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

Фодор Андрій

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

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

**Завдання (варіант № 21)**

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.

**Варіант 21. Суперклас – ПІСТОЛЕТ, Похідний – ВОДНИЙ ПІСТОЛЕТ**

**Вихідний код програми**

**Файл CBullet.py:**

class Bullet:

def \_\_init\_\_(self, count=20):

"""Constructor for the Bullet class."""

if count <= 20:

self.bullet = count

else:

self.bullet = 0

print("Pistol can consist of a maximum of 20 bullets")

def reduce(self):

"""Method to simulate using a bullet."""

if self.bullet == 0:

print("Your gun is unloaded")

return False

elif self.bullet == 1:

print("It was your last bullet")

self.bullet -= 1

return True

else:

self.bullet -= 1

return True

def reload(self):

"""Method to reload bullets to the maximum."""

self.bullet = 20

def get\_bullets(self):

"""Method to get the current bullet count."""

return self.bullet

**Файл CServiceability.py:**

import random

class Serviceability:

def \_\_init\_\_(self, model=None):

"""Constructor for the Serviceability class."""

self.available\_models = ["FN Five-seven", "Glock 17", "HK45 Tactical", "Sig Sauer M11-A1 9mm", "Beretta M9"]

self.model = model if model else random.choice(self.available\_models)

self.serv = 100

def get\_model(self):

"""Method to get the model of the pistol."""

return self.model

def get\_serv(self):

"""Method to get the current pistol serviceability."""

return self.serv

def reduce(self):

"""Method to simulate reducing serviceability with usage."""

self.serv -= random.randint(0, 2)

if self.serv < 10:

print("WARNING: Serviceability less than 10%")

def repair(self, money):

"""Method to repair the weapon for a specified cost."""

if money >= 500:

res = random.randint(0, 9)

if res >= 4:

print("Your weapon is repaired successfully")

self.serv = 100

return True

else:

print("Sorry, your weapon could not be repaired")

return False

else:

print("Repairs cost 500 units")

return False

**Файл CShooting.py:**

import random

class Shooting:

def \_\_init\_\_(self):

"""Constructor for the Shooting class."""

self.hit\_shoots = 0

self.missed\_shoots = 0

def get\_hit\_shoots(self):

"""Method to get the count of hit shoots."""

return self.hit\_shoots

def get\_missed\_shoots(self):

"""Method to get the count of missed shoots."""

return self.missed\_shoots

def get\_accuracy\_of\_shots(self):

"""Method to get the accuracy of shots."""

total\_shots = self.hit\_shoots + self.missed\_shoots

return f"{round((self.hit\_shoots / total\_shots) \* 100)}%"

def make\_shoot(self, distance, serviceability, bullets):

"""Method to simulate one shoot."""

if serviceability.get\_serv() <= 0:

print("Your weapon is broken")

return False

elif not bullets.reduce():

return False

else:

serviceability.reduce()

res = distance \* random.randint(0, 9)

if res < 50:

self.hit\_shoots += 1

return True

else:

self.missed\_shoots += 1

return False

**Файл CPistol.py:**

from .CBullet import Bullet

from .CShooting import Shooting

from .CServiceability import Serviceability

class Pistol:

count\_without\_param = 0

count\_with\_param = 0

def \_\_init\_\_(self, bullet\_count=None, model=None):

"""Constructor for the Pistol class."""

if bullet\_count is None and model is None:

self.bullets = Bullet()

self.serviceability = Serviceability()

self.shoot = Shooting()

Pistol.count\_without\_param += 1

else:

self.bullets = Bullet(bullet\_count)

self.serviceability = Serviceability(model)

self.shoot = Shooting()

Pistol.count\_with\_param += 1

def make\_some\_shoots(self, how\_many, distance):

"""Method to simulate shooting multiple times."""

print("-----You started shooting-----")

while how\_many != 0:

if self.shoot.make\_shoot(distance, self.serviceability, self.bullets):

print("You hit the target")

else:

print("You missed the target")

how\_many -= 1

def get\_hit\_shoots(self):

"""Method to get the count of hit shoots."""

return self.shoot.get\_hit\_shoots()

def get\_missed\_shoots(self):

"""Method to get the count of missed shoots."""

return self.shoot.get\_missed\_shoots()

def get\_accuracy(self):

"""Method to get the accuracy of shots."""

return self.shoot.get\_accuracy\_of\_shots()

def repair\_gun(self, money):

"""Method to repair the gun."""

if self.serviceability.repair(money):

print("-----You have repaired your gun-----")

else:

print("-----You failed to repair your gun-----")

def get\_guns\_serviceability(self):

"""Method to get the serviceability of the pistol."""

return self.serviceability.get\_serv()

def get\_guns\_model(self):

"""Method to get the model of the pistol."""

return self.serviceability.get\_model()

def dispose(self):

"""Method to release used resources."""

pass

def get\_guns\_bullet\_count(self):

"""Method to get the current bullet count."""

return self.bullets.get\_bullets()

def reload\_gun(self):

"""Method to reload the gun."""

self.bullets.reload()

print("-----You have reloaded your gun-----")

**Файл CWaterPistol.py:**

from .CPistol import Pistol

# Клас, що описує можливість використання водних пострілів

class waterable:

def realize\_waterable(self):

pass

waterable = True

# Клас, який розширює інтерфейс Waterable та описує можливість обчислення тиску води в зброї

class waterPressure(waterable):

k = 2

def calculate\_wp(self):

pass

class WaterPistol(Pistol, waterPressure):

def \_\_init\_\_(self, bullet\_count=5, model="", hole\_area=5):

super().\_\_init\_\_(bullet\_count, model)

self.hole\_area = hole\_area

def realize\_waterable(self):

self.water = self.waterable

return self.get\_guns\_model()

def calculate\_wp(self):

self.pressure = self.k \* 50 / self.hole\_area

return self.pressure

def make\_some\_shoots(self, how\_many, distance):

if self.water and self.pressure != 0:

super().make\_some\_shoots(how\_many, int(distance \* (20 / self.pressure)))

print("Water shooting is successful")

else:

print("Water shooting cannot be started because you don't realize waterAble or don't calculate pressure")

def get\_guns\_model(self):

return super().get\_guns\_model() + " water version"

def make\_settings(self):

self.calculate\_wp()

self.realize\_waterable()

return True

def dispose(self):

pass

**Файл \_\_init\_\_.py:**

from .CWaterPistol import WaterPistol

**Файл main.py:**

import classesPac

def main():

wp = classesPac.WaterPistol()

if wp.make\_settings():

wp.make\_some\_shoots(4, 10)

print("--- Weapon info ---\nModel = " + wp.get\_guns\_model() +

";\nServiceability = " + str(wp.get\_guns\_serviceability()) +

";\nBullets = " + str(wp.get\_guns\_bullet\_count()))

print("Pressure =", wp.calculate\_wp())

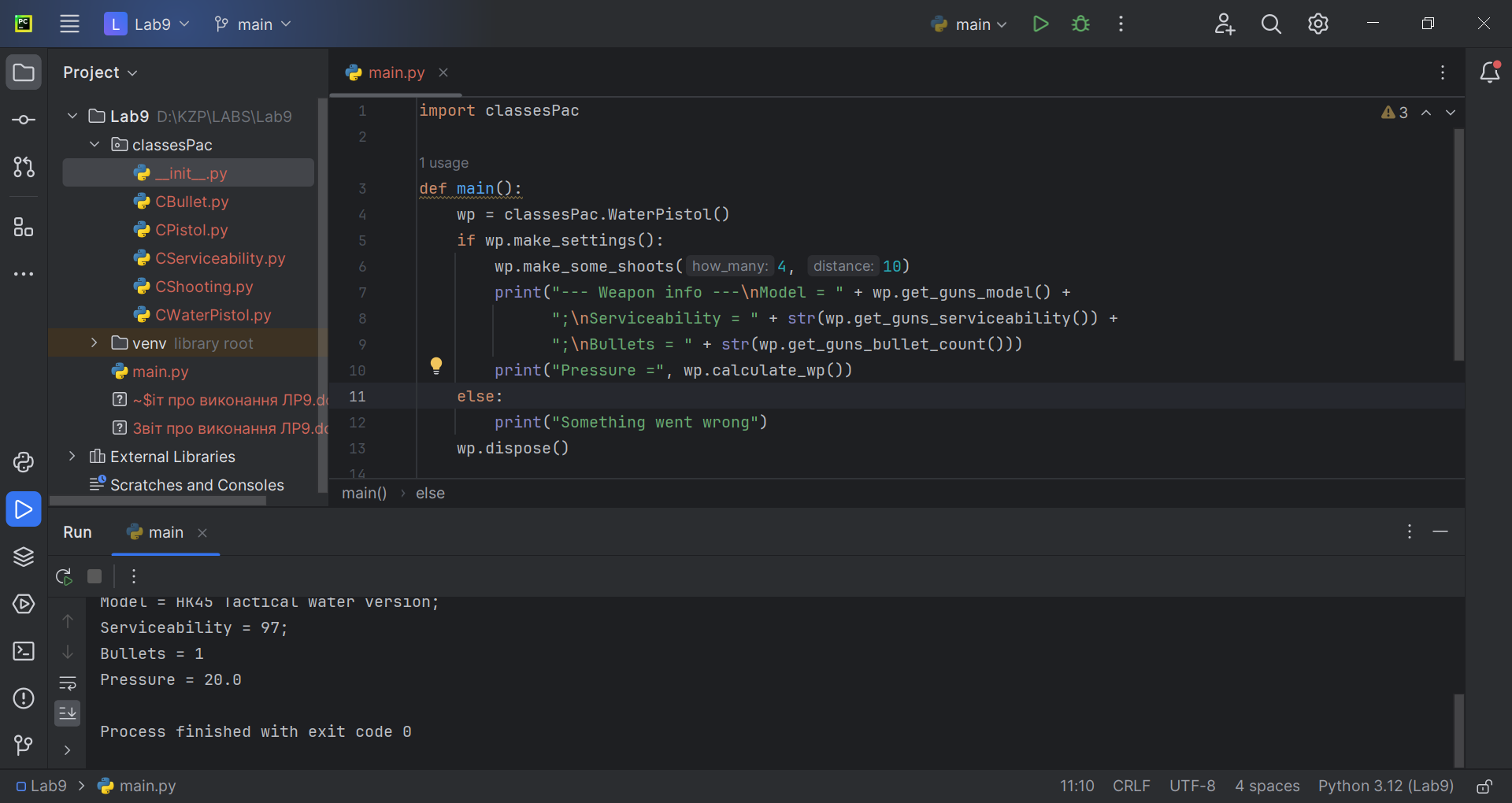
else:

print("Something went wrong")

wp.dispose()

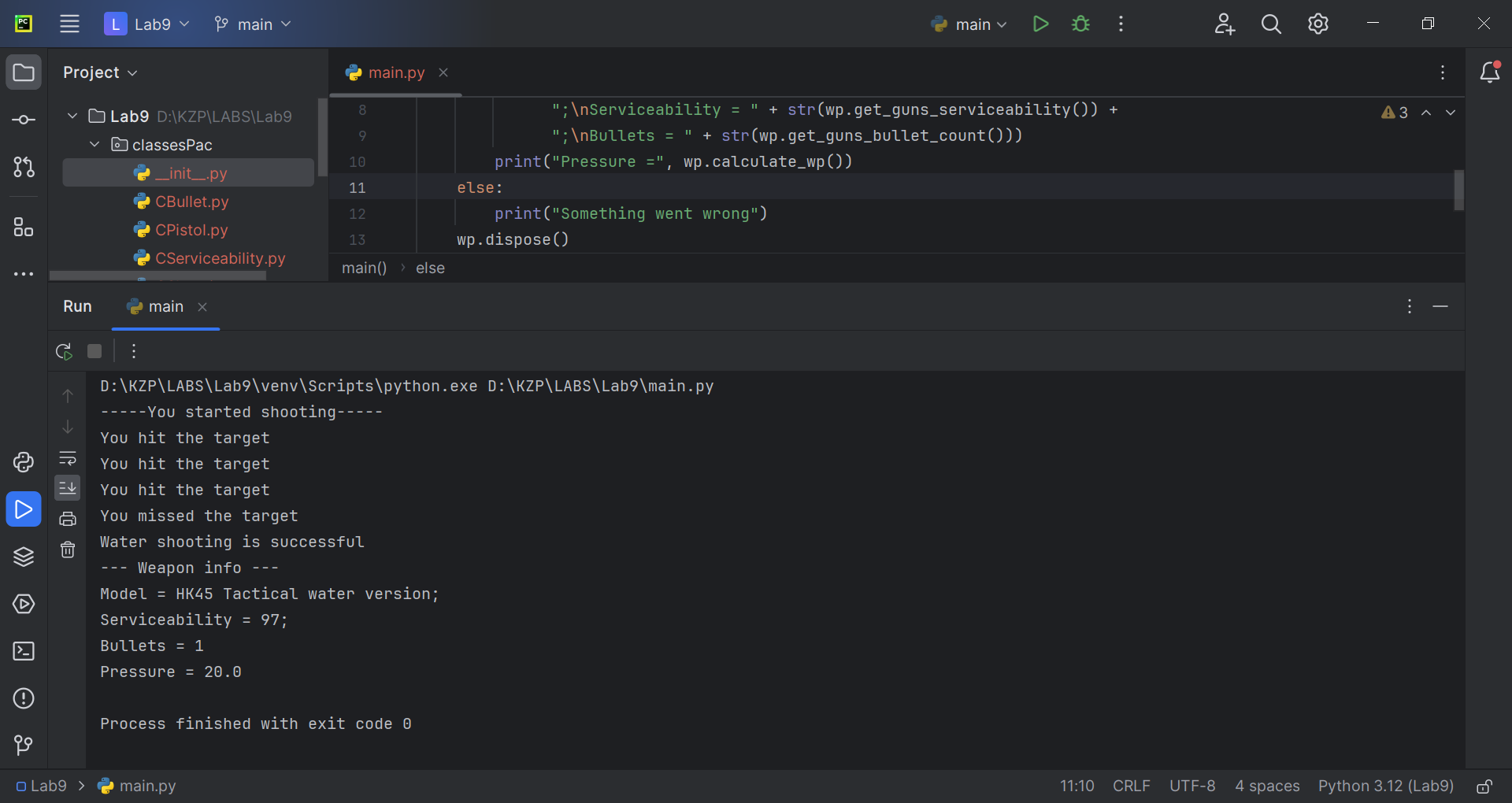
if \_\_name\_\_ == "\_\_main\_\_":

main()



Як видно з цього скріншота, всі класи знаходяться в окремих модулях, які знаходяться в окремому пакеті. Сама ж програма main.ру не знаходяться в цьому пакеті, але використовує клас з нього для демонстраціїї його роботи

**Результат виконання програми**



Як видно з наведеного скріншота, програма працює коректно

**Висновок:** на цій лабораторній роботі я оволодів навиками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python. Закріпив отримані знання на прикладі виконання індивідуального завдання, реалізувавши поліморфізм, множинне наслідування, спадкування та інкапсуляцію