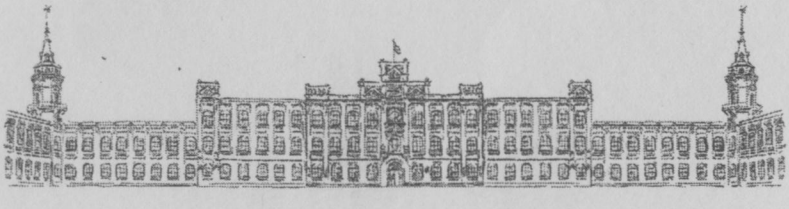
Міністерство освіти і науки, молоді та спорту України
Національний технічний університет України
«Київський політехнічний інститут»

Інженерія програмного забезпечення

Методичні вказівки до виконання лабораторних робіт

Частина ІІ

Шаблони поведінки та породжувальні шаблони



Київ-2011

Міністерство освіти і науки, молоді та спорту України
Національний технічний університет України
«Київський політехнічний інститут»

Інженерія програмного забезпечення

Методичні вказівки
до виконання лабораторних робіт
для студентів напряму підготовки
6.050102 «Комп'ютерна інженерія»

Частина ІІ

Шаблони поведінки та породжувальні шаблони

Рекомендовано Методичною радою НТУУ «КПІ»

Київ
НТУУ «КПІ»
2011

Інженерія програмного забезпечення [Текст] : метод. вказівки до викон. лаборатор. робіт для студ. напряму підготов. 6.050102 «Комп'ютерна інженерія» / Уклад.: А. О. Болдак, О. Н. Абу Усбах. – К.: НТУУ «КПІ», 2011. – Ч. II. Шаблони поведінки та породжувальні шаблони. – 44 с.

Гриф надано Методичною радою НТУУ «КПІ»
(Протокол № 5 від 03.02.2011 р.)

Навчальне видання

Інженерія програмного забезпечення

Методичні вказівки

до виконання лабораторних робіт
для студентів напряму підготовки
6.050102 «Комп'ютерна інженерія»

Частина II

Шаблони поведінки та породжувальні шаблони

Укладачі: *Болдак Андрій Олександрович,* канд. техн. наук, доц.
Абу Усбах Олексій Нідалійович, канд. техн. наук, доц.

Відповідальний редактор: *О. В. Бузовський,* д-р техн. наук, проф.

Рецензент: *І. А. Дичка,* д-р техн. наук, проф.

За редакцією укладачів
Надруковано з оригінал-макета замовника

Темплан 2010 р., поз. 2-131

Підп. до друку 16.03.2011. Формат 60×84 1/16. Папір офс. Гарнітура Times.
Спосіб друку – ризографія. Ум. друк. арк. 2,56. Обл.-вид. арк. 4,25. Наклад 50 пр. Зам. № 11-48.

НТУУ «КПІ» ВПІ ВПК «Політехніка»
Свідоцтво ДК № 1665 від 28.01.2004 р.
03056, Київ, вул. Політехнічна, 14, корп. 15
тел./факс (044) 406-81-78

ЗМІСТ

ВСТУП

Дисципліна «Інженерія програмного забезпечення» призначена для вивчення методів та засобів програмування, зокрема шаблонів проектування структурного рівня. Студенти, які приступають до вивчення даної дисципліни уже засвоїли курс «Програмування» і в достатній мірі володіють навичками створення простих програм за допомогою мов програмування Pascal, Object Pascal і Java.

Практична частина курсу складається з дев'яти лабораторних робіт і призначена для отримання практичних навичок використання шаблонів проектування при розробці програмного забезпечення. Всі лабораторні роботи виконуються в інтегрованому середовищі для розробки програм Eclipse 3.5. Роботи послідовно логічно впорядковані за складністю і охоплюють всі теми, що вивчаються в курсі.

Матеріал до кожної лабораторної роботи містить мету, теоретичні довідки та рекомендації, загальне завдання, варіанти індивідуальних завдань, список питань для самоперевірки, зміст звіту про виконання лабораторних робіт, а також список рекомендованих інформаційних джерел для підготовки і виконання лабораторних робіт.

Друга частина видання містить методичні вказівки для лабораторних робіт №5–№.9.

5. Шаблони поведінки. Шаблони Iterator, Mediator, Observer.
6. Шаблони поведінки - 2. Шаблони Strategy, Chain of Responsibility.
7. Шаблони поведінки - 3. Шаблони Memento, State, Command, Interpreter.
8. Породжувальні шаблони. Шаблони Prototype, Singleton та Factory Method.
9. Породжувальні шаблони - 2. Шаблони Abstract Factory, Builder.

ЛАБОРАТОРНА РОБОТА №5. ШАБЛОНИ ПОВЕДІНКИ. ШАБЛОНИ ITERATOR, MEDIATOR, OBSERVER

Мета: Вивчення шаблонів поведінки. Отримання базових навичок з застосування шаблонів Iterator, Mediator та Observer.

Довідка

Iterator

Проблема. Композитний об'єкт, наприклад, список, повинен надавати доступ до своїх елементів (об'єктів), не розкриваючи їх внутрішню структуру, причому перебирати список потрібно по-різному в залежності від завдання.

Рішення. Створюється клас "Iterator", який визначає інтерфейс для доступу і перебору елементів, "ConcreteIterator" реалізує інтерфейс класу "Iterator" і стежить за поточною позицією при обході "Aggregate"."Aggregate" визначає інтерфейс для створення об'єкту - ітератора. "ConcreteAggregate" реалізує інтерфейс створення ітератора і повертає екземпляр класу "ConcreteIterator", "ConcreteIterator" відстежує поточний об'єкт в агрегаті і може обчислити наступний об'єкт при переборі.

Шаблон підтримує різні способи перебору агрегату, одночасно можуть бути активні кілька переборів.

Mediator

Проблема. Забезпечити взаємодію великої кількості об'єктів, забезпечивши при цьому слабку зв'язаність і позбавивши об'єкти від необхідності явно посилатися один на одного.

Рішення. Створити об'єкт, що інкапсулює спосіб взаємодії великої кількості об'єктів.

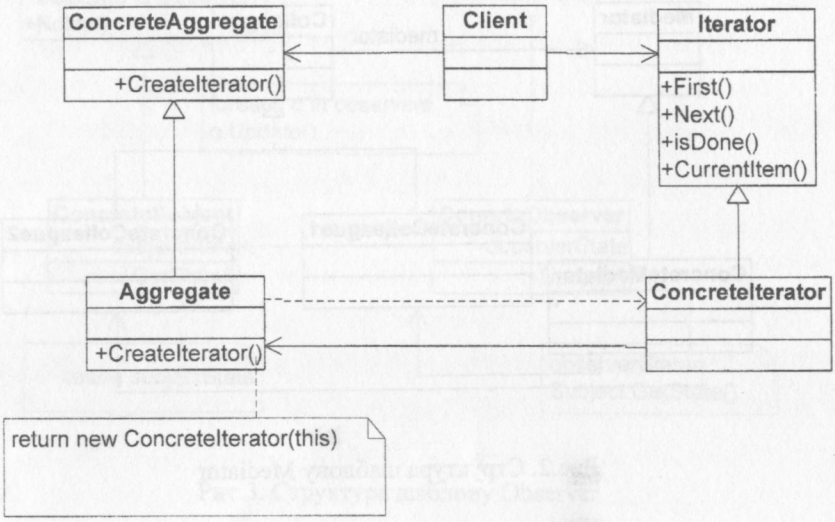


Рис.1. Структура шаблону Iterator

"Mediator" визначає інтерфейс для обміну інформацією з об'єктами "Colleague", "ConcreteMediator" координує дії об'єктів "Colleague". Кожен клас "Colleague" знає про свій об'єкт "Mediator", всі "Colleague" обмінюються інформацією тільки з посередником, при його відсутності їм довелося б обмінюватися інформацією безпосередньо."Colleague" посилають запити посередникові та отримують запити від нього. "Mediator" реалізує кооперативну поведінку, пересилаючи кожен запит одному або декільком "Colleague".

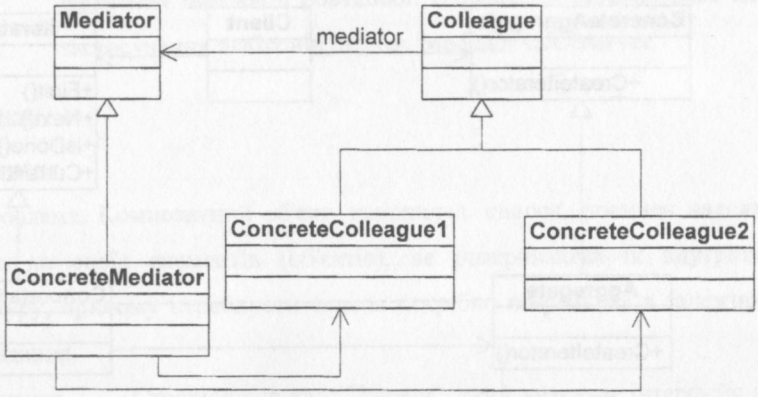


Рис.2. Структура шаблону Mediator

Шаблон усуває зв'язаність між "Colleague", централізуючи управління

Observer

Проблема. Один об'єкт ("Observer") повинен знати про зміну станів або деякі події іншого об'єкта. При цьому необхідно підтримувати низький рівень зв'язування з об'єктом - "Observer".

Рішення. Визначити інтерфейс "Observer". Об'єкти "ConcreteObserver" - передплатники реалізують цей інтерфейс та динамічно реєструються для отримання інформації про деяку подію в "Subject". Потім при реалізації обумовленої події в "ConcreteSubject" сповіщаються всі об'єкти — передплатники.

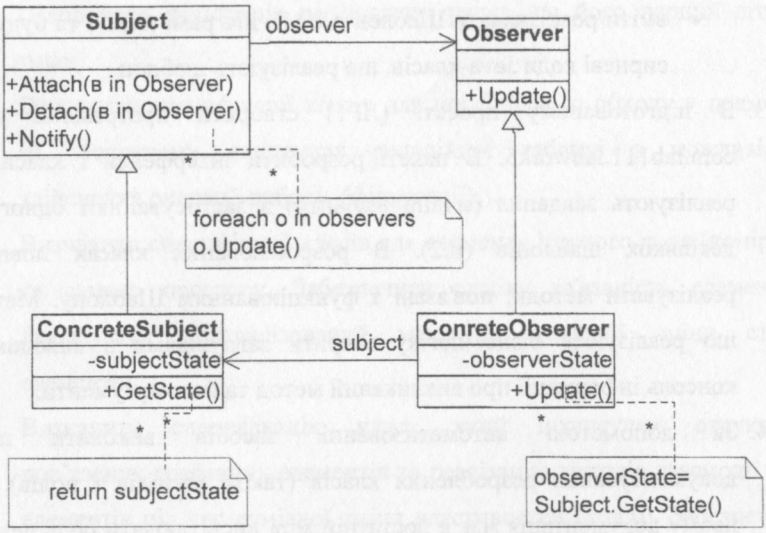


Рис.3. Структура шаблону Observer

Завдання

1. Вивчити шаблони поведінки для проектування ПЗ. Знати загальну характеристику шаблонів поведінки та призначення кожного з них.
2. Детально вивчити шаблони поведінки для проектування ПЗ - Iterator, Mediator та Observer. 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.labwork5. 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 9.

0. Визначити специфікації класів, які інкапсулюють лінійний список об'єктів та реалізують можливість послідовного обходу у прямому та зворотному напрямках оминаючи порожні елементи цієї структури та не розкриваючи її сутності перед користувачем.
1. Визначити специфікації класів, які інкапсулюють лінійний список цілих чисел та реалізують можливість звичайного послідовного обходу та послідовного обходу в упорядкованій структурі.
2. Визначити специфікації класів які інкапсулюють лінійний список символьних рядків та реалізують можливість звичайного послідовного обходу та обходу з додатковою фільтрацію агрегату

(Наприклад фільтрація по довжині рядка, по його першій літері, тощо).

3. Визначити специфікації класів для послідовного обходу у прямому та зворотному напрямках реляційної таблиці з можливістю здійснення операції вибору (фільтрації).
4. Визначити специфікації класів для елементу ігрового поля (комірки) та самого простору. Забезпечити слабку зв'язаність елементів. Реалізувати централізований механізм сумісної зміни стану елементів.
5. Визначити специфікацію класу, який інкапсулює структуру пов'язаних графічних елементів та реалізацію методів взаємодії цих елементів під час сумісної зміни властивостей (колір). Забезпечити слабку зв'язаність елементів.
6. Визначити специфікації класів для подання реляційної таблиці та обмеження зовнішнього ключа з можливістю його перевірки під час зміни значень полів. Забезпечити слабку зв'язаність елементів.
7. Визначити специфікації класів для подання елементів графічного інтерфейсу користувача — GUI (вікна, кнопки, текстові області). Реалізувати механізм реакції на події в будь-якому з елементів.
8. Визначити специфікації класів для подання реляційної таблиці. Реалізувати механізм тригерів — виконання додаткових дій при зміні елемента.

Питання для самостійної перевірки

1. Класифікація шаблонів проектування ПЗ.
2. Призначення шаблонів поведінки для проектування ПЗ.
3. Коротка характеристика кожного шаблону поведінки.
4. Назви, призначення та мотивація шаблону Iterator.

5. Структура шаблону Iterator та його учасники.
6. Особливості реалізації шаблону Iterator. Результат використання шаблону.
7. Назви, призначення та мотивація шаблону Mediator.
8. Структура шаблону Mediator та його учасники.
9. Особливості реалізації шаблону Mediator. Результат використання шаблону.
10. Назви, призначення та мотивація шаблону Observer.
11. Структура шаблону Observer та його учасники.
12. Особливості реалізації шаблону Observer. Результат використання шаблону.
13. Шаблони, які використовуються сумісно з Iterator, Mediator та Observer.

Протокол

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

Список рекомендованих інформаційних джерел

Шаблони проектування програмного забезпечення

- Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес Приемы объектно-ориентированного проектирования. Паттерны проектирования = Design Patterns: Elements of Reusable Object-Oriented Software. — СПб: «Питер», 2007. — С. 366. — ISBN 978-5-469-01136-1 (также ISBN 5-272-00355-1)
- Марк Гранд Шаблоны проектирования в JAVA. Каталог популярных шаблонов проектирования, проиллюстрированных

при помощи UML = Patterns in Java, Volume 1. A Catalog of Reusable Design Patterns Illustrated with UML. — М.: «Новое знание», 2004. — С. 560. — ISBN 5-94735-047-5

- Шаблони проектування програмного забезпечення – http://uk.wikipedia.org/wiki/Шаблони проектування програмного забезпечення
- Обзор паттернов проектирования – http://citforum.ru/SE/project/pattern/
- Объектно-ориентированное проектирование, паттерны проектирования (Шаблоны) – http://www.javenue.info/themes/ood/
- David Gallardo. Шаблоны проектирования Java - http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/jdp101/index.html

Шаблони поведінки

- Шаблони поведінки – http://uk.wikipedia.org/wiki/Шаблони поведінки
- Behavioral pattern – http://en.wikipedia.org/wiki/Behavioral_pattern
- Шаблоны поведения – http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/jdp101/part6.html

ЛАБОРАТОРНА РОБОТА №6. ШАБЛОНИ ПОВЕДІНКИ. ШАБЛОНИ STRATEGY, CHAIN OF RESPONSIBILITY, VISITOR

Мета: Вивчення шаблонів поведінки. Отримання базових навичок з застосування шаблонів Strategy, Chain of Responsibility та Visitor.

Довідка

Strategy

Проблема. Спроектувати змінювані, але надійні алгоритми або стратегії.

Рішення. Визначити для кожного алгоритму або стратегії окремий клас зі стандартним інтерфейсом “Strategy”.

Створюється декілька класів “ConcreteStrategy”, кожен з яких містить один і той же поліморфний метод "AlgorithmInterface". Об'єкт стратегії зв'язується з контекстним об'єктом (тим об'єктом, до якого застосовується алгоритм).

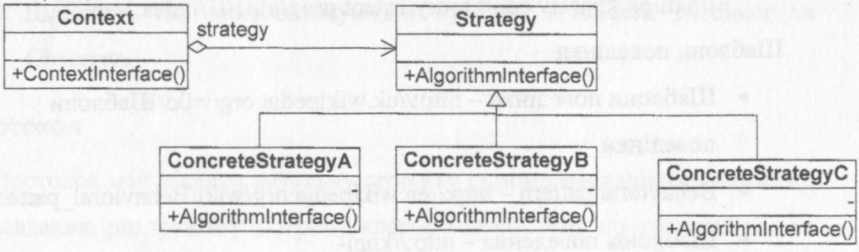


Рис.4. Структура шаблону Strategy

Chain of Responsability

Проблема. Запит повинен бути оброблений декількома об'єктами.

Рішення. Зв'язати об'єкти - одержувачі запиту в ланцюжок і передавати запит вздовж цього ланцюжка, поки він не буде оброблений. "Handler" визначає інтерфейс для обробки запитів, і, можливо, реалізує зв'язок з наступником. "ConcreteHandler" обробляє запит, за який відповідає. Маючи доступ до свого наступника "ConcreteHandler" направляє запит до нього, якщо не може обробити запит сам.

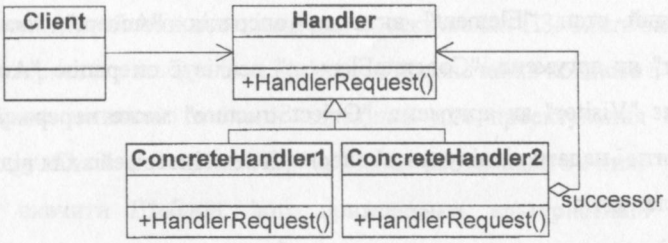


Рис.5. Структура шаблону Chain of Responsability

Логічним є застосування цього шаблону, якщо є більше одного об'єкта, здатного обробити запит і обробник заздалегідь невідомий (і повинен бути знайдений автоматично) або якщо весь набір об'єктів, які здатні обробити запит, повинен задаватися автоматично.

Шаблон послаблює зв'язаність (об'єкт не зобов'язаний "знати", хто саме опрацює його запит). Але немає гарантій, що запит буде оброблений, оскільки він не має явного одержувача.

Visitor

Проблема. Над кожним об'єктом деякої структури виконується операція. Визначити нову операцію, не змінюючи класи об'єктів.

Рішення. Клієнт, який використовує даний шаблон, повинен створити об'єкт класу "ConcreteVisitor", а потім відвідати кожен елемент структури. "Visitor" оголошує операцію "VisitConcreteElement" для кожного класу "ConcreteElement" (ім'я та сигнатура даної операції ідентифікують клас, елемент якого відвідує "Visitor" - тобто, відвідувач може звертатися до елементу безпосередньо). "ConcreteVisitor" реалізує всі операції, оголошення в класі "Visitor". Кожна операція реалізує фрагмент алгоритму, визначеного для класу відповідного об'єкта в структурі. Клас

"ConcreteVisitor" надає контекст для цього алгоритму і зберігає його локальний стан. "Element" визначає операцію "Accept", яка приймає "Visitor" як аргумент, "ConcreteElement" реалізує операцію "Accept", яка приймає "Visitor" як аргумент. "ObjectStructure" може перерахувати свої аргументи і надати відвідувачеві високорівнева інтерфейс для відвідування своїх елементів.

Логічним є використання цього шаблону, якщо в структурі присутні об'єкти багатьох класів з різними інтерфейсами, і необхідно виконати над ними операції, що залежать від конкретних класів, або якщо класи, що визначають структуру об'єктів змінюються рідко, але нові операції над цією структурою додаються часто.

Шаблон спрощує додавання нових операцій, об'єднує споріднені операції в класі "Visitor". При цьому ускладнюється додавання нових класів "ConcreteElement", оскільки потрібно оголошення нової абстрактної операції в класі "Visitor".

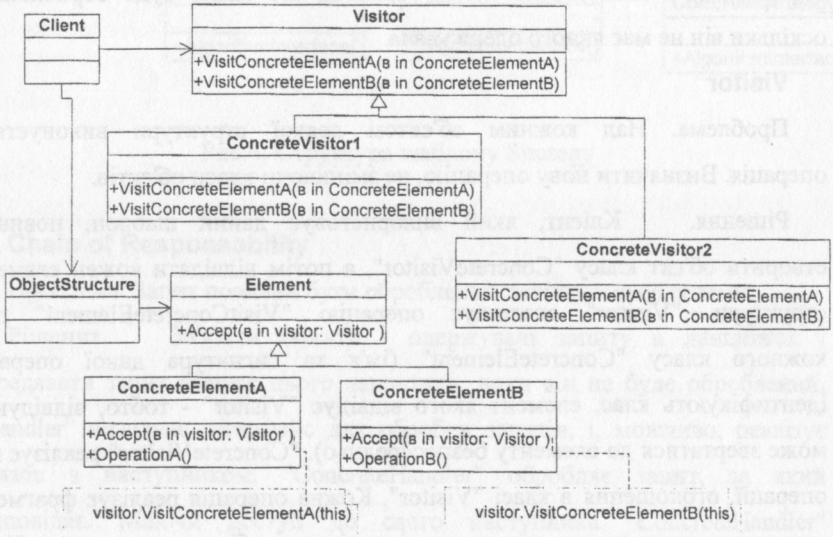


Рис.6. Структура шаблону Visitor

Завдання

1. Повторити шаблони поведінки для проектування ПЗ. Знати загальну характеристику шаблонів поведінки та призначення кожного з них.
2. Детально вивчити шаблони поведінки для проектування ПЗ — Strategy, Chain of Responsibility та Visitor. 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.labwork6. 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 10.

0. Визначити специфікації класу, який містить масив цілих чисел та метод його сортування. Забезпечити можливість динамічної зміни алгоритму та напрямку сортування шляхом зовнішньої параметризації.
1. Визначити специфікації класу, який містить таблицю та метод її відображення у вигляді діаграми. Забезпечити можливість динамічної зміни типу діаграми шляхом зовнішньої параметризації.
2. Визначити специфікації класу, який містить математичну функцію та метод її відображення у вигляді графіка. Забезпечити можливість динамічної зміни системи координат графіка (декартова, полярна тощо) шляхом зовнішньої параметризації.
3. Визначити специфікації класів, що реалізують контейнери для цілих чисел та текстових строк з можливістю їх сортування. Забезпечити можливість динамічної зміни алгоритму сортування шляхом зовнішньої параметризації. Реалізація алгоритму сортування має бути незалежною від типу даних, що сортуються.
4. Визначити специфікації класів, що реалізують елементи графічного інтерфейсу користувача — панелі (композит) та кнопки (компонент). Реалізувати децентралізований механізм обробки події переміщення курсору миші. Кількість компонентів інтерфейсу, які реагують на цю подію, може змінюватись динамічно.
5. Визначити специфікації класів, що реалізують обробку HTTP-запитів різних типів (наприклад GET та POST). Реалізувати можливість

динамічної зміни кількості обробників. Забезпечити децентралізацію та слабку зв'язаність обробників.

6. Визначити специфікації класів для елементу ігрового поля (комірки) та самого простору. Забезпечити слабку зв'язаність елементів. Реалізувати децентралізований механізм сумісної зміни стану елементів.
7. Визначити специфікації класів, що реалізують елементи графічного інтерфейсу користувача — панелі (композит) та кнопки (компонент). Реалізувати механізм додаткових операцій над структурою графічного інтерфейсу без зміни її елементів. В якості ілюстрації такого механізму розробити операцію підрахунку кількості елементів одного типу.
8. Визначити специфікації класів, що реалізують елементи структури комп'ютера (процесор, пам'ять, відеокарта тощо). Реалізувати механізм додаткових операцій над структурою комп'ютера без зміни її елементів. В якості ілюстрації такого механізму розробити операцію визначення потужності, що потребляє комп'ютер.
9. Визначити специфікації класів, що реалізують елементи мережевої структури (кабель, сервер, робоча станція). Реалізувати механізм додаткових операцій над мережевою структурою без зміни її елементів. В якості ілюстрації такого механізму розробити операцію визначення кошторису такої структури.

Питання для самостійної перевірки

1. Класифікація шаблонів проектування ПЗ.
2. Призначення шаблонів поведінки для проектування ПЗ.
3. Коротка характеристика кожного шаблону поведінки.
4. Назви, призначення та мотивація шаблону Strategy.

5. Структура шаблону Strategy та його учасники.
6. Особливості реалізації шаблону Strategy. Результат використання шаблону.
7. Назви, призначення та мотивація шаблону Chain of Responsibility.
8. Структура шаблону Chain of Responsibility та його учасники.
9. Особливості реалізації шаблону Chain of Responsibility. Результат використання шаблону.
10. Назви, призначення та мотивація шаблону Visitor.
11. Структура шаблону Visitor та його учасники.
12. Особливості реалізації шаблону Visitor. Результат використання шаблону.
13. Шаблони, які використовуються сумісно з Strategy, Chain of Responsibility та Visitor.

Протокол

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

Список рекомендованих інформаційних джерел

Шаблони проектування програмного забезпечення

- Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес Приемы объектно-ориентированного проектирования. Паттерны проектирования = Design Patterns: Elements of Reusable Object-Oriented Software. — СПб: «Питер», 2007. — С. 366. — ISBN 978-5-469-01136-1 (также ISBN 5-272-00355-1)
- Марк Гранд Шаблоны проектирования в JAVA. Каталог популярных шаблонов проектирования, проиллюстрированных

при помощи UML = Patterns in Java, Volume 1. A Catalog of Reusable Design Patterns Illustrated with UML. — М.: «Новое знание», 2004. — С. 560. — ISBN 5-94735-047-5

- Шаблони проектування програмного забезпечення – http://uk.wikipedia.org/wiki/Шаблони проектування програмного забезпечення
- Обзор паттернов проектирования – http://citforum.ru/SE/project/pattern/
- Объектно-ориентированное проектирование, паттерны проектирования (Шаблоны) – http://www.javenue.info/themes/ood/
- David Gallardo. Шаблоны проектирования Java – http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/jdp101/index.html

Шаблони поведінки

- Шаблони поведінки – http://uk.wikipedia.org/wiki/Шаблони поведінки
- Behavioral pattern – http://en.wikipedia.org/wiki/Behavioral_pattern
- Шаблоны поведения – http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/jdp101/part6.html

ЛАБОРАТОРНА РОБОТА №7. ШАБЛОНИ ПОВЕДІНКИ. ШАБЛОНИ MEMENTO, STATE, COMMAND, INTERPRETER

Мета: Вивчення шаблонів поведінки. Отримання базових навичок з застосування шаблонів Memento, State, Command та Interpreter.

Довідка

Memento

Проблема. Необхідно зафіксувати стан об'єкту для реалізації, наприклад, механізму відкату.

Рішення. Зафіксувати і винести (не порушуючи інкапсуляції) за межі об'єкта його внутрішній стан так, щоб згодом можна було його відновити в об'єкті. "Memento" зберігає внутрішній стан об'єкта "Originator" і забороняє доступ до себе всім іншим об'єктам окрім "Originator", який має доступ до всіх даних для відновлення в колишньому стані. "Caretaker" може лише передавати "Memento" іншим об'єктам. "Originator" створює "Memento", що містить знімок поточного внутрішнього стану і використовує "Memento" для відновлення внутрішнього стану. "Caretaker" відповідає за збереження "Memento", при цьому не робить ніяких операцій над "Memento" і не досліджує його внутрішній вміст. "Caretaker" запитує "Memento" у "Originator", деякий час тримає його у себе, а потім повертає "Originator".

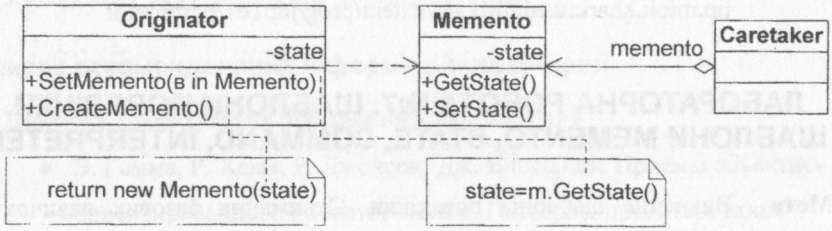


Рис.7. Структура шаблону Memento

При використанні шаблону не розкривається інформація, яка доступна тільки "Originator", спрощується структура "Originator". Але з

використанням "Memento" можуть бути пов'язані значні витрати, якщо "Originator" повинен копіювати великий обсяг інформації, або якщо копіювання повинно проводитися часто.

State

Проблема. Варіювати поведінку об'єкта в залежності від його внутрішнього стану.

Рішення. Клас "Context" делегує запити, які залежать від стану, поточному об'єкту "ConcreteState" (зберігає екземпляр підкласу "ConcreteState", яким визначається поточний стан), і визначає інтерфейс, що представляє інтерес для клієнтів. "ConcreteState" реалізує поведінку, асоційовану з якимось станом об'єкта "Context". "State" визначає інтерфейс для інкапсуляції поведінки, асоційованої з конкретним екземпляром "Context".

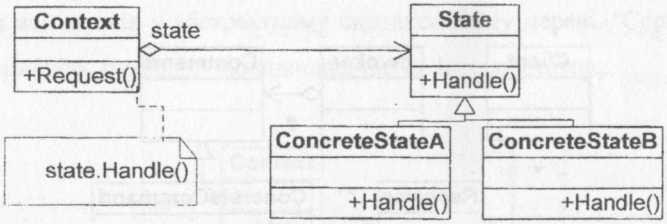


Рис.8. Структура шаблону State

Шаблон локалізує залежну від стану поведінку і ділить її на частини, що відповідають станам, переходи між станами стають явними.

Command

Проблема. Необхідно надіслати об'єкту запит, не знаючи про те, виконання якої операції запитане і хто буде одержувачем.

Рішення. Інкапсулювати запит як об'єкт. "Client" створює об'єкт "ConcreteCommand", який викликає операції одержувача для виконання запиту, "Invoker" відправляє запит, виконуючи операцію "Command" Execute(). "Command" оголошує інтерфейс для виконання операції, "ConcreteCommand" визначає зв'язок між об'єктом "Receiver" і операцією Action(), і, крім того, реалізує операцію Execute() шляхом виклику відповідних операцій об'єкта "Receiver". "Client" створює екземпляр класу "ConcreteCommand" і встановлює його одержувача, "Invoker" звертається до команди для виконання запиту, "Receiver" (будь-який клас) має інформацію про способи виконання операцій, необхідних для виконання запиту.

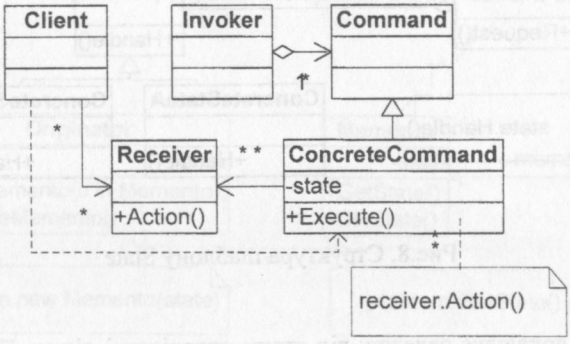


Рис.9. Структура шаблону Command

Шаблон Command розриває зв'язок між об'єктом, який ініціює операції, і об'єктом, що має інформацію про те, як її виконати, крім того

створюється об'єкт "Command", який можна розширювати і маніпулювати ним як об'єктом.

Interpreter

Проблема. Існує підвладна змінам задача, що зустрічається часто.

Рішення. Створити інтерпретатор, який вирішує цю задачу.

Завдання пошуку рядків за зразком може бути вирішено за допомогою створення інтерпретатора, що визначає граматику мови. "Client" будує речення у вигляді абстрактного синтаксичного дерева, у вузлах якої знаходяться об'єкти класів "TerminalExpression" і "NonterminalExpression" (рекурсивне), потім "Client" ініціалізує контекст і викликає операцію Interpret(Context). На кожному вузлі типу "TerminalExpression" реалізується операція Interpret(Context) для кожного підвиразу. Для класу "NonterminalExpression" операція Interpret(Context) визначає базу рекурсії. "AbstractExpression" визначає абстрактну операцію Interpret(Context), загальну для всіх вузлів у абстрактному синтаксичному дереві. "Context" містить інформацію, глобальну по відношенню до інтерпретатору.

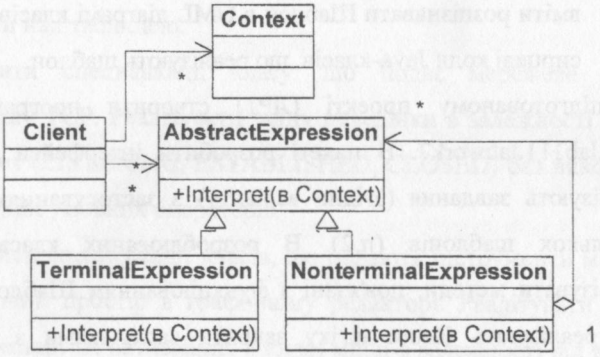


Рис.10. Структура шаблону Interpreter

Завдяки шаблону граматику стає легко розширювати і змінювати, реалізації класів, що описують вузли абстрактного синтаксичного дерева схожі (легко кодуються). Можна легко змінювати спосіб обчислення виразів. Але важко супроводжувати граматику з великим числом правил.

Завдання

1. Повторити шаблони поведінки для проектування ПЗ. Знати загальну характеристику шаблонів поведінки та призначення кожного з них.
2. Детально вивчити шаблони поведінки для проектування ПЗ - Memento, State, Command та Interpreter. 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.labwork7. 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 8.

0. Визначити специфікації класів, які подають у векторному редакторі графічні елементи (коло, трикутник тощо) з різними атрибутами (колір, позиція, розмір тощо). Реалізувати механізм збереження/встановлення стану елемента.
1. Визначити специфікації класу, що подає персонажа в ігровому просторі з необхідними атрибутами (позиція персонажу, склад артефактів, рівень "здоров'я" тощо). Реалізувати механізм збереження/встановлення стану персонажа.
2. Визначити специфікації класів, які подають операції над таблицю в БД. Реалізувати механізм організації транзакцій при виконанні операцій над таблицею.
3. Визначити специфікації класу, що подає мережеве з'єднання протоколу TCP. Реалізувати зміну поведінки в залежності від стану з'єднання (LISTENING, ESTABLISHED, CLOSED) без використання громіздких умовних операторів.
4. Визначити специфікації класів, що подають інструменти малювання та робочий простір в графічному редакторі. Реалізувати механізм зміни реакції на натискання кнопки миші в залежності від вибраного інструменту. Уникати використання громіздких умовних конструкцій.

5. Визначити специфікації класів, що подають чергу HTTP-запитів на обробку. Реалізувати можливість виключення запитів з черги без обробки, та зміни позиції запиту через зміну значення пріоритету.
6. Визначити специфікації класів, що подають реакції на натискання пунктів меню та кнопок інструментальної панелі. Забезпечити можливість динамічної зміни реакції, а також формування макро-реакцій (послідовність з наперед заданих реакцій).
7. Визначити специфікації класів для розбору алгебраїчних виразів з операціями +, -, *, /.

Питання для самостійної перевірки

1. Класифікація шаблонів проектування ПЗ.
2. Призначення шаблонів поведінки для проектування ПЗ.
3. Коротка характеристика кожного шаблону поведінки.
4. Назви, призначення та мотивація шаблону Memento.
5. Структура шаблону Memento та його учасники.
6. Особливості реалізації шаблону Memento. Результат використання шаблону.
7. Назви, призначення та мотивація шаблону State.
8. Структура шаблону State та його учасники.
9. Особливості реалізації шаблону State. Результат використання шаблону.
10. Назви, призначення та мотивація шаблону Command.
11. Структура шаблону Command та його учасники.
12. Особливості реалізації шаблону Command. Результат використання шаблону.
13. Назви, призначення та мотивація шаблону Interpreter.
14. Структура шаблону Interpreter та його учасники.

15. Особливості реалізації шаблону Interpreter. Результат використання шаблону.
16. Шаблони, які використовуються сумісно з Memento, State, Command та Interpreter.

Протокол

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

Список рекомендованих інформаційних джерел

Шаблони проектування програмного забезпечення

- Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес Приемы объектно-ориентированного проектирования. Паттерны проектирования = Design Patterns: Elements of Reusable Object-Oriented Software. — СПб: «Питер», 2007. — С. 366. — ISBN 978-5-469-01136-1 (также ISBN 5-272-00355-1)
- Марк Гранд Шаблоны проектирования в JAVA. Каталог популярных шаблонов проектирования, проиллюстрированных при помощи UML = Patterns in Java, Volume 1. A Catalog of Reusable Design Patterns Illustrated with UML. — М.: «Новое знание», 2004. — С. 560. — ISBN 5-94735-047-5
- Шаблони проектування програмного забезпечення – http://uk.wikipedia.org/wiki/Шаблони проектування програмного забезпечення
- Обзор паттернов проектирования – http://citforum.ru/SE/project/pattern/

- Объектно-ориентированное проектирование, паттерны проектирования (Шаблоны) – http://www.javenue.info/themes/ood/
- David Gallardo. Шаблоны проектирования Java - http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/jdp101/index.html

Шаблони поведінки

- Шаблони поведінки – http://uk.wikipedia.org/wiki/Шаблони поведінки
- Behavioral pattern – http://en.wikipedia.org/wiki/Behavioral_pattern
- Шаблоны поведения – http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/jdp101/part6.html

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".

Singleton

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

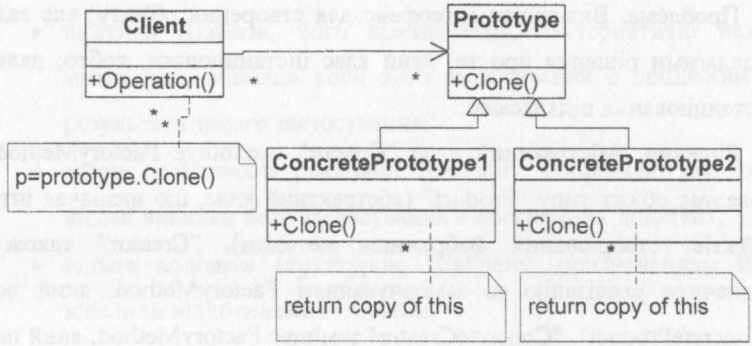


Рис.11. Структура шаблону Prototype

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

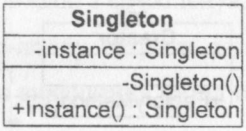


Рис.12. Структура шаблону Singleton

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".

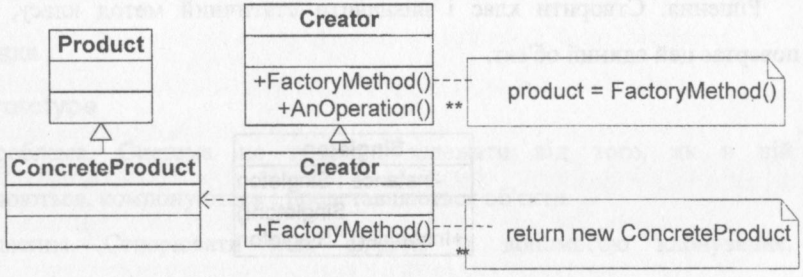


Рис.13. Структура шаблону Factory Method

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 функціонуванням 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).

Питання для самостійної перевірки

1. Класифікація шаблонів проектування ПЗ.
2. Призначення породжувальних шаблонів.
3. Коротка характеристика кожного породжувального шаблону.
4. Назви, призначення та мотивація шаблону Prototype.
5. Структура шаблону Prototype та його учасники.
6. Особливості реалізації шаблону Prototype. Результат використання шаблону.
7. Назви, призначення та мотивація шаблону Singleton.
8. Структура шаблону Singleton та його учасники.
9. Особливості реалізації шаблону Singleton. Результат використання шаблону.
10. Назви, призначення та мотивація шаблону Factory Method.
11. Структура шаблону Factory Method та його учасники.

12. Особливості реалізації шаблону Factory Method. Результат використання шаблону.
13. Шаблони, які використовуються сумісно з Prototype, Singleton та Factory Method.
14. Глибоке та поверхневе клонування.

Протокол

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

Список рекомендованих інформаційних джерел

Шаблони проектування програмного забезпечення

- Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес Приемы объектно-ориентированного проектирования. Паттерны проектирования = Design Patterns: Elements of Reusable Object-Oriented Software. — СПб: «Питер», 2007. — С. 366. — ISBN 978-5-469-01136-1 (также ISBN 5-272-00355-1)
- Марк Гранд Шаблоны проектирования в JAVA. Каталог популярных шаблонов проектирования, проиллюстрированных при помощи UML = Patterns in Java, Volume 1. A Catalog of Reusable Design Patterns Illustrated with UML. — М.: «Новое знание», 2004. — С. 560. — ISBN 5-94735-047-5
- Шаблони проектування програмного забезпечення – http://uk.wikipedia.org/wiki/Шаблони проектування програмного забезпечення
- Обзор паттернов проектирования – http://citforum.ru/SE/project/pattern/

- Объектно-ориентированное проектирование, паттерны проектирования (Шаблоны) – http://www.javenue.info/themes/ood/
- David Gallardo. Шаблоны проектирования Java - http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/jdp101/index.html

Породжувальні шаблони

- Твірні шаблони – http://uk.wikipedia.org/wiki/Твірні шаблони
- Creational pattern – http://en.wikipedia.org/wiki/ Creational_pattern
- Шаблоны создания – http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/jdp101/part4.html

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.

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.

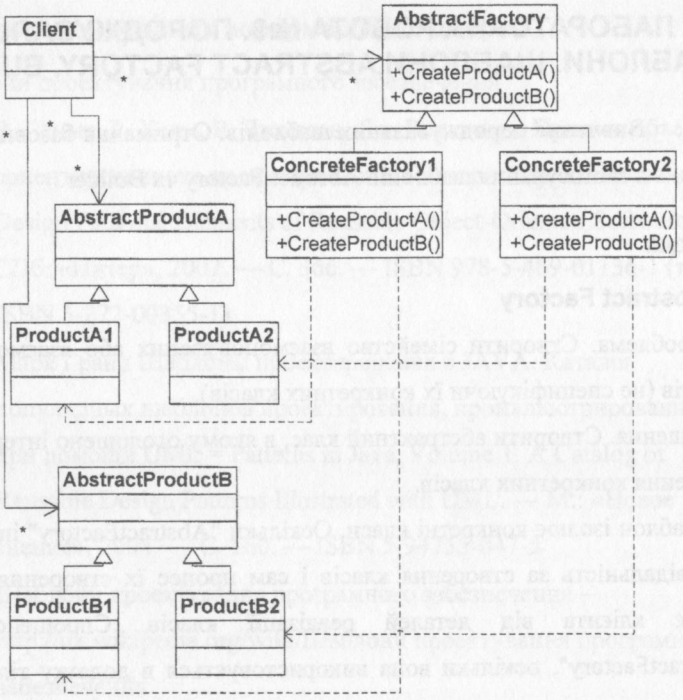


Рис.14. Структура шаблону Abstract Factory

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.

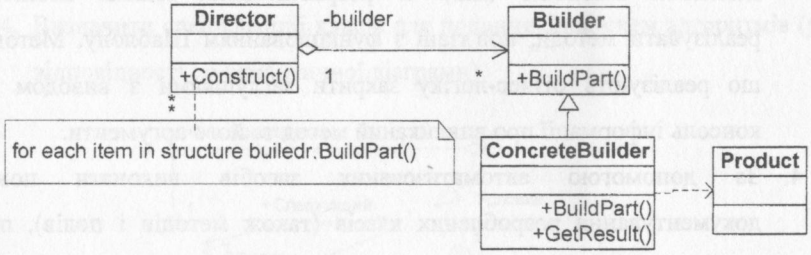


Рис.15. Структура шаблону Builder

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)

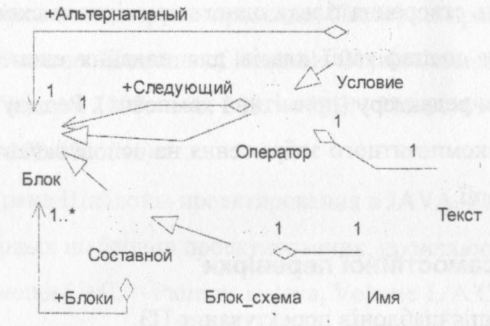


Рис.16. Семантична діаграма блок-схем алгоритмів

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.

Питання для самостійної перевірки

1. Класифікація шаблонів проектування ПЗ.
2. Призначення породжувальних шаблонів.
3. Коротка характеристика кожного породжувального шаблону.
4. Назви, призначення та мотивація шаблону Abstract Factory.
5. Структура шаблону Abstract Factory та його учасники.

6. Особливості реалізації шаблону Abstract Factory. Результат використання шаблону.
7. Назви, призначення та мотивація шаблону Builder.
8. Структура шаблону Builder та його учасники.
9. Особливості реалізації шаблону Builder. Результат використання шаблону.
10. Шаблони, які використовуються сумісно з Abstract Factory та Builder.

Протокол

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

Список рекомендованих інформаційних джерел

Шаблони проектування програмного забезпечення

- Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес Приемы объектно-ориентированного проектирования. Паттерны проектирования = Design Patterns: Elements of Reusable Object-Oriented Software. — СПб: «Питер», 2007. — С. 366. — ISBN 978-5-469-01136-1 (также ISBN 5-272-00355-1)
- Марк Гранд Шаблоны проектирования в JAVA. Каталог популярных шаблонов проектирования, проиллюстрированных при помощи UML = Patterns in Java, Volume 1. A Catalog of Reusable Design Patterns Illustrated with UML. — М.: «Новое знание», 2004. — С. 560. — ISBN 5-94735-047-5
- Шаблони проектування програмного забезпечення – http://uk.wikipedia.org/wiki/Шаблони проектування програмного забезпечення

- Обзор паттернов проектирования – http://citforum.ru/SE/project/pattern/
- Объектно-ориентированное проектирование, паттерны проектирования (Шаблоны) – http://www.javenue.info/themes/ood/
- David Gallardo. Шаблоны проектирования Java - http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/jdp101/index.html

Породжувальні шаблони

- Твірні шаблони – http://uk.wikipedia.org/wiki/Твірні шаблони
- Creational pattern – http://en.wikipedia.org/wiki/ Creational_pattern
- Шаблоны создания – http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/jdp101/part4.html