Національний університет «Львівська Політехніка»

Інститут комп’ютерних технологій, автоматики та метрології

Кафедра електронних обчислювальних машин



Звіт

Про виконання лабораторної роботи №9

З дисципліни «Кросплатформлені засоби програмування»

**Виконав:**

студент групи КІ-305

Ключко Д.С.

**Перевірив:**

Доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

**Тема:** Основи об’єктно-орієнтованого програмування у Python.

**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.

**Завдання згідно варіанту:**

**Варіант: 12**

***Базовий клас:***



***Похідний клас:***



**Виконання:**

**Код програми:**

***main.py***

from my\_Sea import Sea  
  
# основна програма  
if \_\_name\_\_ == "\_\_main\_\_":  
 # об'єкт на основі класу море  
 black = Sea(27, 11, 34, 4, 0)  
  
 # використання методів класу за допомогою створеного об'єкта  
 black.Shippingbalance()  
 black.Printer("Сальдо суднолавства:", black.getShippingbalance(), "DWT")  
 black.Printer("Приріст кораблів:", black.Shipgrowth(), "GRT")  
 black.setParametersSquare(11.5, 32)  
 black.Printer("Площа водойми:", black.getCalcSquare(), "m^2")  
 black.Printer("Максимальна глибина:", black.getMaxDepth(), "m")  
 black.Printer("Поточна глибина:", black.getCurrentDepth(), "m")  
 black.setParametersforEcologysituation("Зелений", 20)  
 black.CurentEcologysituation()  
 black.setParametersforInformation("Україна", "Закарпаття", "Озеро")  
 black.getInformation()

***my\_EcologyWater.py***

# клас екологічна ситуація водойми  
class EcologyWater:  
 # конструктор  
 def \_\_init\_\_(self, Colorwater="default", Transparency=0):  
 self.\_\_colorwater = Colorwater  
 self.\_\_transparency = Transparency  
 # метод повернення кольору води  
 def getColorwater(self):  
 return self.\_\_colorwater  
 # метод повернення прозорості води  
 def getTransparency(self):  
 return self.\_\_transparency  
 # метод встановлення кольору води  
 def setColorwater(self, Colorwater):  
 self.\_\_colorwater = Colorwater  
 # метод встановлення прозорості води  
 def setTransparency(self, Transparency):  
 self.\_\_transparency = Transparency  
 # метод для визначення екологічної ситуації водойми  
 def Ecologysituation(self):  
 if(self.\_\_colorwater == "Зелений" or self.\_\_colorwater == "Жовтий" or self.\_\_colorwater == "Коричневий" ):  
 print("Колір води є незадовільним! В даній водоймі брудна вода!!!")  
 else:  
 print("За результатами перевірки кольору води, водойма є чистою!")  
 if(self.\_\_transparency < 60):  
 print("Вода не є прозорою! В даній водоймі брудна вода!!!")  
 else:  
 print("Вода прозора,отже водойма є чистою!")

***my\_Fullinformation.py***

# клас повна інформація про водойму  
class Fullinformation:  
 # конструктор  
 def \_\_init\_\_(self, Country="default", Region="default", Type="default"):  
 self.\_\_country = Country  
 self.\_\_region = Region  
 self.\_\_type = Type  
 # метод повернення країни  
 def getCountry(self):  
 return self.\_\_country  
 # метод повернення регіону  
 def getRegion(self):  
 return self.\_\_region  
 # метод повернення типу  
 def getType(self):  
 return self.\_\_type  
 # метод встановлення країни  
 def setCountry(self, Country ):  
 self.\_\_country = Country  
 # метод встановлення регіону  
 def setRegion(self, Region):  
 self.\_\_region = Region  
 # метод встановлення типу   
 def setType(self, Type):  
 self.\_\_type = Type

***my\_square.py***

# клас площа водойми  
class Square:  
#конструктор  
 def \_\_init\_\_(self, xwidth=0, xlength=0):  
 self.\_\_width = xwidth  
 self.\_\_length = xlength  
#метод повернення ширини  
 def getWidth(self):  
 return self.\_\_width  
#метод повернення довжини  
 def getLength(self):  
 return self.\_\_length  
#метод встановлення ширини  
 def setWidth(self, xwidth):  
 self.\_\_width = xwidth  
#метод встановлення довжини  
 def setLength(self, xlength):  
 self.\_\_length = xlength  
#метод для обрахунку площі водойми  
 def CalcSquare(self):  
 square = self.\_\_width \* self.\_\_length  
 return square  
#метод для обрахунку глибини водойми  
 def CalcDepth(self):  
 depth = 4\*(self.\_\_width) / (self.\_\_length)  
 return depth

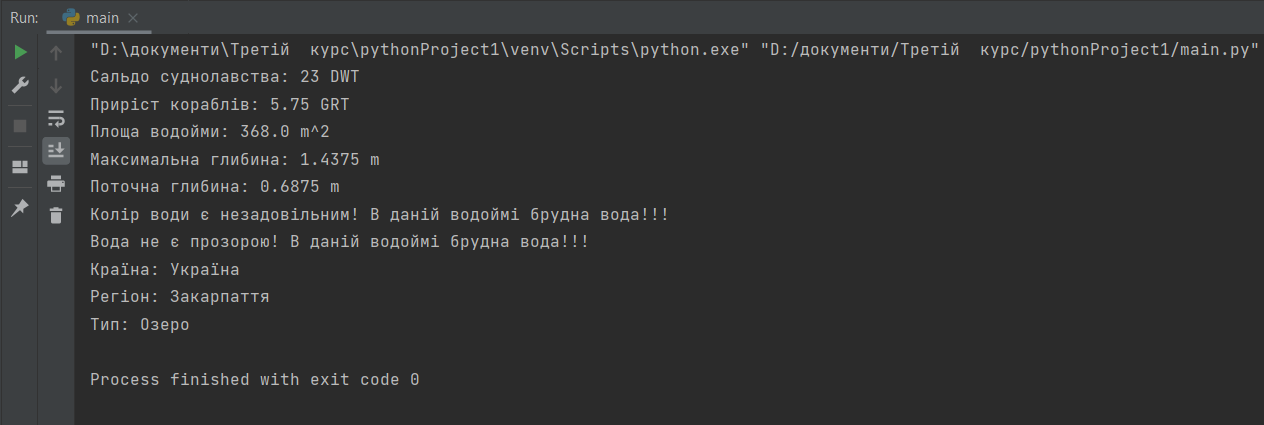
***my\_Water.py***

from my\_square import Square  
from my\_EcologyWater import EcologyWater  
from my\_Fullinformation import Fullinformation  
  
#клас водойма  
class Water:  
 # конструктор  
 def \_\_init\_\_(self, xtemperature=0):  
 self.\_\_squareofwater = Square()  
 self.\_\_ecologysytuation = EcologyWater()  
 self.\_\_information = Fullinformation()  
 self.\_\_temperature = xtemperature  
  
 # метод для отримання площі водойми  
 def getCalcSquare(self):  
 return self.\_\_squareofwater.CalcSquare()  
  
 # метод для отримання поточної глибини  
 def getCurrentDepth(self):  
 currentdepth = 0.0  
 if (self.\_\_temperature >= 25 and self.\_\_temperature <= 45):  
 currentdepth = self.\_\_squareofwater.CalcDepth() - 0.75  
 return currentdepth  
 else:  
 currentdepth = self.\_\_squareofwater.CalcDepth()  
 return currentdepth  
  
 # метод для встановлення параметрів для обчислення площі  
 def setParametersSquare(self, xwidth, xlength):  
 self.\_\_squareofwater.setWidth(xwidth)  
 self.\_\_squareofwater.setLength(xlength)  
  
 # метод для поверненя ширини  
 def getWidthforWater(self):  
 return self.\_\_squareofwater.getWidth()  
  
 # метод для повернення довжини  
 def getLengthforWater(self):  
 return self.\_\_squareofwater.getLength()  
  
 # метод для встановлення параметрів щоб оцінити екологічну ситуацію водойми  
 def setParametersforEcologysituation(self, Colorwater, Transparency):  
 self.\_\_ecologysytuation.setColorwater(Colorwater)  
 self.\_\_ecologysytuation.setTransparency(Transparency)  
  
 # метод для оцінки екологічної ситуації водойми  
 def CurentEcologysituation(self):  
 return self.\_\_ecologysytuation.Ecologysituation()  
  
 # метод для поверненя максимальної глибини водойми  
 def getMaxDepth(self):  
 return self.\_\_squareofwater.CalcDepth()  
  
 # метод для встановлення повної інформації про водойму  
 def setParametersforInformation(self, Country, Region, Type):  
 self.\_\_information.setCountry(Country)  
 self.\_\_information.setRegion(Region)  
 self.\_\_information.setType(Type)  
  
 # метод для отримання повної інформації про водойму  
 def getInformation(self):  
 print("Країна:", self.\_\_information.getCountry())  
 print("Регіон:", self.\_\_information.getRegion())  
 print("Тип:", self.\_\_information.getType())

***my\_Sea.py***

from my\_Water import Water  
  
#клас море  
class Sea(Water):  
 #конструктор  
 def \_\_init\_\_(self, xtemperature=0, Xshipsaresailing=0, Xshipsareinports=0, Xdata=0, Xshippinbalance=0):  
 super().\_\_init\_\_(xtemperature)  
 self.\_\_shipsaresailing = Xshipsaresailing  
 self.\_\_shipsareinports = Xshipsareinports  
 self.\_\_data = Xdata  
 self.\_\_shippinbalance = Xshippinbalance  
 #метод обраховує сальдо судноплавства  
 def Shippingbalance(self):  
 self.\_\_shippinbalance = self.\_\_shipsareinports - self.\_\_shipsaresailing  
  
 #метод повертає сальдо судноплавства  
 def getShippingbalance(self):  
 return self.\_\_shippinbalance  
 #метод повертає приріст кораблів  
 def Shipgrowth(self):  
 return self.\_\_shippinbalance / self.\_\_data  
 # метод виводить на екран властивості класу  
 def Printer(self, line,line1, line3):  
 print(line, line1, line3)

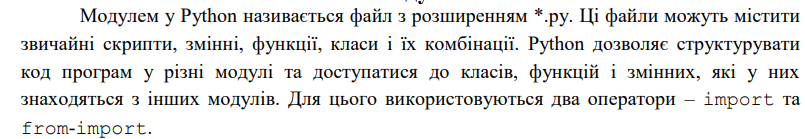
**Результат роботи програми у консолі:**

****

**КОНТРОЛЬНІ ПИТАННЯ**

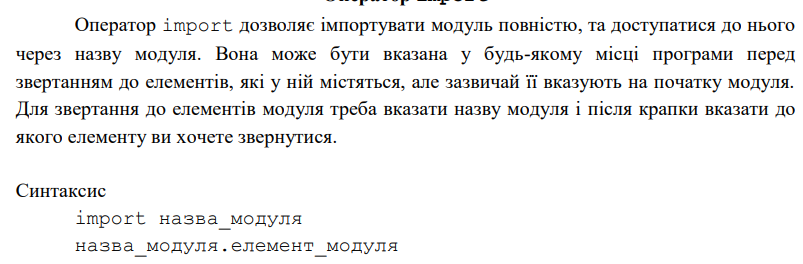
1. Що таке модулі?

**Відповідь:**

****

1. Як імпортувати модуль?

**Відповідь:**



1. Як оголосити клас?

**Відповідь:**

Клас оголошується за допомогою ключового слова class після якого йде назва класу.



1. Що може міститися у класі?

**Відповідь:**

Клас може містити:

- дані, які належать класу (статичні дані-члени класу);

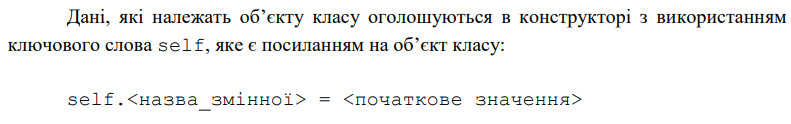
- дані, які належать об’єкту класу;

- методи, які належать класу (статична методи);

- методи, які належать об’єкту класу.

1. Як називається конструктор класу?

**Відповідь:**

****

****

1. Як здійснити спадкування?

**Відповідь:**

Для реалізації спадкування класи, які слід успадкувати вказуються у круглих дужках через кому після назви класу, який оголошується:



1. Які види спадкування існують?

**Відповідь:**

Python дозволяє реалізовувати як одинарне так і множинне спадкування.

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

**Відповідь:**

Конфлікти імен: Якщо два батьківські класи мають методи чи атрибути з однаковими іменами, унаслідуваний клас не зможе однозначно визначити, який саме метод чи атрибут використовувати.

Складність: Чим більше класів використовує множинне спадкування, тим складніше стає розуміння програми та відстеження поведінки класів

Залежність: Існує ризик зміни в одному з батьківських класів, що може призвести до зміни унаслідованого класу, що відображається на усіх його підкласах.

1. Що таке класи-домішки?

**Відповідь:**

Клас-домішка або mixin class – це клас, який використовується у цьому шаблоні, надаючи функціональні можливості (методи), але не призначений для самостійного використання у вигляді об’єктів класу. В ідеальному випадку класи-домішки не мають власної ієрархії спадкування і не мають полів, а мають лише методи.

1. Яка роль функції super() при спадкуванні?

**Відповідь:**

Функція super() використовується в мові програмування Python для доступу до методів та атрибутів батьківського класу з підкласу. Коли клас успадковується від іншого класу, він може використовувати super() для виклику методів батьківського класу.

Роль super() полягає в тому, щоб забезпечити можливість отримати доступ до методів батьківського класу у випадку, коли в підкласі визначено свої власні методи з тими ж іменами.

**Висновок:** на цій лабораторній роботі я оволодів навиками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python. Створив клас водойма, який наслідує деякі властивості класів площа водойми, поточна екологічна ситуація водойми та повна інформація про водойму. Також створив клас море, на основі класу водойми і розробив програму-драйвер, у якій створив об’єкт на основі цього класу та протестував його методи.