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-301

*Солтисяк О.Р.*

**Прийняв:**

доцент кафедри ЕОМ

*Іванов Ю. С.*

Львів – 2024

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

**Завдання (варіант № 26)**

1. Написати та налагодити програму на мові Python згідно варіанту **(**26. Шлюпка на веслах**🡪 Похідний клас** Моторний човен**)**. 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.

1. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
2. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
3. Дати відповідь на контрольні запитання.

**Вихідний код програми:**

from boats.motor\_boat import MotorBoat

from boats.sail\_boat import SailBoat

from boats.yacht import Yacht

def main():

# Creating an instance of MotorBoat

motor\_boat = MotorBoat(engine\_power=100, oars=2, material='aluminum', fuel\_type='diesel', max\_speed=40)

# Using RowBoat functionality

print(motor\_boat.row())

motor\_boat.set\_oars\_count(4)

motor\_boat.set\_material('fiberglass')

print(motor\_boat.row())

print(motor\_boat.dock())

# Using MotorBoat functionality

print(motor\_boat.start\_engine())

motor\_boat.set\_engine\_power(150)

motor\_boat.set\_fuel\_type('electric')

print(motor\_boat.start\_engine())

print(motor\_boat.accelerate())

print(motor\_boat.anchor())

print(motor\_boat.stop\_engine())

# Creating an instance of SailBoat

sail\_boat = SailBoat(sail\_area=50, material='canvas')

print(sail\_boat.hoist\_sail())

print(sail\_boat.lower\_sail())

print(sail\_boat.dock())

# Creating an instance of Yacht

yacht = Yacht(sail\_area=80, engine\_power=200, material='carbon fiber', luxury\_level='high')

print(yacht.hoist\_sail())

print(yacht.start\_engine())

print(yacht.entertain\_guests())

print(yacht.stop\_engine())

print(yacht.lower\_sail())

print(yacht.dock())

if \_\_name\_\_ == "\_\_main\_\_":

main()

***Файл yacht.py***

# boats/yacht.py

from .sail\_boat import SailBoat

class Yacht(SailBoat):

def \_\_init\_\_(self, sail\_area, engine\_power, material='fiberglass', luxury\_level='high'):

super().\_\_init\_\_(sail\_area, material)

self.engine\_power = engine\_power # Engine power in horsepower

self.luxury\_level = luxury\_level # Level of luxury: low, medium, high

def start\_engine(self):

return f"Starting the yacht's engine with {self.engine\_power} horsepower."

def stop\_engine(self):

return "Stopping the yacht's engine."

def get\_luxury\_level(self):

return self.luxury\_level

def set\_luxury\_level(self, level):

self.luxury\_level = level

def entertain\_guests(self):

return f"Entertaining guests in a {self.luxury\_level} luxury level yacht."

***Файл sailboat.py***

# boats/sail\_boat.py

class SailBoat:

def \_\_init\_\_(self, sail\_area, material='wood'):

self.sail\_area = sail\_area # Sail area in square meters

self.material = material

def hoist\_sail(self):

return f"Hoisting the sail with an area of {self.sail\_area} square meters."

def lower\_sail(self):

return "Lowering the sail."

def get\_sail\_area(self):

return self.sail\_area

def set\_sail\_area(self, sail\_area):

self.sail\_area = sail\_area

def get\_material(self):

return self.material

def set\_material(self, material):

self.material = material

def dock(self):

return "The sailboat is docking."

***Файл motorboat.py***# boats/motor\_boat.py

from .row\_boat import RowBoat

class MotorBoat(RowBoat):

def \_\_init\_\_(self, engine\_power, oars=2, material='wood', fuel\_type='gasoline', max\_speed=30):

super().\_\_init\_\_(oars, material)

self.engine\_power = engine\_power # Engine power in horsepower

self.fuel\_type = fuel\_type # Type of fuel used by the motor

self.max\_speed = max\_speed # Maximum speed in knots

def start\_engine(self):

return f"Starting the engine with {self.engine\_power} horsepower, running on {self.fuel\_type}."

def stop\_engine(self):

return "Stopping the engine."

def get\_engine\_power(self):

return self.engine\_power

def set\_engine\_power(self, power):

self.engine\_power = power

def get\_fuel\_type(self):

return self.fuel\_type

def set\_fuel\_type(self, fuel\_type):

self.fuel\_type = fuel\_type

def accelerate(self):

return f"The motorboat is accelerating to a maximum speed of {self.max\_speed} knots."

def anchor(self):

return "The motorboat is anchoring."  
  
  
***Файл rowboat.py***

# boats/row\_boat.py

class RowBoat:

def \_\_init\_\_(self, oars=2, material='wood'):

self.oars = oars

self.material = material

def row(self):

return f"Rowing the boat with {self.oars} oars made of {self.material}."

def get\_oars\_count(self):

return self.oars

def set\_oars\_count(self, count):

self.oars = count

def get\_material(self):

return self.material

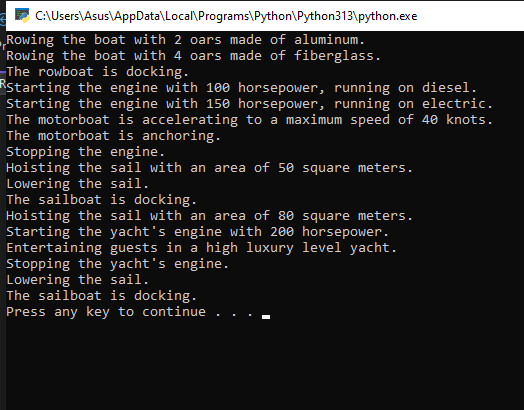
def set\_material(self, material):

self.material = material

def dock(self):

return "The rowboat is docking."

**Результат виконання програми:**



**Відповіді на контрольні запитання:**

1. Що таке модулі?

Модулі в Python - це файли, які містять код Python. Вони використовуються для організації коду на логічні блоки і полегшення управління.

1. Як імпортувати модуль?

Модуль можна імпортувати за допомогою ключового слова import. Наприклад:

import my\_module

1. Як оголосити клас?

Клас оголошується за допомогою ключового слова class. Наприклад:

class MyClass:

# тіло класу

1. Що може міститися у класі?

У класі можуть міститися атрибути (змінні) і методи (функції), що дозволяє організувати дані та пов'язані з ними операції.

1. Як називається конструктор класу?

Конструктор класу називається \_\_init\_\_. Це спеціальний метод, який викликається автоматично при створенні нового об'єкта класу.

1. Як здійснити спадкування?

Спадкування здійснюється шляхом вказання батьківського класу у визначенні нового класу. Наприклад:

class ChildClass(ParentClass):

# тіло класу

1. Які види спадкування існують?

У Python існують єдине спадкування, коли клас може успадковувати властивості лише від одного батьківського класу.

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

Небезпека множинного спадкування полягає в можливому конфлікті імен та складнощах з розумінням коду. Для уникнення цього слід стежити за чітким розподілом відповідальностей і уникати зайвого спадкування.

1. Що таке класи-домішки?

Класи-домішки в Python - це класи, які містять функціональність для багатьох класів. Об'єкти цих класів можуть включати в себе функціональність, яка необов'язково визначається в класах, які успадковуються від них.

1. Яка роль функції super() при спадкуванні?

Функція super() використовується для отримання доступу до методів батьківського класу. Вона дозволяє вам викликати методи батьківського класу, щоб доповнити або замінити їх у підкласі.

**Висновок:**

На даній лабораторній роботі я оволодів навичками використання засобів об'єктно-орієнтованого програмування мовою Python. Також ознайомився з ключовими аспектами цієї парадигми, включаючи створення та використання класів, роботу з об'єктами, та використання спадкування для покращення ефективності програми.