Міністерство освіти і науки України

Національний університет «Львівська Політехніка»

Кафедра ЕОМ

ЗВІТ

з лабораторної роботи №9

з дисципліни: “Кросплатформенні засоби програмування”

на тему: **«Основи об’єктно-орієнтованого програмування Python»**

Варіант - **19**

Виконав:

ст. гр. КІ-305

Лихограй А. В.

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. Дати відповіді на контрольні запитання:
   - що таке модулі?
   - як імпортувати модуль?
   - як оголосити клас?
   - що може міститися у класі?
   - як називається конструктор класу?
   - як здійснити спадкування?
   - які види спадкування існують?
   - які небезпеки є при множинному спадкуванні, як їх уникнути?
   - що таке класи-домішки?
   - яка роль функції super() при спадкуванні?

Вхідний варіант: «Принтер» - «Мультифункціональний пристрій».

Вихідний код програми:

PaperTray.py:

```
import os
import struct
import sys
class PaperTray:
    maxPaperLevel = 250
    def __init__(self):
        self.paperCount = 0
        self.maxPaperLevel = 250

    def __init__(self, count):
        if count <= self.maxPaperLevel:
            self.paperCount = count
        else:
            print("\nПаперу забагато!!!\n")
```

```
    def add_paper(self):
        temp = int(input("Кількість нового паперу: "))
        if temp <= self.maxPaperLevel:
            self.paperCount = temp
        else:
            print("\nПаперу забагато!!!\n")

    def use_paper(self):
        self.paperCount -= 1

    def get_paper_count(self):
        return self.paperCount

    def reset_paper_count(self):
        self.paperCount = 0

    def is_paper_empty(self):
        return self.paperCount == 0

    def is_paper_full(self):
        return self.paperCount == self.maxPaperLevel

    def set_paper_count(self, count):
        self.paperCount = count
```

InkCatridge.py:

```
class InkCartridge:
    def __init__(self):
        self.inkLevel = 0

    def __init__(self, inkLevel):
        self.inkLevel = inkLevel

    def add_ink(self):
        inkLevel = int(input("Кількість нового чорнила: "))

    def use_ink(self, length):
        if length > 2000:
            self.inkLevel -= 3
        elif 1000 > length:
            self.inkLevel -= 2
        else:
            self.inkLevel -= 1

    def get_ink_level(self):
        return self.inkLevel

    def reset_ink_level(self):
        self.inkLevel = 0

    def is_ink_empty(self):
        return self.inkLevel == 0

    def is_ink_full(self):
        return self.inkLevel == 100

    def set_ink_level(self, level):
        self.inkLevel = level
```

InkCatridge.py:

```
from PaperTray import PaperTray
from InkCatridge import InkCartridge
import os


class Printer:
    # The default constructor.
    def __init__(self):
        self.model = "unknow"
        self.resourse = 0
        self.work = False
        self.paperTray = PaperTray()
        self.inkCartridge = InkCatridge()

    # The constructor with parameters.
    def __init__(self, model, inkCartridge, paperTray, resourse):
        self.model = model
        self.resourse = resourse
        self.work = False
        self.paperTray = PaperTray(paperTray)
        self.inkCartridge = InkCartridge(inkCartridge)

    # Print document.
    def print_document(self):
        self.file_name = input("Введіть назву документу: ")
        count = int(input("Введіть кількість копій: "))
        if not self.is_work() or count == 0:
            print("Пристрій не ввімкнуто!!!")
            self.write_logs("Printer has problem!\n")
        else:
            print("Розпочато друк документу:", self.file_name)
            if self.paperTray.get_paper_count() >= count and
self.inkCartridge.get_ink_level() > 10:
                for i in range(1, count + 1):
                    print("\nДрук", i, "сторінки.")
                    self.work_printer()
                    self.use_ink_paper(self.file_name)
            else:
                print("Замало чорнила або паперу!")

    # Print text to the console.
    def work_printer(self):
        try:
            if os.path.exists(self.file_name):
                with open(self.file_name, 'r') as file:
                    result = str(file.read())
            else:
                raise FileNotFoundError(f"Файл {self.file_name} не
знайдено.")
        except FileNotFoundError as e:
            print(e)
        print(result)

    # Count ink for printing.
    def count_ink(self, text):

        if not text:
            return 0
        return len(text)

    # Use ink and paper.
    def use_ink_paper(self, text):
        self.write_logs("Use ink and paper.\n")
```

```
        self.paperTray.use_paper()
        self.inkCartridge.use_ink(self.count_ink(text))

    # Check if printer is work.
    def is_work(self):
        self.write_logs("Is printer work?" + str(self.work) + ".\n")
        return self.work

    # Power on.
    def power_on(self):
        self.write_logs("Printer is on.\n")
        if self.resourse == 0:
            print("Пристрій потрібно відкалібрувати!")
        else:
            print("Пристрій ввімкнено!")
            self.work = True
            self.resourse -= 1

    # Power off.
    def power_off(self):
        self.write_logs("Printer is off.\n")
        print("Принтер вимкнено!")
        self.work = False
        self.networkConnection.disconnect()

    # Calibrate printer.
    def calibrate_printer(self):
        self.write_logs("Calibrate printer\n")
        print("Принтер було відкалібровано!")
        self.inkCartridge.reset_ink_level()
        self.paperTray.reset_paper_count()
        self.resourse = 100

    # Clean printer heads.
    def clean_print_heads(self):
        self.write_logs("Clear printer\n")
        self.paperTray.reset_paper_count()
        self.inkCartridge.reset_ink_level()
        self.resourse = 0
        print("Принтер очищено!")

    # Print "About printer".
    def about_printer(self):
        self.write_logs("About printer.\n")
        print("\n\nПро принтер:")
        print("Назва:", self.get_printer_name())
        self.check_paper_level()
        self.check_ink_level()

    # Check ink level.
    def check_ink_level(self):
        self.write_logs("Check ink level.\n")
        print("Залишилось чорнила:", self.inkCartridge.get_ink_level())

    # Check paper level.
    def check_paper_level(self):
        self.write_logs("Check paper level.\n")
        print("Залишилось паперу:", self.paperTray.get_paper_count())

    # Add new paper.
    def add_paper(self):
        self.write_logs("Add paper.\n")
        self.paperTray.add_paper()

    # Add new ink level.
    def add_ink_level(self):
```

```
        self.write_logs("Add ink.\n")
        self.inkCartridge.add_ink()

    # Set printer resource.
    def set_printer_resource(self, res):
        self.write_logs("Set printer recourse.\n")
        self.resourse = res

    # Set printer name.
    def set_printer_name(self, name):
        self.write_logs("Set printer name.\n")
        self.model = name

    # Get printer name.
    def get_printer_name(self):
        self.write_logs("Get printer name.\n")
        return self.model

    # Replace papre.
    def replace_paper_tray(self, new_paper_tray):
        new_paper_tray.reset_paper_count()
        new_paper_tray.set_paper_count(250)

    # Replace ink
    def replace_ink_cartridge(self, new_ink_cartridge):
        new_ink_cartridge.reset_ink_level()
        new_ink_cartridge.set_ink_level(100)

    # Write logs.
    def write_logs(self, text_to_write):
        try:
            with open("Logs.txt", "a") as log_file:
                log_file.write(text_to_write)
        except IOError as e:
            print("Сталася помилка при записі до файлу:", e)

    # Get resource.
    def get_resource(self):
        return self.resourse
```

MultiFunctionDevice.py:

```
from Printer import Printer
import os
import sys

class MultifunctionDevice(Printer):
    # The default constructor.
    def __init__(self):
        super().__init__()
    # The constructor with parameters.
    def __init__(self, model, paperTray, inkCatridge, resource):
        super().__init__(model, paperTray, inkCatridge, resource)
    # Scan document.
    def scanDocument(self):
        print("Розпочато сканування " + self.file_name + "\n\n");
        try:
            if os.path.exists(self.file_name):
                with open(self.file_name, 'r') as file:
                    result = str(file.read())
            else:
                raise FileNotFoundError(f"Файл {self.file_name} не
знайдено.")
        except FileNotFoundError as e:
            print(e)
        return result
```

```
    # Print document.
    def print_document(self, text):
        count = int(input("Введіть кількість копій: "))
        if not(self.is_work()) or count == 0:
            print("Пристрій не ввімкнуто!!!\n");
        else:
            print("Розпочато друк відсканованого документу: \n");
            if self.paperTray.get_paper_count() >= count and
self.inkCartridge.get_ink_level() > 10:
                print("Розпочато друк відсканованого документу: \n")
                for i in range(1, count + 1):
                    print("\nДрук", i, "сторінки.")
                    sys.stdout.write(text)
                    self.use_ink_paper(text)
            else:
                print("Замало чорнила або паперу!\n")
```

main.py:

```
from MultifunctionDevice import MultifunctionDevice

if __name__ == '__main__':
    myPrinter = MultifunctionDevice("HP", 110, 2, 3)
    myPrinter.power_on()
    myPrinter.about_printer()
    myPrinter.file_name = str(input("Введіть назву файлу, який потрібно
зісканувати: "))
    myPrinter.print_document(myPrinter.scanDocument())
    myPrinter.about_printer()
    myPrinter.add_paper()
    myPrinter.about_printer()
```

Результат виконання програми:

```
Пристрій ввімкнено!


Про принтер:
Назва: HP
Залишилось паперу: 2
Залишилось чорнила: 110
Введіть назву файлу, який потрібно зісканувати: MyFile.txt
Розпочато сканування MyFile.txt


Введіть кількість копій: 1
Розпочато друк відсканованого документу:

Розпочато друк відсканованого документу:


Друк 1 сторінки.
My text.
My text.
My text.
My text.
My text.
My text.


Про принтер:
Назва: HP
Залишилось паперу: 1
Залишилось чорнила: 108
Кількість нового паперу: 20


Про принтер:
Назва: HP
Залишилось паперу: 20
Залишилось чорнила: 108
```

Рис. 1. Результат роботи програми.

Відповіді на контрольні запитання:

1. Модулі:

- Модулі в Python - це файли, що містять Python код. Вони дозволяють групувати код для легшого управління та використання.

2. Імпортування модуля:

- Використовуйте ключове слово `import`:

```python
import module_name
```

3. Оголошення класу:

- Використовуйте ключове слово `class`:

```python
class ClassName:
    # тіло класу
```

4. Зміст класу:

- Клас може містити атрибути (змінні), методи (функції, що належать класу), конструктори, властивості і багато іншого.

5. Конструктор класу:

- Конструктор класу має назву `__init__` і викликається автоматично при створенні нового об'єкта класу.

6. Спадкування:

- Спадкування в Python реалізується через ключове слово `class` при визначенні нового класу.

7. Види спадкування:

- Одиночне спадкування (один клас успадковує від іншого).

- Множинне спадкування (клас успадковує від декількох класів).

8. Небезпеки при множинному спадкуванні та як їх уникнути:

- Проблема "алмаза спадкування" може виникнути, коли клас успадковує від двох класів, які мають спільний батьківський клас. Для уникнення цієї проблеми слід уникати зайвого множинного спадкування.

9. Класи-домішки:

- Клас-домішка (mixin) - це клас, який містить функціональність, яку можна використовувати в інших класах, але не представляє собою окремої ієрархії спадкування.

10. Роль функції `super()` при спадкуванні:

- Функція `super()` використовується для виклику методів батьківського класу. Вона може викликатися в конструкторі або інших методах підкласу, щоб викликати методи батьківського класу і отримати доступ до його функціональності.

Висновок: під час виконання лабораторної роботи я оволодіти навиками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python. Реалізував предметну область згідно варіанту.