

*Присвячується «Банді Чотирьох»,
чия прозорливість і майстерність
у формулюванні та описі патернів проектування
назавжди змінили галузь проектування програмних архітектур
і поліпили життя розробників у всьому світі.*

[Купити книгу на сайті kniga.biz.ua >>>](http://kniga.biz.ua)

Про авторів цієї книжки

Ерік Фрімен



Ерік, за словами Кеті Сьєрра, співавторки серії *Head First*, є «однією з тих рідкісних осіб, які добре розбираються в мові, практиці застосування та культурі в найрізноманітніших галузях, техно-хіпстер, віцепрезидент, інженер, аналітик».

Ерік пропрацював майже десять років на посаді технічного директора *Disney Online & Disney.com* у *The Walt Disney Company*. Зараз Ерік присвячує свій час *WickedlySmart* — стартапу, заснованому спільно з Елізабет.

За освітою Ерік є фахівцем із комп'ютерних технологій; він займався дослідженнями разом із корифеєм галузі Девідом Гелернтером під час роботи над дисертацією в Єльському університеті. Його дисертація стала однією з фундаментальних праць в галузі альтернатив для інтерфейсів, що реалізують метафору робочого столу, а також першою реалізацією потоків активності — концепції, розробленої ним спільно з доктором Гелернтером.

У вільний час Ерік серйозно займається музикою; останній проект Еріка, створений разом із одним з піонерів напрямку «ембієнт» Стівом Роучем, доступний в магазині *App Store* під назвою *Immersion Station*.

Ерік живе з дружиною і маленькою донькою в Остіні (штат Техас). Його донька часто навідується до студії Еріка; їй подобається крутити ручок синтезаторів.

Пишіть йому за адресою eric@wickedlysmart.com або відвідайте сайт <https://www.wickedlysmart.com/>.

Елізабет Робсон



Елізабет — програмістка, письменниця і викладачка. Вона закохана в свою роботу ще з часів навчання в Єльському університеті, де отримала ступінь магістра в галузі комп'ютерних технологій, а також створила паралельну візуальну мову програмування і програмну архітектуру.

Елізабет брала участь в створенні популярного сайту *Ada Project* — одного з перших, що допомагають жінкам знайти інформацію про роботу та освіту в галузі комп'ютерних технологій.

Вона стала одним із засновників *WickedlySmart* — компанії, що працює в сфері онлайн-освіти на базі веб-технологій. Тут вона створює книжки, статті, відеокурси тощо. Раніше Елізабет була на посаді директорки зі спеціальних проектів у видавничій компанії *O'Reilly Media* і розробляла семінари та курси дистанційного навчання в різних технічних галузях, які допомагають людям розібратися в новітніх технологіях. До приходу в *O'Reilly* Елізабет працювала в *The Walt Disney Company*, де керувала дослідженнями і розробками в сфері цифрових мультимедійних технологій.

Коли Елізабет не сидить за комп'ютером, вона займається велоспортом або веслуванням, фотографує та готує вегетаріанські страви.

Із нею можна зв'язатися за адресою beth@wickedlysmart.com або відвідати її блог за адресою <http://elisabethrobson.com>.

Творці серії Head First (і співавтори цієї книжки)

Кеті Сьєрра



Берт Бейтс



Кеті зацікавилася теорією навчання ще відтоді, коли почала створювати ігри (для *Virgin*, *MGM* й *Amblin*). Велика частина формату *Head First* була розроблена нею під час викладання курсу *New Media Authoring* для програми *UCLA Extension's Entertainment Studies*. Останнім часом вона готує фахівців для *Sun Microsystems*, навчає інструкторів Java мистецтву викладання новітніх Java-технологій; брала участь в розробці декількох сертифікаційних іспитів *Sun Microsystems*. Разом із Бертом Бейтсом використовувала концепції *Head First* для навчання тисяч розробників. Кеті є засновником сайту javaranch.com, який у 2003 і 2004 роках завойовував нагороду *Jolt Cola Productivity Award* від журналу *Software Development*. Іноді вона викладає Java на курсах *Java Jam Geek Cruise* (geekcruises.com).

Любить: бігати, кататися на лижах, скейтбординг, гратися з власною ісландською конячкою, а також дива науки. Не любить: ентропію.

Часто буває на [javaranch](http://javaranch.com), іноді веде блог на seriouspony.com.

Пишіть їй за адресою kathy@wickedlysmart.com.

Берт — досвідчений програміст і проектувальник. Його десятирічні дослідження в сфері штучного інтелекту викликали в нього підвищений інтерес до теорії і технології навчання. Відтоді він займається підвищенням кваліфікації програмістів. Останнім часом очолював групу розробки сертифікаційних іспитів із мови *Java* для корпорації *Sun*.

Перше десятиліття своєї кар'єри програміста Берт подорожував по всьому світі, допомагаючи ЗМІ — таким, як *Radio New Zealand*, *Weather Channel* і *Arts & Entertainment Network (A&E)*. Одним із його улюблених проектів того часу стала побудова повного імітатора залізничної мережі для *Union Pacific Railroad*.

Берт — завзятий прихильник гри в го і давно працює над її програмуванням. Непогано грає на гітарі та банджо.

Шукайте його на [javaranch](http://javaranch.com) або на го-сервері *IGS*. Ви також можете написати йому листа за адресою terrapin@wickedlysmart.com.

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 «Роуху»..... 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 проектування. 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 проектування?

- 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

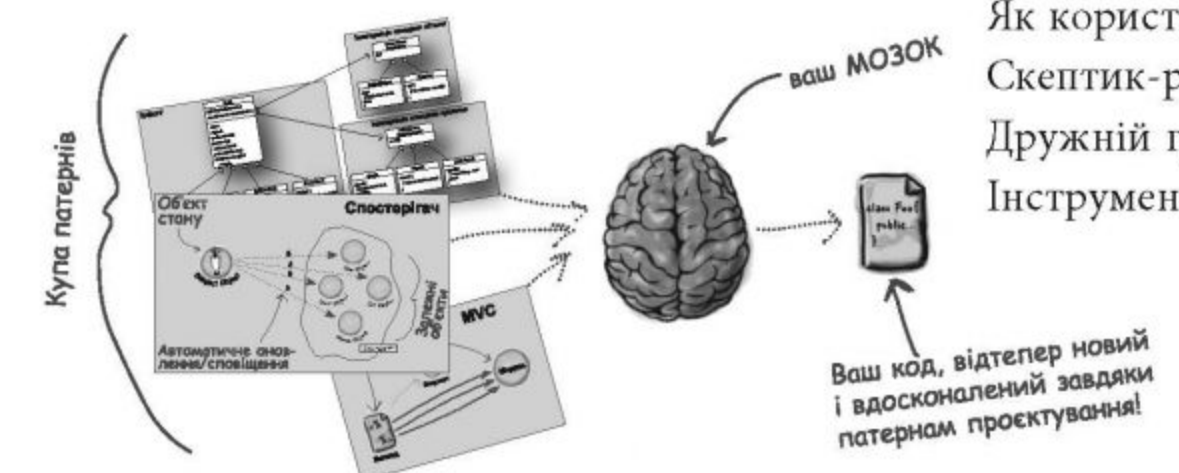
Знайомство з патернами проектування

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 проектування, познайомимося з ключовими принципами ОО-проектування 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
- Проектування 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 проектування?..... 65
- Скептик-розробник програм..... 66
- Дружній гуру патернів 66
- Інструменти для вашої проектувальної панелі 68

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



2 Патерн Спостерігач Тримайте об'єкти в курсі подій

Не прогайте, 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.

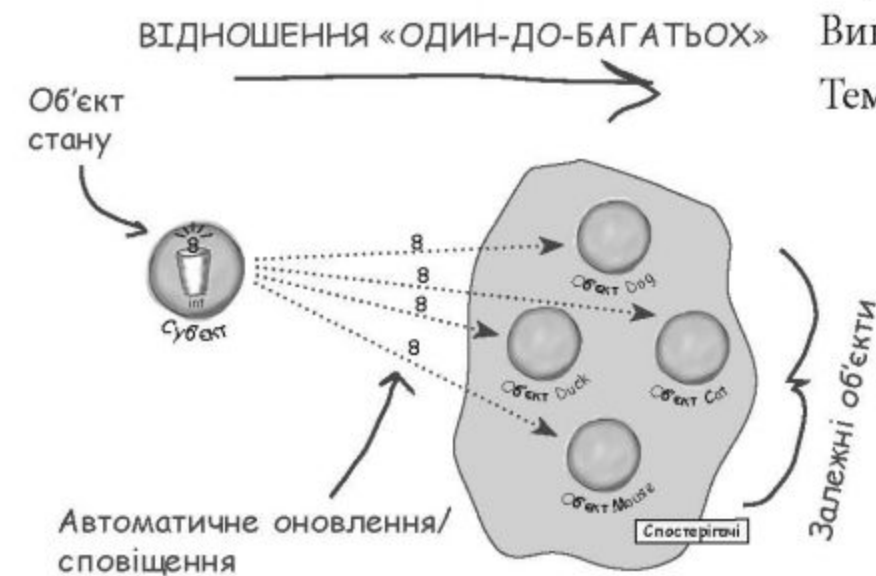
OO КОНЦЕПЦІЇ

Абстракція
Інкапсуляція
Поліморфізм
Спільне походження

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.

Огляд застосунку для моніторингу погоди Weather Monitoring	73
Знайомство з патерном Спостерігач	78
Видавці + Передплатники = патерн Спостерігач	79
Один день із життя патерна Спостерігач	80
П'ятихвилинна драма: суб'єкт для спостереження	82
Минуло два тижні... ..	84
Визначення патерна Спостерігач.....	85
Патерн Спостерігач визначено: діаграма класів.....	86
Сила слабких зв'язків	87
Розмова в офісі.....	89
Реалізація Weather Station	91
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 коду	104
Темна сторона java.util.Observable.....	105
Інші приклади використання патерна Спостерігач в JDK.....	106
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.

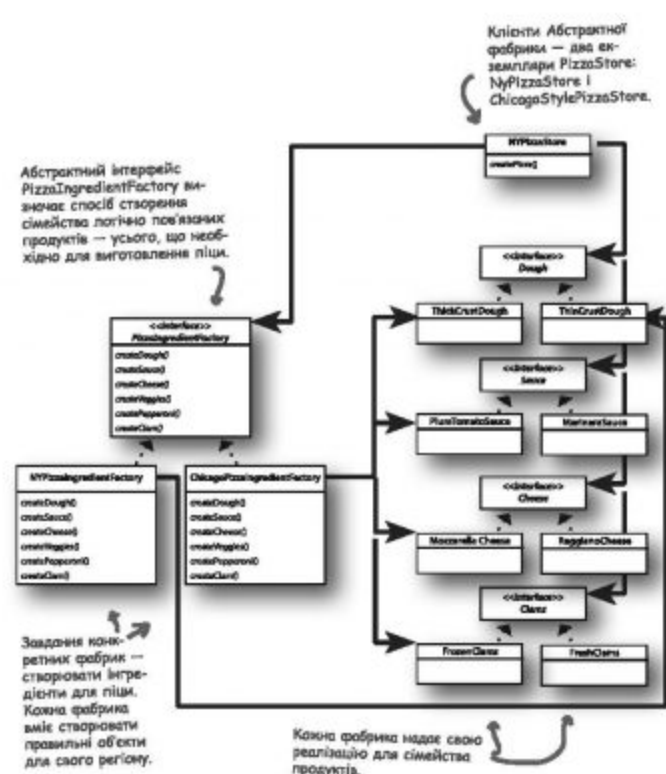
Ласкаво просимо до «Starbuzz Coffee»!	114
Принцип відкритості/закритості	120
Знайомство з патерном Декоратор.....	122
Побудова замовленого напою за допомогою декораторів.....	123
Визначення патерна Декоратор.....	125
Прикрашаємо наші напої	126
Розмова в офісі.....	127
Навчання бариста	128
Пишемо код 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 варіюються 146
- Додання нових типів піци 147
- Інкапсуляція створення об'єктів 148
- Побудова простої фабрики піци..... 149
- Переробка класу PizzaStore..... 150
- Визначення простої фабрики 151
- Франчайзинг піцерії..... 152
- Інфраструктура для піцерії 154
- Дозвіл підкласам вирішувати 155
- Давайте створимо PizzaStore 157
- Оголошення фабричного методу 159
- Не вистачає лише одного: ПІЦЦІ!..... 162
- Ви вже зачекалися. Настав час для піци!..... 164
- Настав час познайомитися з патерном Фабричний метод..... 165
- Інша точка зору: паралельні ієрархії класів 166
- Визначення патерна Фабричний метод 168
- PizzaStore із сильними залежностями..... 171
- Розгляньмо залежності між об'єктами..... 172
- Принцип інверсії залежностей 173
- Застосування принципу 174
- 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 фабрика 194
- Інструменти для вашої проектувальної панелі 196

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.

Халлявне залізо! Давайте перевіримо пульт дистанційного керування.....223

Ознайомлення з класами постачальників.....224

Розмова в офісі225

А тим часом у їдальні... або Стиглий вступ до патерна Команда..... 227

Розглянемо ці взаємодії трохи детальніше... ..228

Ролі та обов'язки в їдальні Об'єктівіля.....229

Від їдальні до патерна Команда231

Наш перший об'єкт команди233

Використання об'єкта команди234

Створення простого тесту для користування пультом дистанційного керування234

Визначення патерна Команда236

Визначення патерна Команда: Діаграма класів237

Пов'язування команд зі слотами239

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 вентилятором251

Переходимо до тестування вентилятора.....252

Подальше тестування253

На кожному пульті має бути Режим Вечірки!.....254

Використання макрокоманд255

Патерн Команда означає безліч класів команд258

Спрощення коду RemoteControl із допомогою лямбда-виразів ... 259

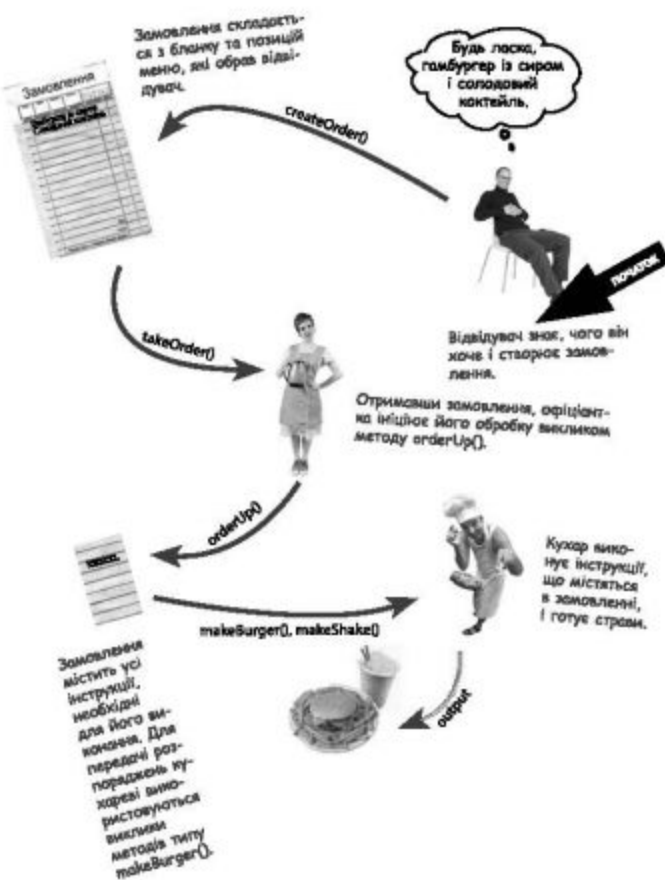
Ще більше спрощення завдяки посиланням на методи261

Тест-драйв команд із використанням лямбда-виразів.....262

Додаткові можливості патерна Команда: черги запитів265

Додаткові можливості використання патерна Команда: логування запитів.....266

Інструменти для вашої проектувальної панелі267



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.

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

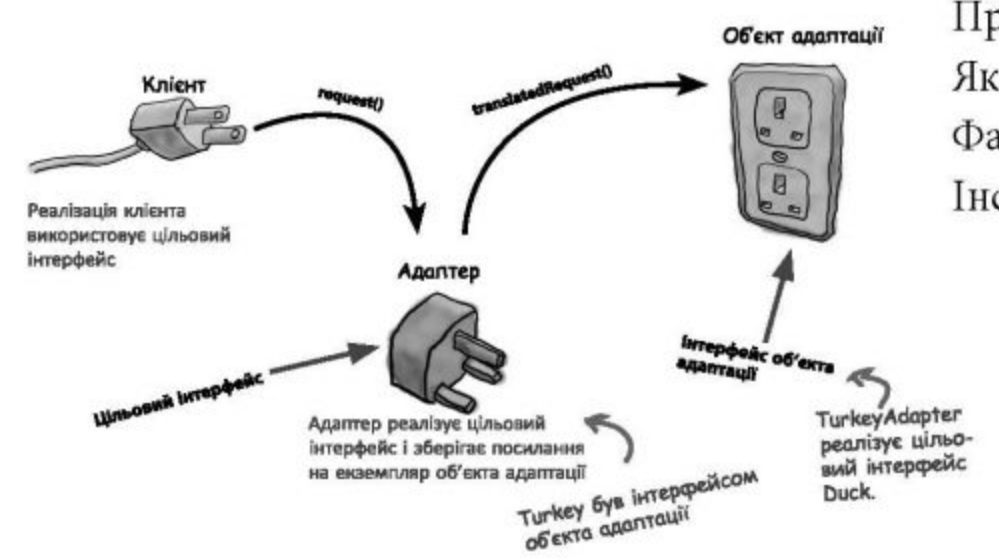
Визначення патерна Фасад..... 300

Принцип мінімальної поінформованості 301

Як НЕ заводити «друзів» і впливати на об'єкти..... 302

Фасад і принцип мінімальної поінформованості.... 305

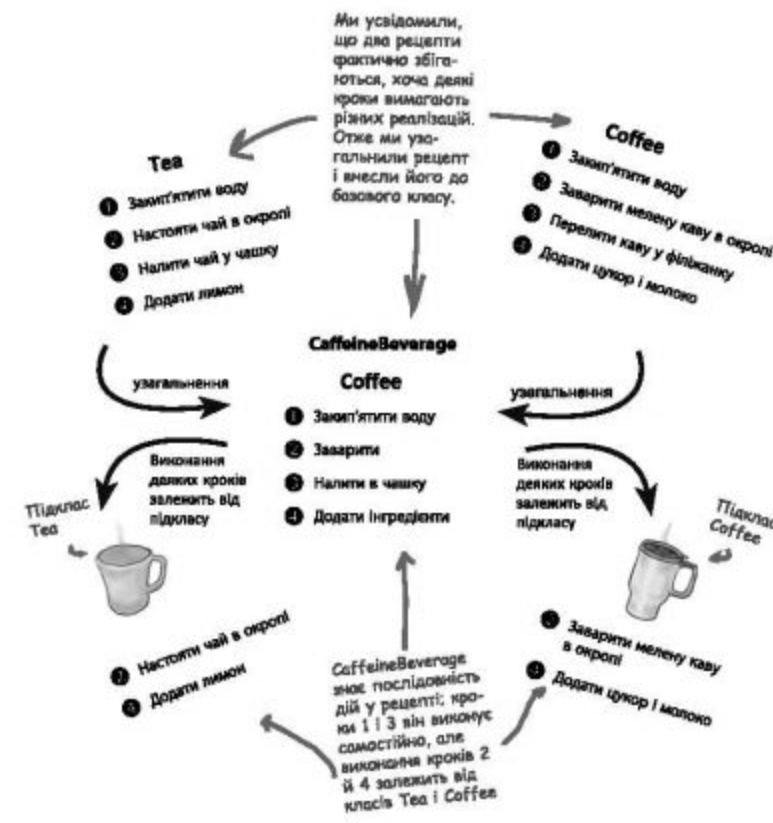
Інструменти для вашої проектувальної панелі..... 306



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 підкласти могли підключатися до обробки будь-якої миті. Ми навіть дізнаємося про ще один принцип проектування, натхнений практикою Голлівуда.



Час додати трохи кофєїну..... 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

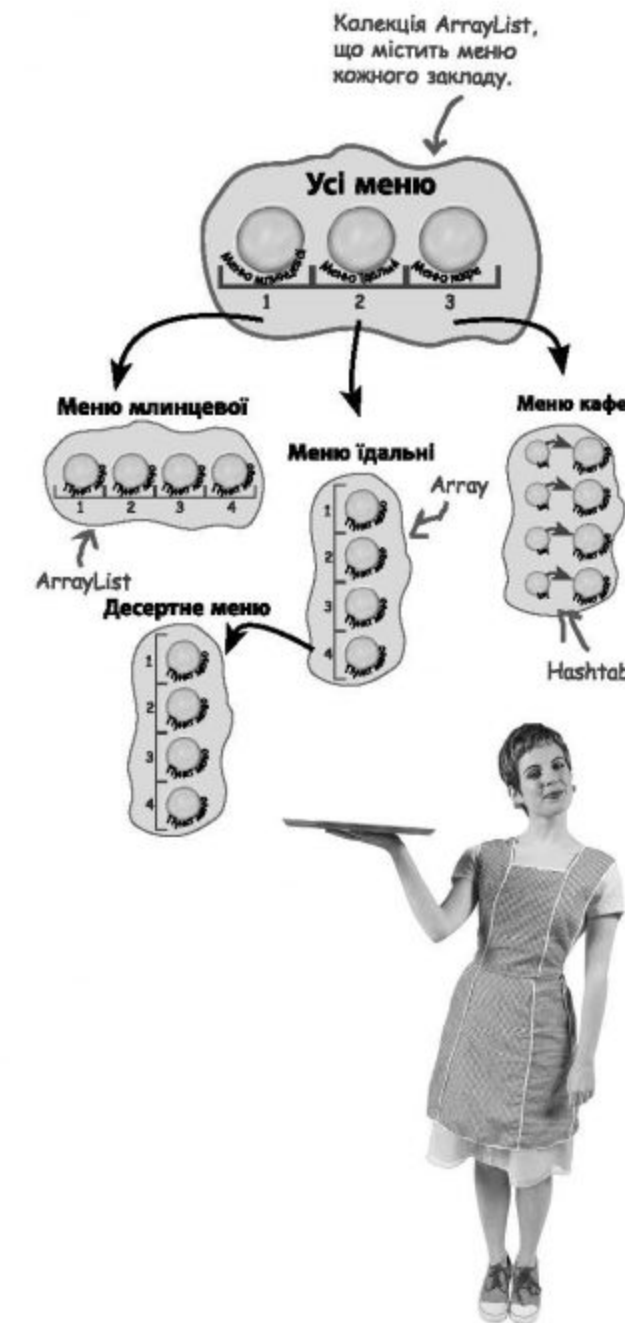
Аплети..... 341

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 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 способи їх зберігання. Також тут будуть описані способи створення суперколекцій. А якщо цього недостатньо, ви дізнаєтеся дещо нове про функції об'єктів.



Сенсація: об'єктівські 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

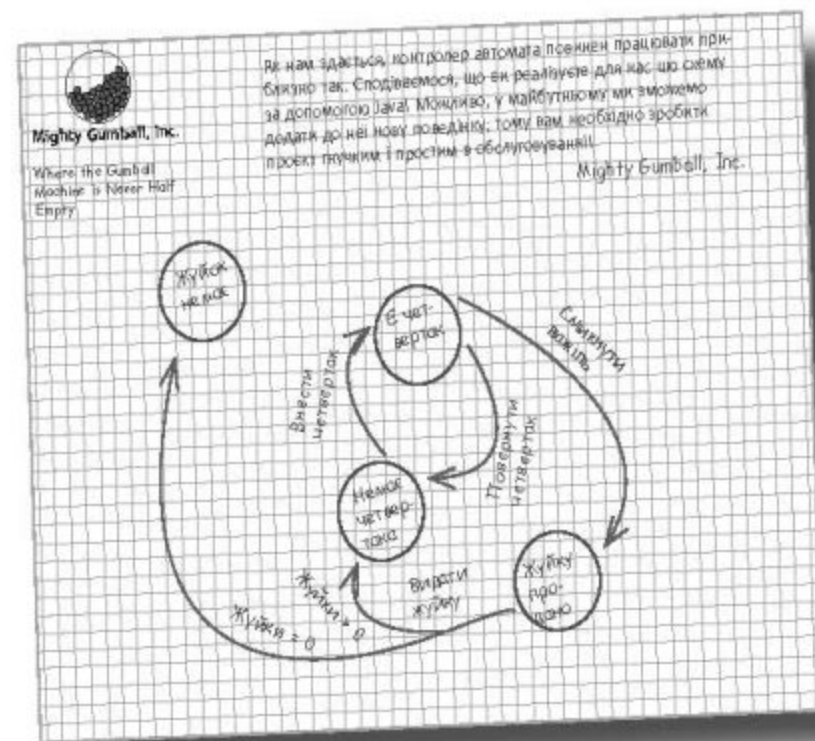
Реалізація компонентів меню 392
 Реалізація MenuItem 393
 Реалізація комбінаційного меню 394
 Готуємося до тестування... 396
 Тепер можна тестувати... 397
 Готуємося до тестового запуску... 398
 Повернення до ітератора 400
 Ітератор CompositeIterator 401
 Ітератор Null 404
 Дайте мені веганське меню! 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
 Машини 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

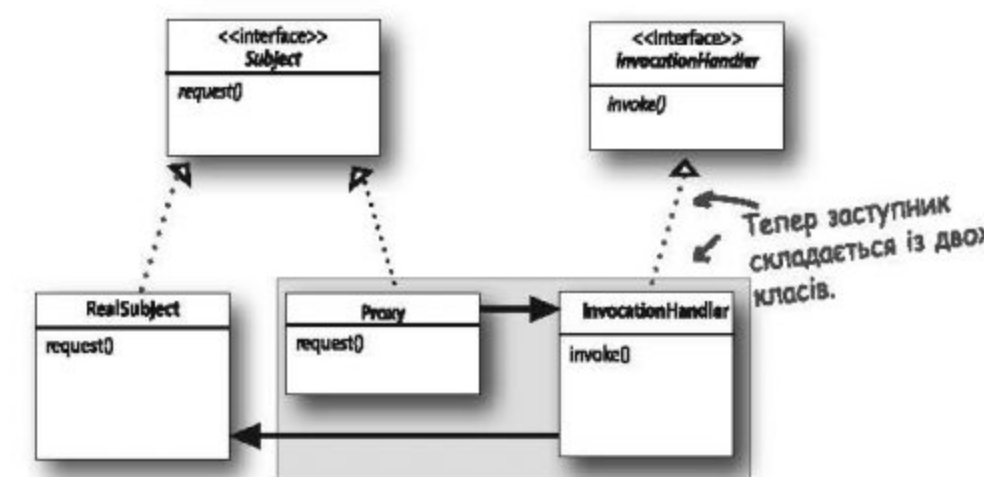


11

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 гумкою..... 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

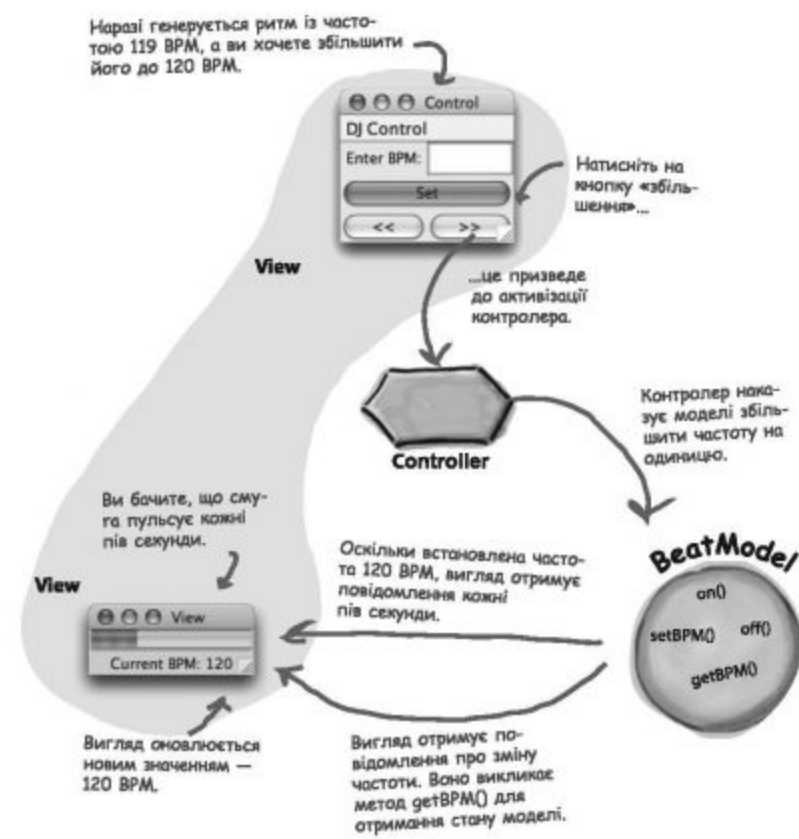


П'ятихвилинна драма: захист клієнтів 507
 Загальна картина: динамічний заступник для PersonBean..... 508
 Тестування служби знайомств 513
 Запуск коду..... 514
 Зоопарк Proxy..... 516
 Інструменти для вашої проектувальної панелі 518
 Код для перегляду обкладинки CD..... 521

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.



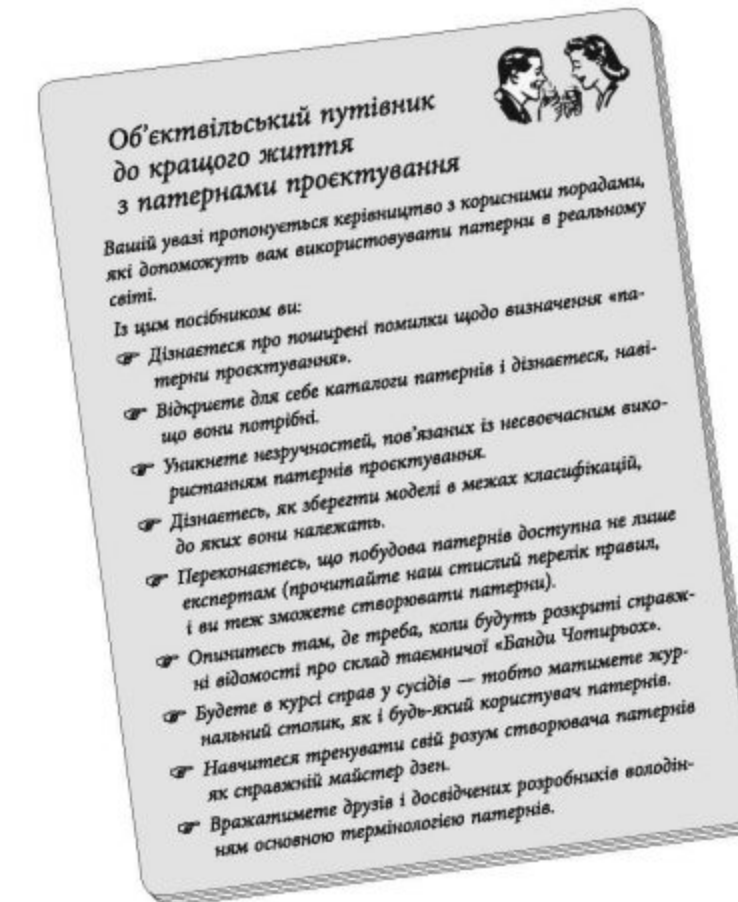
Працюючи разом.....	526
І знову качки.....	527
Що ми зробили?.....	549
Діаграма класів качиним оком.....	550
MVC: король складних патернів.....	552
Знайомство з Model-View-Controller.....	557
Детальніший погляд.....	558
MVC із точки зору патернів.....	560
Використання MVC для контролю за ритмом.....	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...

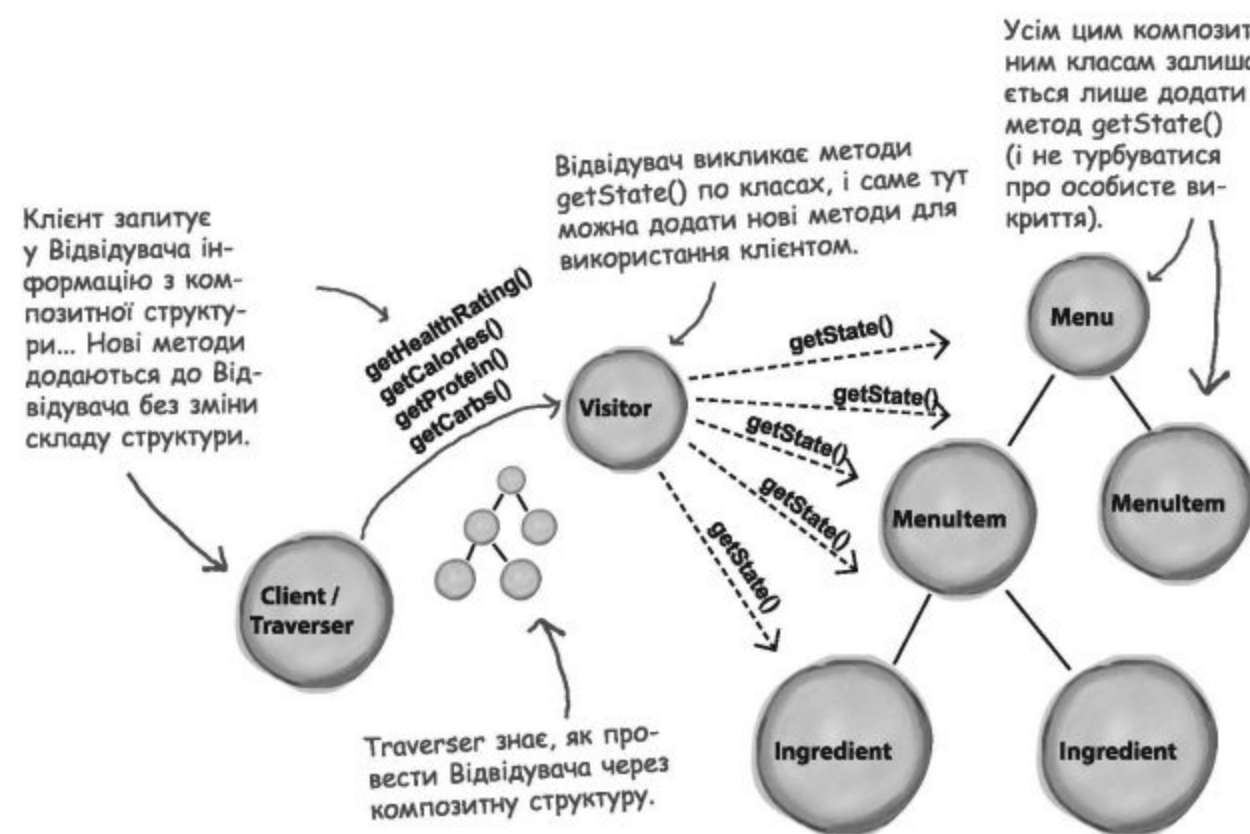
Визначення патерна проектування.....	607
Детальніше про визначення патерна проектування.....	609
Нехай сила буде з тобою!.....	610
Отже, ви хочете створювати патерни.....	615
Класифікація патернів проектування.....	617
Мислити патернами.....	622
Ваш розум у патернах.....	625
Не забувайте про силу єдиної 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'єктивілі.....	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.

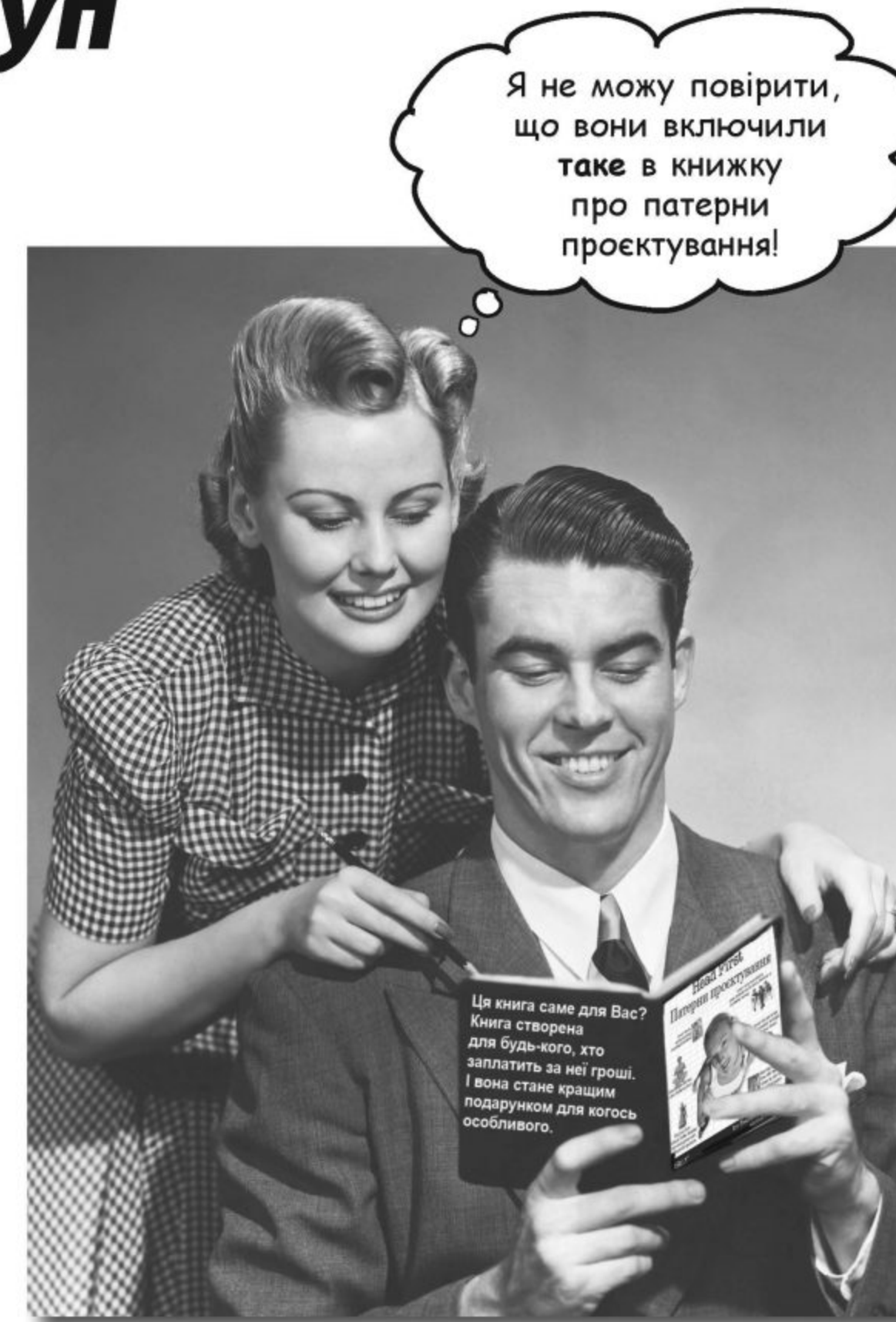


Міст	640
Будівельник	642
Ланцюжок обов'язків	644
Легковаговик	646
Інтерпретатор	648
Посередник	650
Знімок	652
Прототип	654
Відвідувач	656

Показчик	659
----------------	-----

Як користуватися цією книжкою

Вступ



У цьому розділі ми відповімо на нагальне питання: «Так навіщо вони включили таке в книжку про патерни проектування?»

Для кого написано цю книжку?

Якщо ви відповісте «так» на всі наступні питання:

- 1 Ви знаєте **Java**? (Бути знавцем не обов'язково.)
- 2 Ви хочете **вивчити, зрозуміти, запам'ятати і застосовувати** патерни, а також принципи ОО-проектування, на яких ґрунтуються патерни проектування?
- 3 Ви надаєте перевагу **жвавій бесіді** перед сухими, нудними академічними лекціями?

Імовірно, буде теж добре, якщо замість цього ви володієте C#.

...Тоді ця книжка для вас.

Кому ця книжка не підійде?

Якщо ви відповісте «так» на будь-яке з наступних питань:

- 1 **Ви абсолютно не знаєте Java?** (Бути знавцем не обов'язково. Причому навіть якщо ви не знаєте Java, але володієте C#, імовірно, ви зрозумієте не менше 80 % прикладів. Можливо, підійде й досвід програмування на C++.)
- 2 Ви є до біса крутим ОО-проектувальником / розробником, якому потрібен **довідник**?
- 3 Ви займаєтеся архітектурним проектуванням і шукаєте інформацію про патерни **корпоративного** рівня?
- 4 **Ви боїтеся спробувати щось нове?** Імовірніше підете до стоматолога, аніж надягнете смугасте з картатим? Чи вважаєте, що книжка, у якій компоненти Java зображені у вигляді чоловічків, серйозною бути не може?

...Тоді ця книжка не для вас.



взагалі-то ця книжка для всіх, хто має кредитку.

Ми знаємо, про що ви думаєте

«Хіба ця книжка з програмування є серйозною?»

«І чому тут стільки малюнків?»

«Чи можна так чогось навчитися?»

І ми знаємо, про що думає ваш мозок

Мозок прагне нових вражень. Він постійно шукає, аналізує, очікує чогось неймовірного. Він так улаштований, і це допомагає нам вижити.

У наші дні ви навряд чи станете обідом для тигра. Однак ваш мозок постійно перебуває наготові. Просто ви про це не здогадуєтесь.

Яким чином ваш мозок поводить з усіма звичайними, повсякденними речами? Він докладно зусиль для захисту від них, аби вони не заважали його справжній роботі — збереженню того, що дійсно важливо. Це не заважає збереженню нудної інформації; вона не відфільтровується крізь фільтр під назвою «очевидно несуттєве».

Але як же мозок дізнається, що є дійсно важливим? Уявіть, що ви виїхали на прогулянку, і раптом прямо перед вами з'являється тигр. Що відбувається у вашій голові та тілі?

Активізуються нейрони. Спалахують емоції. Відбуваються хімічні реакції.

Саме таким чином ваш мозок розуміє...

Це важливо! Не забувайте про це!

А тепер уявіть, що ви перебуваєте вдома або у бібліотеці. Це затишне, тепле місце, де тигри не водяться. Ви вчитеся, готуетесь до іспиту. Або намагаєтеся опанувати складну технічну тему, на яку ваш керівник вам виділив тиждень... максимум десять днів.

І тут виникає проблема: ваш мозок намагається надати вам послугу. Він намагається зробити так, аби на цю очевидно несуттєву інформацію не витрачалися дорожці ресурси. Їх краще витратити на щось важливе. На тигрів, наприклад. Або на те, що вогонь небезпечний. Або що не варто знов кататися в футболці та шортах на лижах.

Немає простого способу сказати своєму мозку: «Послухай, мозку, я тобі, звісно, вдячний, але хоч якою б нудною була ця книжка, і нехай за шкалою емоцій Ріхтера я перебуваю на нулі, я хочу запам'ятати те, що тут написано».

Ваш мозок вважає, що ЦЕ важливо.



Чудово. Ще 654 сухі, нудні сторінки.

Ваш мозок вважає, що ЦЕ не варто запам'ятовувати.



Ми вважаємо читача «Head First» учнем

Яким чином ми щось *дізнаємося*? Спочатку потрібно це «щось» *зрозуміти*, а потім *не забути*. Заштовхати в голову якнайбільше фактів недостатньо. Згідно з даними новітніх досліджень у галузі когнітивістики, нейробіології та психології навчання, для *засвоєння* матеріалу потрібно щось більше, ніж простий текст на сторінці. Ми знаємо, як змусити ваш мозок працювати.

Основні принципи серії «Head First»:

Зробити це наочним. Графіка запам'ятовується значно краще за звичайний текст і значно підвищує ефективність сприйняття інформації (до 89 % поліпшення пригадування та відтворення вивченого). До того ж, матеріал стає зрозумілішим.

Якщо розташувати відповідний текст на малюнках або поряд із ними, а не під ними або на сусідній сторінці, учні будуть мати вдвічі більше шансів зрозуміти зміст.



Використовувати розмовний та персоналізований стиль викладу.

Нещодавні дослідження показали, що при особистому розмовному стилі викладу матеріалу замість формальних лекцій, поліпшення результатів на підсумковому тестуванні становило до 40 %. Розповідайте історію замість того, аби читати лекцію. Не ставтеся до себе занадто серйозно. Що імовірніше приверне вашу увагу: цікава бесіда за столом або лекція?

Погано бути абстрактним методом. Доводиться обходитися без тіла.



`abstract void roam();`

Метод немає тіла! Не забудьте поставити крапку з комою.

Змусити учня мислити глибше. Поки ви не почнете напружувати звичини, у вашій голові нічого не відбувається. Читач повинен бути зацікавлений у результаті; він повинен натхненно вирішувати завдання, формулювати висновки і генерувати нові знання. А для цього необхідні вправи і каверзні питання, у вирішенні яких задіяні обидві півкулі мозку і різні почуття.

Чи можна сказати, що ванна є ванною кімнатою? Ванна кімната — це ванна? Або це взаємопов'язані речі?



Привернути увагу читача і зберегти її. Ситуація, знайома кожному: «Я дуже хочу вивчити це, але занадто на першій сторінці». Мозок звертає увагу на цікаве, дивне, привабливе, несподіване. Вивчення складної технічної теми не має бути нудним. Ваш мозок осягне це набагато швидше, якщо це буде цікаво викладено.

Залучення емоцій. Відомо, що наша здатність запам'ятовувати значною мірою залежить від емоційного співпереживання. Ми запам'ятовуємо те, що нам не байдуже. Ми запам'ятовуємо, коли щось відчуваємо. Ні, ми не маємо на увазі хвилюючі історії про хлопчика та його собаку. Мова йде про такі емоції, як здивування, цікавість, інтерес і почуття «Так, я крутий!» при вирішенні завдання, яке всі інші вважають складним, або коли ви розумієте, що розбираєтеся в темі краще за всезнайку Боба з технічного відділу.



Мета пізнання: мислення про мислення

Якщо ви дійсно хочете вчитися, і хочете дізнаватися швидше й глибше, зверніть увагу на те, як ви звертаєте увагу. Подумайте, яким чином ви мислите. Дізнайтеся, як ви навчаєтеся.

Більшість із нас не вивчали теорію метапізнання або навчання під час навчання. Ми *повинні* вчитися, але нас рідко *цього* навчають.

Проте, оскільки ви тримаєте цю книжку, то, імовірно, дійсно хочете вивчати патерни проектування. І ви, мабуть, не хочете витратити забагато часу на це. І ви хочете *запам'ятати* прочитане і застосувати нове знання на практиці. І для цього ви повинні *зрозуміти* це. Аби отримати максимум користі від цієї книжки, або будь-якої книжки чи досвіду навчання, візьміть на себе відповідальність за роботу власного мозку. Вашого мозку у цьому контенті.

Хитрість полягає в тому, аби змусити ваш мозок бачити новий матеріал, який ви вивчаєте, як дійсно важливий. Як той, що має вирішальне значення для вашого благополуччя. Не менш важливе, ніж тигр. Інакше вам доведеться постійно боротися, і ваш мозок зробить усе можливе, аби ухилитися від запам'ятовування нової інформації.

Цікаво, яким чином мені змусити власний мозок запам'ятати цей матеріал...



Отже, як переконати ваш мозок, що патерни проектування так само важливі, як і тигр?

Існує спосіб повільний і нудний, а є швидкий і ефективний. Перший ґрунтується на безглуздому повторюванні. Усім відомо, що навіть найнуднішу інформацію можна запам'ятати, якщо повторювати її знову й знову. При достатній кількості повторень ваш мозок розмірковує: «Відчувається ніби несуттєве для мене, але якщо те саме повторюється знов, і знов, і знов... гаразд, нехай буде так».

Швидший спосіб ґрунтується на **стимулюванні активності мозку**, особливо на поєднанні різних *видів* мозкової діяльності. Доведено, що всі фактори, перераховані на попередній сторінці, допомагають вашому мозку працювати на вас. Наприклад, дослідження показали, що розміщення слів *усередині* малюнків (а не в підписах, в основному тексті тощо) змушує мозок аналізувати зв'язок між текстом і графікою, а це призводить до активізації більшої кількості нейронів. Більше нейронів — вища ймовірність того, що інформація буде визнана важливою й вартою того, аби запам'ятати її.

Розмовний стиль теж важливий: зазвичай люди приділяють більше уваги, коли беруть участь в розмові, бо їм доводиться стежити за перебігом бесіди і висловлювати свою думку. Причому мозок абсолютно *не цікавить*, що ви «розмовляєте» з книжкою! З іншого боку, якщо текст сухий і формальний, то мозок відчуває те саме, що відчуваєте ви на нудній лекції в ролі пасивного слухача. Його хилить у сон.

Проте малюнки і розмовний стиль — це лише початок.

Ось що зробили МИ:

Ми використовували **малюнки**, бо мозок краще пристосований для сприйняття графіки, ніж тексту. Із точки зору мозку малюнок коштує 1024 слова. А коли текст комбінується з графікою, ми вбудовуємо текст безпосередньо в малюнки, бо мозок працює ефективніше, коли текст розташовується в межах речі, на яку посилається текст, на відміну від підпису або фрагмента тексту.

Ми використовуємо **дублювання**: повторюємо те саме кілька разів, застосовуючи різні засоби передачі інформації, звертаємося до *різних почуттів* і робимо все для підвищення ймовірності того, що матеріал буде закодований в кількох ділянках вашого мозку.

Ми використовуємо концепції та малюнки дещо **несподіваним** чином, бо мозок краще сприймає нову інформацію, і ми використовуємо фотографії, оскільки вони зазвичай мають **емоційний зміст**, бо мозок звертає увагу на біохімію емоцій. Те, що змушує нас *відчувати*, краще запам'ятовується, навіть якщо це невеликий **жарт**, **здивування** або **інтерес**.

Ми використовуємо персоналізований **розмовний стиль**, бо мозок краще сприймає інформацію, коли ви берете участь в розмові, а не пасивно слухаєте лекцію. Це відбувається й під час *читання*.

До книжки включено чимало **вправ** і творчих **завдань**, бо мозок краще запам'ятовує, коли ви щось **робите**, а не просто читаете про щось. Ми постаралися зробити їх непростими, але цікавими — саме цьому віддає перевагу більшість *читачів*.

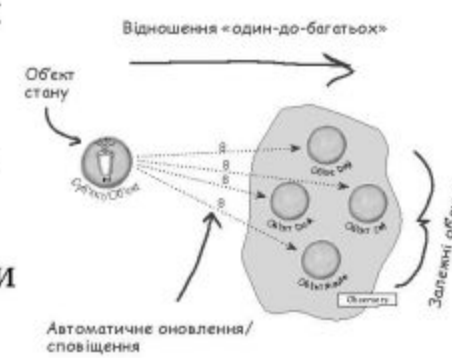
Ми поєднали **кілька стилів навчання**, бо одні читачі надають перевагу покроковим описам, у той час як інші прагнуть спочатку уявити «загальну картину», а третім вистачає фрагмента коду. Проте, незалежно від ваших особистих уподобань, корисно бачити кілька варіантів подання одного матеріалу.

Ми постаралися задіяти **обидві півкулі вашого мозку**, бо що більше ділянок мозку задіяно, то вища ймовірність засвоєння матеріалу і зосередженості на навчанні. Поки одна півкуля мозку працює, інша часто має можливість відпочити; це підвищує ефективність навчання протягом тривалого часу.

А ще книжка містить **історії** та вправи, що **відтворюють інші точки зору**, оскільки мозок якісніше засвоює інформацію, коли йому доводиться оцінювати і робити припущення.

У книжці часто зустрічаються **завдання** і **питання**, на які не завжди можна дати просту відповідь, бо мозок швидше вчиться і запам'ятовує, коли йому доводиться щось робити. Пам'ятайте, що неможливо накачати м'язи, спостерігаючи за тим, як займаються інші. Однак ми подбали про те, щоби зусилля читачів були спрямовані у правильному напрямку. Вам **не доведеться ламати голову** над незрозумілими прикладами або розбиратися в складному, перенасиченому технічним жаргоном або занадто лаконічному тексті.

Ми використовували як приклади **людей**. У матеріалах, завданнях, малюнках тощо, — бо *ви* теж людина. І ваш мозок приділяє більше уваги *людям*, ніж *речам*. Ми керувалися принципом **80/20**. Якщо ви збираєтеся здобути ступінь доктора в галузі розробки ПЗ, то однієї книжки все одно недостатньо, тому ми не намагалися розповісти про *все*. Йтиметься лише про **найнеобхідніше**.



Гуру патернів



КЛЮЧОВІ МОМЕНТИ

Головоломки



Виріжте та прилаштуйте на холодильник.

Ось що ВИ можете зробити, аби змусити свій мозок підкорятися

Отже, ми свою справу зробили. Решта за вами. Ці поради стануть відправною точкою; прислухайтеся до свого мозку і визначте, що вам підходить, а що не підходить. Спробуйте нове.

1 Не поспішайте. Що більше ви зрозумієте, то менше доведеться запам'ятовувати.

Просто читати недостатньо. Коли книжка ставить вам запитання, не переходьте до відповіді. Уявіть, що хтось дійсно ставить вам запитання. Що глибше ваш мозок буде думати, то швидше ви зрозумієте й запам'ятаєте матеріал.

2 Виконуйте вправи. Робіть власні нотатки.

Ми включили вправи до книжки, але виконувати їх за вас не збираємося. І не просто *розглядайте* вправи. **Беріть олівець і пишіть**. Фізична активність *під час* навчання підвищує його ефективність.

3 Читайте «Не існує безглузвих запитань».

Це означає: читайте все. Врізки, написи на малюнках — **це частина основного матеріалу!** Не проминайте їх.

4 Не читайте інші книжки після цієї перед сном.

Частина навчання (особливо перенесення інформації в довгострокову пам'ять) відбувається *після* того, як ви відкладаєте книжку. Ваш мозок не одразу засвоює інформацію. Якщо під час обробки надійде нова інформація, частина того, про що ви дізналися раніше, може бути втрачена.

5 Пийте воду. І якомога більше.

Мозок найкраще працює в умовах високої вологості. Дегідратація (яка може статися ще до того, як ви відчуєте спрагу) знижує когнітивні функції.

6 Говоріть про це.

Мова активізує інші ділянки мозку. Якщо ви намагаєтеся щось зрозуміти або краще запам'ятати, скажіть уголос. А ще краще спробуйте пояснити комусь іншому. Ви будете швидше засвоювати матеріал і, можливо, відкриєте для себе щось нове.

7 Прислухайтеся до свого мозку.

Слідкуйте за тим, коли ваш мозок починає втомлюватися. Якщо ви починаєте поверхнево сприймати матеріал або забуваєте щойно прочитане — настав час зробити перерву. Якщо ви цього не зробите, може настати певна мить, коли ви не зможете пришвидшитись у навчанні, намагаючись засвоїти більше, а, навпаки, — лише зашкодите процесу.

8 Відчувайте!

Ваш мозок повинен знати, що матеріал книжки *дійсно важливий*. Переймайтеся перебігом наших історій. Вигадуйте власні підписи до фотографій. Поморщитися від невдалого жарту все одно краще за відсутність будь-яких відчуттів.

9 Проекуйте щось!

Застосуйте нові знання до проекту, над яким ви працюєте, або переробіть старий проект. Просто зробіть *хоч щось*, аби набути практичного досвіду за рамками вправ. Усе, що для цього потрібно — олівець і відповідна проблема, що потребує вирішення... проблема, що може отримати зиск від застосування одного чи більшої кількості патернів проектування.