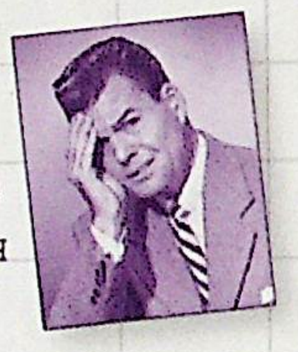
Оновлено для Java 8

Head First

Патерни проєктування

Легкий для сприйняття довідник



Уникайте безглуздих помилок через наслідування

Дізнайтеся секрети гуру проєктування патернів

Дізнайтеся, як Starbuzz Coffee подвоїла ціну акцій завдяки патернам проєктування

Дізнайтеся, чому все, що знали ваші друзі про патерни, імовірно, помилкове

Оволодійте важливими для вас патернами максимально ефективно

Перевірте, як покращиться любовне життя Джима, коли він позбудеться старих шаблонів

Ерік Фрімен, Елізабет Робсон
за участю Кеті Сьєрра й Берта Бейтса

ВИДАВНИЦТВО ФАБУЛА #PRO

«Патерни проєктування» — ваша книжка, якщо вам хоч колись доводилося стикатися з проблемами дизайну програмного забезпечення. Вам не доведеться «винаходити колесо» — просто скористайтеся зведеним докупи величезним досвідом розробників і можливістю використовувати найкращі практики. Ви дізнаєтеся, навіщо потрібні патерни, побачите, як вони виглядають і працюють у «дикій природі», а призвичаївшись до їх використання, зможете витратити вільний час на щось більш складне і цікаве. Автори чудово показали, як патерни використовуються у Java API і як застосувати вбудовану підтримку патернів Java у вашому власному коді. Заразом ви засвоїте реальні принципи ОО-програмування, що залишаться з вами і тоді, коли доведеться працювати без патернів.

Візуально насичений формат книжки розроблений із використанням новітніх досліджень в галузях нейробіології, когнітивної науки і теорії навчання. Тому весь її матеріал миттєво запам'ятовується, і невдовзі ви зможете з легкістю спілкуватися з іншими членами вашої команди на «таємній мові» патернів проєктування.

Зміст (коротко)

Зміст (докладно)

Вступ

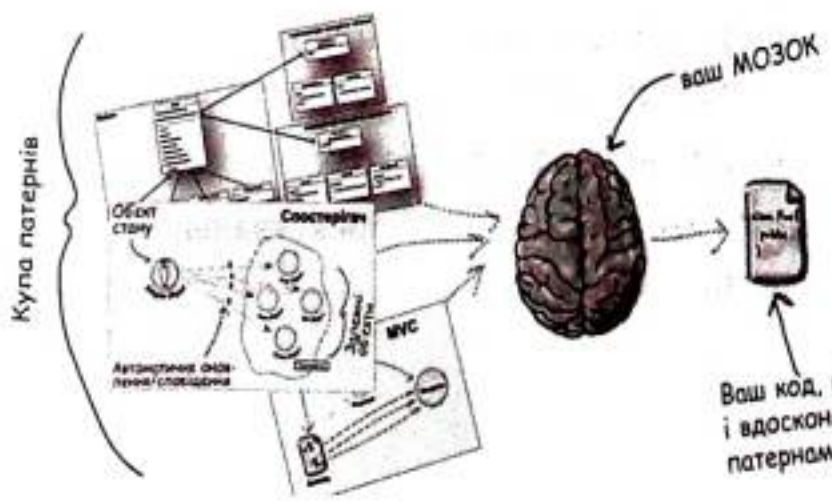
Налаштуйте свій мозок на патерни проєктування. Ось що вам знадобиться, коли ви намагаєтеся щось засвоїти, у той час як ваш мозок не хоче сприймати інформацію. Ваш мозок думає: «Краще перейматися важливішими речами, наприклад, небезпечними дикими тваринами або питанням, чому не можна голяка покататися на сноуборді». Як же змусити свій мозок думати, що ваше життя залежить від оволодіння патернами проєктування?

1 Знайомство з патернами проєктування

Ласкаво просимо до світу патернів

Хтось вже вирішив ваші проблеми. У цьому розділі ви дізнаєтеся, чому (і як) слід використовувати досвід інших розробників, які вже стикалися з аналогічними завданнями і успішно впоралися з ними. Наразі ми поговоримо про використання та переваги патернів проєктування, познайомимося з ключовими принципами ОО-проєктування і розберемо приклад одного з патернів. Найкращий спосіб використання патернів полягає в тому, аби запам'ятати їх, а потім навчитися розпізнавати ті місця ваших архітектур та наявних застосунків, де їх доречно використати. Отже, замість програмного коду ви повторно скористаєтеся чужим досвідом.

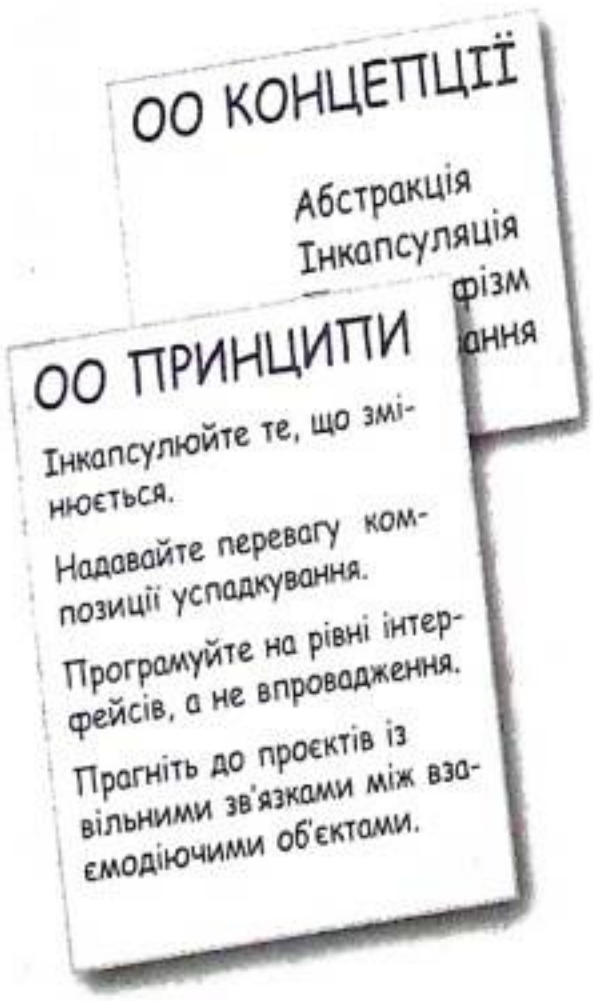
Пам'ятайте, що знання таких концепцій, як абстракція, успадкування та поліморфізм, ще не робить із вас вдалого об'єктно-орієнтованого проєктувальника. Істинний гуру-проєктувальник докладає зусиль для забезпечення гнучкої архітектури, здатної зберігатися та адаптуватися до змін.

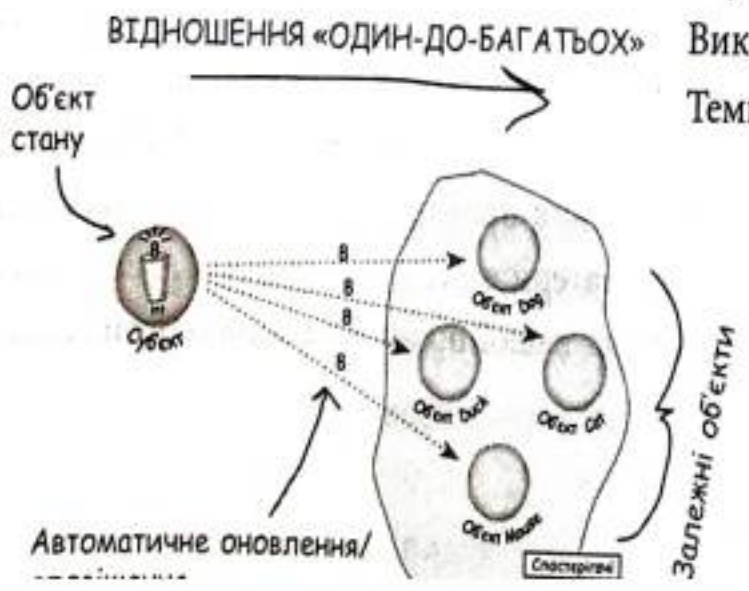


2 Патерн Спостерігач

Тримайте об'єкти в курсі подій

Не проґавте, коли відбувається щось цікаве! Наш наступний патерн сповіщає об'єкти про настання деяких подій, що можуть представляти для них інтерес. При цьому об'єкти навіть можуть вирішувати під час виконання, чи бажають вони і далі отримувати інформацію. Патерн Спостерігач надзвичайно корисний і належить до найбільш часто використовуваних патернів JDK. Наостанок у цьому розділі будуть розглянуті зв'язки типу «один-до-багатьох» і слабкі зв'язки (саме так, ми сказали — зв'язки). За допомогою патерну Спостерігач ви станете душею Спілки Патернів.





3 Патерн Декоратор

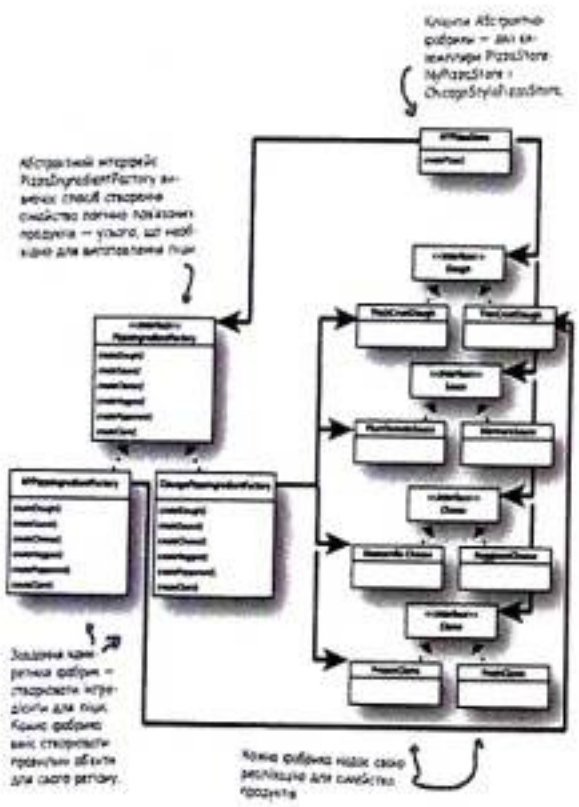
Оздоблення об'єктів

Цей розділ можна назвати «Погляд на архітектуру для любителів успадкування». Ми проаналізуємо типові зловживання у сфері успадкування, і ви навчитеся декорувати свої класи під час виконання з використанням різновидів композиції. Навіщо? Для того аби цей прийом дозволив вам наділити свої (чи чужі) об'єкти новими можливостями без модифікації коду основних класів.

4 Патерн Фабрика

Домашня ОО-випічка

Приготуйтеся зайнятися випічкою об'єктів у слабкопов'язаних ОО-архітектурах. Створення об'єктів аж ніяк не зводиться до простого виклику оператора **new**. Виявляється, створення екземплярів не завжди має здійснюватися відкрито; воно часто створює *проблеми сильного зв'язування*. Але ж ви цього не хочете, чи не так? Патерн Фабрика врятує вас від неприємних залежностей.



5 Патерн Одинак

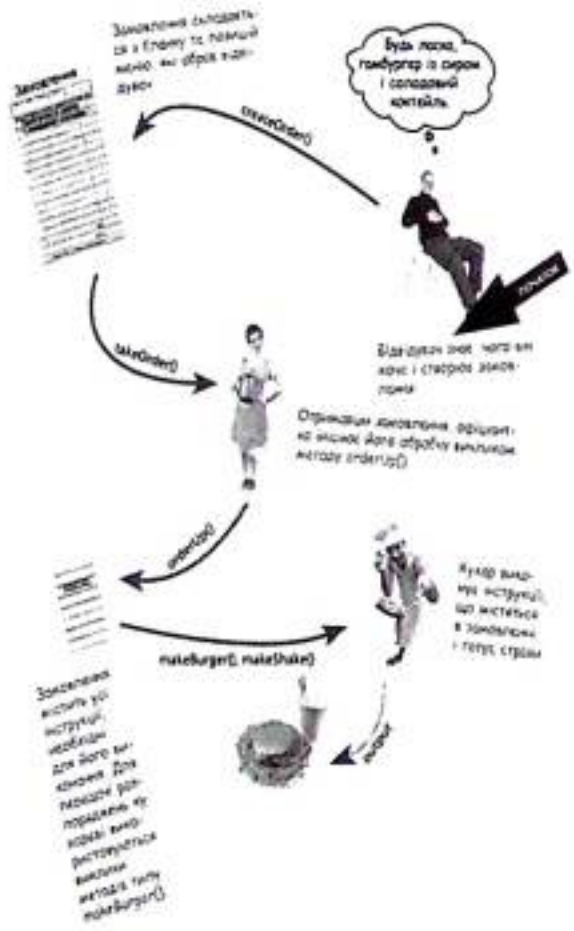
Унікальні об'єкти

Патерн Одинак спрямований на створення унікальних об'єктів, що існують лише в одному екземплярі. З усіх патернів Одинак має найпростішу діаграму класів; власне, уся діаграма складається з одного-єдиного класу! Однак не варто розслаблятися: незважаючи на простоту з погляду архітектури класів, у його реалізації криється чимало пасток. Отже, пристебніть паски!

6 Патерн Команда

Інкапсуляція виклику

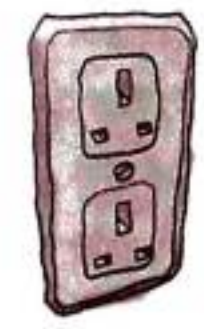
У цьому розділі ми виходимо на новий рівень інкапсуляції — цього разу будуть інкапсулюватися виклики методів. Саме так — викликаючому об'єкту не потрібно турбуватися про те, як саме виконуватимуться його запити. Він просто використовує інкапсульований метод для виконання свого завдання. З цими інкапсульованими викликами методів також можна робити деякі аж надто прості речі на кшталт їх логування або скасування операцій.

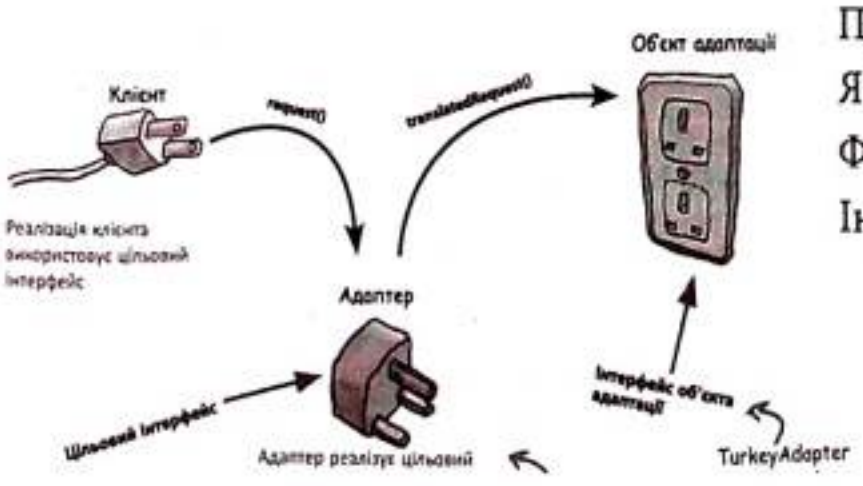


7 Патерни Адаптер і Фасад

Уміння пристосовуватися

У цьому розділі ми займемося всілякими неймовірними трюками — переважно будемо затикати круглі дірки квадратними пробками. Це неможливо, скажете ви? Тільки не з патернами проєктування. Пам'ятаєте патерн Декоратор? Ми пакували об'єкти, аби розширити їхні можливості. А у цьому розділі ми займемося пакуванням об'єктів з іншою метою: щоб імітувати інтерфейс, яким вони насправді не володіють. Навіщо? Із метою адаптування архітектури, розрахованої на один інтерфейс, для класу, що реалізує інший інтерфейс. Але і це ще не все: у цьому розділі буде описаний інший патерн, у якому об'єкти пакуються для спрощення їхнього інтерфейсу.





8 Патерн Шаблонний метод

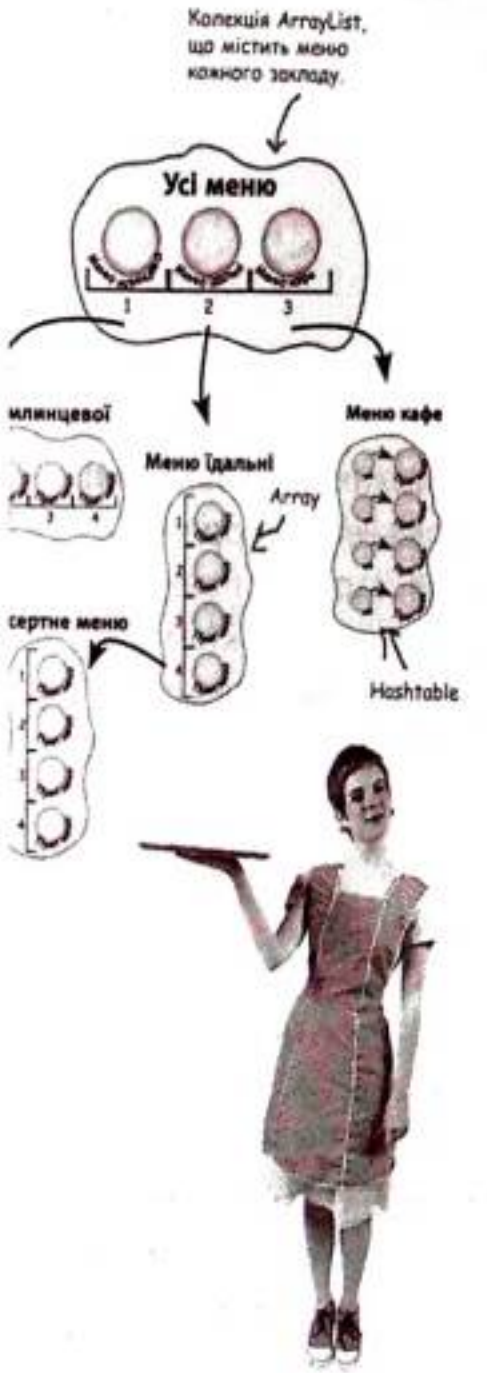
Інкапсуляція алгоритмів

Ми вже є знавцями механізмів інкапсуляції створення об'єктів, викликів методу, складних інтерфейсів, качок, піци... Що ж далі? Ми збираємося перейти до інкапсуляції блоків алгоритмів, аби підкласи могли підключатися до обробки будь-якої миті. Ми навіть дізнаємося про ще один принцип проєктування, натхнений практикою Голлівуда.

9 Патерни Ітератор і Компонувальник

Добре керовані колекції

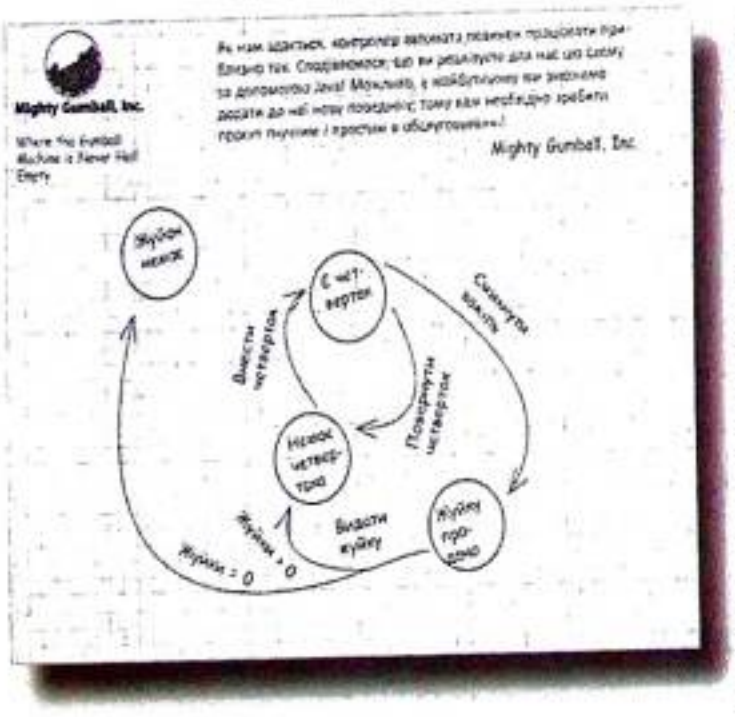
Існує чимало способів розмістити об'єкти в колекції. Розміщуйте об'єкти в контейнерах Array, Stack, List, Hashtable — як вам заманеться. Кожен із цих способів має свої переваги і недоліки. Але якоїсь миті клієнту заманеться піддати ітерації всі ці об'єкти, і коли це станеться, чи збираєтеся ви розкривати реалізацію колекції? Сподіваємося, ні! Це було б украй непрофесійно. Вам не варто ризикувати своєю кар'єрою. У цьому розділі ви дізнаєтеся, як надати клієнту механізм ітерації об'єктів без розкриття інформації про спосіб їх зберігання. Також тут будуть описані способи створення суперколекцій. А якщо цього недостатньо, ви дізнаєтеся дещо нове про функції об'єктів.



10 Патерн Стан

Стан речей

Маловідомий факт: патерни Стратегія і Стан — близнюки, розлучені при народженні. Як відомо, патерн Стратегія продовжував створення успішного бізнесу у сфері взаємозамінних алгоритмів. Заразом патерн Стан обрав, мабуть, найблагороднішій шлях, допомагаючи об'єктам контролювати власну поведінку за допомогою зміни їхнього внутрішнього стану. Він часто звертається до своїх клієнтів: «Просто повторюйте за мною: я достатньо добрий, я достатньо розумний і просто геніальний...»

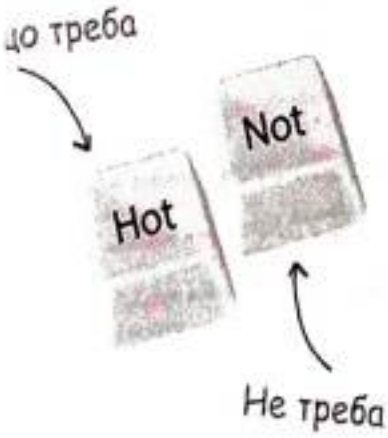


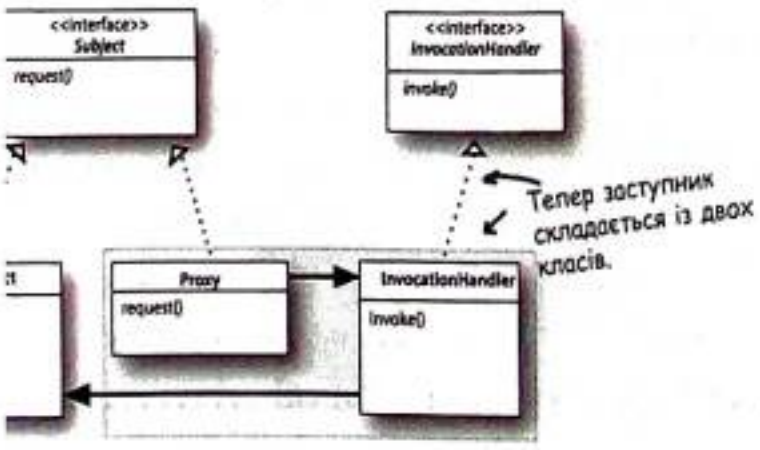
11

Патерн Заступник «Proxy»

Контроль доступу до об'єктів

Чи доводилося вам колись розігрувати сценку «хороший поліцейський — поганий поліцейський»? Ви — хороший поліцейський і виконуєте службові обов'язки люб'язно та у дружній манері, але не хочете марнувати час на дрібниці, надаючи послуги всім поспіль. Тому ви обзаводитесь «поганим поліцейським», який керує доступом до вас. Саме цим і займаються Заступники: вони керують доступом. Незабаром ви переконаєтеся, що існує безліч способів взаємодії Заступників з об'єктами, які вони обслуговують. Відомо, що Заступники пересилають Інтернетом цілі виклики методів, а іноді просто терпляче чекають на місці, зображуючи дуже ліниві об'єкти.

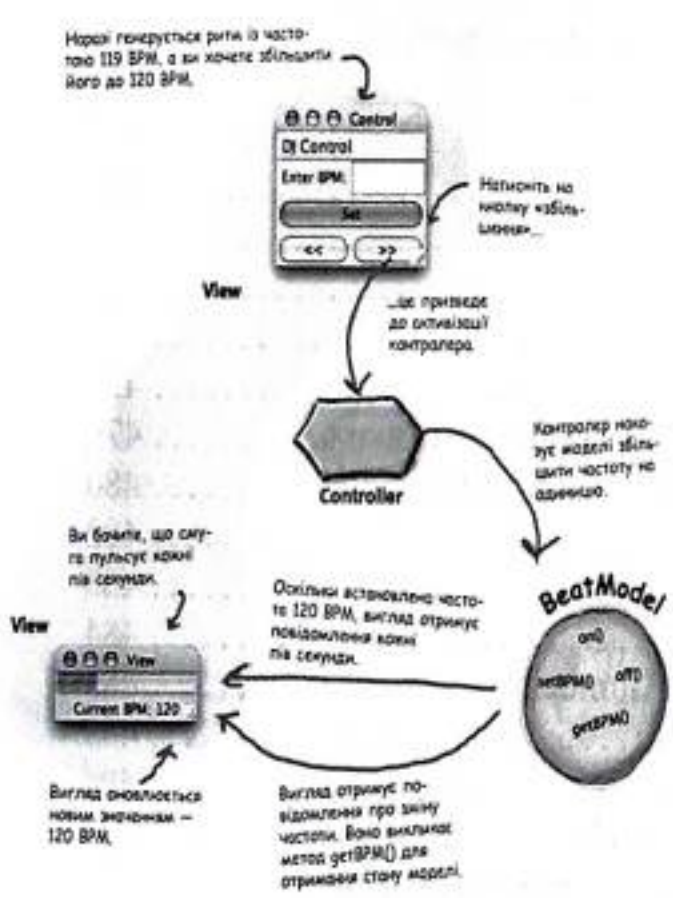




12 Складені патерни

Патерни патернів

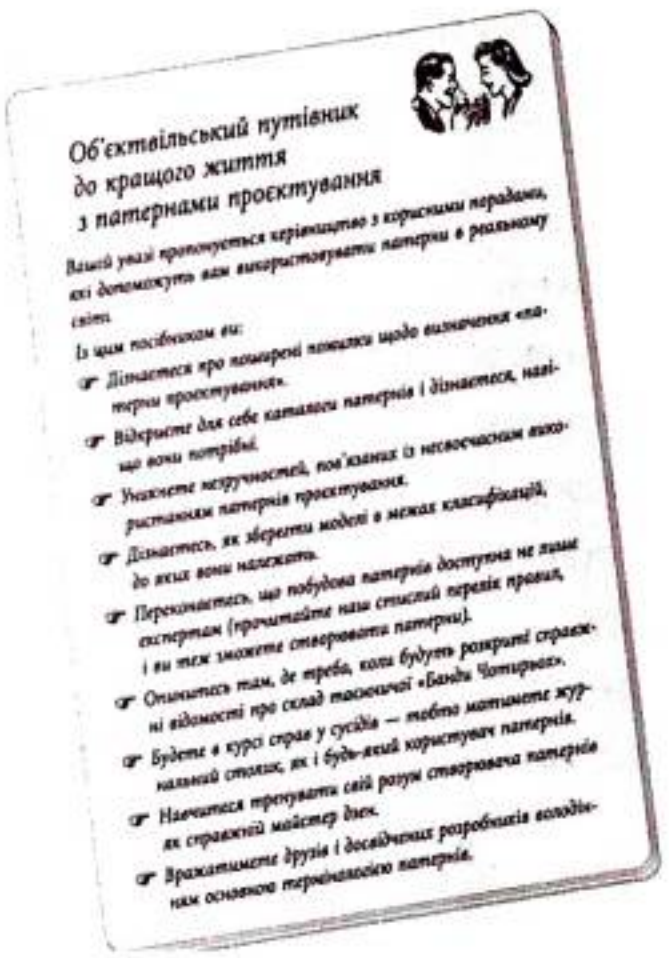
Хто міг би здогадатися, що патерни можуть працювати пліч-о-пліч? Ви вже були свідками запеклих суперечок у «Бесідах біля каміна» (і ви ще не бачили тих «смертельних двобоїв», які редактор змусив нас вилучити з книжки), тому хто б міг подумати, що мирне співіснування дійсно можливе! Хочете вірте, хочете — ні, але деякі з найпотужніших ОО-архітектур використовують комбінації кількох патернів. Будьте готові, аби перенести навички вашого патерна на наступний рівень: настає час для складання патернів.



13 Патерни для кращого життя

Патерни в реальному світі

Нарешті Ви готові поринути в яскравий новий світ, сповнений патернів проєктування. Але перш ніж відкрити ці двері до нових можливостей, бажано засвоїти деякі технічні тонкощі, із якими ви можете зіткнутися в реальному світі, оскільки справжнє життя трохи складніше, ніж в Об'єктвілі. На щастя, ви маєте путівник, що спростить вам перші кроки...



14 Додаток: Інші патерни

Не кожному судилося залишатися на піку популярності. За останнє десятиліття багато що змінилося. Із моменту виходу першого видання книжки «Design Patterns: Elements of Reusable Object-Oriented Software» розробники тисячі разів застосовували ці патерни в своїх проєктах. У цьому додатку подано повноцінні, першосортні патерни від «Банди Чотирьох» — вони використовуються не так часто, як інші патерни, що наводилися раніше. Однак ці патерни нічим не гірші, і якщо вони доречні у вашій ситуації — застосовуйте їх без жодних вагань. У цьому додатку ми постараємося дати загальне уявлення про суть цих патернів.