ЛАБОРАТОРНА РОБОТА №8. ПОРОДЖУВАЛЬНІ ШАБЛОНИ. ШАБЛОНИ PROTOTYPE, SINGLETON, FACTORY METHOD

Мета: Вивчення породжувальних шаблонів. Отримання базових навичок з застосування шаблонів Prototype, Singleton та Factory Method.

Довідка

Prototype

Проблема. Система не повинна залежати від того, як в ній створюються, компонуються і представляються об'єкти.

Рішення. Створювати нові об'єкти за допомогою клонування. "Prototype" оголошує інтерфейс для клонування самого себе. "Client" створює новий об'єкт, звертаючись до "Prototype" з запитом клонувати "Prototype".

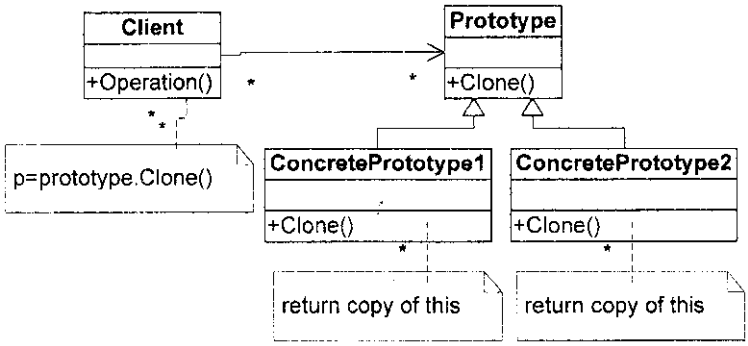

Рис. 1. Структура шаблону Prototype

Singleton

Проблема. Необхідний лише один примірник спеціального класу, різні об'єкти повинні звертатися до цього примірника через єдину точку доступу.

Рішення. Створити клас і визначити статичний метод класу, який повертає цей єдиний об'єкт.

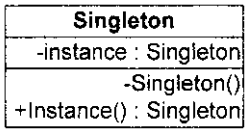

Рис. 2. Структура шаблону Singleton

Раціональніше створювати саме статичний примірник спеціального класу, а не оголосити необхідні методи статичними, оскільки при використанні методів примірника можна застосувати механізм наслідування й створювати підкласи. Статичні методи в мовах програмування не поліморфні і не допускають перекриття в похідних класах. Рішення на основі створення екземпляра є більш гнучким, оскільки згодом може знадобитися вже не єдиний екземпляр об'єкта, а декілька.

Factory Method

Проблема. Визначити інтерфейс для створення об'єкту, але залишити підкласами рішення про те, який клас інстанціювати, тобто, делегувати інстанціювання підкласами.

Рішення. Абстрактний клас "Creator" оголошує FactoryMethod, який повертає об'єкт типу "Product" (абстрактний клас, що визначає інтерфейс об'єктів, створюваних фабричним методом). "Creator" також може визначити реалізацію за замовчуванням FactoryMethod, який повертає "ConcreteProduct". "ConcreteCreator" заміщає FactoryMethod, який повертає об'єкт "ConcreteProduct". "Creator" "покладається" на свої підкласи в реалізації FactoryMethod, що повертає об'єкт "ConcreteProduct".

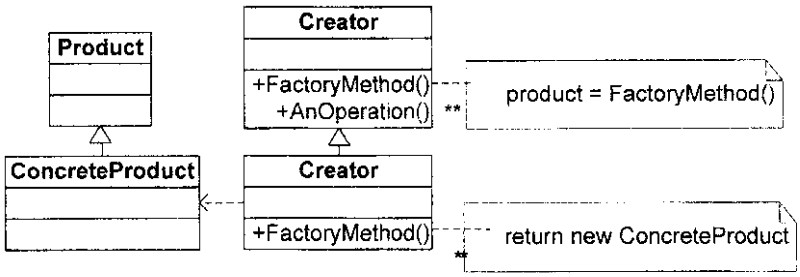

Рис. 3. Структура шаблону Factory Method

Шаблон позбавляє проектувальника від необхідності вбудовувати в код залежні від програми класи. При цьому виникає додатковий рівень підкласів.

Завдання

1. Вивчити породжувальні шаблони. Знати загальну характеристику породжувальних шаблонів та призначення кожного з них.
2. Детально вивчити породжувальні шаблони – Prototype, Singleton та Factory Method. 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.labwork8. 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 композитної структури ігрового простору. Реалізувати глибоке та поверхневе клонування такої структури.
1. Визначити специфікації класів та реалізацію методів для механізму клонування графічних елементів у редакторі векторної графіки. Забезпечити можливість як глибокого так і поверхневого клонування.
2. Визначити специфікації класів, які подають графічні елементи (примітиви та їх композиції) у редакторі векторної графіки. Реалізувати механізм клонування елементів з параметром глибини.
3. Визначити специфікації класів для подання файлової системи у вигляді дерева об'єктів (файл - листовий об'єкт, каталог - вузловий). Реалізувати механізм клонування таких об'єктів з параметром глибини.
4. Визначити специфікації класів для подання розпорядника гри, який складається з ігрового простору та списку ігрових фішок. Забезпечити можливість створення тільки одного примірника розпорядника.
5. Визначити специфікації класів для подання реляційної таблиці, схеми бази даних та валідатора запитів до таблиці. Забезпечити можливість створення тільки одного примірника схеми бази даних.
6. Визначити специфікації класів для подання композитної структури алгебраїчного виразу, карти змінних та обчислювача виразу. Забезпечити існування тільки одної карти змінних.
7. Визначити специфікації класів, які інкапсулюють лінійний список об'єктів та ітератор послідовного обходу у прямому та зворотньому напрямках для цієї структури.
8. Визначити специфікації класів, які інкапсулюють лінійний список цілих чисел та ітератор послідовного обходу у прямому напрямку в упорядкованій структурі.
9. Визначити специфікації класів для реалізації агрегату та його ітератору, який реалізує можливість зміни алгоритму пошуку наступного елементу під час виконання програми.
10. Визначити специфікації класів для реалізації композиту та його ітераторів — для обходу структури методами пошуку в глибину (DFS) та ширину (BFS).

Протокол

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