Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра ЕОМ

Лабораторна робота №9

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «ОСНОВИ ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ У PYTHON »

Варіант № 25

Виконав:
ст. гр. КІ-305
Федусь Н.В.
Прийняв:
Іванов Ю. С.

ЛЬВІВ – 2023

МЕТА

Оволодіти навиками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python

ЗАВДАННЯ(ВАРІАНТ 25)

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 завдання: Базовий клас: Кондиціонер, похідний: Пристрій кліматконтролю

Код програми:

main.py

```
from climate_control import Climate_Controle, Conditioner
if __name__ == '__main__':
conditioner=Conditioner(
"Aerol",
"55TSg7",
16,
True
)
conditioner.get_info()
print("-----------------------------------------------")
conditioner.set_power_status(False)
conditioner.set_model("4444")
conditioner.get_info()
print("-----------------------------------------------")
```

```
climate_control=Climate_Controle(
"Gusing",
"44Gty4",
20,
True,
True,
1000,
True
)
climate_control.get_info()
print("-------------------------------------------------")

climate_control.set_temperature(30)
climate_control.set_wifi_enabled(False)
climate_control.set_brand("Sumsung")
climate_control.get_info()
print("-------------------------------------------------")
```

conditioner.py

```
class Conditioner:
def __init__(self, brand, model, temperature, power_status):
self._brand = brand
self._model = model
self._temperature = temperature
self._power_status = power_status

# Getter methods
def get_brand(self):
return self._brand

def get_model(self):
return self._model

def get_temperature(self):
return self._temperature

def get_power_status(self):
return self._power_status

# Setter methods
def set_brand(self, brand):
self._brand = brand

def set_model(self, model):
```

```
self._model = model

def set_temperature(self, temperature):
self._temperature = temperature

def set_power_status(self, power_status):
self._power_status = power_status
def get_info(self):
print(f"Brand {self._brand}")
print(f"Model {self._model}")
print(f"Temperature {self._temperature}")
print(f"Power status {self._power_status}")
```

climate_control.py

```
from conditioner import Conditioner
class Climate_Controle(Conditioner):
def __init__(self, brand, model, temperature, power_status, wifi_enabled, air_quality_sensor,
energy_saving_mode):
# Call the constructor of the parent class
super().__init__(brand, model, temperature, power_status)
self._wifi_enabled = wifi_enabled
self._air_quality_sensor = air_quality_sensor
self._energy_saving_mode = energy_saving_mode

# Getter methods for additional parameters
def get_wifi_enabled(self):
return self._wifi_enabled

def get_air_quality_sensor(self):
return self._air_quality_sensor

def get_energy_saving_mode(self):
return self._energy_saving_mode

# Setter methods for additional parameters
def set_wifi_enabled(self, wifi_enabled):
self._wifi_enabled = wifi_enabled

def set_air_quality_sensor(self, air_quality_sensor):
self._air_quality_sensor = air_quality_sensor

def set_energy_saving_mode(self, energy_saving_mode):
self._energy_saving_mode = energy_saving_mode

# Override the get_info method to include additional information
```

```
def get_info(self):
# Call the get_info method of the parent class
super().get_info()
# Include additional information
print("Wi-Fi Enabled:", self._wifi_enabled)
print("Air Quality Sensor:", self._air_quality_sensor)
print("Energy Saving Mode:", self._energy_saving_mode)
```

Результат виконання програми:

```
Brand Aerol
Model 55TSg7
Temperature 16
Power status True
--------------------------------------------------
Brand Aerol
Model 4444
Temperature 16
Power status False
--------------------------------------------------
Brand Gusing
Model 44Gty4
Temperature 20
Power status True
Wi-Fi Enabled: True
Air Quality Sensor: 1000
Energy Saving Mode: True
--------------------------------------------------
Brand Sumsung
Model 44Gty4
Temperature 30
Power status True
Wi-Fi Enabled: False
Air Quality Sensor: 1000
Energy Saving Mode: True
--------------------------------------------------
```

Висновок:

Виконавши лабораторну роботу, я ознайомився з основними принципами мови програмування python та оволодів навиками застосування них.