ЛАБОРАТОРНА РОБОТА №9. ПОРОДЖУВАЛЬНІ ШАБЛОНИ. ШАБЛОНИ ABSTRACT FACTORY, BUILDER

Мета: Вивчення породжувальних шаблонів. Отримання базових навичок з застосування шаблонів Abstract Factory та Builder.

Довідка

Abstract Factory

Проблема. Створити сімейство взаємопов'язаних або взаємозалежних об'єктів (не специфікуючи їх конкретних класів).

Рішення. Створити абстрактний клас, в якому оголошено інтерфейс для створення конкретних класів.

Шаблон ізолює конкретні класи. Оскільки "AbstractFactory" інкапсулює відповідальність за створення класів і сам процес їх створення, то вона ізолює клієнта від деталей реалізації класів. Спрощено заміну "AbstractFactory", оскільки вона використовується в додатку тільки один раз при інстанціюванні.

Слід зазначити, що інтерфейс "AbstractFactory" фіксує набір об'єктів, які можна створити. Це в певній мірі ускладнює розширення "AbstractFactory" для виготовлення нових об'єктів.

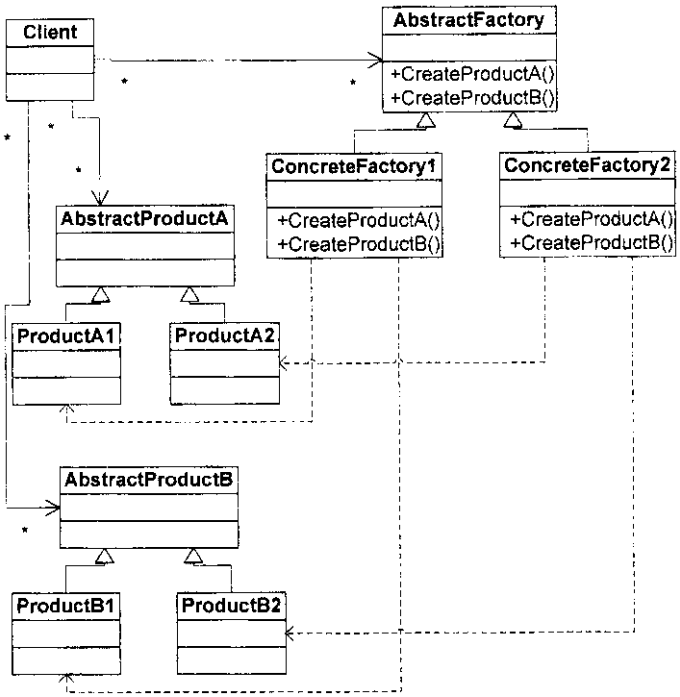

Рис. 14. Структура шаблону Abstract Factory

Builder

Проблема. Відокремити конструювання складного об'єкту від його подання, так щоб в результаті одного і того ж конструювання могли виходити різні подання. Алгоритм створення складного об'єкта не повинен залежати від

того, з яких частин складається об'єкт і як вони стикуються між собою.

Рішення. "Client" створює об'єкт - розпорядник "Director" і конфігурує його об'єктом - "Builder". "Director" повідомляє "Builder" про те, що потрібно побудувати чергову частину "Product"."Builder" обробляє запити "Director" і додає нові частини до "Product", потім "Client" забирає "Product" у "Builder".

Об'єкт "Builder" надає об'єкту "Director" абстрактний інтерфейс для конструювання "Product", за яким може приховати подання та внутрішню структуру продукту, та, крім того, процес складання "Product". Для зміни внутрішнього представлення "Product" досить визначити новий вид "Builder". Даний шаблон ізолює код, що реалізує створення об'єкта та його подання.

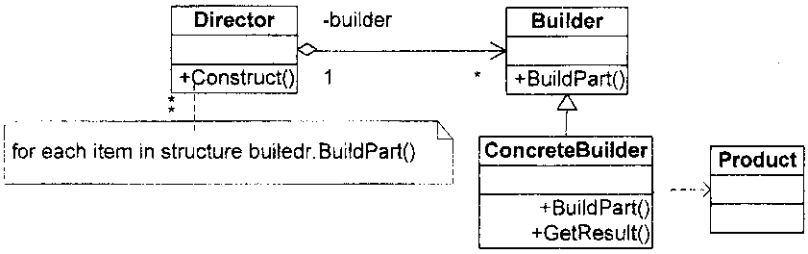

Рис. 15. Структура шаблону Builder

Завдання

1. Повторити породжувальні шаблони. Знати загальну характеристику породжувальних шаблонів та призначення кожного з них.
2. Детально вивчити породжувальні шаблони – Abstract Factory та Builder. Для кожного з них:
   - вивчити Шаблон, його призначення, альтернативні назви, мотивацію, випадки коли його застосування є доцільним та результати такого застосування;
   - знати особливості реалізації Шаблону, споріднені шаблони, відомі випадки його застосування в програмних додатках;
   - вільно володіти структурою Шаблону, призначенням його класів та відносинами між ними;
   - вміти розпізнавати Шаблон в UML діаграмі класів та будувати сирцеві коди Java-класів, що реалізують шаблон.
3. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork9. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.2). В розроблюваних класах повністю реалізувати методи, пов'язані з функціюванням Шаблону. Методи, що реалізують бізнес-логіку закрити заглушками з виводом на консоль інформації про викликаний метод та його аргументи.
4. За допомогою автоматизованих засобів виконати повне документування розроблених класів (також методів і полів), при цьому документація має в достатній мірі висвітлювати роль певного класу в загальній структурі Шаблону та особливості конкретної реалізації.

Варіанти завдання

Номер варіанту завдання обчислюється як залишок від ділення номеру залікової книжки на 11.

0. Визначити специфікації класів для подання сімейства віджетів графічного інтерфейсу користувача з реалізацією на різних API (WinAPI, GTK). Забезпечити можливість прозорого для клієнта розширення реалізацією для інших API (Qt, OSX).
1. Визначити специфікації класів для подання сімейства інструментів роботи з об'єктними даними через різні API (DB, File). Забезпечити можливість прозорого для клієнта розширення реалізацією для інших API (WebService).
2. Визначити специфікації класів для подання сімейства інструментів універсального інтерактивного середовища розробки (Validator, Compiler, Debugger) з їх реалізацією для різних мов (Java, C++). Забезпечити можливість прозорого для клієнта розширення реалізацією для мов (ObjectPascal).
3. Визначити специфікації класів для прямокутного ігрового простору та завантажувача його конфігурації із зовнішнього файлу.
4. Визначити специфікації класів для подання блок-схем алгоритмів (у відповідності до семантичної діаграми)

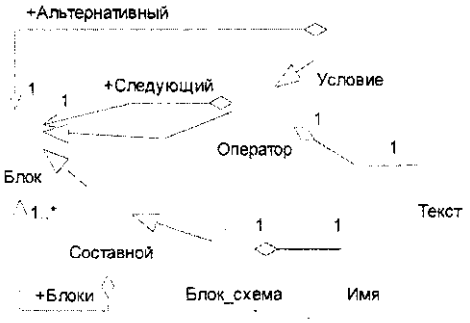

Рис. 16. Семантична діаграма блок-схем алгоритмів та її завантажувача із зовнішнього файлу.

5. Визначити специфікації класів для будівника дерева розбору складного виразу (у відповідності до БНФ) на основі його символьного подання.
   ```
   <вираз>::=<простий вираз> | <складний вираз>
   <простий вираз>::=<константа> | <змінна>
   <константа>::=(<число>)
   <змінна>::=(<ім'я>)
   <складний вираз>::=(<вираз><знак операції><вираз>)
   <знак операції>::=+|-|*|/
   ```
6. Визначити специфікації класів для подання запису, реляційної таблиці та її завантажувача із зовнішнього файлу.
7. Визначити специфікації класів для подання реляційної таблиці та будівника прямого добутку таблиць.
8. Визначити специфікації класів для подання реляційної таблиці та

будівника проекції таблиць.

9. Визначити специфікації класів для подання реляційної таблиці, схеми бази даних та відповідного завантажувача. Забезпечити можливість створення тільки одного примірника схеми бази даних.
10. Визначити специфікації класів для подання елементів векторного графічного редактору (примітив і композит). Реалізувати можливість побудови композитного зображення на основі завантаженого файлу-специфікації.

Протокол

Протокол має містити титульну сторінку (з номером залікової книжки), завдання, роздруківку діаграми класів, розроблений програмний код та згенеровану документацію в форматі JavaDoc.