

Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



## **Звіт**

З лабораторної роботи №9

Варіант – 10

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Основи об’єктно-орієнтованого програмування у Python»

Виконав: ст. гр. КІ-306

Згурський Т.С.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів 2023

**Мета роботи:** оволодіти навиками реалізації парадигм об'єктно-орієнтованого програмування використовуючи засоби мови Python.

## ЗАВДАННЯ

1. Написати та налагодити програму на мові Python згідно варіанту. Програма має задовольняти наступним вимогам:

- класи програми мають розміщуватися в окремих модулях в одному пакеті;
- точка входу в програму (main) має бути в окремому модулі;
- мають бути реалізовані базовий і похідний класи предметної області згідно варіанту;
- програма має містити коментарі.

2. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

4. Дати відповідь на контрольні запитання.

## Варіант завдання:

Базовий клас: Будинок

Похідний клас: Офісний центр

Код програми:

House.py:

```
import datetime
```

```
class House:
```

```
    def __init__(self, address="No information", number_of_floors=0, has_garden=False):
```

```
        self.address = address
```

```
        self.number_of_floors = number_of_floors
```

```
        self.has_garden = has_garden
```

```
        self.writer = None
```

```
    def open_log_file(self):
```

```
        try:
```

```
            self.writer = open("log.txt", "a")
```

```
        except IOError as e:
```

```
            print(f"Помилка при відкритті файлу для запису: {e}")
```

```
    def close_log_file(self):
```

```
        try:
```

```
        if self.writer:
            self.writer.close()
    except IOError as e:
        print(f"Помилка при закритті файлу: {e}")

def display_details(self):
    print(f"Адреса будинку: {self.address}")
    print(f"Кількість поверхів: {self.number_of_floors}")
    print(f"Наявність саду: {'Так' if self.has_garden else 'Ні'}")

def log_message(self, message):
    try:
        if self.writer:
            timestamp = datetime.datetime.now()
            self.writer.write(f"[{timestamp}] {message} - {self}\n")
    except IOError as e:
        print(f"Помилка при записі до файлу: {e}")

def set_number_of_floors(self, number_of_floors):
    self.number_of_floors = number_of_floors
    self.log_message(f"Встановлено кількість поверхів: {number_of_floors}")

def set_address(self, address):
    self.address = address
    self.log_message(f"Оновлено адресу: {address}")

def set_has_garden(self, has_garden):
    self.has_garden = has_garden
    self.log_message("Оновлено інформацію про сад.")

def add_floor(self):
    self.number_of_floors += 1
    self.log_message("Додано поверх.")
```

```
def remove_floor(self):
    if self.number_of_floors > 0:
        self.number_of_floors -= 1
        self.log_message("Видалено поверх.")
    else:
        self.log_message("Не можна видалити поверх. Кількість поверхів вже мінімальна.")

def get_number_of_floors(self):
    self.log_message("Дана інформація про кількість поверхів.")
    return self.number_of_floors

def get_address(self):
    return self.address

def has_garden(self):
    self.log_message("Дана інформація про наявність саду.")
    return self.has_garden
```

HouseApp.py:

```
from house import House
```

```
class HouseApp:
```

```
    def main(self):
        # Create a house
        house = House("123 Main St.", 2, True)

        # Display house details
        house.display_details()

        # Update house details
        house.set_number_of_floors(3)
```

```
house.set_address("456 Elm St.")
house.set_has_garden(False)

# Display updated house details
house.display_details()

house.to_string()

if __name__ == "__main__":
    app = HouseApp()
    app.main()
```

OfficeCenter.py:

```
from house import House
import datetime
```

```
class OfficeCenter(House):
    def __init__(self, address, number_of_floors, office_space, has_meeting_room, number_of_desks,
has_projector, has_whiteboard):
        super().__init__(address, number_of_floors)
        self.office_space = office_space
        self.has_meeting_room = has_meeting_room
        self.number_of_desks = number_of_desks
        self.has_projector = has_projector
        self.has_whiteboard = has_whiteboard

    def get_office_space(self):
        return self.office_space

    def set_office_space(self, office_space):
        self.office_space = office_space
```

```
def allocate_office_space(self, square_meters):
    self.office_space += square_meters

def equip_meeting_room(self, projector, whiteboard):
    self.has_meeting_room = True
    self.has_projector = projector
    self.has_whiteboard = whiteboard

def add_desks(self, desks_to_add):
    self.number_of_desks += desks_to_add

def remove_desks(self, desks_to_remove):
    if desks_to_remove > self.number_of_desks:
        print("Cannot remove more desks than available.")
    else:
        self.number_of_desks -= desks_to_remove

def add_projector(self, has_projector):
    self.has_projector = has_projector

def add_whiteboard(self, has_whiteboard):
    self.has_whiteboard = has_whiteboard

def add_projector2(self, has_projector):
    print("Projector was added")
    has_projector = True

def write_address_to_file(self):
    address = self.get_address()

    try:
        with open("Address.txt", "w") as writer:
            writer.write(address)
    except IOError as e:
```

```
print(f"Помилка при записі у файл: {e}")
```

```
return address
```

```
def __str__(self):  
    return self.write_address_to_file()
```

OfficeInterface.py:

```
class OfficeInterface:
```

```
    def get_office_space(self):  
        pass
```

```
    def set_office_space(self, office_space):  
        pass
```

```
    def allocate_office_space(self, square_meters):  
        pass
```

```
    def equip_meeting_room(self, projector, whiteboard):  
        pass
```

```
    def add_desks(self, desks_to_add):  
        pass
```

```
    def remove_desks(self, desks_to_remove):  
        pass
```

```
    def add_projector(self, has_projector):  
        pass
```

```
    def add_whiteboard(self, has_whiteboard):  
        pass
```

```
def write_address_to_file(self):  
    pass
```

OfficeInterface2.py:

```
class OfficeInterface2:  
    def add_projector2(self, has_projector):  
        pass  
  
    def add_whiteboard2(self, has_whiteboard):  
        pass
```

### **Результати роботи програми:**

```
Кількість поверхів: 2  
Наявність саду: Так  
Кількість поверхів: 3  
Адреса будинку: New address  
Наявність саду: True
```

### **Відповіді на контрольні запитання**

1. Що таке модулі?

- Модулі в Python - це файли, які містять Python-код. Вони використовуються для організації коду у логічні групи, і можуть містити функції, класи, змінні та інші об'єкти.

2. Як імпортувати модуль?

- import модуль

3. Як оголосити клас?

- class МійКлас: # Тіло класу

4. Що може міститися у класі?

- атрибути (змінні), методи (функції), конструктори, спеціальні методи (наприклад, \_\_init\_\_, \_\_str\_\_), властивості та інше.

5. Як називається конструктор класу?



- Конструктор класу має ім'я `__init__`. Він викликається при створенні нового об'єкта класу і використовується для ініціалізації атрибутів об'єкта.

6. Як здійснити спадкування?

- `class ПідКлас(БазовийКлас):` # Тіло підкласу

7. Які види спадкування існують?

- одиночне спадкування (коли підклас успадковує лише один базовий клас) та множинне спадкування (коли підклас успадковує більше одного базового класу).

8. Які небезпеки є при множинному спадкуванні, як їх уникнути?

- Небезпеки при множинному спадкуванні включають в себе можливі конфлікти імен методів або атрибутів між базовими класами, що може призвести до непередбачуваної поведінки. Для уникнення цих проблем можна використовувати аліаси, викликати методи базових класів безпосередньо або використовувати композицію замість спадкування.

9. Що таке класи-домішки?

- це класи, які містять певний функціонал і можуть бути використані для розширення функціональності інших класів. Вони не призначені для створення об'єктів, але можуть бути включені у інші класи за допомогою спадкування, щоб надати їм певну функціональність.

10. Яка роль функції `super()` при спадкуванні?

- для виклику методів базового класу з підкласу. Вона допомагає уникнути явного вказівання імен базових класів та робить код більш гнучким при зміні структури спадкування. Наприклад, `super().__init__()` викликає конструктор базового класу.

## Висновок

У ході виконання даної лабораторної роботи, я здобув важливі навички об'єктноорієнтованого програмування мовою Python. Ознайомився з ключовими аспектами цієї парадигми, включаючи створення та використання класів, роботу з об'єктами, та використання спадкування та поліморфізму для покращення ефективності програм.