МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТУ “ЛЬВІВСЬКА ПОЛІТЕХНІКА“

Кафедра систем штучного інтелекту

Лабораторна робота №4

з дисципліни

«Системний аналіз»

Виконав:

Студент групи КН-214

Добрій Назарій

Викладач:

Зімоха.І.О.

Львів – 2021р.

Кроки реалізації

1. Приведіть усі створювані продукти до загального інтерфейсу.

2. Створіть порожній фабричний метод у класі, який виробляє продукти. В якості типу, що повертається, вкажіть загальний інтерфейс продукту.

3. Пройдіться по коду класу й знайдіть усі ділянки, що створюють продукти. По черзі замініть ці ділянки викликами фабричного методу, переносячи в нього код створення різних продуктів. Можливо, доведеться додати до фабричного методу декілька параметрів, що контролюють, який з продуктів потрібно створити. Імовірніше за все, фабричний метод виглядатиме гнітюче на цьому етапі. В ньому житиме великий умовний оператор, який вибирає клас створюваного продукту. Але не хвилюйтеся, ми ось-ось все це виправимо.

4. Для кожного типу продуктів заведіть підклас і перевизначте в ньому фабричний метод. З суперкласу перемістіть туди код створення відповідного продукту.

5. Якщо створюваних продуктів занадто багато для існуючих підкласів творця, ви можете подумати про введення параметрів до фабричного методу, аби повертати різні продукти в межах одного підкласу

Виконання

Фабричний метод — це породжувальний патерн проектування, який визначає загальний інтерфейс для створення об’єктів у суперкласі, дозволяючи підкласам змінювати тип створюваних об’єктів

Функція та сервіс :

№ вар	Предмета область
5	Видання книг

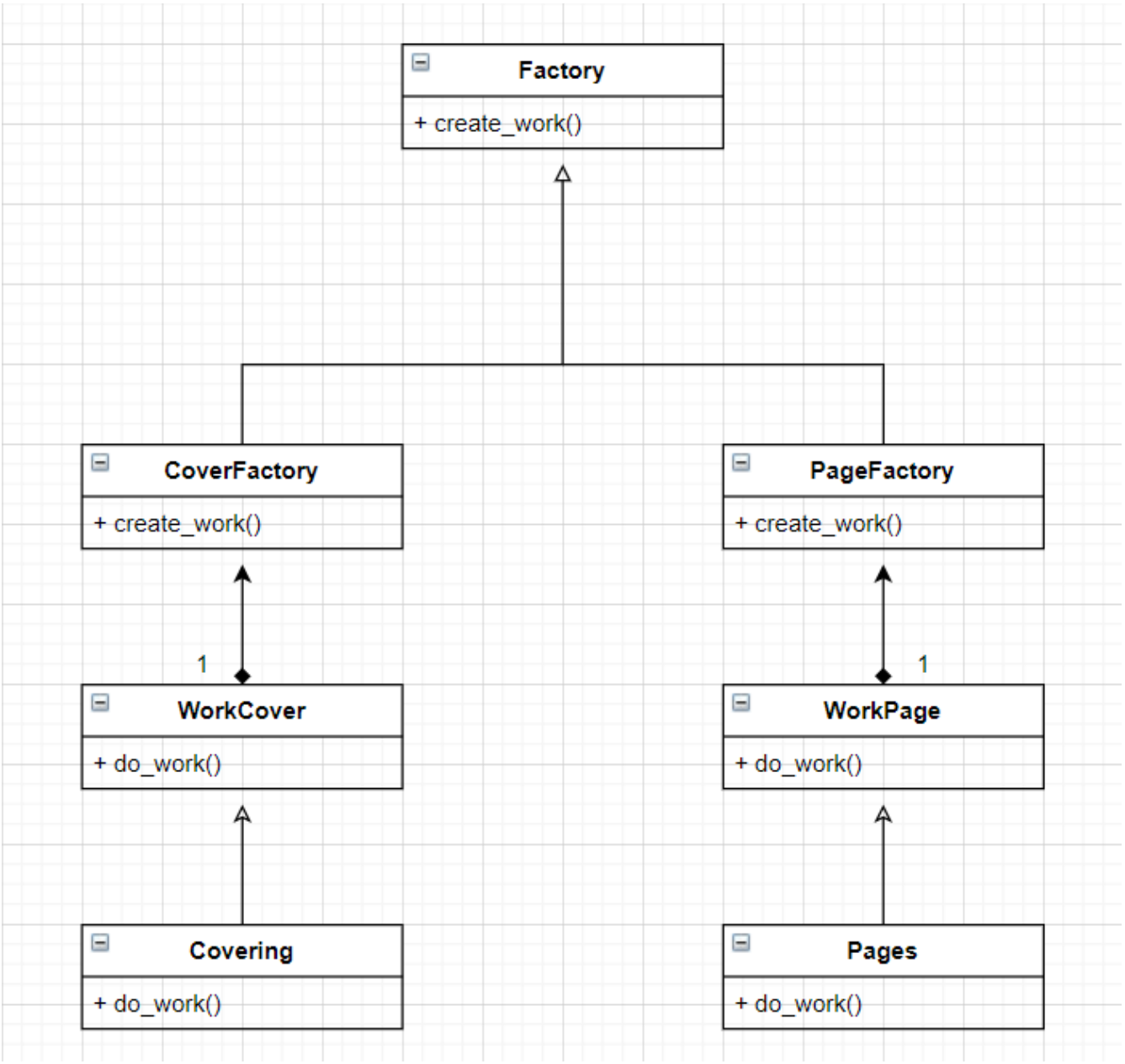

Рис. 1. UML-діаграма патерну “Фабричний метод”

Код програми на мові Python:

```
from abc import ABC, abstractmethod


class WorkCover(ABC):
    @abstractmethod
    def do_work(self):
        pass


class WorkPage(ABC):
    @abstractmethod
    def do_work(self):
        pass
```

```python
class Covering(WorkCover):
    def do_work(self):
        print("Виготовляємо палітурку для книжки")


class Pages(WorkPage):
    def do_work(self):
        print("Виготовляємо сторінки для книжки")


class Factory(ABC):
    @abstractmethod
    def create_work(self, type):
        pass


class CoverFactory(Factory):
    def create_work(self, type):
        return Covering()


class PageFactory(Factory):
    def create_work(self, type):
        return Pages()


cover_factory = CoverFactory()
page_factory = PageFactory()


work_cover = cover_factory.create_work('Створення палітурки')

work_cover.do_work()

work_page = page_factory.create_work('Створення сторінок')

work_page.do_work()
```

Рис. 2.1. Код реалізації програми.

```
Виготовляємо палітурку для книжки
Виготовляємо сторінки для книжки


Process finished with exit code 0
```

Рис. 2.2. Виконання програми.

Висновок: я навчився використовувати патерн «фабричний метод». Реалізував його програмно на основі предметної області «Видання книг».