

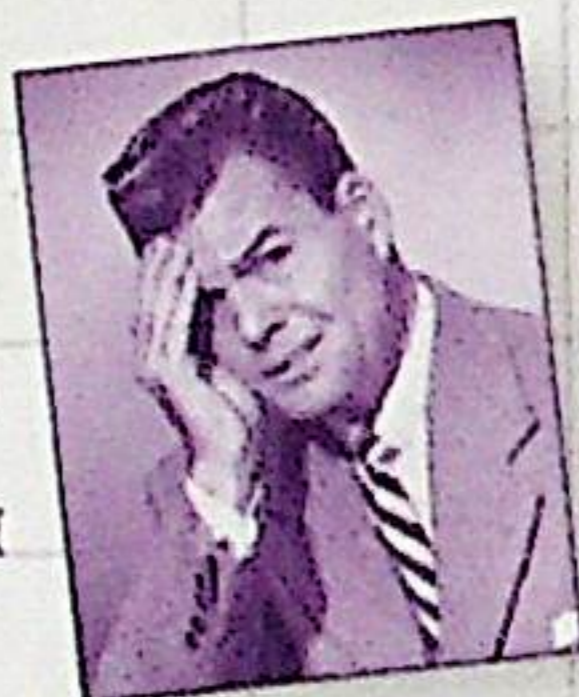
91572475

Оновлено
для Java 8

Head First Патерни проектування

Легкий для сприйняття довідник

Уникайте
безглузких
помилочок через
наслідування



Дізнайтеся секрети
гуру проектування
патернів



Дізнайтеся,
як Starbuzz Coffee
подвоїла ціну акцій
завдяки патернам
проектування



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



Оволодійте важливими
для вас патернами
максимально
ефективно



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

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

ВИДАВНИЦТВО
ФАБУЛА
#PRO

«Патерни проектування» — ваша книжка, якщо вам хоч колись доводилося стикатися з проблемами дизайну програмного забезпечення. Вам не доведеться «винаходити колесо» — просто скористайтеся зведеним до купи величезним досвідом розробників і можливістю використовувати найкращі практики. Ви дізнаєтеся, навіщо потрібні патерни, побачите, як вони виглядають і працюють у «дикій природі», а при звичаївшись 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 кодi. 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 проектування.

Зміст (коротко)

Вступ	25
1. Ласкаво просимо до світу патернів: вступ	37
2. Тримайте об'єкти в курсі подій: патерн Спостерігач	71
3. Оздоблення об'єктів: патерн Декоратор	113
4. Домашня ОО-випічка: патерн Фабрика	143
5. Унікальні об'єкти: патерн Одинак	201
6. Інкапсуляція виклику: патерн Команда	221
7. Уміння пристосовуватися: патерни Адаптер і Фасад	271
8. Інкапсуляція алгоритмів: патерн Шаблонний метод	309
9. Добре керовані колекції: патерни Ітератор і Компонувальник	347
10. Стан речей: патерн Стан	415
11. Контроль доступу до об'єктів: патерн Заступник «Proху»	459
12. Патерни патернів: складені патерни	525
13. Патерни в реальному світі: патерни для кращого життя	605
14. Додаток: інші патерни	639

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?

Для кого написано цю книжку?	26
Ми знаємо, про що ви думаєте	27
І ми знаємо, про що думає ваш мозок	27
Ми вважаємо читача «Head First» учнем	28
Мета пізнання: мислення про мислення	29
Ось що зробили МИ:	30
Ось що ви можете зробити, аби змусити свій мозок підкорятися	31
Прочитай мене	32
Технічні рецензенти	34
Подяки	35
Ще більше подяк	36

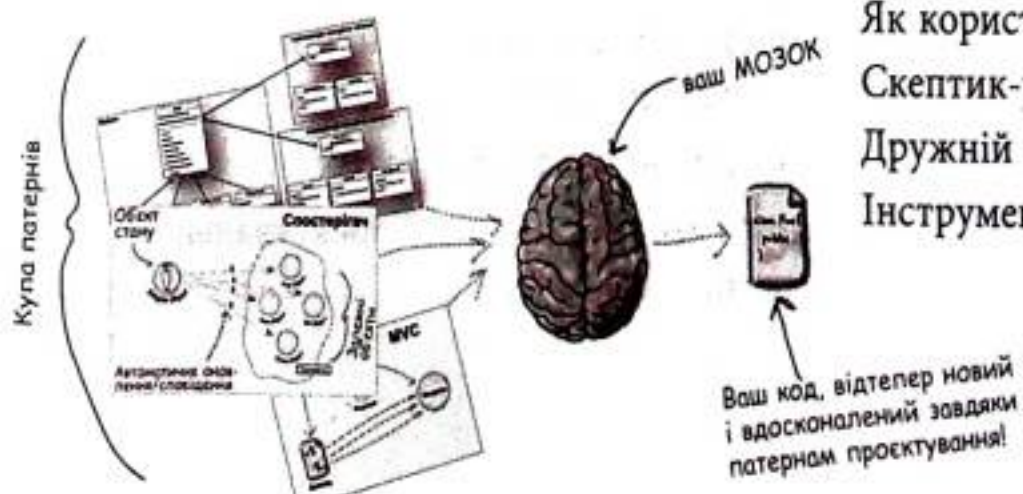
1 Знайомство з патернами проєктування

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.

Усе розпочалося з простого застосунку SimUDuck.....	38
Тепер нам потрібні качки, що будуть ЛІТАТИ.....	39
Але тут усе пішло шкереберть... ..	40
Джо розмірковує про успадкування.....	41
Як щодо інтерфейсу?.....	42
А якої ви думки про цю архітектуру?.....	42
А як би ви діяли на місці Джо?.....	43
Єдина константа в розробці програмного забезпечення.....	44
Зосереджуємося на проблемі.....	45
Відокремлюємо змінне від постійного.....	46
Проєктування поведінки качок.....	47
Реалізація поведінки качок.....	49
Інтеграція поведінки з класом Duck.....	51
Тестування коду Duck.....	54
Динамічна зміна поведінки.....	56
Інкапсуляція поведінки: загальна картина.....	58
Відношення «МІСТИТЬ» бувають зручнішими ніж відношення «Є».....	59
До речі, про патерни... ..	60
Підслухане в місцевій їдальні.....	62
Підслухане в сусідньому офісі.....	63
Сила загальної номенклатури патернів.....	64
Як користуватися патернами проєктування?.....	65
Скептик-розробник програм.....	66
Дружній гуру патернів.....	66
Інструменти для вашої проєктувальної панелі.....	68

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



2

Патерн Спостерігач

Тримайте об'єкти в курсі подій

Не прогавайте, 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.

OO КОНЦЕПЦІЇ

Абстракція
Інкапсуляція

OO ПРИНЦИПИ

Інкапсулюйте те, що змінюється.

Надавайте перевагу композиції успадкування.

Програмуйте на рівні інтерфейсів, а не впровадження.

Прагніть до проектів із вільними зв'язками між взаємодіючими об'єктами.

фіз
мання

Огляд застосунку для моніторингу погоди
Weather Monitoring73

Знайомство з патерном Спостерігач78

Видавці + Передплатники = патерн Спостерігач79

Один день із життя патерна Спостерігач80

П'ятихвилинна драма: суб'єкт для спостереження82

Минуло два тижні...84

Визначення патерна Спостерігач.....85

Патерн Спостерігач визначено: діаграма класів86

Сила слабких зв'язків87

Розмова в офісі.....89

Реалізація Weather Station91

Реалізація інтерфейсу Subject у WeatherData.....92

Настав час перейти до візуальних елементів.....93

Запуск метеостанції.....94

Використання вбудованого в Java патерна Спостерігач...98

Як вбудована підтримка патерна працює в Java.....99

Переробляємо Weather Station за допомогою вбудованої підтримки.....101

Виконання нового коду104

Темна сторона java.util.Observable.....105

Інші приклади використання патерна Спостерігач в JDK106

Нарешті — програмний код.....107

Інструменти для вашої проектувальної панелі.....109



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.

Ласкаво просимо до «Starbuzz Coffee»!	114
Принцип відкритості/закритості	120
Знайомство з патерном Декоратор	122
Побудова замовленого напою за допомогою декораторів	123
Визначення патерна Декоратор	125
Прикрашаємо наші напої	126
Розмова в офісі	127
Навчання бариста	128
Пишемо код для Starbuzz	129
Програмуємо класи напоїв	130
Програмування заправок	131
Готуємо каву	132
Декоратори в реальному світі: ввід/вивід у мові Java	134
Декорування класів java.io	135
Написання власного декоратора вводу/виводу Java	136
Тестування декоратора вводу/виводу	137
Інструменти для вашої проектувальної панелі	139

Раніше я думав, що справжні чоловіки використовують лише розподіл на підкласи. Це тривало доти, доки я не усвідомив можливості динамічного розширення на стадії виконання, а не під час компіляції. Подивіться, яким я став зараз!

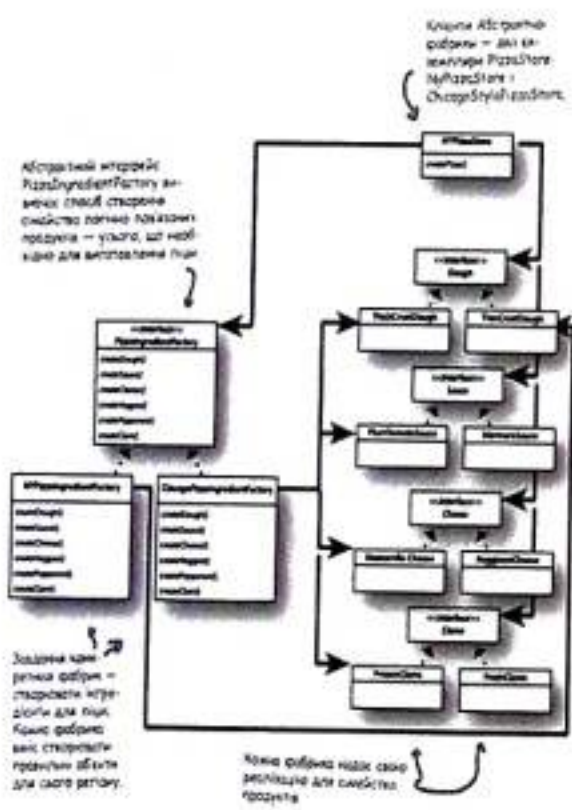


4

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.



- Визначення аспектів, що варіюються146
- Додання нових типів піци147
- Інкапсуляція створення об'єктів148
- Побудова простої фабрики піци.....149
- Переробка класу PizzaStore.....150
- Визначення простої фабрики151
- Франчайзинг піцерії.....152
- Інфраструктура для піцерії154
- Дозвіл підкласам вирішувати155
- Давайте створимо PizzaStore157
- Оголошення фабричного методу159
- Не вистачає лише одного: ПІЦИ!.....162
- Ви вже зачекалися. Настав час для піци!.....164
- Настав час познайомитися з патерном Фабричний метод.....165
- Інша точка зору: паралельні ієрархії класів166
- Визначення патерна Фабричний метод168
- PizzaStore із сильними залежностями.....171
- Розгляньмо залежності між об'єктами.....172
- Принцип інверсії залежностей173
- Застосування принципу174
- Інвертування мислення.....176
- Кілька порад щодо застосування принципу.....177
- Тим часом повертаємося до піцерії...178
- Сімейства інгредієнтів.....179
- Побудова фабрик інгредієнтів.....180
- Побудова фабрики інгредієнтів для Нью-Йорка.....181
- Переробляємо класи піци...183
- Переробка класів піци триває...184
- Повертаємося до наших піцерій.....186
- Чого ми досягли?.....187
- Більше піци для Ітана і Джоела!.....188
- Визначення патерна Абстрактна фабрика.....190
- Порівняння патернів Фабричний метод і Абстрактна фабрика194

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!



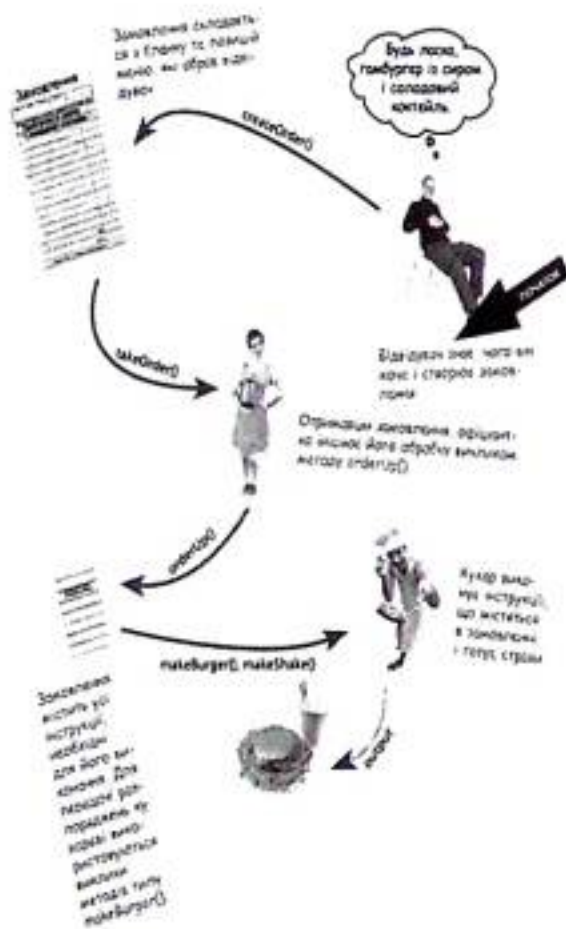
Розбір класичної реалізації патерна Одинак	205
Шоколадна фабрика	207
Визначення патерна Одинак.....	209
Маємо проблему.....	210
Уявіть, що ви — Java Virtual Machine (JVM)	211
Вирішення проблеми багатопотокового доступу.....	212
Чи можна вдосконалити багатопотокову реалізацію?	213
Тим часом на шоколадній фабриці... ..	215
Інструменти для вашої проектувальної панелі.....	218

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.



Халявне залізо! Давайте перевіримо пульт дистанційного керування.....	223
Ознайомлення з класами постачальників	224
Розмова в офісі	225
А тим часом у їдальні... або Стислий вступ до патерна Команда.....	227
Розглянемо ці взаємодії трохи детальніше... ..	228
Ролі та обов'язки в їдальні Об'єктивіля.....	229
Від їдальні до патерна Команда	231
Наш перший об'єкт команди	233
Використання об'єкта команди	234
Створення простого тесту для користування пультом дистанційного керування	234
Визначення патерна Команда	236
Визначення патерна Команда: Діаграма класів	237
Пов'язування команд зі слотами	239
Реалізація пульта дистанційного керування.....	240
Реалізація команд.....	241
Перевіряємо пульт дистанційного керування в роботі.....	242
Настав час скласти документацію... ..	245
Що, власне, слід зробити?	246
Настав час протестувати кнопку скасування!	249
Використання стану для реалізації скасування.....	250
Реалізація скасування в командах керування вентилятором	251
Переходимо до тестування вентилятора.....	252
Подальше тестування	253
На кожному пульті має бути Режим Вечірки!	254
Використання макрокоманд	255
Патерн Команда означає безліч класів команд	258
Спрощення коду RemoteControl із допомогою лямбда-виразів ...	259
Ще більше спрощення завдяки посиланням на методи	261
Тест-драйв команд із використанням лямбда-виразів.....	262
Додаткові можливості патерна Команда: черги запитів	265
Додаткові можливості використання патерна Команда: логування запитів.....	266
Інструменти для вашої проектувальної панелі	267

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 проектування. 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.

Британська розетка



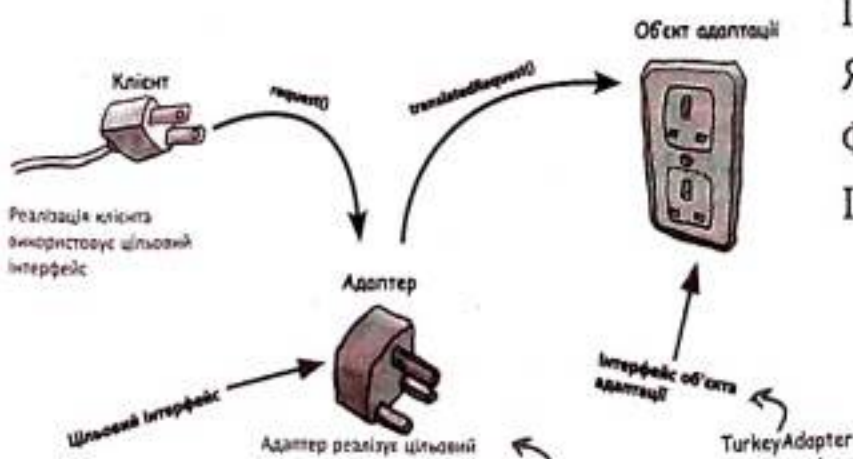
Адаптер



Стандартна вилка



Адаптери навколо нас	272
Об'єктно-орієнтовані адаптери.....	273
Якщо він ходить, як качка, і крякає, як качка, то може бути індичкою, запакованою в адаптер качки....	274
Тестування адаптера	276
Як працює патерн Адаптер	277
Визначення патерна Адаптер.....	279
Адаптери об'єктів і класів	280
Реальні адаптери	284
Адаптація Перерахування до Ітератора.....	285
А зараз про щось інше.....	290
Домашній, мій милий домашній кінотеатр.....	291
Перегляд фільму (складний спосіб).....	292
Світло, камера, фасад!	294
Побудова фасаду для домашнього кінотеатру	297
Реалізація спрощеного інтерфейсу	298
Перегляд фільму (простий спосіб).....	299
Визначення патерна Фасад.....	300
Принцип мінімальної поінформованості	301
Як НЕ заводити «друзів» і впливати на об'єкти.....	302
Фасад і принцип мінімальної поінформованості	305
Інструменти для вашої проектувальної панелі.....	306



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 проектування, натхнений практикою Голлівуда.



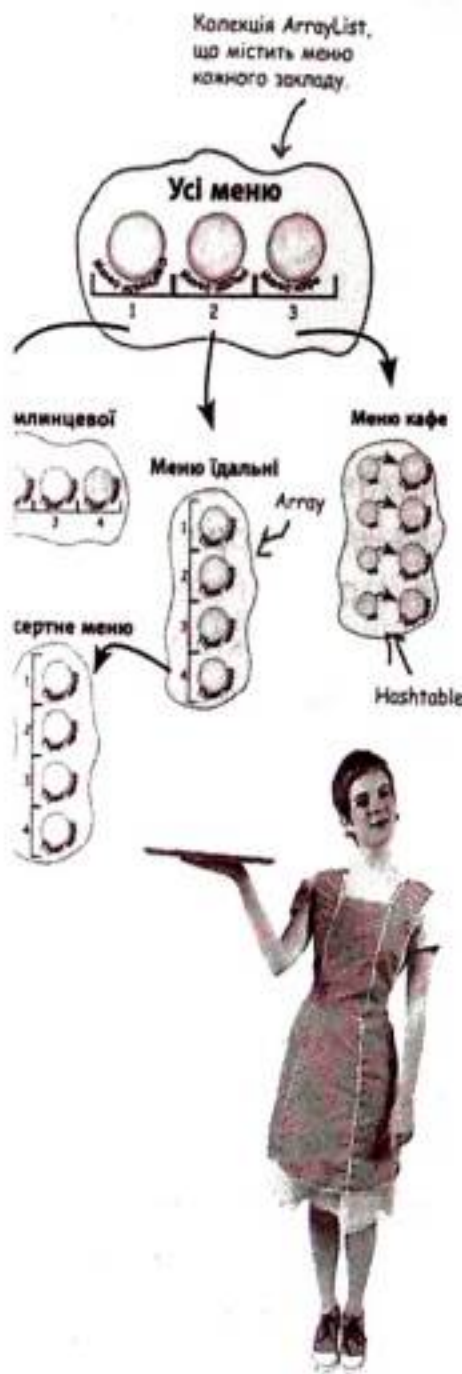
Час додати трохи кофеїну.....	310
Приготування деяких класів кави та чаю (мовою Java).....	311
А зараз — чай... ..	312
Сер, чи можу я абстрагувати ваші Coffee і Tea?.....	314
Продовжуємо проектування.....	315
Абстрагування prepareRecipe().....	316
Що ми зробили?.....	319
Патерн Шаблонний метод.....	320
Давайте приготуємо чай... ..	321
Що надає нам Шаблонний метод?.....	322
Визначення патерна Шаблонний метод.....	323
Перехоплення в Шаблонному методі.....	326
Використання перехоплювачів.....	327
Пробний запуск коду.....	328
Голлівудський принцип.....	330
Голлівудський принцип і Шаблонний метод.....	331
Шаблонні методи в природних умовах.....	333
Сортування за Шаблонним методом.....	334
Сортуємо качок.....	335
Що робить метод compareTo()?.....	335
Порівняння об'єктів Duck.....	336
Давайте відсортуємо кілька об'єктів Duck.....	337
Виготовлення сортувальної машини для Duck.....	338
Розгойдування в рамках.....	340
Аплети.....	341
Інструменти для вашої проектувальної панелі.....	344

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 розкрити 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.



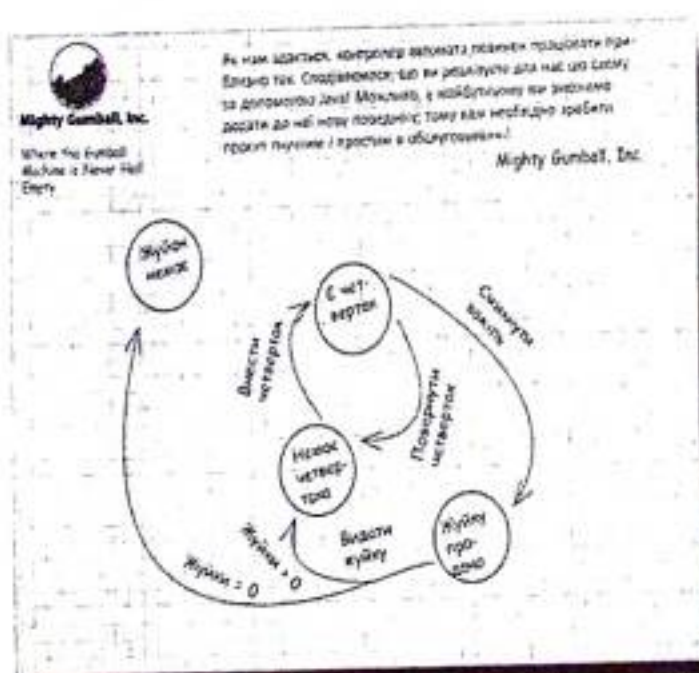
Сенсація: об'єктвільські їдальня і млинцева об'єднуються!	348
Перевіряємо елементи меню	349
Реалізація меню Лу та Мела	350
Які проблеми створює наявність двох різних реалізацій меню?	352
Що далі?	354
Чи можемо ми інкапсулювати перебирання?	356
Зустрічайте патерн Ітератор!	358
Додання ітератора до DinerMenu	359
Переробка DinerMenu за допомогою ітератора	360
Виправлення коду офіціантки	361
Тестування нашого коду	362
Що ми зробили?	363
Що ми наразі маємо... ..	364
Упроваджуємо вдосконалення... ..	365
Удосконалення з java.util.Iterator	366
Роботу майже завершено	367
Що нам це дає?	368
Визначення патерна Ітератор	369
Єдина відповідальність	372
Знайомство з класом CafeMenu	375
Переробка коду CafeMenu	376
Додання CafeMenu до класу Waitress	377
Сніданок, обід і вечеря	378
Що ми зробили?	379
Ми відокремили офіціантку від реалізації... ..	379
...І розширили Waitress	380
Але це ще не все!	380
Ітератори і колекції	381
Чи готова офіціантка до прайм-тайму?	383
А коли ми вже тріумфували перемогу... ..	385
Що нам потрібно?	386
Визначення патерна Компонувальник	388
Проектування меню з патерном Компонувальник	391

Реалізація компонентів меню	392
Реалізація MenuItem	393
Реалізація комбінаційного меню	394
Готуємося до тестування... ..	396
Тепер можна тестувати... ..	397
Готуємося до тестового запуску... ..	398
Повернення до ітератора	400
Ітератор CompositeIterator	401
Ітератор Null	404
Дайте мені веганське меню!	405
Спільна магія патернів Ітератор і Компонувальник	406
Інструменти для вашої проектувальної панелі.....	411

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, найблагородніший 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...»



Java-запобіжники	416
Розмова в офісі	417
Машина стану 101	418
Написання коду	420
Внутрішнє тестування	422
Ви здогадувалися, що так відбудеться... запит на зміну!	424
Безладний СТАН справ... ..	426
Нова архітектура.....	428
Визначення інтерфейсу State і класів	429
Реалізація класів станів	431
Переробка класу GumballMachine	432
А тепер — повний код класу GumballMachine.....	433
Реалізація більшої кількості станів.....	434
Що ми зробили досі.....	437
Патерн Стан визначено!	440
Нам ще треба реалізувати гру «1 із 10».....	443
Завершення гри.....	444
Демо для генерального директора «Mighty Gumball, Inc.»....	445
Перевірка розумності... ..	447
Мало не забули!	450
Інструменти для вашої проектувальної панелі.....	453

11

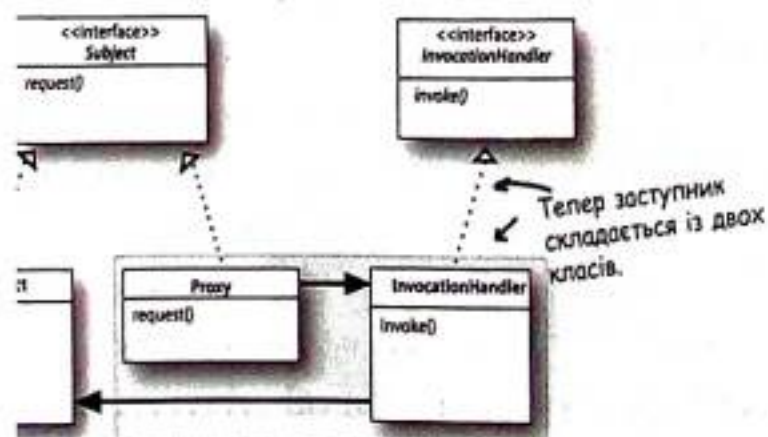
Патерн Заступник «Proxy»

Контроль доступу до об'єктів

Чи доводилося вам колись розігравати 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.



Програмування моніторингу.....	461
Тестування моніторингу	462
Роль віддаленого заступника.....	464
Включення віддаленого заступника в код моніторингу	
GumballMachine	466
Дистанційні виклики методів.....	467
Java RMI: загальна картина	470
Створення віддаленої служби.....	471
Повний код серверної частини.....	475
Як клієнт отримує об'єкт заглушки?	476
Повний клієнтський код.....	478
Віддалений заступник для автомата із жувальною gumкою.....	479
Перетворення GumballMachine на віддалений сервіс	480
Реєстрація в реєстрі RMI.....	482
А тепер — клієнт GumballMonitor.....	483
Тестова програма для монітора	484
Нова демонстрація для CEO «Mighty Gumball»	485
Визначення патерна Proxy (Заступник)	489
Будьте готові до віртуального Proxy.....	491
Відображення обкладинок компакт-дисків.....	492
Проектування перегляду обкладинки CD із віртуальним заступником.....	493
Клас ImageProxy	494
Тестування програми перегляду обкладинок.....	498
Що ми зробили?	499
Створення захисного заступника засобами Java API.....	503
Служба знайомств в Об'єктивлі	504
Реалізація PersonBean	505



П'ятихвилинна драма: захист клієнтів.....	507
Загальна картина: динамічний заступник для PersonBean.....	508
Тестування служби знайомств	513
Запуск коду.....	514
Зоопарк Proxy.....	516
Інструменти для вашої проектувальної панелі	518
Код для перегляду обкладинки CD.....	521

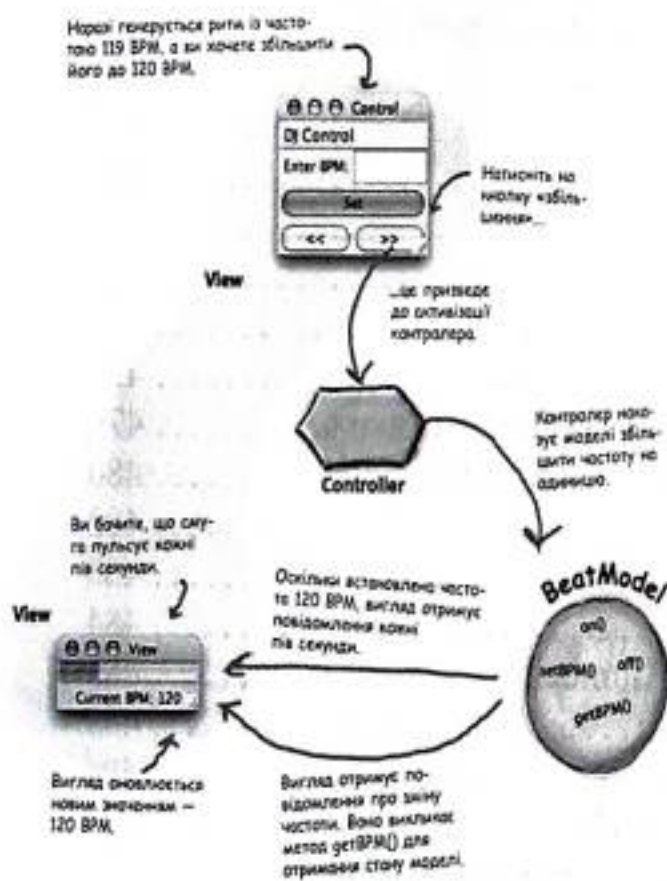
12

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.



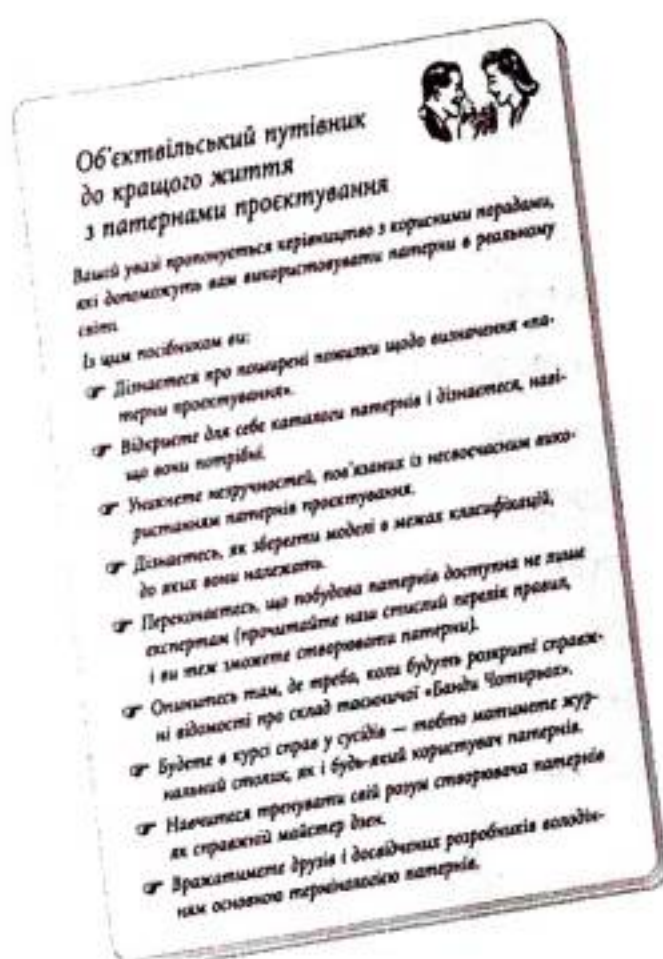
Працюючи разом.....	526
І знову качки.....	527
Що ми зробили?	549
Діаграма класів качиним оком	550
MVC: король складних патернів	552
Знайомство з Model-View-Controller	557
Детальніший погляд.....	558
MVC із точки зору патернів	560
Використання MVC для контролю за ритмом	562
Складаємо все разом	564
Будівництво фрагментів	565
Подивимось на конкретний клас BeatModel.....	566
Вигляд.....	567
Реалізація вигляду	568
Контролер	570
Складаємо все разом	572
Дослідження стратегії	573
Адаптація моделі.....	574
Можна переходити до HeartController	575
Тестовий запуск.....	576
MVC і Web.....	577
Модель 2: діджей з мобільного	579
Крок 1: Модель.....	580
Крок 2: Сервлет-контролер	580
Наразі нам потрібен вигляд!	582
Тестування Моделі 2... ..	583
Патерни проектування і Модель 2.....	585
Спостерігач	585
Стратегія	586
Компонувальник	586
Інструменти для вашої проектувальної панелі	588

13

Патерни для кращого життя

Патерни в реальному світі

Нарешті Ви готові поринути в яскравий новий світ, сповнений патернів проектування. 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'єктивілі. На щастя, ви маєте путівник, що спростить вам перші кроки...

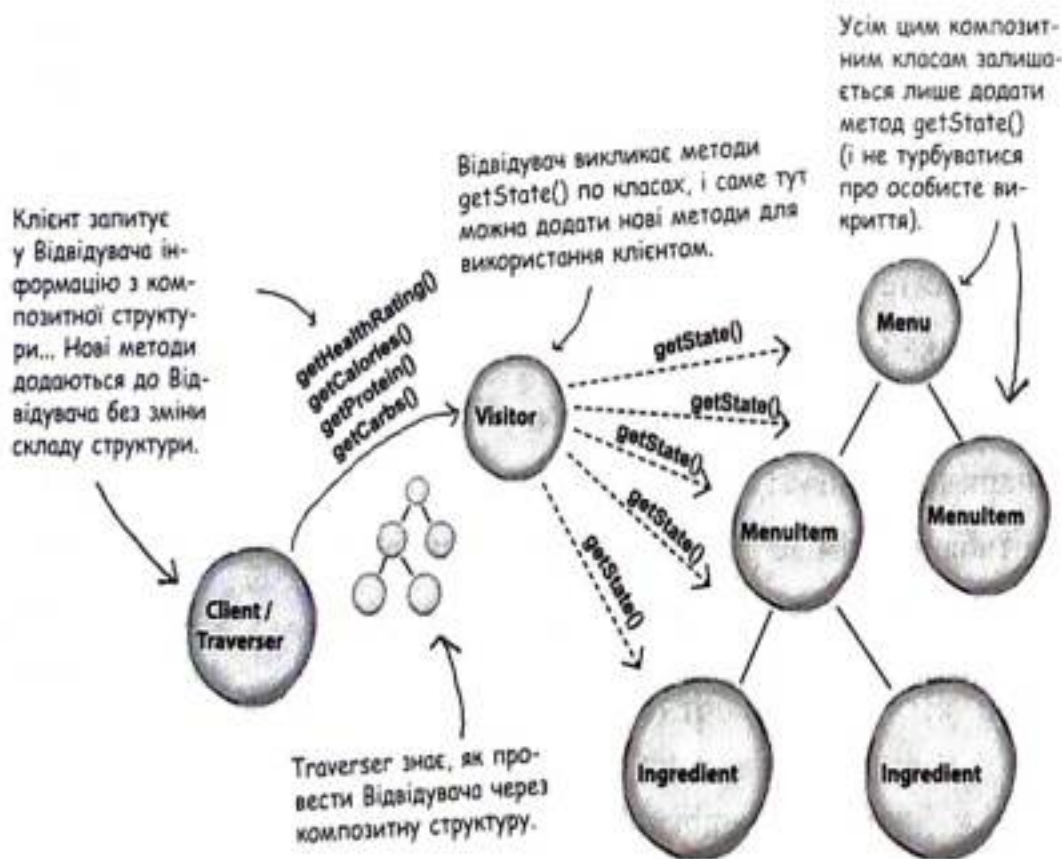


Визначення патерна проектування	607
Детальніше про визначення патерна проектування	609
Нехай сила буде з тобою!	610
Отже, ви хочете створювати патерни.....	615
Класифікація патернів проектування.....	617
Мислити патернами	622
Ваш розум у патернах.....	625
Не забувайте про силу єдиної номенклатури	627
Прогулянка Об'єктивілем із «Бандою Чотирьох»	629
Ваша подорож тільки розпочалася.....	630
Зоопарк патернів	632
Боротьба зі злом за допомогою анти-патернів	634
Інструменти для вашої проектувальної панелі	636
Залишаючи Об'єктивіль.....	637

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 — застосуйте їх без жодних вагань. У цьому додатку ми постараємося дати загальне уявлення про суть цих патернів.



Міст	640
Будівельник	642
Ланцюжок обов'язків	644
Легковаговик	646
Інтерпретатор	648
Посередник	650
Знімок	652
Прототип	654
Відвідувач	656

Показчик	659
----------------	-----