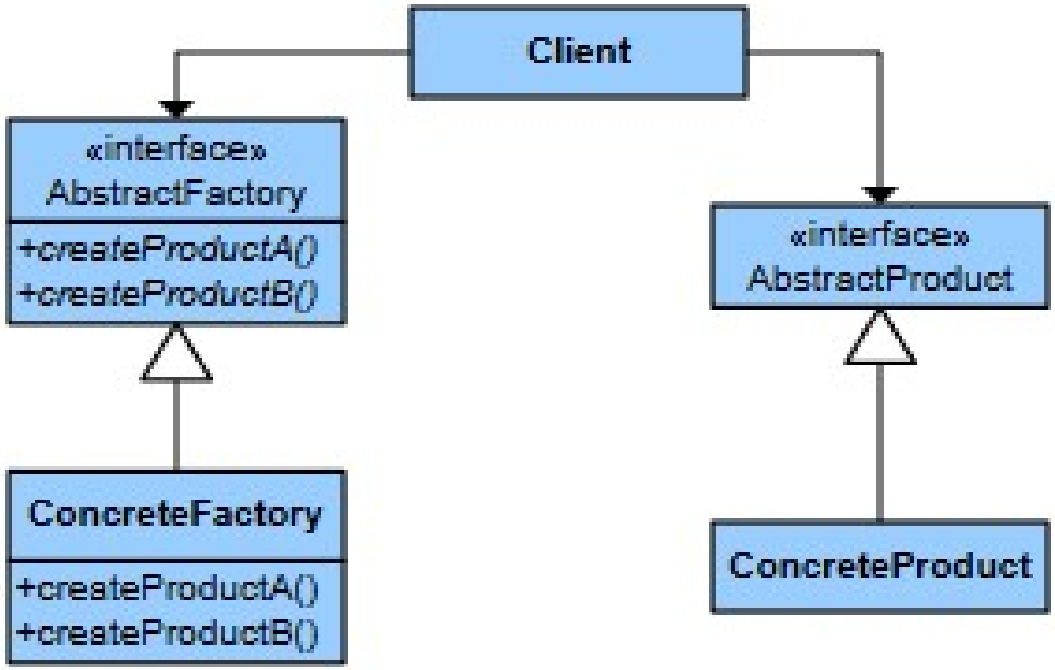
ПАТЕРН ПРОЕКТУВАННЯ «ABSTRACT FACTORY»

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, реалізувавши 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.

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.

ПОРІВНЯННЯ

Паттерн	Дозволяє	Використовується для
Абстрактна фабрика	Дозволяє створювати сімейства пов'язаних об'єктів без прив'язки до конкретних класів.	Використовується, коли система повинна працювати зі сімейством об'єктів, які пов'язані або взаємозалежні.
Фабричний метод	Дозволяє відкладати створення об'єктів до підкласів.	Використовується, коли клас має визначити, який конкретний підклас використовувати для створення об'єктів, або коли класи підкласів можуть бути різними.
Будівельник	Дозволяє складати складні об'єкти крок за кроком.	Використовується, коли процес створення об'єкта складається з кількох кроків, або коли потрібно створювати різні варіації об'єктів з одного набору компонентів.

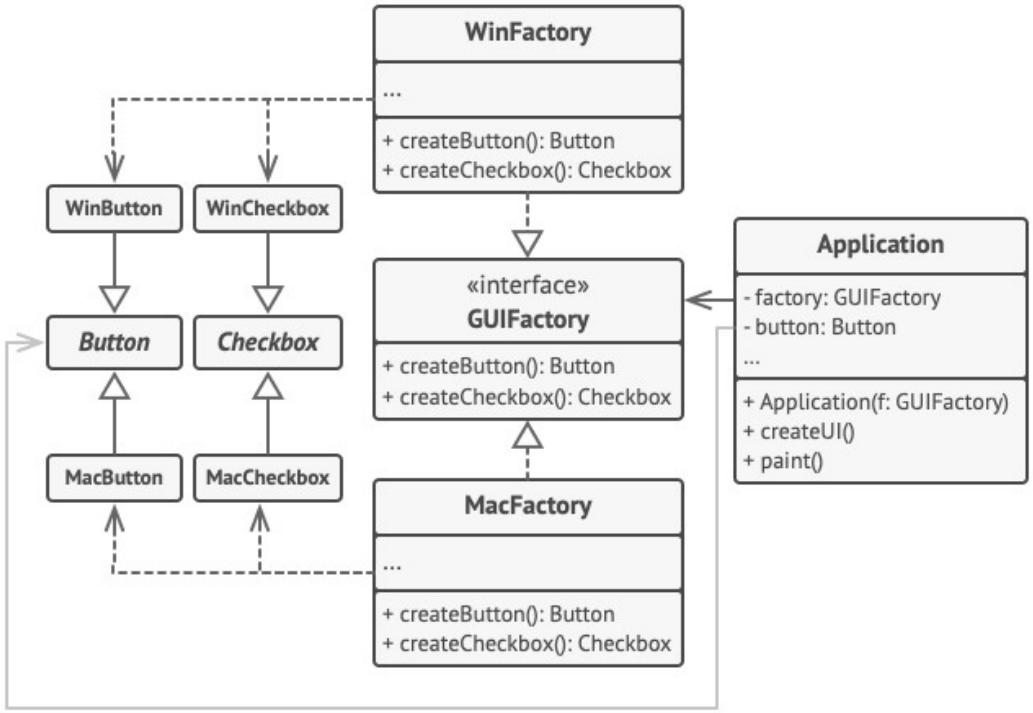
ПОРІВНЯННЯ

Паттерн	Дозволяє	Використовується для
Прототип	Дозволяє клонувати об'єкти, замість того, щоб створювати нові.	Використовується, коли процес створення об'єктів є важким або займає багато ресурсів, або коли потрібно створити новий об'єкт на основі існуючого з невеликими змінами.
Одиночний	Гарантує, що в класі існує тільки один екземпляр, і надає глобальну точку доступу до цього екземпляра.	Використовується, коли в системі потрібно мати єдиний об'єкт певного класу та цей об'єкт повинен бути доступним з будь-якого місця.

ABSTRACT FACTORY

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

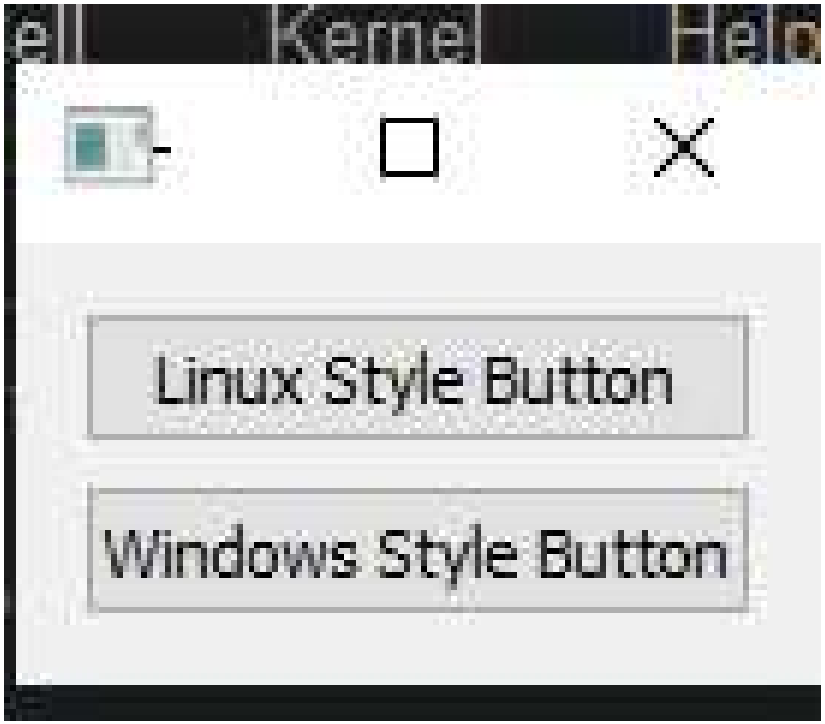
Крос-платформова програма може відображати одні й ті самі елементи інтерфейсу по-різному, в залежності від обраної операційної системи. Важливо, щоб у такій програмі всі створювані елементи завжди відповідали поточній операційній системі.



ПРИКЛАД ВИКОРИСТАННЯ

```python
# Абстрактні класи для фабрики та продуктів
class ButtonFactory:
    def create_button(self, window):
        pass

class Button:
    def paint(self):
        pass

# Конкретні класи для фабрики та продуктів для Linux
class LinuxButton(Button):
    def paint(self):
        return "Render a button in a Linux style"

    def draw(self, window):
        button = QPushButton("Linux Style Button", window)
        return button

class LinuxButtonFactory(ButtonFactory):
    def create_button(self, window):
        return LinuxButton().draw(window)

# Конкретні класи для фабрики та продуктів для Windows
class WindowsButton(Button):
    def paint(self):
        return "Render a button in a Windows style"

    def draw(self, window):
        button = QPushButton("Windows Style Button", window)
        return button

class WindowsButtonFactory(ButtonFactory):
    def create_button(self, window):
        return WindowsButton().draw(window)
```

ПРИКЛАД ВИКОРИСТАННЯ




```python
# Клас, який використовує фабрику
class Application:
    def __init__(self, factory, window):
        self.factory = factory
        self.window = window

    def create_button(self):
        return self.factory.create_button(self.window)


if __name__ == "__main__":
    app = QApplication([])

    window = QWidget()
    layout = QVBoxLayout()
    window.setLayout(layout)

    linux_factory = LinuxButtonFactory()
    linux_app = Application(linux_factory, window)
    linux_button = linux_app.create_button()
    layout.addWidget(linux_button)

    windows_factory = WindowsButtonFactory()
    windows_app = Application(windows_factory, window)
    windows_button = windows_app.create_button()
    layout.addWidget(windows_button)

    window.show()
    app.exec_()
```

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- Еріх Ґамма та інш., Патерни проєктування. Прийоми ООП
- https://refactoring.guru/uk/design-patterns/abstract-factory