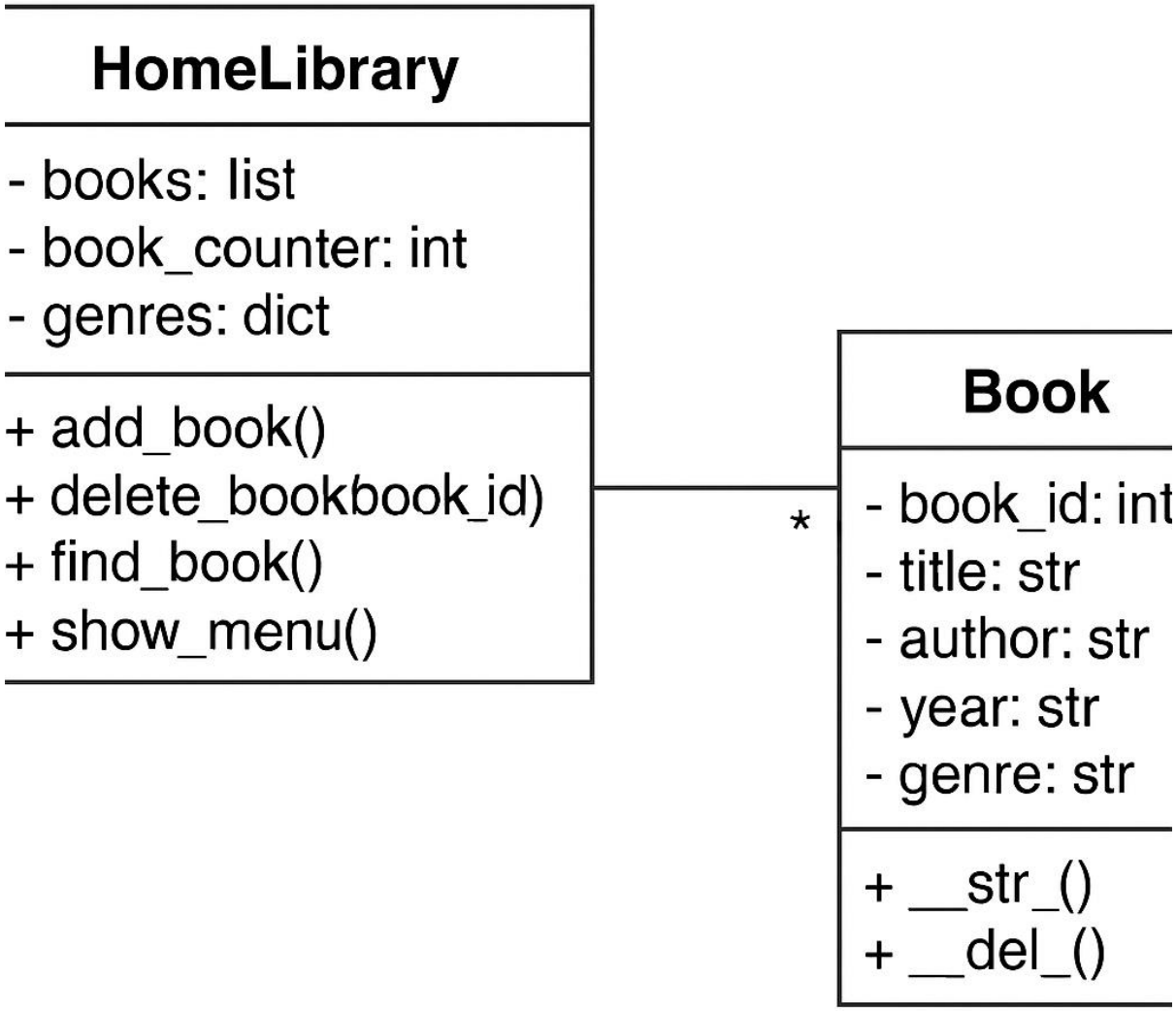
Лістинг класу:

```
class HomeLibrary:
    """
    Represents a home library with book management functionality.
    """
    def __init__(self):
        self.books = []
        self.book_counter = 0
        self.genres = {
            "1": "Fiction",
            "2": "Non-Fiction",
            "3": "Science",
            "4": "History",
            "5": "Fantasy",
            "6": "Biography",
            "7": "Romance",
            "8": "Thriller",
            "9": "Horror",
            "0": "Other"
        }

    def add_book(self):
        """
        Adds a new book to the library.
        Returns: None
        """
        book_id = self.book_counter
        self.book_counter += 1

        while True:
            title = input("Enter book title: ")
            if title:
                break
            print("Book title cannot be empty.")

        while True:
            author = input("Enter author name: ")
            if author:
                break
            print("Author name cannot be empty.")

        while True:
            year = input("Enter publication year: ")
```

```
            if year.isdigit() and 1800 <= int(year) <= 2025:
                break
            print("Invalid year. Please enter a valid year between 1800 and
2025.")

        while True:
            print("Choose genre:")
            for key, value in self.genres.items():
                print(f"{key}: {value}")
            choice = input("Enter genre: ")
            if choice in self.genres:
                genre = self.genres[choice]
                break
            print("Invalid genre. Please try again.")

        book = Book(book_id, title, author, year, genre)
        self.books.append(book)
        print(f"Book ID {book_id}: '{book.title}' by {book.author} has been
added.")

    def delete_book(self, book_id):
        """
        Deletes a book from the library by book ID.
        Returns: None
        """
        for book in self.books:
            if book.book_id == book_id:
                self.books.remove(book)
                return
        print(f"Book {book_id} not found.")

    def find_book(self):
        """
        Finds a book in the library based on user input.
        Returns: None
        """
        try:
            found_books = self.books
            book_id = input("Enter book ID (or press Enter to skip): ")
            if book_id:
                book_id = int(book_id)
                found_books = [book for book in found_books if book.book_id ==
book_id]
            else:
                author = input("Enter author name (or press Enter to skip): ")
                year = input("Enter year (or press Enter to skip): ")
                genre = input("Enter genre (or press Enter to skip): ")
```

```
                if author: found_books = [book for book in found_books if
book.author == author]
                if year: found_books = [book for book in found_books if
book.year == year]
                if genre: found_books = [book for book in found_books if
book.genre == genre]

            if found_books:
                print("Found books:")
                for book in found_books:
                    print(book)
            else:
                print("No books found with the given criteria.")
        except ValueError:
            print("Invalid book ID. Please enter a valid number.")

    def show_menu(self):
        """
        Displays the home library menu and handles user input for book
management.
        Returns: None
        """
        while True:
            print("\n1 -> Add book")
            print("2 -> Find book")
            print("3 -> Delete book")
            print("0 -> Back to main menu")

            match input("Enter your choice: "):
                case "1": self.add_book()
                case "2": self.find_book()
                case "3":
                    book_id = int(input("Enter book ID to delete: "))
                    self.delete_book(book_id)
                case "0": return
                case _: print("Invalid choice. Please try again.")
```

Діаграма класів:



Результати роботи:

```
1 -> Add book
2 -> Find book
3 -> Delete book
0 -> Back to main menu
Enter your choice: 1
Enter book title: metamorphosis
Enter author name: Kafka
Enter publication year: 1915
Choose genre:
1: Fiction
2: Non-Fiction
3: Science
4: History
5: Fantasy
6: Biography
7: Romance
8: Thriller
9: Horror
0: Other
Enter genre: 1
Book ID 0: 'metamorphosis' by Kafka has been added.
```

```
Enter your choice: 2
Enter book ID (or press Enter to skip):
Enter author name (or press Enter to skip):
Enter year (or press Enter to skip): 1915
Enter genre (or press Enter to skip): Fiction
Found books:
ID: 0, Title: metamorphosis, Author: Kafka, Year: 1915, Genre: Fiction
```

```
Enter your choice: 3
Enter book ID to delete: 0
Book 'metamorphosis' by Kafka has been deleted.
```

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Як створити клас у мові Python?

За допомогою ключового слова class:

```
class Book:
    def __init__(self, title):
        self.title = title
```

2. Що таке інкапсуляція?

Це принцип ООП, який об’єднує дані (атрибути) і методи в одному класі, приховуючи внутрішню реалізацію.

У класі Book інкапсульовано title, author, year і методи.

3. Що таке об’єкт?

Об’єкт — це екземпляр класу, що має власні значення атрибутів і доступ до методів класу.

```
book1 = Book(1, "1984", "Orwell", 1949, "Fiction")
```

4. Що таке атрибут?

Змінна, що зберігається в об’єкті:

```
self.title = title  # title — це атрибут класу Book
```

5. У чому полягає відмінність методу від функції?

Метод — це функція, яка належить класу і викликається через об’єкт:

```
book1.__str__()  # метод
print("Hello")   # звичайна функція
```