

Содержание (сводка)

	Введение	25
1	Добро пожаловать в мир паттернов: <i>знакомство с паттернами</i>	37
2	Объекты в курсе событий: <i>паттерн Наблюдатель</i>	71
3	Украшение объектов: <i>паттерн Декоратор</i>	111
4	Домашняя ОО-выпечка: <i>паттерн Фабрика</i>	141
5	Уникальные объекты: <i>паттерн Одиночка</i>	199
6	Инкапсуляция вызова: <i>паттерн Команда</i>	219
7	Умение приспосабливаться: <i>паттерны Адаптер и Фасад</i>	269
8	Инкапсуляция алгоритмов: <i>паттерн Шаблонный Метод</i>	307
9	Управляемые коллекции: <i>паттерны Итератор и Композитор</i>	345
10	Состояние дел: <i>паттерн Состояние</i>	413
11	Управление доступом к объектам: <i>паттерн Заместитель</i>	457
12	Паттерны паттернов: <i>составные паттерны</i>	523
13	Паттерны в реальном мире: <i>паттерны для лучшей жизни</i>	599
14	Приложение: <i>другие паттерны</i>	633

Содержание (настоящее)

Введение

Настройте свой мозг на дизайн паттернов. Вот что вам понадобится, когда вы пытаетесь что-то выучить, в то время как ваш мозг не хочет воспринимать информацию. Ваш мозг считает: «Лучше уж я подумаю о более важных вещах, например об опасных диких животных или почему нельзя голышом прокатиться на сноуборде». Как же заставить свой мозг думать, что ваша жизнь зависит от овладения дизайном паттернов?

Для кого написана эта книга?	26
Мы знаем, о чем вы думаете	27
Метапознание	29
Заставь свой мозг повиноваться	31
Технические рецензенты	34
Благодарности	35

1 Знакомство с паттернами

Добро пожаловать в мир паттернов

Наверняка вашу задачу кто-то уже решал. В этой главе вы узнаете, почему (и как) следует использовать опыт других разработчиков, которые уже сталкивались с аналогичной задачей и успешно решили ее. Заодно мы поговорим об использовании и преимуществах паттернов проектирования, познакомимся с ключевыми принципами ООП и разберем пример одного из паттернов. Лучший способ использовать паттерны — *запомнить их*, а затем научиться *распознавать* те места ваших архитектур и существующих приложений, где их уместно *применить*. Таким образом, вместо программного кода вы повторно используете чужой *опыт*.

Знание таких концепций, как абстракция, наследование и полиморфизм, еще не делает из вас хорошего ОО-проектировщика. Истинный гурู проектирования стремится создавать гибкие архитектуры, способные адаптироваться к изменениям.



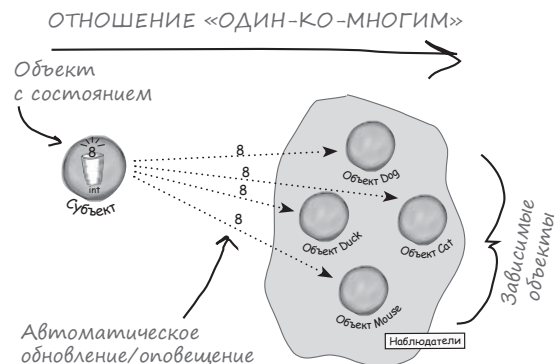
Приложение SimUDuck	38
Джо думает о наследовании...	41
Как насчет интерфейса?	42
Единственная константа в программировании	44
Отделяем переменное от постоянного	46
Реализация поведения уток	49
Тестирование кода Duck	54
Динамическое изменение поведения	56
Инкапсуляция поведения: общая картина	58
Отношения СОДЕРЖИТ бывают удобнее отношений ЯВЛЯЕТСЯ	59
Паттерн Стратегия	60
Сила единой номенклатуры	64
Как пользоваться паттернами?	65
Новые инструменты	68
Ответы к упражнениям	69

2 Паттерн Наблюдатель

Объекты в курсе событий

Не упустите, когда происходит что-то интересное! Наш следующий паттерн оповещает объекты о наступлении неких событий, которые могут представлять для них интерес, — причем объекты даже могут решать во время выполнения, желают ли они и дальше получать информацию. Паттерн Наблюдатель чрезвычайно полезен и принадлежит к числу наиболее часто используемых паттернов JDK. Также в этой главе будут рассмотрены связи типа «один-ко-многим» и слабые связи. С паттерном Наблюдатель вы станете душой Общества Паттернов.

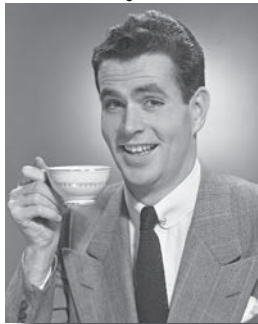
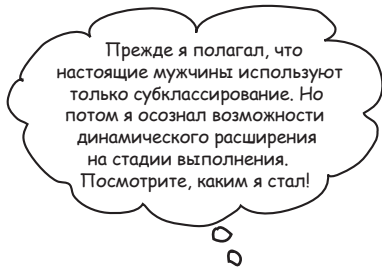
Обзор приложения Weather Monitoring	73
Знакомство с паттерном Наблюдатель	78
Издатели + Подписчики = Паттерн Наблюдатель	79
Пятиминутная драма: субъект для наблюдения	82
Определение паттерна Наблюдатель	85
Сила слабых связей	87
Проектирование Weather Station	89
Реализация Weather Station	90
Встроенная реализация в языке Java	97
Темная сторона java.util.Observable	104
Новые инструменты	108
Ответы к упражнениям	110



Паттерн Декоратор

Украшение объектов

Эту главу можно назвать «Взгляд на архитектуру для любителей наследования». Мы проанализируем типичные злоупотребления из области наследования, и вы научитесь декорировать свои классы во время выполнения с использованием разновидности композиции. Зачем? Затем, что этот прием позволяет вам наделить свои (или чужие) объекты новыми возможностями без модификации кода классов.

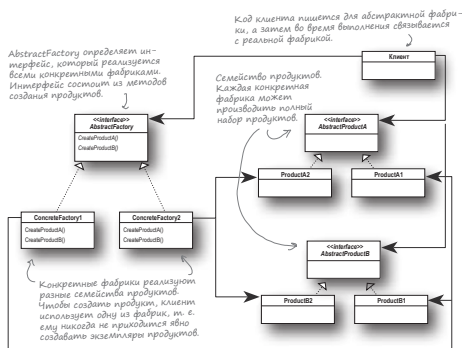


Добро пожаловать в Starbuzz	112
Принцип открытости/закрытости	118
Знакомство с паттерном Декоратор	120
Построение заказанного напитка	121
Определение паттерна Декоратор	123
Декораторы и напитки	124
Пишем код для Starbuzz	127
Программируем классы напитков	128
Программирование дополнений	129
Декораторы в реальном мире: ввод/вывод в языке Java	132
Написание собственного декоратора ввода/вывода	134
Новые инструменты	137
Ответы к упражнениям	138

4 Паттерн Фабрика

Домашняя ОО-выпечка

Приготовьтесь заниматься выпечкой объектов в слабосвязанных ОО-архитектурах. Создание объектов отнюдь не сводится к простому вызову оператора *new*. Оказывается, создание экземпляров не всегда должно осуществляться открыто; оно часто создает проблемы *сильного связывания*. А ведь вы *этого* не хотите, верно? Паттерн Фабрика спасет вас от неприятных зависимостей.



Видим <i>new</i> — подразумеваем <i>конкретный</i>	142
Пицца Объектвила	144
Инкапсуляция создания объектов	146
Построение Простой Фабрики для пиццы	147
Определение Простой Фабрики	149
Инфраструктура для пиццерии	152
Принятие решений в subclasses	153
Subclasses PizzaStore	155
Объявление Фабричного Метода	157
Пора познакомиться с паттерном Фабричный Метод	163
Параллельные иерархии классов	164
Определение паттерна Фабричный Метод	166
PizzaStore с сильными зависимостями	169
Зависимости между объектами	170
Принцип инверсии зависимостей	171
Вернемся в пиццерию...	176
Семейства ингредиентов...	177
Построение фабрик ингредиентов	178
Рассмотрим Абстрактную Фабрику	185
За сценой	186
Определение паттерна Абстрактная Фабрика	188
Сравнение паттернов Фабричный Метод и Абстрактная Фабрика	192
Новые инструменты	194
Ответы к упражнениям	195

5 Паттерн Одиночка

Уникальные объекты

Паттерн Одиночка направлен на создание уникальных объектов, существующих только в одном экземпляре. Из всех паттернов Одиночка имеет самую простую диаграмму классов; собственно, вся диаграмма состоит из одного-единственного класса! Однако не стоит расслабляться; несмотря на всю его простоту с точки зрения архитектуры классов, в его реализации кроется немало ловушек. Так что пристегните ремни!



Единственный и неповторимый	200
Вопросы и ответы	201
Классическая реализация паттерна Одиночка	203
Признания Одиночки	204
Шоколадная фабрика	205
Определение паттерна Одиночка	207
Кажется, у нас проблемы...	208
Представьте, что вы – JVM	209
Решение проблемы многопоточного доступа	210
Одиночка. Вопросы и ответы	214
Новые инструменты	216
Ответы к упражнениям	217

6 Паттерн Команда

Инкапсуляция вызова

В этой главе мы выходим на новый уровень инкапсуляции — на этот раз будут инкапсулироваться вызовы методов. Да, все верно — вызывающему объекту не нужно беспокоиться о том, как будут выполняться его запросы. Он просто использует инкапсулированный метод для решения своей задачи. Инкапсуляция позволяет решать и такие нетривиальные задачи, как регистрация или отмена вызовов.

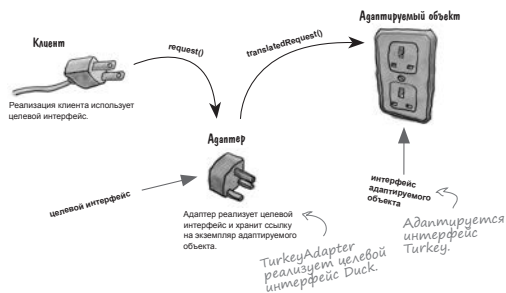
Автоматизируй дом или проиграешь	220
Пульт домашней автоматизации	221
Классы управления устройствами	222
Краткое введение в паттерн Команда	225
Рассмотрим взаимодействия чуть более подробно...	226
Роли и обязанности в кафе Объектвия	227
От кафе к паттерну Команда	229
Наш первый объект команды	231
Определение паттерна Команда	234
Связывание команд с ячейками	237
Реализация пульта	238
Проверяем пульт в деле	240
Пора писать документацию...	243
Реализация отмены с состоянием	248
На каждом пульте должен быть Режим Вечеринки!	252
Использование макрокоманд	253
Паттерн Команда означает множество классов команд	256
Упрощение кода RemoteControl с лямбда-выражениями	257
Тест-драйв команд с использованием лямбда-выражений	260
Расширенные возможности паттерна Команда: очереди запросов	263
Расширенные возможности паттерна Команда: регистрация запросов	264
Новые инструменты	265
Ответы к упражнениям	266

Паттерны Адаптер и Фасад

Умение приспосабливаться

В этой главе мы займемся всякими невозможными трюками — будем затыкать круглые дырки квадратными пробками. Невозможно, скажете вы? Только не с паттернами проектирования. Помните паттерн Декоратор? Мы «упаковывали» объекты, чтобы расширить их возможности. А в этой главе мы займемся упаковкой объектов с другой целью: чтобы имитировать интерфейс, которым они в действительности не обладают. Для чего? Чтобы адаптировать архитектуру, рассчитанную на один интерфейс, для класса, реализующего другой интерфейс. Но и это еще не все; попутно будет описан другой паттерн, в котором объекты упаковываются для упрощения их интерфейса.

Адаптеры вокруг нас	270
Объектно-ориентированные адаптеры	271
Как работает паттерн Адаптер	275
Определение паттерна Адаптер	277
Адаптеры объектов и классов	278
Беседа у камина: Адаптер объектов и Адаптер классов	281
Практическое применение адаптеров	282
Адаптация перечисления к итератору	283
Беседа у камина: паттерн Декоратор и паттерн Адаптер	286
Домашний кинотеатр	289
Свет, камера, фасад!	292
Построение фасада для домашнего кинотеатра	295
Определение паттерна Фасад	298
Принцип минимальной информированности	299
Новые инструменты	304
Ответы к упражнениям	305



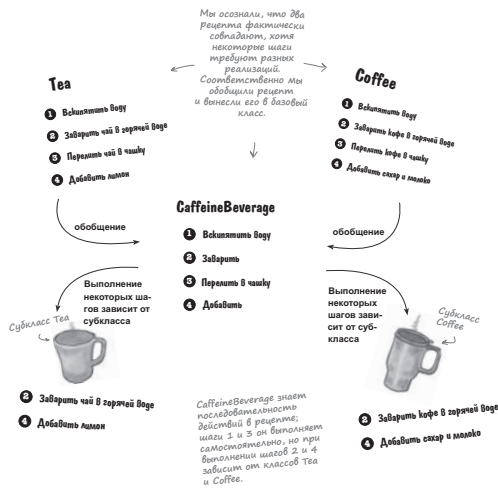
8

Паттерн Шаблонный Метод

Инкапсуляция алгоритмов

Мы уже «набили руку» на инкапсуляции; мы инкапсулировали создание объектов, вызовы методов, сложные интерфейсы, уток, пиццу... Что дальше? Следующим шагом будет инкапсуляция алгоритмических блоков, чтобы subclasses могли в любой момент связаться с нужным алгоритмом обработки. В этой главе даже будет описан принцип проектирования, вдохновленный Голливудской практикой.

Что мы сделали?

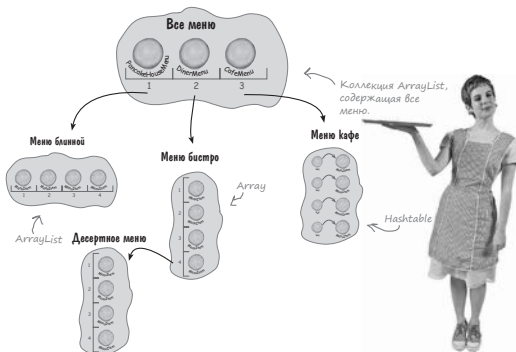


Кофе и чай (на языке Java)	309
Абстрактный кофе и чай	312
Продолжаем переработку...	313
Абстрагирование prepareRecipe()	314
Что мы сделали?	317
Паттерн Шаблонный Метод	318
Готовим чай...	319
Что дает Шаблонный Метод?	320
Определение паттерна Шаблонный Метод	321
Код под увеличительным стеклом	322
Перехватчики в паттерне Шаблонный Метод	324
Использование перехватчиков	325
Проверяем, как работает код	326
Голливудский принцип	328
Голливудский принцип и Шаблонный Метод	329
Шаблонные методы на практике	331
Сортировка на базе Шаблонного Метода	332
Сортируем уток...	333
Сравнение объектов Duck	334
Как сортируются объекты Duck	336
Шаблонный метод в JFrames	338
Аплеты	339
Беседа у камина: Шаблонный Метод и Стратегия	340
Новые инструменты	342
Ответы к упражнениям	343

9 Паттерны Итератор и Компоновщик

Управляемые коллекции

Существует много способов создания коллекций. Объекты можно разместить в контейнере Array, Stack, List, Hashtable — выбирайте сами. Каждый способ обладает своими достоинствами и недостатками. Но в какой-то момент клиенту потребуется перебрать все эти объекты, и, когда это произойдет, собираетесь ли вы раскрывать реализацию коллекции? Надеемся, нет! Это было бы крайне непрофессионально. В этой главе вы узнаете, как предоставить клиенту механизм перебора объектов без раскрытия информации о способе их хранения. Также в ней будут описаны способы создания суперколлекций. А если этого недостаточно, вы узнаете кое-что новое относительно обязанностей объектов.



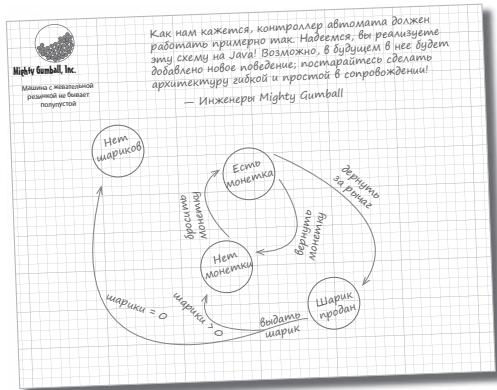
Бистро объединяется с блинной!	346
Сравниваем две реализации	348
Как инкапсулировать перебор элементов?	354
Паттерн Итератор	356
Добавление итератора в DinerMenu	357
Рассмотрим архитектуру	362
Интеграция с java.util.Iterator	364
Что нам это дает?	366
Определение паттерна Итератор	367
Принцип одной обязанности	370
Итераторы и коллекции	379
А когда мы уже торжествовали победу...	383
Определение паттерна Компоновщик	386
Проектирование меню с использованием паттерна Компоновщик	389
Реализация комбинационного меню	392
Возвращение к итераторам	398
Пустой итератор	402
Магия итераторов и композиций	404
Новые инструменты	409
Ответы к упражнениям	410

10

Паттерн Состояние

Состояние дел

Малоизвестный факт: паттерны Стратегия и Состояние — близнецы, разлученные при рождении. Как известно, паттерн Стратегия организовал успешный бизнес в области взаимозаменяемых алгоритмов. Паттерн Состояние выбрал, пожалуй, более благородный путь: он помогает объектам управлять своим поведением посредством изменения внутреннего состояния.



Работа с диаграммой состояния	414
Краткий курс конечных автоматов	416
Программирование	418
Кто бы сомневался... запрос на изменение!	422
Печальное СОСТОЯНИЕ дел...	424
Определение интерфейса State и классов	427
Реализация классов состояний	429
Переработка класса Gumball Machine	430
Определение паттерна Состояние	438
Состояние vs Стратегия	439
Проверка разумности	445
Чуть не забыли!	448
Новые инструменты	451
Ответы к упражнениям	452

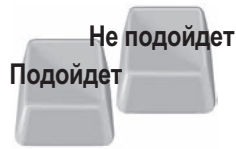


11

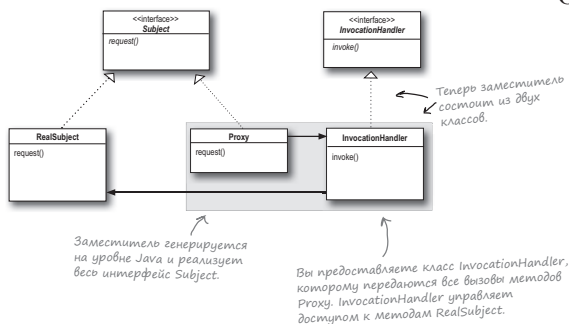
Паттерн Заместитель

Управление доступом к объектам

Когда-нибудь разыгрывали сценку «хороший полицейский, плохой полицейский»? Вы — «хороший полицейский», вы общаетесь со всеми любезно и по-дружески, но не хотите, чтобы все обращались к вам за каждым пустяком. Поэтому вы обзаводитесь «плохим полицейским», который управляет доступом к вам. Именно этим и занимаются заместители: они управляют доступом. Как вы вскоре увидите, способы взаимодействия заместителей с обслуживаемыми объектами. Иногда заместители пересылают по Интернету целые вызовы методов, а иногда просто терпеливо стоят на месте, изображая временно отсутствующие объекты.



Монитор и автомат с жевательной резинкой	459
Роль «удаленного заместителя»	462
Введение в RMI	464
Удаленные вызовы методов	465
Удаленный заместитель. За сценой	485
Определение паттерна Заместитель	487
Знакомьтесь: Виртуальный Заместитель	489
Проектирование виртуального заместителя	491
Виртуальный заместитель. За сценой	497
Создание защитного заместителя средствами Java API	501
Пятиминутная драма: защита клиентов	505
Динамический заместитель	506
Разновидности заместителей	514
Новые инструменты	516
Ответы к упражнениям	517

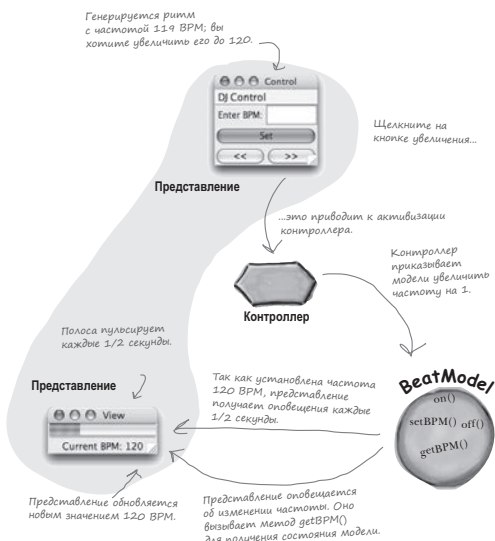


12

Составные паттерны

Паттерны паттернов

Кто бы мог предположить, что паттерны порой работают рука об руку? Вы уже были свидетелями ожесточенных перепалок в «Беседах у камина» (причем вы не видели «Смертельные поединки» паттернов — редактор заставил нас исключить их из книги!). И после этого оказывается, что мирное сосуществование все же возможно. Хотите верить, хотите нет, но некоторые из самых мощных ОО-архитектур строятся на основе комбинаций нескольких паттернов.

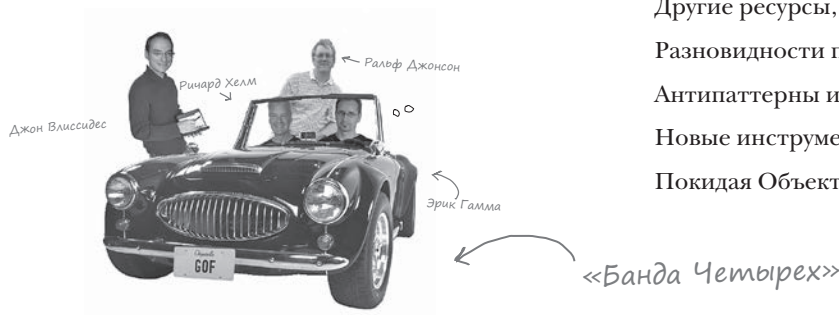
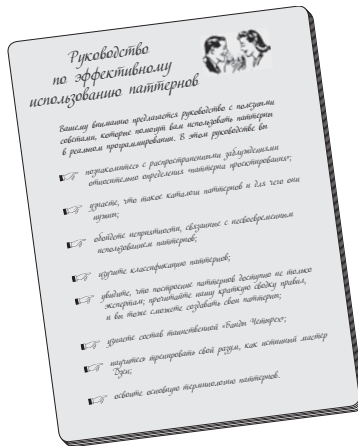


Совместная работа паттернов	524
И снова утки	525
Добавление Адаптера	528
Добавление Декоратора	530
Добавление Фабрики	532
Добавление Компоновщика и Итератора	537
Добавление Наблюдателя	540
И все вместе	547
Диаграмма классов с высоты утиноного полета	548
Паттерны проектирования — ключ к MVC	550
MVC с точки зрения паттернов	554
Использование MVC для управления ритмом...	556
Модель	559
Представление	561
Контроллер	564
Анализ паттерна Стратегия	567
Адаптация модели	568
Можно переходить к HeartController	569
MVC и Веб	571
Паттерны проектирования и Модель 2	579
Новые инструменты	582
Ответы к упражнениям	583

13 Паттерны для лучшей жизни

Паттерны в реальном мире

Вы стоите на пороге дивного нового мира, населенного паттернами проектирования. Но прежде чем открывать дверь, желательно изучить некоторые технические тонкости, с которыми вы можете столкнуться — в реальном мире жизнь немного сложнее, чем здесь, в Объективле. К счастью, у вас имеется хороший путеводитель, который упростит ваши первые шаги...

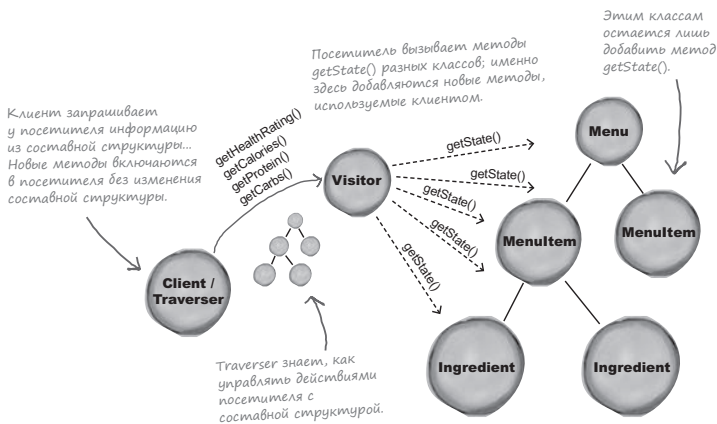


Руководство по использованию паттернов	600
Определение паттерна проектирования	601
Подробнее об определении паттерна проектирования	603
Да пребудет с вами Сила	604
Каталоги паттернов	605
Как создавать паттерны	608
Хотите создавать паттерны?	609
Классификация паттернов проектирования	611
Мыслить паттернами	616
Разум и паттерны	619
И не забудьте о единстве номенклатуры	621
Пять способов использования единой номенклатуры	622
Прогулка по Объективлю с «Бандой Четырех»	623
Наше путешествие только начинается...	624
Другие ресурсы, посвященные паттернам	625
Разновидности паттернов	626
Антипаттерны и борьба со злом	628
Новые инструменты	630
Покидая Объективль...	631

14

Приложение: Другие паттерны

Не каждому суждено оставаться на пике популярности. За последние 10 лет многое изменилось. С момента выхода первого издания книги «Банды Четырех» разработчики тысячи раз применяли эти паттерны в своих проектах. В этом приложении представлены полноценные, первосортные паттерны от «Банды Четырех» — они используются реже других паттернов, которые рассматривались ранее. Однако эти паттерны ничем не плохи, и если они уместны в вашей ситуации — применяйте их без малейших сомнений. В этом приложении мы постараемся дать общее представление о сути этих паттернов.



Мост	634
Строитель	636
Цепочка Обязанностей	638
Приспособленец	640
Интерпретатор	642
Посредник	644
Хранитель	646
Прототип	648
Посетитель	650