Міністерство освіти і науки України

Національний університет «Львівська політехніка»



Звіт

до лабораторної роботи №9

з дисципліни: “Кросплатформні засоби програмування”

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

Варіант №12

Виконала:

ст.гр. КІ-307

Коваль К. О.

Прийняв:

Іванов Ю. С.

**Львів 2023**

**Мета:** 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:

1. Що таке модулі?

2. Як імпортувати модуль?

3. Як оголосити клас?

4. Що може міститися у класі?

5. Як називається конструктор класу?

6. Як здійснити спадкування?

7. Які види спадкування існують?

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

9. Що таке класи-домішки?

10. Яка роль функції super() при спадкуванні?

**Варіант 12:**

**Базовий клас: Водойма**

**Похідний клас: Море**

**Виконання:**

**Main.py:**

from Sea import Sea  
  
# main program  
if \_\_name\_\_ == "\_\_main\_\_":  
 river = Sea()  
 print("Initial Details:")  
 river.printDetails()  
  
 river.setName("Amazon")  
 river.setArea(7000)  
 river.setType("River")  
  
 river.depthChange(10)  
 river.absorbWater(50)  
 river.evaporate(20)  
  
 river.pollute()  
  
 print("\nModified Details:")  
 river.printDetails()  
 print("\nNew Mods:")  
 river.dirtyWater()  
 # river.clearWater()

**Sea.py:**

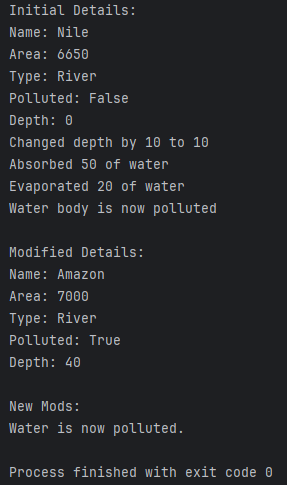
from WaterBody import WaterBody  
  
#class Sea  
class Sea(WaterBody):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()

**WaterBody.py:**

#class WaterBody  
class WaterBody:  
 def \_\_init\_\_(self):  
 self.\_\_name = "Nile"  
 self.\_\_area = 6650  
 self.\_\_type = "River"  
 self.\_\_isPolluted = False  
 self.\_\_depth = 0  
 def setName(self,name):  
 self.\_\_name = name  
 def setArea(self,area):  
 self.\_\_area = area  
 def setType(self,type):  
 self.\_\_type = type  
 def getName(self):  
 print("Accessed name: " + str(self.\_\_name))  
 return self.\_\_name  
 def getArea(self):  
 print("Accessed area: " + str(self.\_\_area))  
 return self.\_\_area  
 def getType(self):  
 print("Acessed type: " + str(self.\_\_type))  
 return self.\_\_type  
 def increaseArea(self,additionalArea):  
 self.\_\_area += additionalArea  
 print("Increased area by " + str(additionalArea) + "to " + str(self.\_\_area))  
 def changeType(self,newType):  
 self.\_\_type = newType  
 print("Changed type to " + str(newType))  
 def printDetails(self):  
 print("Name: " + str(self.\_\_name))  
 print("Area: " + str(self.\_\_area))  
 print("Type: " + str(self.\_\_type))  
 print("Polluted: " + str(self.\_\_isPolluted))  
 print("Depth: " + str(self.\_\_depth))  
 def rainfallEffect(self,rainfall):  
 if rainfall>50:  
 self.increaseArea(rainfall)  
 else:  
 print("Rainfall was not sufficient to increase area")  
 def pollute(self):  
 self.\_\_isPolluted = True  
 print("Water body is now polluted")  
 def clean(self):  
 self.\_\_isPolluted = False  
 print("Water body is now clean")  
 def setDepth(self,depth):  
 self.\_\_depth = depth  
 print("Set depth to " + str(depth))  
 def depthChange(self,change):  
 self.\_\_depth += change  
 print("Changed depth by " + str(change) + " to " + str(self.\_\_depth))  
 def evaporate(self, amount):  
 if amount<self.\_\_depth:  
 self.\_\_depth -= amount  
 print("Evaporated " + str(amount) + " of water")  
 else:  
 self.\_\_depth = 0  
 print("Evaporated all water, depth is now 0")  
 def absorbWater(self,amount):  
 self.\_\_depth += amount  
 print("Absorbed " + str(amount) + " of water")  
 def dirtyWater(self):  
 self.\_\_isPolluted = True  
 print("Water is now polluted.")  
 def clearWater(self):  
 self.\_\_isPolluted = False  
 print("Water is now clear.")

self.\_\_intact=intact

**Результат:**

****

**Відповіді на контрольні запитання:**

1. \*\*Модулі в Python:\*\*

- Модуль — це файл, який містить Python-код. Він може містити змінні, функції та класи.

2. \*\*Імпорт модуля в Python:\*\*

- Використовується ключове слово `import`. Наприклад:

```python

import module\_name

```

3. \*\*Оголошення класу в Python:\*\*

- Використовується ключове слово `class`. Наприклад:

```python

class MyClass:

# тіло класу

```

4. \*\*Що може міститися у класі:\*\*

- Змінні класу (атрибути)

- Методи класу

- Конструктор (спеціальний метод `\_\_init\_\_`)

- Інші спеціальні методи (наприклад, `\_\_str\_\_` для зручного виводу об'єкта)

5. \*\*Конструктор класу в Python:\*\*

- Конструктор класу має ім'я `\_\_init\_\_` і викликається автоматично при створенні нового об'єкта класу.

```python

class MyClass:

def \_\_init\_\_(self, param1, param2):

self.param1 = param1

self.param2 = param2

```

6. \*\*Спадкування в Python:\*\*

- Використовується ключове слово `class` з ім'ям батьківського класу у визначенні нового класу.

```python

class ChildClass(ParentClass):

# тіло класу-спадкоємця

```

7. \*\*Види спадкування в Python:\*\*

- Одинарне спадкування: клас успадковує властивості лише від одного батьківського класу.

- Багатошарове спадкування: клас успадковує властивості від більше ніж одного класу.

8. \*\*Небезпеки при множинному спадкуванні і їх уникнення:\*\*

- Конфлікти імен методів або атрибутів у різних батьківських класах. Можна використовувати роздільні простори імен (`super()`).

9. \*\*Класи-домішки в Python:\*\*

- Класи, які містять функціональність для використання в інших класах. Їх можна успадковувати багато разів.

10. \*\*Роль функції `super()` при спадкуванні в Python:\*\*

- `super()` використовується для звернення до методів батьківського класу. Наприклад, `super().\_\_init\_\_()` викликає конструктор батьківського класу.

**Висновок:** на даній лабораторній роботі я оволоділа навиками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python.