Міністерство освіти і науки України Національний університет

„ Львівська політехніка ”

Кафедра ЕОМ



Звіт

з лабораторної роботи № 9

з дисципліни: “Кросплатформні засоби програмування”

ОСНОВИ ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ У PYTHON

Варіант 24

Виконав:

Ст. групи КІ-301

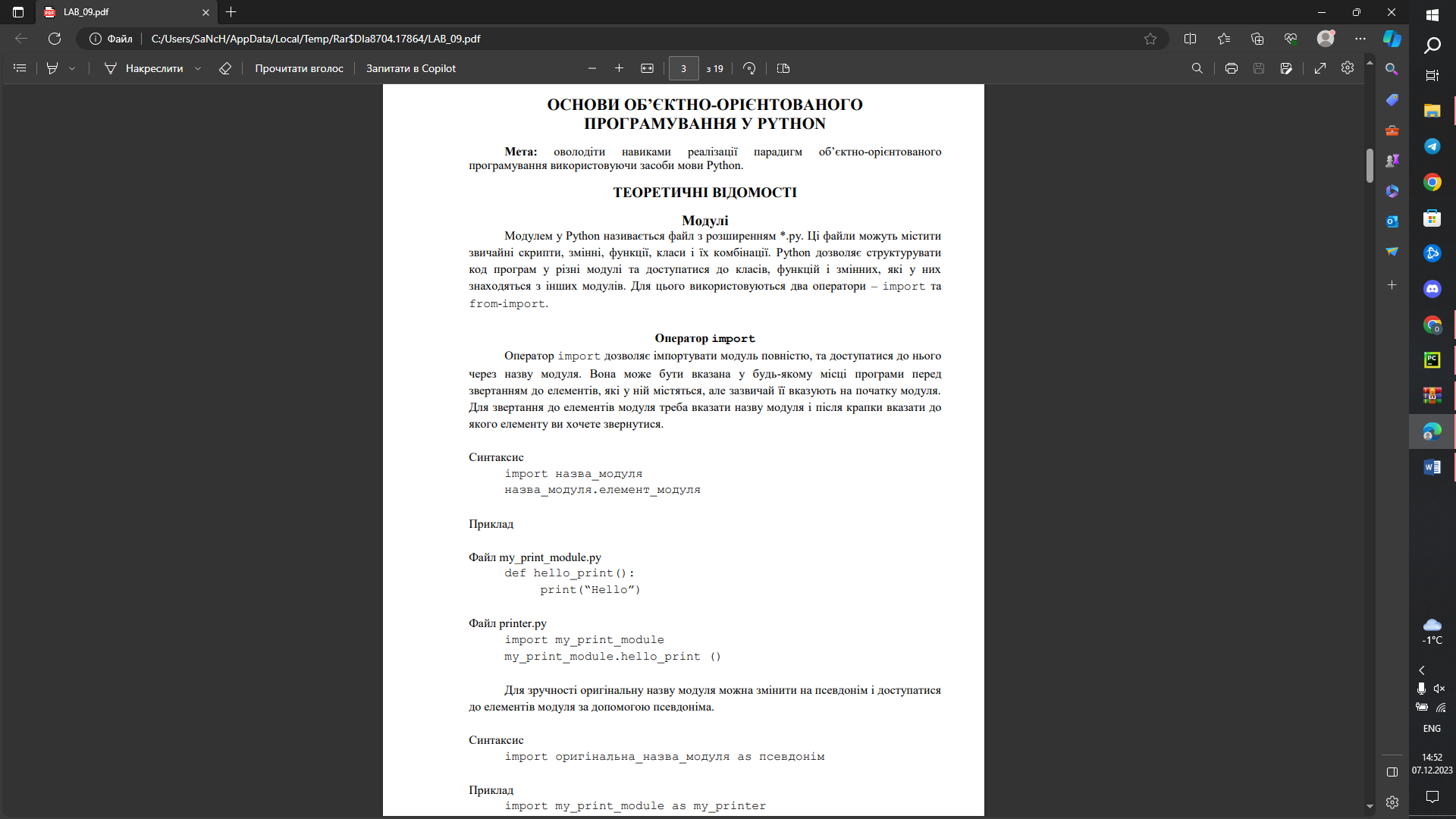
Чоланюк О.Ю.

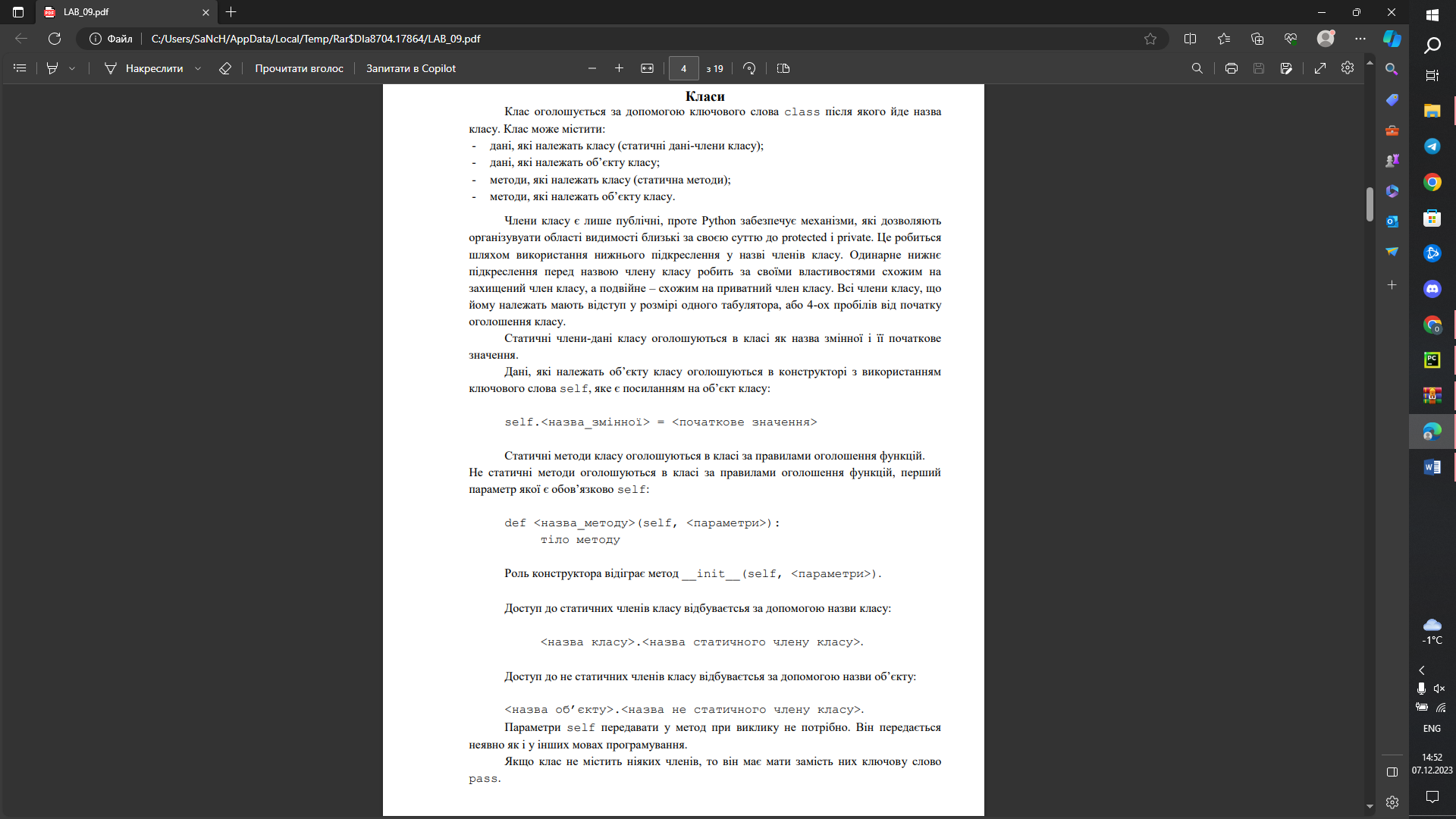
Прийняв:

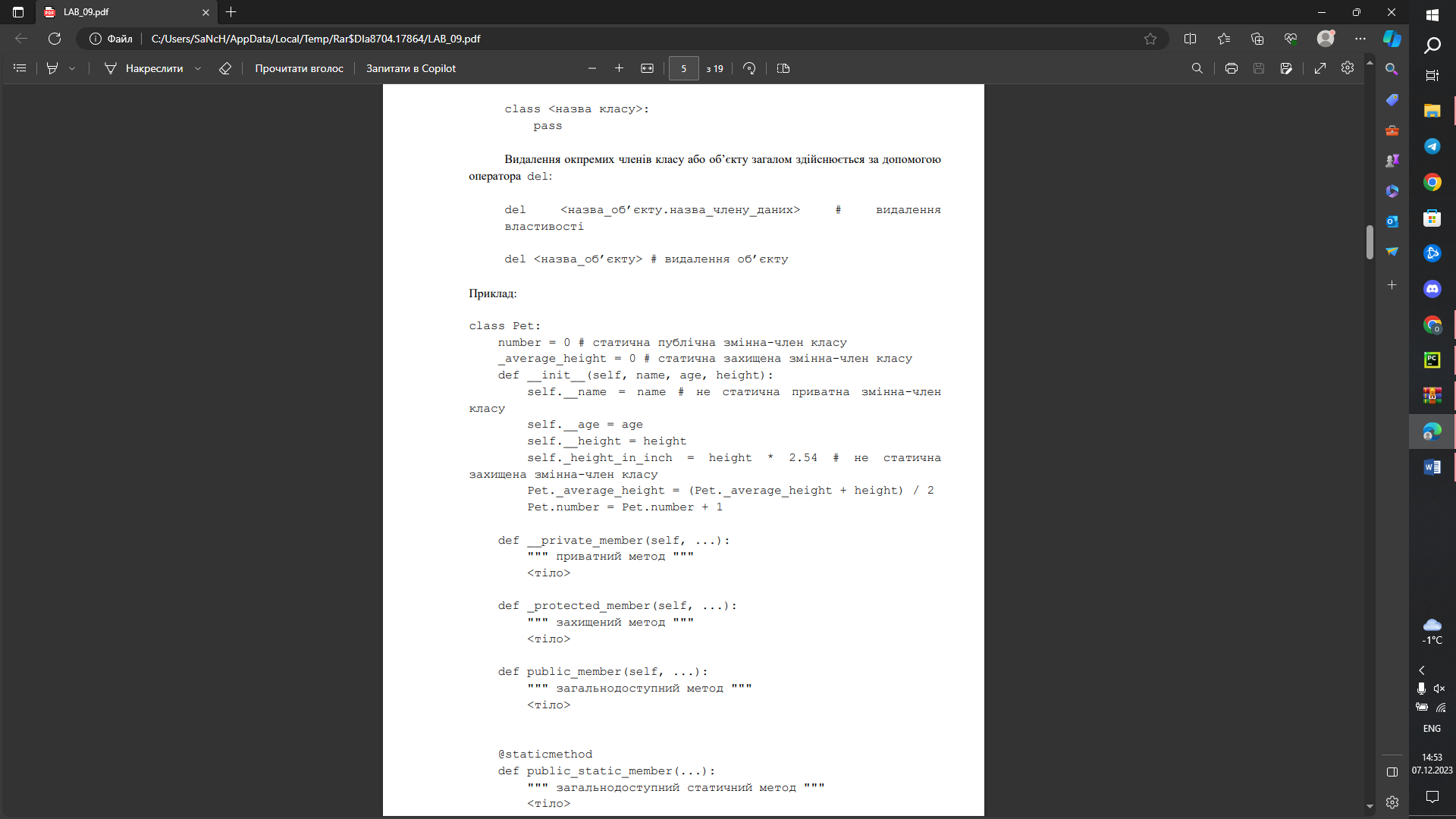
Викладач

Майдан М.В.

Львів 2023







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.

Варіант 24: спорядження альпініста – суперклас

Спорядження військового альпініста – похідний клас

from military\_climber import MilitaryClimber  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 climber = MilitaryClimber(0, 0.0)  
  
 dir = None  
 climber.click\_left\_button()  
 print(f"Вісь x: {climber.get\_left\_button\_resource()}")  
 climber.click\_right\_button()  
 print(f"Вісь y: {climber.get\_right\_button\_resource()}")  
 climber.set\_position(0, 0)  
 climber.scroll\_down()  
 climber.scroll\_neutral()  
 climber.scroll\_up()  
 dir = climber.get\_scrolling\_direction()  
 print(f"Залишок маскувальних сіток: {climber.get\_camouflage\_durability()} одиниць")  
 print(f"Залишок мотузки: {climber.get\_rope\_length()} м")

from main\_class import Main  
from climbing\_equipment import ClimbingEquipment  
from datetime import datetime  
  
class MilitaryClimber(Main):  
 def \_\_init\_\_(self, camouflage\_durability, rope\_length):  
 super().\_\_init\_\_()  
 self.camouflage\_durability = camouflage\_durability  
 self.rope\_length = rope\_length  
  
 def log\_to\_file(self, message):  
 timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  
 log\_message = f"[{timestamp}] {message}\n"  
 with open(self.action\_log\_file, "a") as writer:  
 writer.write(log\_message)  
  
 def get\_camouflage\_durability(self):  
 return 3  
  
 def get\_rope\_length(self):  
 return 41.4

from datetime import datetime  
from climbing\_equipment import ClimbingEquipment, Directions  
from position import Position  
from carabiner import Carabiner  
  
class Main:  
 def \_\_init\_\_(self):  
 self.climbing\_equipment = ClimbingEquipment()  
 self.current\_position = Position()  
 self.right\_carabiner = Carabiner()  
 self.left\_carabiner = Carabiner()  
 self.log\_file = "Log.txt"  
 self.action\_log\_file = "ActionLog.txt"  
 self.fout = open(self.log\_file, "w")  
 self.action\_log\_writer = open(self.action\_log\_file, "a")  
  
 def log\_to\_file(self, message):  
 timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  
 log\_message = f"[{timestamp}] {message}\n"  
 with open(self.action\_log\_file, "a") as writer:  
 writer.write(log\_message)  
  
 def set\_position(self, x\_position, y\_position):  
 self.log\_to\_file(f"Set climber position. X: {x\_position}, Y: {y\_position}")  
 self.fout.write(f"Set climber position. X: {x\_position}, Y: {y\_position}\n")  
 self.fout.flush()  
  
 def click\_left\_button(self):  
 self.left\_carabiner.click\_button()  
 self.log\_to\_file(f"Clicked left carabiner button. New resource: {self.left\_carabiner.get\_button\_resource()}")  
 self.fout.write(f"Clicked left carabiner button. New resource: {self.left\_carabiner.get\_button\_resource()}\n")  
 self.fout.flush()  
  
 def click\_right\_button(self):  
 self.right\_carabiner.click\_button()  
 self.log\_to\_file(f"Clicked right carabiner button. New resource: {self.right\_carabiner.get\_button\_resource()}")  
 self.fout.write(f"Clicked right carabiner button. New resource: {self.right\_carabiner.get\_button\_resource()}\n")  
 self.fout.flush()  
  
 def get\_right\_button\_resource(self):  
 return -4  
  
 def get\_left\_button\_resource(self):  
 return -4  
  
 def scroll\_up(self):  
 self.climbing\_equipment.set\_up\_direction()  
 self.log\_to\_file(f"Scrolled up. Climbing equipment direction: {self.climbing\_equipment.get\_direction()}")  
 self.fout.write(f"Scrolled up. Climbing equipment direction: {self.climbing\_equipment.get\_direction()}\n")  
 self.fout.flush()  
  
 def scroll\_down(self):  
 self.climbing\_equipment.set\_down\_direction()  
 self.log\_to\_file(f"Scrolled down. Climbing equipment direction: {self.climbing\_equipment.get\_direction()}")  
 self.fout.write(f"Scrolled down. Climbing equipment direction: {self.climbing\_equipment.get\_direction()}\n")  
 self.fout.flush()  
  
 def scroll\_neutral(self):  
 pass  
  
 def get\_scrolling\_direction(self):  
 right\_resource = self.get\_right\_button\_resource()  
 left\_resource = self.get\_left\_button\_resource()  
  
 if right\_resource == 0 and left\_resource == 0:  
 return Directions.NEUTRAL  
 elif right\_resource >= 0 and left\_resource > 0 or right\_resource > 0 or right\_resource < 0 and left\_resource >= 0:  
 return Directions.UP  
 elif right\_resource < 0:  
 return Directions.DOWN  
 else:  
 return Directions.NEUTRAL

class Carabiner:  
 def \_\_init\_\_(self):  
 self.btn\_resource = 10  
  
 def click\_button(self):  
 self.btn\_resource -= 1  
  
 def get\_button\_resource(self):  
 return self.btn\_resource

class Position:  
 def \_\_init\_\_(self):  
 self.x = 0  
 self.y = 0  
  
 def get\_x\_position(self):  
 return self.x  
  
 def get\_y\_position(self):  
 return self.y  
  
 def get\_position(self, obj):  
 obj.x = self.x  
 obj.y = self.y

from enum import Enum  
  
class Directions(Enum):  
 NEUTRAL = 0  
 UP = 1  
 DOWN = 2  
  
class ClimbingEquipment:  
 def \_\_init\_\_(self):  
 self.direction = Directions.NEUTRAL  
  
 def set\_up\_direction(self):  
 self.direction = Directions.UP  
  
 def set\_down\_direction(self):  
 self.direction = Directions.DOWN  
  
 def set\_neutral\_direction(self):  
 self.direction = Directions.NEUTRAL  
  
 def get\_direction(self):  
 return self.direction

