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
Довганюк О.С.
**Прийняв:**
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

**Мета роботи:** 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.

**Варіант 6:**

**6. Літак**

**6. Бомбардувальник**

**Вихідний код програми**

**airplane.py**

```
from datetime import datetime


# Class representing an airplane
class Airplane:
    def __init__(self, make, model):
        # The make of the airplane
        self.make = make
        # The model of the airplane
        self.model = model
        # The fuel level of the airplane, initially set to 100
        self.fuel_level = 100
```

```
    # Method to make the airplane fly
    def fly(self):
        # If the fuel level is greater than 0, the airplane flies and fuel level decreases by 10
        if self.fuel_level > 0:
            print(f"[{datetime.now()}][{self.make}][{self.model}] is flying")
            self.fuel_level -= 10
        # If the fuel level is 0 or less, the airplane cannot fly and a message is printed
        else:
            print("Not enough fuel. Please refuel.")

    # Method to make the airplane land
    def land(self):
        print(f"[{datetime.now()}][{self.make}][{self.model}] is landing")

    # Method to refuel the airplane
    def refuel(self):
        print(f"[{datetime.now()}][{self.make}][{self.model}] is being refueled")
        # The fuel level is set to 100
        self.fuel_level = 100

    # Method to check the fuel level of the airplane
    def check_fuel_level(self):
        print(f"[{datetime.now()}][{self.make}][{self.model}] fuel level: {self.fuel_level}%")
```

## bomber.py

```
from airplane import Airplane
from datetime import datetime

# Class representing a bomber, which is a type of airplane
class Bomber(Airplane):
    def __init__(self, make, model, weapon):
        # Inherits from the Airplane class
        super().__init__(make, model)
        # The weapon used by the bomber
        self.weapon = weapon
        # The number of bombs the bomber has, initially set to 10
```

```
        self.bombs = 10

    # Method to make the bomber drop bombs
    def drop_bombs(self):
        # If there are bombs available, the bomber drops a bomb and the count decreases by 1
        if self.bombs > 0:
            print(f"[{datetime.now()}][{self.make}][{self.model}] is drop bombs")
            self.bombs -= 1
        # If there are no bombs available, the bomber cannot drop bombs and a message is printed
        else:
            print("Out of bombs. Please reload.")

    # Method to make the bomber defend
    def defend(self):
        print(f"[{datetime.now()}][{self.make}][{self.model}] is defending")

    # Method to reload the bomber's weapon
    def reload_weapon(self):
        print(f"[{datetime.now()}][{self.make}][{self.model}] weapon is being reloaded")
        # The number of bombs is set to 10
        self.bombs = 10

    # Method to check the number of bombs the bomber has
    def check_ammo_count(self):
        print(f"[{datetime.now()}][{self.make}][{self.model}] bombs count: {self.bombs}")
```

## LAB_9.py

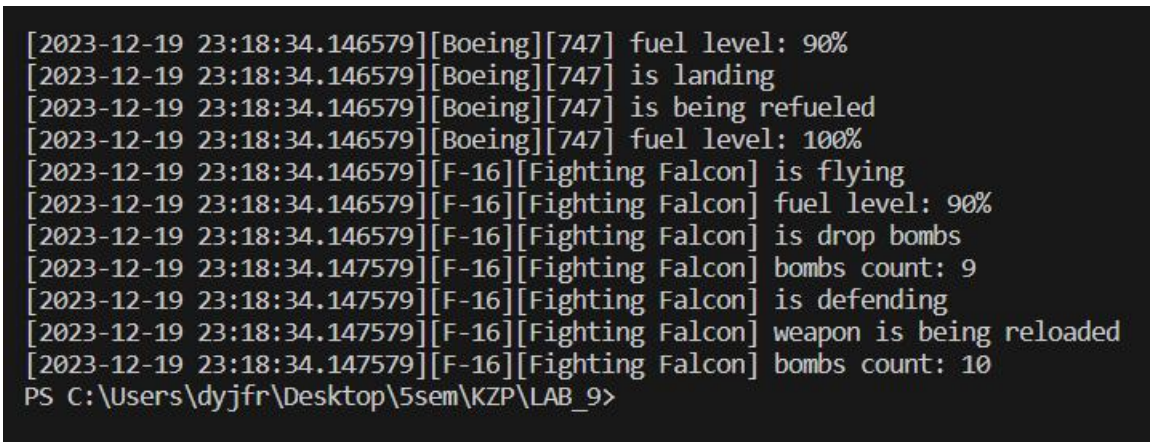
```
from airplane import Airplane
from bomber import Bomber

# Створення екземпляра базового класу "Літак"
plane1 = Airplane("Boeing", "747")
plane1.fly()
plane1.check_fuel_level()
plane1.land()
plane1.refuel()
```

```
plane1.check_fuel_level()


# Створення екземпляра похідного класу "бомбардувальник"
bomber1 = Bomber("F-16", "Fighting Falcon", "Missiles")
bomber1.fly()
bomber1.check_fuel_level()
bomber1.drop_bombs()
bomber1.check_ammo_count()
bomber1.defend()
bomber1.reload_weapon()
bomber1.check_ammo_count()
```

## Результат виконання програми




## Висновок

Під час виконання даної лабораторної роботи я оволодіd навиками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python.