Міністерство освіти і науки України

Національний університет "Львівська політехніка"

Звіт з лабораторної роботи №9

з курсу “Кросплатформні засоби програмування”

Основи об'єктно орієнтованого програмування у Python

Виконав: студент гр. KI-306

Шаповал Віталій

Прийняв: к.т.н. Олексів М.В.

Львів 2024 р.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Модулем у Python називається файл з розширенням *.py. Ці файли можуть містити звичайні скрипти, змінні, функції, класи і їх комбінації. Python дозволяє структурувати код програм у різні модулі та доступатися до класів, функцій і змінних, які у них знаходяться з інших модулів. Для цього використовуються два оператори – import та from-import.

Оператор import дозволяє імпортувати модуль повністю, та доступатися до нього через назву модуля. Вона може бути вказана у будь-якому місці програми перед звертанням до елементів, які у ній містяться, але зазвичай її вказують на початку модуля. Для звертання до елементів модуля треба вказати назву модуля і після крапки вказати до якого елементу ви хочете звернутися.

Варіант № 28 Лампа --> Енергозбергіаюча лампа

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.

Код програми:

Файл ./lamp.py:

```
class Lamp:
    def __init__(self, wattage, voltage, socket, batteryCapacity):
        """
        Initializes the Lamp object with wattage, voltage, socket type and
battery capacity

        Parameters:
        wattage (float): The wattage of the lamp
        voltage (float): The voltage of the lamp
        socket (str): The type of socket the lamp uses
        batteryCapacity (float): The capacity of the lamp's battery
```

```
    """
    self._wattage = wattage
    self._voltage = voltage
    self._socket = socket
    self._batteryCapacity = batteryCapacity

def getWattage(self):
    return self._wattage

def getVoltage(self):
    return self._voltage

def getSocket(self):
    return self._socket

def getBatteryCapacity(self):
    return self._batteryCapacity

def setWattage(self, wattage):
    self._wattage = wattage

def setVoltage(self, voltage):
    self._voltage = voltage

def setSocket(self, socket):
    self._socket = socket

def setBatteryCapacity(self, batteryCapacity):
    self._batteryCapacity = batteryCapacity

def unchargeBattery(self, amount):
    """
    Decreases the battery capacity of the lamp by the given amount

    Parameters:
    amount (float): The amount to decrease the battery capacity by

    Returns:
    bool: True if the battery capacity was decreased, False otherwise
    """
    if self._batteryCapacity > amount:
        self._batteryCapacity -= amount
        return True
    else:
        return False

def chargeBattery(self, amount):
    """
    Increases the battery capacity of the lamp by the given amount

    Parameters:
    amount (float): The amount to increase the battery capacity by
```

```
        Returns:
        bool: True if the battery capacity was increased, False otherwise
        """
        if self._batteryCapacity < amount:
            return False
        else:
            self._batteryCapacity -= amount
            return True
```

Файл ./energy_saving_lamp.py:

```
import lamp

class EnergySavingLamp(lamp.Lamp):
    def __init__(self, wattage, voltage, socket, batteryCapacity, efficiency,
lifespan, lampType):
        """
        Initializes the EnergySavingLamp object with additional properties

        Parameters:
        wattage (float): The wattage of the lamp
        voltage (float): The voltage of the lamp
        socket (str): The type of socket the lamp uses
        batteryCapacity (float): The capacity of the lamp's battery
        efficiency (float): Energy efficiency rating of the lamp
        lifespan (int): Lifespan of the lamp in hours
        lampType (str): Type of the lamp (CFL or LED)
        """
        super().__init__(wattage, voltage, socket, batteryCapacity)
        self._efficiency = efficiency  # Energy efficiency rating
        self._lifespan = lifespan        # Lifespan in hours
        self._lampType = lampType        # Type of lamp (CFL or LED)
        self._isWarmingUp = False        # Indicates if the lamp is warming up

    def getEfficiency(self):
        return self._efficiency

    def getLifespan(self):
        return self._lifespan

    def getLampType(self):
        return self._lampType

    def isWarmingUp(self):
        return self._isWarmingUp

    def turnOn(self):
        print("Energy-saving lamp is turned on.")
        self._isWarmingUp = True
        # Logic for warming up can be implemented here

    def turnOff(self):
        print("Energy-saving lamp is turned off.")
        self._isWarmingUp = False
```

```
    def updateBrightness(self):
        if self._isWarmingUp:
            # Logic to gradually increase brightness
            print("Lamp is warming up...")
            # Update brightness logic can be added here
```

Файл ./main.py:

```
import energy_saving_lamp as esl

def main():
    """
    Головна функція, яка демонструє роботу класу EnergySavingLamp.
    """

    # Створення об'єкта енергозберігаючої лампи
    energy_saving_lamp = esl.EnergySavingLamp(
        wattage=15.0,
        voltage=220.0,
        socket='E27',
        batteryCapacity=100.0,
        efficiency=0.85,  # 85% ефективність
        lifespan=10000,   # 10000 годин
        lampType='LED'    # Тип лампи
    )

    # Виведення початкових даних
    print("Енергозберігаюча лампа:")
    print(f"Потужність: {energy_saving_lamp.getWattage()} Вт")
    print(f"Напруга: {energy_saving_lamp.getVoltage()} В")
    print(f"Тип розетки: {energy_saving_lamp.getSocket()}")
    print(f"Ємність акумулятора: {energy_saving_lamp.getBatteryCapacity()} мАг")
    print(f"Ефективність: {energy_saving_lamp.getEfficiency() * 100}%")
    print(f"Термін служби: {energy_saving_lamp.getLifespan()} годин")
    print(f"Тип лампи: {energy_saving_lamp.getLampType()}")

    # Увімкнення лампи
    energy_saving_lamp.turnOn()

    # Симуляція розряду акумулятора
    discharge_amount = 20.0
    if energy_saving_lamp.unchargeBattery(discharge_amount):
        print(f"Акумулятор розряджений на {discharge_amount} мАг. Нова ємність:
{energy_saving_lamp.getBatteryCapacity()} мАг")
    else:
        print("Не вдалося розрядити акумулятор.")

    # Симуляція зарядки акумулятора
    charge_amount = 10.0
    if energy_saving_lamp.chargeBattery(charge_amount):
        print(f"Акумулятор заряджений на {charge_amount} мАг. Нова ємність:
{energy_saving_lamp.getBatteryCapacity()} мАг")
    else:
```

```
        print("Не вдалося зарядити акумулятор.")

    # Перевірка стану нагрівання
    if energy_saving_lamp.isWarmingUp():
        print("Лампа нагрівається...")
        energy_saving_lamp.updateBrightness()

    # Вимкнення лампи
    energy_saving_lamp.turnOff()

if __name__ == "__main__":
    main()
```

Виконання програми:

Енергозберігаюча лампа:

Потужність: 15.0 Вт

Напруга: 220.0 В

Тип розетки: E27

Ємність акумулятора: 100.0 мАг

Ефективність: 85.0%

Термін служби: 10000 годин

Тип лампи: LED

Energy-saving lamp is turned on.

Акумулятор розряджений на 20.0 мАг. Нова ємність: 80.0 мАг

Акумулятор заряджений на 10.0 мАг. Нова ємність: 70.0 мАг

Лампа нагрівається...

Lamp is warming up...

Energy-saving lamp is turned off.

Висновок

У цій лабораторній роботі я створив програму на мові Python, яка реалізує базовий та похідний класи предметної області відповідно до варіанту. Програма була структурована у вигляді пакету, де кожен клас знаходився у своєму окремому модулі, а точка входу в програму (main) була реалізована в окремому модулі. Це забезпечило зручність підтримки та масштабованість програми. У процесі роботи я успішно реалізував базовий і похідний класи, що дозволяють опрацьовувати відповідні дані.

Код був завантажений на GitHub згідно з методичними вказівками. Це завдання допомогло мені покращити розуміння об'єктно-орієнтованого програмування в Python та навички організації коду у вигляді модулів і пакетів.