Міністерство освіти і науки України

Національний університет «Львівська політехніка»



**ЗВІТ**

з лабораторної роботи №9

з дисципліни: “Кросплатформні засоби програмування”

на тему: “Основи об’єктно-орієнтованого програмування у Python”

Виконав:

ст. гр. КІ-305

Шевчук М.О.

Прийняв:

Іванов Ю.С.

Львів – 2023

**Мета:** 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 № 28**





**Лістинг програми:**

**main.py**

from EconomLamp import EconomLamp

# основна програма

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

# об'єкт на основі класу газова плита

lamp = EconomLamp()

# використання методів класу за допомогою створеного об'єкта

lamp.getInfoLook()

lamp.repaintLight("Red")

lamp.resizeLight(7)

lamp.getInfoLook()

lamp.getLightPower()

lamp.powerOn()

lamp.powerOff()

lamp.lampWorking(30)

lamp.getPower()

lamp.setPower(15)

lamp.getPower()

**Type.py**

# клас корпус

class Type:

#конструктор

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

self.\_\_color = "White"

self.\_\_radius = 5

# метод встановлює колір світла

def setColor(self, newColor):

self.\_\_color = newColor

# метод повертає колір світла

def getColor(self):

return self.\_\_color

# метод встановлює радіус

def setRadius(self, newRadius):

self.\_\_radius = newRadius

def getRadius(self):

return self.\_\_radius

# метод повертає розмір лампочки

def getSize(self):

return 3.14 \* self.\_\_radius \* self.\_\_radius

**Light.py**

**#** клас комфорка

class Light:

# конструктор

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

self.\_\_power = 20

# метод повертає потужність світла

def getPower(self):

return self.\_\_power

# метод втановлює потужності світла

def setPower(self, newPower):

self.\_\_power = newPower

**Lamp.py**

from Type import Type

from Light import Light

from Button import Button

# клас аудіоплеєр

class Lamp:

# конструктор

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

self.\_\_typeOfLamp = Type()

self.\_\_lightOfLamp = Light()

self.\_\_powerOn = Button()

self.\_\_powerOff = Button()

# метод змінює колір світла

def repaintLight(self, uColor):

self.\_\_typeOfLamp.setColor(uColor)

print("New color of light is " + uColor + ".")

# метод змінює колір світла

def resizeLight(self, uRadius):

self.\_\_typeOfLamp.setRadius(uRadius)

print("New radius of lamp is " + str(uRadius) + " cm.")

# метод повертає повну інформацію про зовнішній вигляд лампочки

def getInfoLook(self):

print("Light color is " + str(self.\_\_typeOfLamp.getColor()))

print("Lamp size is " + str(self.\_\_typeOfLamp.getSize()))

# метод повертає радіус лампочки

def getRadiusLamp(self):

print("Radius of lamp: " + str(self.\_\_typeOfLamp.getRadius()) + " cm.")

# метод повертає потужність світла

def getLightPower(self):

print("Light power is " + str(self.\_\_lightOfLamp.getPower()) + " W.")

# метод виводить ресурс кнопки включання

def PowerOnResource(self):

print("The lamp will work for a while " + str(self.\_\_powerOn.getResource()) + " time units.")

# метод виводить ресурс кнопки виключання

def PowerOffResource(self):

print("The lamp will work for a while " + str(self.\_\_powerOff.getResource()) + " time units.")

# метод симулює натискання на кнопку включення

def powerOn(self):

self.\_\_powerOn.clickButton(True)

print("You tap the power on button.")

# метод симулює натискання на кнопку виключення

def powerOff(self):

self.\_\_powerOff.clickButton(False)

print("You tap the power off button.")

# метод надає інформацію про кількість увімкнень та вимкнень лампочки

def OnOffCount(self):

print("You tap the on/off button " + str(self.\_\_powerOn.getCountClicks()))

**Button.py**

**# клас кнопка**

class Button:

# конструктор

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

self.\_\_remainingWork = 999999999

self.\_\_count = 0

# метод симулює натискання на кнопку

def clickButton(self, type):

if type:

self.\_\_remainingWork -= 1

self.\_\_count += 1

else:

self.\_\_remainingWork -= 1

self.\_\_count -= 1

# метод повертає ресурс кнопки

def getResource(self):

return self.\_\_remainingWork

# метод повертає кількість натисків на кнопку за раз

def getCountClicks(self):

**return self**.\_\_count

**EconomLamp.py**

**from** Lamp import Lamp

# клас економ лампочка

class EconomLamp(Lamp):

# конструктор

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

super().\_\_init\_\_()

self.\_\_theLampIsIntact = True

self.\_\_powerLight = 30

# метод додає потужність світлу

def setPower(self, additionalPow):

self.\_\_powerLight += additionalPow

# метод симулює роботу лампочки

def lampWorking(self, time):

if self.\_\_theLampIsIntact:

print("The lamp is working " + str(time) +" min.")

else:

print("The lamp is NOT working.")

# метод повертає потужність світла

def getPower(self):

print("Power light: " + str(self.\_\_powerLight))

**Результат виконання :**

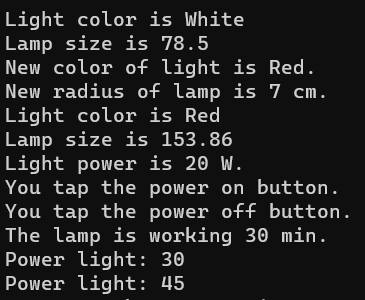
****

Рис.1.Результат виводу на консоль

**Відповіді на контрольні запитання:**

1. **Що таке модулі?**

Модулем у Python називається файл з розширенням \*.py. Ці файли можуть містити звичайні скрипти, змінні, функції, класи і їх комбінації.

1. **Як імпортувати модуль?**

Для цього використовуються два оператори – import та from-import.

1. **Як оголосити клас?**

Клас оголошується за допомогою ключового слова class після якого йде назва класу.

1. **Що може міститися у класі?**

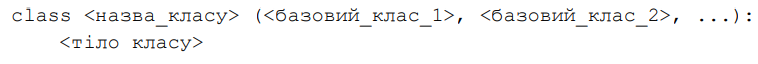
* дані, які належать класу (статичні дані-члени класу);
* дані, які належать об’єкту класу;
* методи, які належать класу (статична методи);
* методи, які належать об’єкту класу.

1. **Як називається конструктор класу?**

Роль конструктора відіграє метод \_\_init\_\_(self, ).

1. **Як здійснити спадкування?**

Для реалізації спадкування класи, які слід успадкувати вказуються у круглих дужках через кому після назви класу, який оголошується:



1. **Які види спадкування існують?**

* Одинарне спадкування
* Багатократне спадкування
* Інтерфейси
* Абстрактні класи
* Множинне спадкування
* Заміщення

1. **Які небезпеки є при множинному спадкуванні, як їх уникнути?**

* Амбігвітність – уникнути можна забезпечивши чітку структуру спадкування
* Складність ієрархій класів – уникнути можна дотримуючись принципу “розділення відповідальностей”
* Залежність від порядку – уникнути можна спроектувавши класи так, вони не залежали від порядку успадкування
* Дублювання коду – уникнути можна використовуючи абстрактні класи або інтерфейси для виділення спільного функціоналу та розділення його між класами
* Складність відлагодження та обслуговування – уникнути можна називаючи класи та методи відповідно до їх функції

1. **Що таке класи-домішки?**

Домішки або Mixin – це шаблон проектування, в якому деякий метод базового класу використовує метод, який не визначається у цьому класі. Цей метод призначений для реалізації іншим базовим класом. Клас-домішка або mixin class – це клас, який використовується у цьому шаблоні, надаючи функціональні можливості (методи), але не призначений для самостійного використання у вигляді об’єктів класу. В ідеальному випадку класи-домішки не мають власної ієрархії спадкування і не мають полів, а мають лише методи

1. **Яка роль функції super() при спадкуванні?**

Функція super() грає важливу роль при спадкуванні в об'єктно-орієнтованому програмуванні. Вона дозволяє виконувати методи батьківського класу в контексті підкласу.

**Висновок:** Я оволодів навиками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python.