**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**Інститут комп’ютерних технологій, автоматики та метрології**

**кафедра “Електронних обчислювальних машин”**

****

Звіт

з лабораторної роботи №9

дисципліни «Кросплатформні засоби програмування»

на тему: **«**ОСНОВИ ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ У PYTHON**»**

Варіант 15

**Виконала:**

студентка групи КІ-303

Сухарник А.А.

**Прийняв:**

Іванов Ю.С.

Львів – 2024

ЛАБОРАТОРНА РОБОТА №9

**ОСНОВИ ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ У PYTHON**

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

**Теоретичний матеріал**

Модулем у Python називається файл з розширенням \*.py. Ці файли можуть містити звичайні скрипти, змінні, функції, класи і їх комбінації. Python дозволяє структурувати код програм у різні модулі та доступатися до класів, функцій і змінних, які у них знаходяться з інших модулів. Для цього використовуються два оператори – import та from-import. Оператор import дозволяє імпортувати модуль повністю, та доступатися до нього через назву модуля. Вона може бути вказана у будь-якому місці програми перед звертанням до елементів, які у ній містяться, але зазвичай її вказують на початку модуля. Для звертання до елементів модуля треба вказати назву модуля і після крапки вказати до якого елементу ви хочете звернутися.

Члени класу є лише публічні, проте Python забезпечує механізми, які дозволяють організувуати області видимості близькі за своєю суттю до protected і private. Це робиться шляхом використання нижнього підкреслення у назві членів класу. Одинарне нижнє підкреслення перед назвою члену класу робить за своїми властивостями схожим на захищений член класу, а подвійне – схожим на приватний член класу. Всі члени класу, що йому належать мають відступ у розмірі одного табулятора, або 4-ох пробілів від початку оголошення класу.

**Завдання (Варіант №15)**

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 (Корабель) і похідний (Фрегат) 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.

**Виконання завдання**

**Frigate.py**

from Ship import Ship  
  
  
class Frigate(Ship):  
 """  
 Клас Frigate розширює функціональність базового класу Ship,  
 додаючи бойові можливості.  
 """  
  
 def \_\_init\_\_(self, name, tonnage, armor\_type, cannon\_count):  
 """  
 Ініціалізує фрегат.  
  
 :param name: Назва фрегата.  
 :param tonnage: Водотоннажність (у тоннах).  
 :param armor\_type: Тип броні.  
 :param cannon\_count: Кількість гармат.  
 """  
 super().\_\_init\_\_(name, tonnage, armor\_type)  
 self.cannon\_count = cannon\_count  
 self.shield\_activated = False  
  
 def fire\_cannon(self, target):  
 """  
 Стріляє з гармат по цілі.  
  
 :param target: Назва цілі.  
 """  
 if self.cannon\_count > 0:  
 print(f"Фрегат {self.name} стріляє з {self.cannon\_count} гармат по цілі: {target}.")  
 else:  
 print(f"Фрегат {self.name} не має гармат для атаки.")  
  
 def prepare\_for\_battle(self):  
 """  
 Готує фрегат до бою.  
 """  
 self.shield\_activated = True  
 print(f"Фрегат {self.name} готовий до бою: щит активовано.")  
  
 def activate\_shield(self):  
 """  
 Активує щит.  
 """  
 self.shield\_activated = True  
 print(f"Щит фрегата {self.name} активовано.")  
  
 def deactivate\_shield(self):  
 """  
 Вимикає щит.  
 """  
 self.shield\_activated = False  
 print(f"Щит фрегата {self.name} вимкнено.")  
  
 def scan\_area(self):  
 """  
 Виконує сканування навколишньої зони.  
 """  
 print(f"Фрегат {self.name} сканує навколишню зону: ворогів не виявлено.")  
  
 def get\_status(self):  
 """  
 Повертає статус фрегата.  
 """  
 basic\_status = super().get\_status()  
 shield\_status = "активовано" if self.shield\_activated else "вимкнено"  
 return f"{basic\_status} Щит: {shield\_status}, гармати: {self.cannon\_count}."

**main.py**

from Frigate import Frigate  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 # Створюємо фрегат  
 frigate = Frigate("Гордість океану", 5000, "Титанова броня", 20)  
  
 # Виводимо початковий статус фрегата  
 print("\n1. Початковий статус:")  
 print(frigate.get\_status())  
  
 # Увімкнення двигуна  
 print("\n2. Увімкнення двигуна:")  
 frigate.start\_engine()  
  
 # Підготовка до бою  
 print("\n3. Підготовка до бою:")  
 frigate.prepare\_for\_battle()  
  
 # Вогонь по цілі  
 print("\n4. Атака цілі:")  
 frigate.fire\_cannon("ворожий корабель")  
  
 # Використання радару  
 print("\n5. Використання радара:")  
 frigate.scan\_area()  
  
 # Захист  
 print("\n6. Активація системи захисту:")  
 frigate.activate\_shield()  
  
 # Переміщення  
 print("\n7. Переміщення фрегата:")  
 frigate.set\_course("південний захід")  
 frigate.move()  
  
 # Ремонт  
 print("\n8. Виконання ремонту:")  
 frigate.repair()  
  
 # Вимкнення двигуна  
 print("\n9. Вимкнення двигуна:")  
 frigate.stop\_engine()  
  
 print("\n10. Фінальний стан:")  
 print(frigate.get\_status())

**Ship.py**

class Ship:  
 """  
 Клас Ship представляє базовий корабель.  
 """  
  
 def \_\_init\_\_(self, name, tonnage, armor\_type):  
 """  
 Ініціалізує корабель.  
  
 :param name: Назва корабля.  
 :param tonnage: Водотоннажність (у тоннах).  
 :param armor\_type: Тип броні.  
 """  
 self.name = name  
 self.tonnage = tonnage  
 self.armor\_type = armor\_type  
 self.is\_engine\_on = False  
 self.is\_moving = False  
 self.current\_course = "невідомий"  
 self.damage\_level = 0 # Від 0 (без ушкоджень) до 100 (критичні ушкодження)  
  
 def start\_engine(self):  
 """  
 Увімкнення двигуна.  
 """  
 self.is\_engine\_on = True  
 print(f"Двигун корабля {self.name} увімкнено.")  
  
 def stop\_engine(self):  
 """  
 Вимкнення двигуна.  
 """  
 self.is\_engine\_on = False  
 print(f"Двигун корабля {self.name} вимкнено.")  
  
 def set\_course(self, course):  
 """  
 Встановлює курс корабля.  
  
 :param course: Новий курс.  
 """  
 self.current\_course = course  
 print(f"Курс корабля {self.name} встановлено на {course}.")  
  
 def move(self):  
 """  
 Починає рух корабля, якщо двигун увімкнено.  
 """  
 if self.is\_engine\_on:  
 self.is\_moving = True  
 print(f"Корабель {self.name} рухається за курсом {self.current\_course}.")  
 else:  
 print("Неможливо почати рух: двигун вимкнено.")  
  
 def stop(self):  
 """  
 Зупиняє рух корабля.  
 """  
 if self.is\_moving:  
 self.is\_moving = False  
 print(f"Корабель {self.name} зупинено.")  
 else:  
 print("Корабель вже стоїть на місці.")  
  
 def repair(self):  
 """  
 Ремонтує корабель, відновлюючи його стан.  
 """  
 self.damage\_level = 0  
 print(f"Корабель {self.name} відремонтовано.")  
  
 def take\_damage(self, damage):  
 """  
 Завдає ушкодження кораблю.  
  
 :param damage: Рівень ушкоджень (від 1 до 100).  
 """  
 self.damage\_level = min(100, self.damage\_level + damage)  
 print(f"Корабель {self.name} отримав ушкодження: {damage}%. Поточний рівень ушкоджень: {self.damage\_level}%.")  
  
 def get\_status(self):  
 """  
 Повертає статус корабля.  
 """  
 engine\_status = "увімкнений" if self.is\_engine\_on else "вимкнений"  
 return f"Корабель {self.name}, тоннаж: {self.tonnage}, броня: {self.armor\_type}, двигун: {engine\_status}, ушкодження: {self.damage\_level}%."

**Відповіді на контрольні питання**

1. Що таке модулі? ***Модулі*** *– це файли, які містять код і дані, які можна повторно використовувати в інших частинах програми.*

2. Як імпортувати модуль? *import module\_name*

3. Як оголосити клас? *class MyClass:*

4. Що може міститися у класі? ***У класі*** *можуть міститися: атрибути (змінні), методи (функції), конструктори, статичні та класові методи*.

5. Як називається конструктор класу? *\_\_init\_\_*

6. Як здійснити спадкування? *class ChildClass(ParentClass):*

7. Які види спадкування існують? ***Види спадкування****: одиничне, множинне, багаторівневе, ієрархічне, гібридне*.

8. Які небезпеки є при множинному спадкуванні, як їх уникнути? ***Небезпеки при множинному спадкуванні****: конфлікти методів та "ромбовидна проблема". Уникати можна через використання* ***MRO (Method Resolution Order)*** *або інтерфейсів/домішок.*

9. Що таке класи-домішки? *невеликі класи, які додають додаткову функціональність і використовуються як додаток до основного класу.*

10. Яка роль функції super() при спадкуванні? *дозволяє викликати методи чи конструктори батьківського класу, що корисно для уникнення дублювання коду при спадкуванні.*

**Висновок:** Я оволодів навиками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python.