

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
Бокало П.М.

Прийняв:

доцент кафедри ЕОМ
Іванов Ю. С.

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

Завдання(варіант №2):

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.

Індивідуальне завдання:

2. Космічний корабель

2. Багаторазовий космічний корабель

Вихідний код програми:

Файл Bokalo.KI_306_1.Lab9.py:

```
from Starship import Starship
from Multi_Starship import Multi_Starship

def main():
    ms = Multi_Starship("Boss", "Soreman")

    ms.stats()

    print("\n")

    for i in range(10):
        ms.fight()

    print("\n")

    ms.stats()

    print("\n")
```

```

ms.repair()

print("\n")

ms.call_captain()

print("\n")

ms.stats()

print("\n")

ms.call_captain()

print("\n")

ms.closer()

print("\n")

if __name__ == "__main__":
    main()

```

Файл Starship.py:

```

class Starship:
    def __init__(self, name="void", c_name="Noname"):
        self.fuel = 100
        self.s_name = name
        self.s_health = 100
        self.staff = 10
        self.captain = Captain(c_name)
        self.is_fighting = False
        self.fout = None

    def closer(self):
        if self.fout is not None:
            self.fout.close()
            self.captain.close()

    def call_captain(self):
        self.captain.Captain_stats()

    def move(self):
        if self.fuel > 10:
            self.fuel -= 10
        else:
            print("Can't move!!!")

    def damage(self):
        if self.s_health > 0 and self.staff > 0:
            self.s_health -= 10
            self.staff -= 1
        elif self.s_health == 0:
            print("Starship is critically damaged!!!")

    def repair(self):
        if self.s_health < 100:
            self.staff += 5
            self.s_health += 10
            coins = self.captain.getCoins()
            coins -= 50
            self.captain.setCoins(coins)
        else:
            print("Repaired")

```

```

def fight(self):
    self.is_fighting = True
    self.damage()
    coins = self.captain.getCoins()
    self.captain.setCoins(coins + 25)

def retreat(self):
    self.is_fighting = False

def stats(self):
    print("Stats:")
    print("Name:", self.s_name)
    print("Fuel:", self.fuel)
    print("Health:", self.s_health)
    print("Staff:", self.staff)
    print("Fighting:", self.is_fighting)

def refuel(self):
    self.fuel = 100
    coins = self.captain.getCoins()
    self.captain.setCoins(coins - 25)

def clearShip_logs(self):
    try:
        with open("Starship_log.txt", "w") as clearWriter:
            clearWriter.close()
    except FileNotFoundError:
        print("Error: File 'Starship_log.txt' cannot be cleared.")

class Captain:
    def __init__(self, name="Noname"):
        self.name = name
        self.coins = 100
        self.health = 100
        self.id = 0
        self.fout = None

    def getCoins(self):
        return self.coins

    def setCoins(self, coins):
        self.coins = coins

    def Captain_stats_file(self):
        print("New Captain created:")
        print("ID:", self.id)
        print("Name:", self.name)
        print("Coins:", self.coins)
        print("Health:", self.health)

    def Captain_stats(self):
        print("Your captain has:")
        print("Name:", self.name)
        print("Coins:", self.coins)
        print("Health:", self.health)

    def close(self):
        if self.fout is not None:
            self.fout.close()

    def clearCaptain_logs(self):
        try:
            with open("Captain_log.txt", "w") as clearWriter:
                clearWriter.close()
        except FileNotFoundError:
            print("Error: File 'Captain_log.txt' cannot be cleared.")

```

Файл Multi_Starship.py:

```
from Starship import Starship

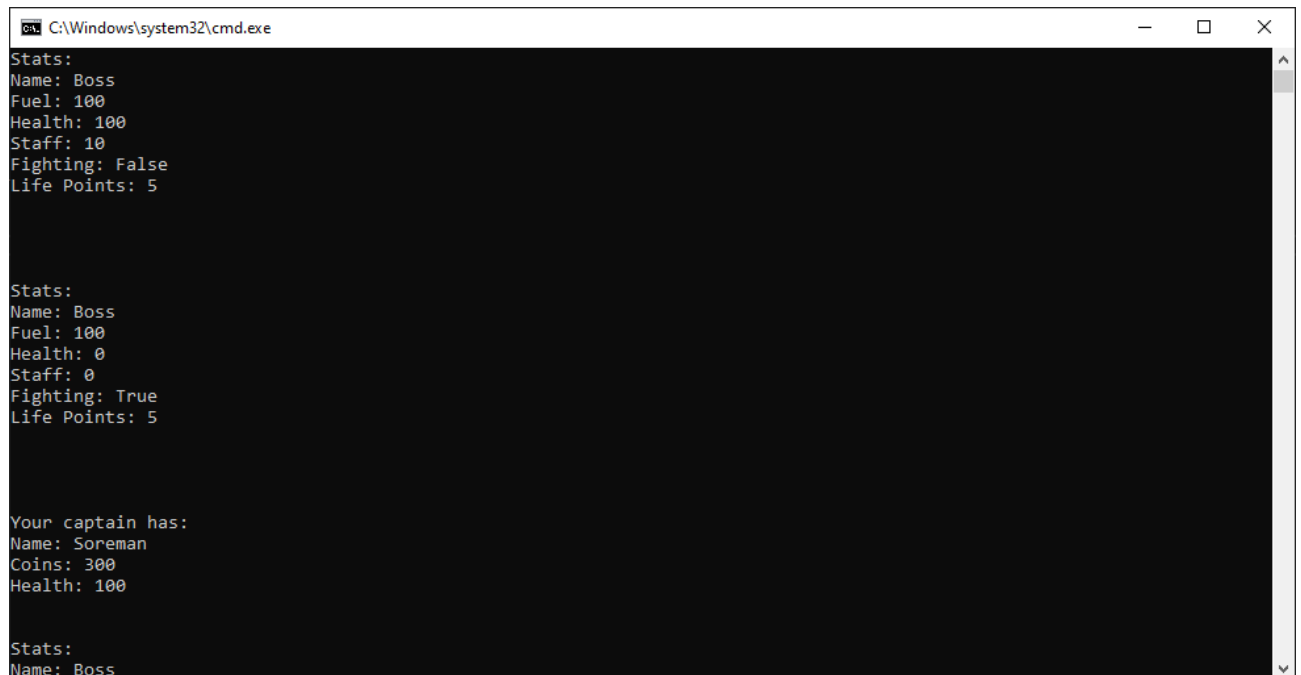
class Multi_Starship(Starship):
    def __init__(self, name="void", c_name="Noname"):
        super().__init__(name, c_name)
        self.Life_points = 5

    def Star_jump(self):
        for _ in range(5):
            self.move()

    def repair(self):
        if self.s_health == 0:
            self.Life_points = self.Life_points - 1 if self.Life_points > 0 else
self.Life_points
            super().repair()
        elif self.Life_points == 0:
            print("Out of life points!!! Get more")
        elif 0 < self.s_health < 100:
            super().repair()

    def stats(self):
        super().stats()
        print("Life Points:", self.Life_points)
```

Результат роботи програми:



```
C:\Windows\system32\cmd.exe

Stats:
Name: Boss
Fuel: 100
Health: 100
Staff: 10
Fighting: False
Life Points: 5

Stats:
Name: Boss
Fuel: 100
Health: 0
Staff: 0
Fighting: True
Life Points: 5

Your captain has:
Name: Soreman
Coins: 300
Health: 100

Stats:
Name: Boss
```

Відповідь на контрольні питання

1. Що таке модулі?

Модулі - це файли, які містять код для використання в інших програмах.

2. Як імпортувати модуль?

Імпорт модуля здійснюється за допомогою ключового слова "import" та імені модуля.

3. Як оголосити клас?

Клас оголошується ключовим словом "class", за яким слідує ім'я класу.

4. Що може міститися у класі?

У класі можуть міститися атрибути (змінні) та методи (функції).

5. Як називається конструктор класу?

Конструктор класу називається "init".

6. Як здійснити спадкування?

Спадкування здійснюється за допомогою наслідування від іншого класу.

7. Які види спадкування існують?

Існують одинарне та багат шарове спадкування.

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

При множинному спадкуванні можуть виникати конфлікти та нерозбіжності, які слід уникнути шляхом коректної організації класів.

9. Що таке класи-домішки?

Класи-домішки - це класи, які містять методи, які можна використовувати у інших класах.

10. Яка роль функції super() при спадкуванні?

Функція super() використовується для виклику методів батьківського класу при спадкуванні.

Висновок: на цій лабораторній роботі, я оволодів навичками реалізації парадигм об'єктно-орієнтованого програмування використовуючи засоби мови Python.