Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра «Електронних обчислювальних машин»

Звіт
з лабораторної роботи № 9
з дисципліни: «Кросплатформенні засоби програмування»
на тему: «Основи Об'єктно-Орієнтованого програмування у Python»

Виконав:
студент групи *КІ-306*
Ярема Максим
Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

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

Завдання (варіант № 29)

29. Протигаз

29. Протигаз командира

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.

Вихідний код програми

Файл Main.py

```
from commander_gasMask import CommanderGasMask
from filter import Filter

def main():
    commander_gas_mask = CommanderGasMask("EN14387", "A2B2E2K2")

    commander_gas_mask.check_status()

    commander_gas_mask.breathe()
    commander_gas_mask.breathe()

    commander_gas_mask.seal_mask()

    commander_gas_mask.breathe()

    commander_gas_mask.turn_night_vision_on()

    commander_gas_mask.breathe()

    commander_gas_mask.replace_filter(Filter("A1B1E1K1"))
```

```
    commander_gas_mask.check_status()

    commander_gas_mask.send_sos_signal()

    commander_gas_mask.clean_and_replace_filter()

if __name__ == "__main__":
    main()
```

Файл filter.py

```
class Filter:
    def __init__(self, type):
        self.type = type
        self.is_effective = True

    def make_ineffective(self):
        self.is_effective = False

    def get_is_effective(self):
        return self.is_effective

    def get_type(self):
        return self.type
```

Файл gasMask.py

```
from filter import Filter

class GasMask:
    def __init__(self, mask_type, filter_type):
        self.type = mask_type
        self.filter = Filter(filter_type)
        self.is_night_vision_on = False
        self.usage_count = 0
        self.is_sealed = False
        self.is_clean = True

    def turn_night_vision_on(self):
        self.is_night_vision_on = True
        print("Night vision mode is turned on.")

    def turn_night_vision_off(self):
        self.is_night_vision_on = False
        print("Night vision mode is turned off.")

    def replace_filter(self, new_filter):
        self.filter = new_filter
        print("Filter replaced with a new one.")

    def breathe(self):
        if self.is_sealed and self.filter.is_effective:
```

```
            self.increment_usage()
            print("Breathed safely using the gas mask.")
        else:
            print("Gas mask is not sealed properly or the filter is not
effective.")

    def seal_mask(self):
        self.is_sealed = True
        print("Gas mask is sealed.")

    def unseal_mask(self):
        self.is_sealed = False
        print("Gas mask is unsealed.")

    def clean_mask(self):
        self.is_clean = True
        print("Gas mask is cleaned.")

    def increment_usage(self):
        self.usage_count += 1

    def clean_and_replace_filter(self):
        if not self.is_clean:
            self.clean_mask()
        self.replace_filter(Filter("A2B2E2K2"))
        print("Gas mask is cleaned and the filter is replaced.")

    def check_status(self):
        status = "Gas Mask Status:\n"
        status += "Type: " + self.type + "\n"
        status += "Filter Type: " + self.filter.get_type() + "\n"
        status += "Is Night Vision On: " + str(self.is_night_vision_on) + "\n"
        status += "Usage Count: " + str(self.usage_count) + "\n"
        status += "Is Sealed: " + str(self.is_sealed) + "\n"
        status += "Is Clean: " + str(self.is_clean) + "\n"
        print(status)

    def get_type(self):
        return self.type
```

Файл commander_gasMask.py

```
from gasMask import GasMask

class CommanderGasMask(GasMask):
    def __init__(self, mask_type, filter_type):
        super().__init__(mask_type, filter_type)

    def get_equipment_type(self):
        return self.get_type()
```

```
    def send_sos_signal(self):
        print("Sending SOS signal...")
```

Результат виконання програми

```
Gas Mask Status:
Type: EN14387
Filter Type: A2B2E2K2
Is Night Vision On: False
Usage Count: 0
Is Sealed: False
Is Clean: True

Gas mask is not sealed properly or the filter is not effective.
Gas mask is not sealed properly or the filter is not effective.
Gas mask is sealed.
Breathed safely using the gas mask.
Night vision mode is turned on.
Breathed safely using the gas mask.
Filter replaced with a new one.
Gas Mask Status:
Type: EN14387
Filter Type: A1B1E1K1
Is Night Vision On: True
Usage Count: 2
Is Sealed: True
Is Clean: True

Sending SOS signal...
Filter replaced with a new one.
Gas mask is cleaned and the filter is replaced.
```

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'єктно-орієнтованого 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.