****

Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №7  
**Технологія розроблення програмного забезпечення**

Тема: ШАБЛОН «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»

Варіант 25

Виконав Перевірив:

студент групи ІА-23 Мягкий Михайло

Калина С. О. Юрійович

Київ 2024

# **Тема.**

# ШАБЛОН «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»

# **Мета.**

Метою лабораторної роботи є поглиблене вивчення та практичне опанування шаблонів проєктування, зокрема шаблонів «Mediator», «Facade», «Bridge» та «Template Method». Реалізація шаблону Bridge згідно обраної теми. Формування навичок застосування об'єктно-орієнтованого підходу при розробці складних програмних систем вдосконалення вмінь проєктування архітектури програмного забезпечення з використанням сучасних патернів.

# **Завдання.**

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми

**Хід роботи**

**25. Installer generator (iterator, builder, factory method, bridge, interpreter, client-server)**

Генератор інсталяційних пакетів повинен мати якийсь спосіб налаштування файлів, що входять в установку, установки вікон з інтерактивними можливостями (галочка - створити ярлик на робочому столі; ввести в текстове поле деякі дані, наприклад, ліцензійний ключ і т.д.). Генератор повинен вивести один файл .exe або .msi.

# **Короткі теоретичні відомості.**

Основні принципи проєктування програмного забезпечення

1. Принцип DRY (Don't Repeat Yourself)

Цей принцип наголошує на необхідності уникнення повторень у вихідному коді. Основні причини дотримання принципу:

* Зменшення обсягу коду
* Полегшення читабельності
* Спрощення підтримки та модифікації
* Мінімізація ризику поширення помилок через копіювання коду

2. Принцип KISS (Keep it Simple, Stupid!)

Філософія простоти в проєктуванні систем:

* Перевага простих компонентів над складними
* Кожен компонент повинен виконувати чітко визначену функцію
* Простота сприяє надійності та зрозумілості коду

3. Принцип YOLO (You Only Load It Once!)

Оптимізація ініціалізації та конфігурації:

* Одноразове завантаження конфігураційних змінних
* Попередження проблем із продуктивністю
* Мінімізація повторних операцій введення-виведення

4. Принцип Парето (80/20)

Статистичний підхід до оптимізації:

* 80% навантаження створюється 20% компонентів
* 80% коду пишеться за 20% часу
* 80% помилок можна усунути, виправивши 20% вад

5. Принцип YAGNI (You Ain't Gonna Need It)

Уникнення зайвої складності:

* Реалізація лише необхідного функціоналу
* Відмова від передчасного узагальнення
* Фокус на актуальних вимогах

Шаблони проєктування

1. Шаблон Mediator (Посередник)

**Призначення:** Централізація взаємодії між компонентами через проміжний об'єкт.

**Основні характеристики:**

* Зменшення прямих залежностей між компонентами
* Спрощення комунікації
* Підвищення гнучкості системи

**Приклад:** Диспетчер керування повітряним рухом, який координує взаємодію літаків.

2. Шаблон Facade (Фасад)

**Призначення:** Створення уніфікованого інтерфейсу для складної підсистеми.

**Основні характеристики:**

* Приховування внутрішніх деталей реалізації

**Приклад:** Співробітник служби підтримки магазину як єдина точка взаємодії для клієнта.

3. Шаблон Bridge (Міст)

**Призначення:** Роз'єднання абстракції та реалізації.

**Основні характеристики:**

* Зменшення кількості класів при комбінуванні властивостей
* Незалежний розвиток ієрархій абстракції та реалізації
* Гнучкість додавання нових абстракцій та реалізацій

**Приклад:** Незалежний розвиток класів фігур та кольорів.

4. Шаблон Template Method (Шаблонний метод)

**Призначення:** Визначення кістяка алгоритму з можливістю зміни окремих кроків.

**Основні характеристики:**

* Винесення загальної логіки в базовий клас
* Можливість перевизначення окремих кроків у підкласах

# **Хід роботи**

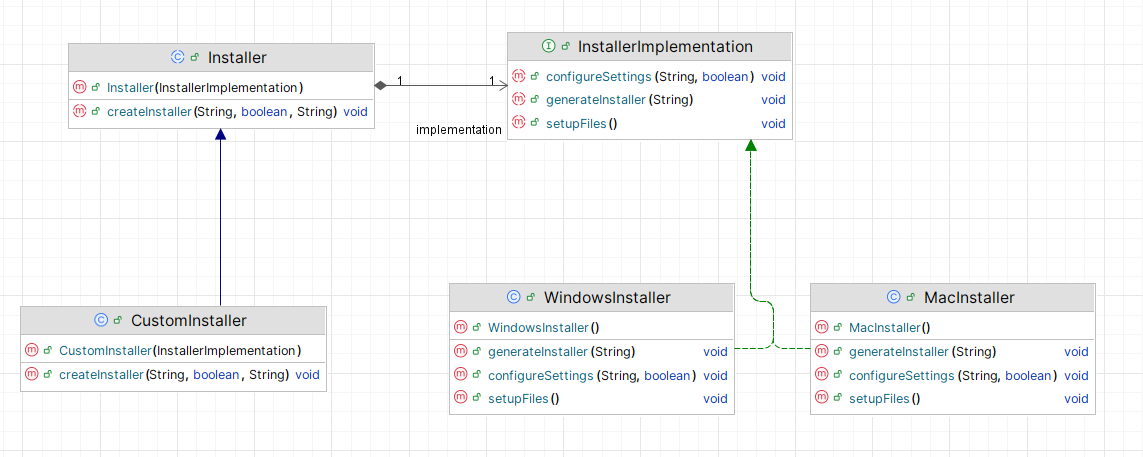


Рисунок №1 – Діаграма класів , згенерована IDE

На діаграмі показано реалізацію шаблону **Bridge**, що складається з чотирьох основних компонентів:

1. Installer (Абстракція)

Опис:

* Абстрактний клас, який має посилання на реалізацію (InstallerImplementation).

Методи:

* Installer(InstallerImplementation implementation): Конструктор, який приймає реалізацію.
* createInstaller(String licenseKey, boolean createShortcut, String outputFormat): Абстрактний метод для створення інсталяційного файлу.

Зв'язок:

* Асоціація з InstallerImplementation, що означає, що Installer делегує функціональність реалізації.

2. InstallerImplementation (Інтерфейс реалізації)

Опис:

* Інтерфейс, який визначає методи, що мають бути реалізовані для специфічних платформ (наприклад, Windows або Mac).

Методи:

* setupFiles(): Налаштування файлів для встановлення.
* configureSettings(String licenseKey, boolean createShortcut): Конфігурація параметрів, таких як ліцензійний ключ та ярлики.
* generateInstaller(String outputFormat): Генерація інсталяційного файлу (наприклад, .exe або .msi).

3. Конкретні реалізації (Специфічна реалізація функціоналу)

3.1. WindowsInstaller

Опис:

* Реалізація інтерфейсу InstallerImplementation для Windows.

Методи:

* setupFiles(): Реалізація налаштування файлів для Windows.
* configureSettings(String licenseKey, boolean createShortcut): Реалізація конфігурації параметрів для Windows.
* generateInstaller(String outputFormat): Генерація Windows-інсталятора у форматі .exe або .msi.

3.2. MacInstaller

Опис:

* Реалізація інтерфейсу InstallerImplementation для Mac.

Методи:

* setupFiles(): Реалізація налаштування файлів для Mac.
* configureSettings(String licenseKey, boolean createShortcut): Реалізація конфігурації параметрів для Mac.
* generateInstaller(String outputFormat): Генерація Mac-інсталятора у форматі .exe або .msi.

4. CustomInstaller (Розширена абстракція)

Опис:

* Конкретний клас, що розширює Installer і реалізує метод createInstaller.

Методи:

* CustomInstaller(InstallerImplementation implementation): Конструктор, який передає реалізацію в базовий клас.
* createInstaller(String licenseKey, boolean createShortcut, String outputFormat):
* Викликає методи setupFiles(), configureSettings() та generateInstaller() з реалізації.

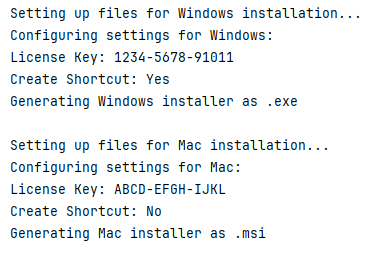
Загальна логіка:

* Абстракція Installer делегує виклик методів реалізації через посилання на InstallerImplementation.
* Специфічні реалізації (WindowsInstaller, MacInstaller) відповідають за конкретні дії для своїх платформ.
* Розширена абстракція CustomInstaller може додавати додаткову поведінку або викликати методи реалізації відповідно до потреб.

Приклад взаємодії:

Коли створюється об'єкт CustomInstaller, йому передається реалізація (WindowsInstaller або MacInstaller), і виклики методів createInstaller делегуються відповідній реалізації через InstallerImplementation.

# **Робота паттерну**



**Висновок**

За результатами виконання лабораторної роботи можна зробити наступні висновки. Проведено детальне вивчення шаблонів проєктування, застосовано патерн Bridge. Розвинуто вміння застосовувати шаблони проєктування у розробці програмного забезпечення.