

Міністерство освіти і науки України  
Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

Звіт  
З практичної роботи 1  
З дисципліни “Архітектура програмного забезпечення”  
з теми:  
«Design patterns»

Виконав ст. гр. ПЗПІ-21-8  
Хотян М.О.

Харків 2023

## 1. Мета роботи

Підготувати і оформити у вигляді презентації доповідь на тему паттерн Abstract factory.

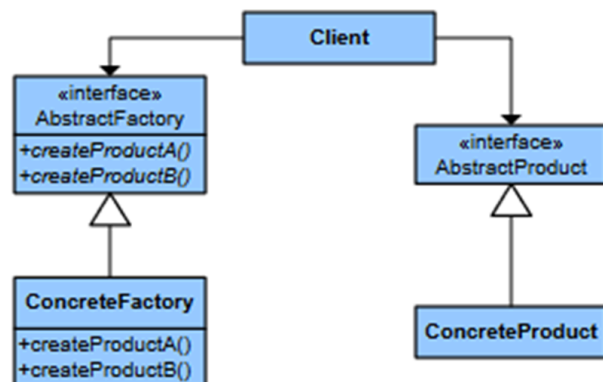
## 2. Висновки

Під час виконання роботи було підготовлено доповідь на тему паттерн Abstract factory.

# ПАТЕРН ПРОЕКТУВАННЯ «ABSTRACT FACTORY»

## ABSTRACT FACTORY

- Abstract factory належить до породжувальних (creational) типів патерну.






## КРОКИ СТВОРЕННЯ ABSTRACT FACTORY

- Патерн Абстрактна фабрика пропонує виділити загальні інтерфейси для окремих продуктів, що складають одне сімейство, і описати в них спільну для цих продуктів поведінку.
- Далі ви створюєте абстрактну фабрику — загальний інтерфейс, який містить методи створення всіх продуктів сімейства. Ці операції повинні повертати абстрактні типи продуктів, представлені інтерфейсами, які ми виділили раніше.
- Для кожної варіації сімейства продуктів ми повинні створити свою власну фабрику, реалізуючи абстрактний інтерфейс. Фабрики створюють продукти однієї варіації.
- Клієнтський код повинен працювати як із фабриками, так і з продуктами тільки через їхні загальні інтерфейси. Це дозволить подавати у ваші класи будь-які типи фабрик і виробляти будь-які типи продуктів, без необхідності вносити зміни в існуючий код.

3



## ВИКОРИСТАННЯ ABSTRACT FACTORY

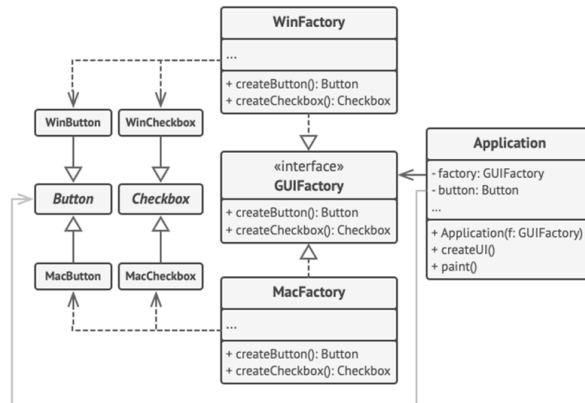
- система повинна бути незалежною від того, як її продукти створені, складені та представлені.
- система повинна бути налаштована з одним із кількох сімейств продуктів.
- сімейство пов'язаних об'єктів продукту розроблено для спільного використання, і вам потрібно забезпечити дотримання цього обмеження.
- ви хочете надати бібліотеку класів продуктів, і ви хочете розкрити лише їхні інтерфейси, а не їхні реалізації.

4

## ПСЕВДОКОД ДЛЯ ABSTRACT FACTORY

Для прикладу розглянемо задачу створення крос-платформові елементи інтерфейсу і стежить за тим, щоб вони відповідали обраній операційній системі.

Крос-платформова програма може відображати одні й ті самі елементи інтерфейсу по-різному, в залежності від обраної операційної системи. Важливо, щоб у такій програмі всі створювані елементи завжди відповідали поточній операційній системі.



5

## ПСЕВДОКОД

```

1 // Цей патерн передбачає, що ви маєте кілька сімейств продуктів,
2 // які знаходяться в окремих ієрархіях класів (Button/Checkbox).
3 // Продукти одного сімейства повинні мати спільний інтерфейс.
4 interface Button is
5     method paint()
6
7 // Сімейства продуктів мають однакові варіації (macOS/Windows).
8 class WinButton implements Button is
9     method paint() is
10         // Відобразити кнопку в стилі Windows.
11
12 class MacButton implements Button is
13     method paint() is
14         // Відобразити кнопку в стилі macOS.
15
16
17 interface Checkbox is
18     method paint()
19
20 class WinCheckbox implements Checkbox is
21     method paint() is
22         // Відобразити чекбокс в стилі Windows.
23
24 class MacCheckbox implements Checkbox is
25     method paint() is
26         // Відобразити чекбокс в стилі macOS.
27
28
29 // Абстрактна фабрика знає про всі абстрактні типи продуктів.
30 interface GUIFactory is
31     method createButton():Button
32     method createCheckbox():Checkbox
33

```

6

## ПСЕВДОКОД

```
// Кожна конкретна фабрика знає лише про продукти своєї варіації  
// і створює лише їх.  
class WinFactory implements GUIFactory is  
| method createButton():Button is  
|   return new WinButton()  
| method createCheckbox():Checkbox is  
|   return new WinCheckbox()  
  
// Незважаючи на те, що фабрики оперують конкретними класами,  
// їхні методи повертають абстрактні типи продуктів. Завдяки  
// цьому фабрики можна замінити одну на іншу, не змінюючи  
// клієнтського коду.  
class MacFactory implements GUIFactory is  
| method createButton():Button is  
|   return new MacButton()  
| method createCheckbox():Checkbox is  
|   return new MacCheckbox()  
  
// Для коду, який використовує фабрику, не важливо, з якою  
// конкретно фабрикою він працює. Всі отримувачі продуктів  
// працюють з ними через загальні інтерфейси.  
class Application is  
| private field factory: GUIFactory  
| private field button: Button  
| constructor Application(factory: GUIFactory) is  
|   this.factory = factory  
| method createUI()  
|   this.button = factory.createButton()  
| method paint()  
|   button.paint()
```

7

## ПСЕВДОКОД

```
// Програма вибирає тип конкретної фабрики й створює її  
// динамічно, виходячи з конфігурації або оточення.  
class ApplicationConfigurator is  
| method main() is  
|   config = readApplicationConfigFile()  
  
|   if (config.OS == "Windows") then  
|     factory = new WinFactory()  
|   else if (config.OS == "Mac") then  
|     factory = new MacFactory()  
|   else  
|     throw new Exception("Error! Unknown operating system.")  
  
|   Application app = new Application(factory)
```

8