

## 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>	265
8	Инкапсуляция алгоритмов: <i>паттерн Шаблонный Метод</i>	303
9	Управляемые коллекции: <i>паттерны Итератор и Компоновщик</i>	341
10	Состояние дел: <i>паттерн Состояние</i>	403
11	Управление доступом к объектам: <i>паттерн Заместитель</i>	447
12	Паттерны паттернов: <i>составные паттерны</i>	513
13	Паттерны в реальном мире: <i>паттерны для лучшей жизни</i>	581
14	Приложение: <i>другие паттерны</i>	615

## 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: наука о мышлении	29
Вот что сделали мы	30
Что можете сделать вы	31
Примите к сведению	32

# 1 Знакомство с паттернами

## Добро пожаловать в мир паттернов

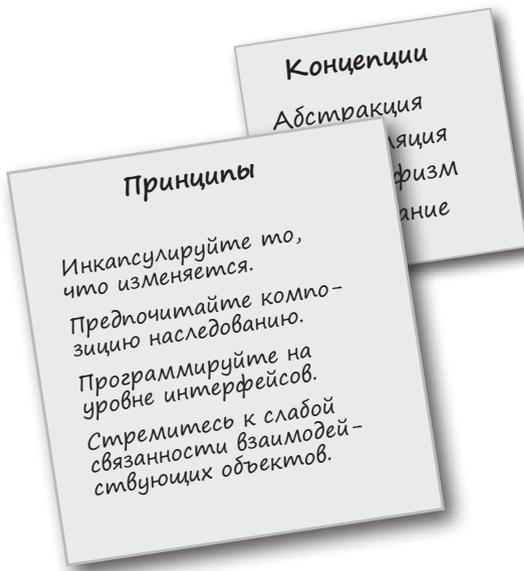
Наверняка вашу задачу кто-то уже решал. В этой главе вы узнаете, почему (и как) следует использовать опыт других разработчиков, которые уже сталкивались с аналогичной задачей и успешно решили ее. Заодно мы поговорим об использовании и преимуществах паттернов проектирования, познакомимся с ключевыми принципами объектно-ориентированного (ОО) проектирования 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*.



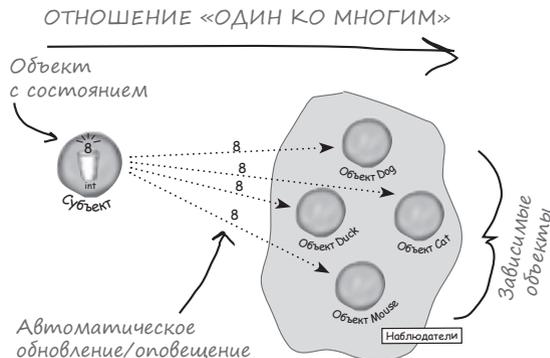
Все началось с простого приложения SimUDuck	38
Джо думает о наследовании...	41
Как насчет интерфейса?	42
Единственная константа в программировании	44
Отделяем переменное от постоянного	46
Проектирование переменного поведения	47
Реализация поведения уток	49
Интеграция поведения с классом Duck	51
Тестирование 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

Вы ведь не хотите оставаться в неведении, когда происходит что-то интересное, верно? У нас есть паттерн, который будет держать ваши объекты в курсе, когда происходит нечто такое, что их *интересует*. Это паттерн Наблюдатель — один из наиболее часто встречающихся паттернов проектирования, и он к тому же невероятно полезен. Мы рассмотрим многие интересные аспекты паттерна Наблюдатель, такие как *отношения типа «один ко многим»* и *слабое связывание*. А когда эти концепции запечатлеются у вас в мозгу, вы наверняка станете душой вечеринки в сообществе паттернов.



Обзор приложения Weather Monitoring	73
Как устроен класс WeatherData	74
Знакомство с паттерном Наблюдатель	78
Издатели + Подписчики = Паттерн Наблюдатель	79
Один день из жизни паттерна Наблюдатель	80
Определение паттерна Наблюдатель	85
Паттерн Наблюдатель: диаграмма классов	86
Сила слабых связей	88
Проектирование Weather Station	91
Реализация Weather Station	92
Реализация интерфейса Subject в WeatherData	93
Переходим к визуальным элементам	94
Тестирование Weather Station	95
Обновленный код с использованием лямбда-выражений	101
Новые инструменты	106
Ответы к упражнениям	108



# 3 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 наделять свои (или чужие) объекты новыми возможностями *без модификации кода классов*.

Прежде я полагал, что настоящие мужчины используют только субклассирование. Но потом я осознал возможности динамического расширения на стадии выполнения. Посмотрите, каким я стал!



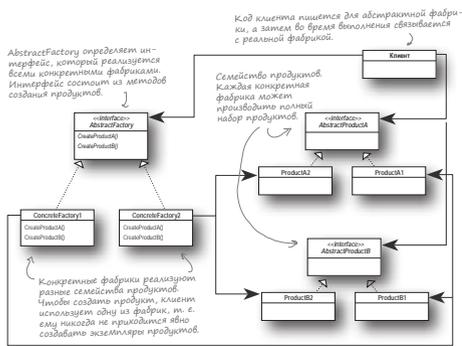
Добро пожаловать в Starbuzz	112
Принцип открытости/закрытости	118
Знакомство с паттерном Декоратор	120
Построение заказанного напитка	121
Определение паттерна Декоратор	123
Декораторы и напитки	124
Пишем код для Starbuzz	127
Программируем классы напитков	128
Программирование дополнений	129
Декораторы 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 OO-выпечка

Приготовьтесь заняться выпечкой объектов в слабосвязанных OO-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..



Определение изменяемых аспектов	144
Инкапсуляция создания объектов	146
Построение простой фабрики для пиццы	147
Переработка класса PizzaStore	148
Определение Простой Фабрики	149
Принятие решений в subclasses	153
Объявление фабричного метода	157
Пора познакомиться с паттерном Фабричный Метод	163
Параллельные иерархии создателей и продуктов	164
Определение паттерна Фабричный Метод	166
Зависимости между объектами	170
Принцип инверсии зависимостей	171
Применение принципа	172
Семейства ингредиентов	177
Построение фабрик ингредиентов	178
Перерабатываем классы пиццы...	181
Возвращаемся к пиццериям	184
Чего мы добились?	185
Определение паттерна Абстрактная Фабрика	188
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! Но не стоит расслабляться; несмотря на простоту с точки зрения структуры классов, его реализация требует довольно серьезных объектно-ориентированных размышлений. Приготовьтесь пошевелить мозгами — и за дело!

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

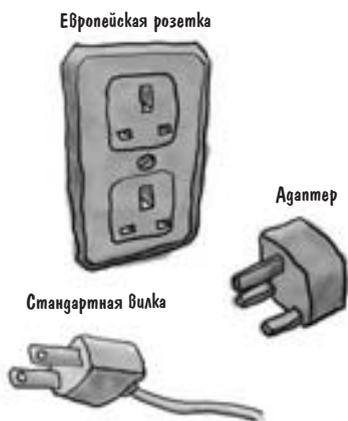
В этой главе мы выходим на новый уровень инкапсуляции — на этот раз будут инкапсулироваться вызовы методов. Да, все верно — вызывающему объекту не нужно беспокоиться о том, как будут выполняться его запросы. Он просто использует инкапсулированный метод для решения своей задачи. Инкапсуляция позволяет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

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 Объектвиля	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
Пора протестировать кнопку отмены!	248
Реализация отмены с состоянием	249
Реализация отмены в командах управления вентилятором	250
Переходим к тестированию вентилятора	251
На каждом пульте должен быть Режим Вечеринки!	253
Использование макрокоманд	254
Расширенные возможности паттерна Команда: очереди запросов	257
Расширенные возможности паттерна Команда: регистрация запросов	258
Новые инструменты	260
Ответы к упражнениям	261

# 7 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	266
Объектно-ориентированные адаптеры	267
Как работает паттерн Адаптер	271
Определение паттерна Адаптер	273
Адаптеры объектов и классов	274
Беседа у камина: Адаптер объектов и Адаптер классов	277
Практическое применение адаптеров	278
Адаптация перечисления к итератору	279
Беседа у камина: паттерн Декоратор и паттерн Адаптер	282
Домашний кинотеатр	285
Просмотр фильма (сложный способ)	286
Свет, камера, фасад!	288
Построение фасада для домашнего кинотеатра	291
Реализация упрощенного интерфейса	292
Просмотр фильма (простой способ)	293
Определение паттерна Фасад	294
Принцип минимальной информированности	295
Новые инструменты	300
Ответы к упражнениям	301



## 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 субклассы могли в любой момент связаться с нужным алгоритмом обработки. В этой главе даже будет описан принцип проектирования, вдохновленный голливудской практикой.

Кофе и чай (на языке Java)	305
Абстрактный кофе и чай	308
Продолжаем переработку...	309
Абстрагирование prepareRecipe()	310
Что мы сделали?	313
Паттерн Шаблонный Метод	314
Готовим чай...	315
Что дает Шаблонный Метод?	316
Определение паттерна Шаблонный Метод	317
Перехватчики в паттерне Шаблонный Метод	320
Голливудский принцип	324
Голливудский принцип и Шаблонный Метод	325
Шаблонные методы на практике	327
Сортировка на базе Шаблонного Метода	328
Сортируем уток...	329
Что делает метод compareTo()?	329
Сравнение объектов Duck	330
Пример сортировки	331
Как сортируются объекты Duck	332
Шаблонный метод в JFrame	334
Специализированные списки и AbstractList	335
Беседа у камина: Шаблонный Метод и Стратегия	336
Новые инструменты	338
Ответы к упражнениям	339

# 9 Паттерны Итератор и Компоновщик

## Управляемые коллекции

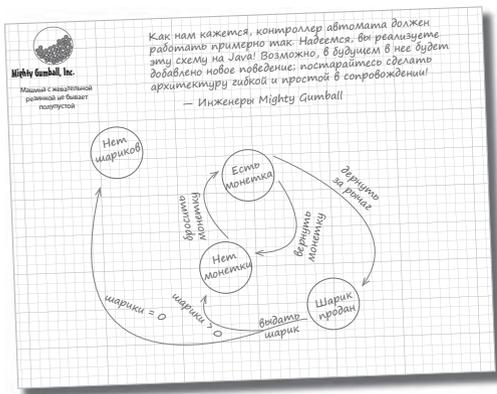
Существует много способов создания коллекций. Объекты можно разместить в контейнере Array, Stack, List, HashMap — выбирайте сами. Каждый способ обладает своими достоинствами и недостатками. Но в какой-то момент клиенту потребуется перебрать все эти объекты, и когда это произойдет, собираетесь ли вы раскрывать реализацию коллекции? Надеюсь, 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.

Быстро объединяется с блинной!	342
Две реализации меню	344
Спецификация официантки с поддержкой Java	346
Как инкапсулировать перебор элементов?	349
Паттерн Итератор	351
Добавление итератора в DinerMenu	352
Переработка DinerMenu с использованием итератора	353
Исправление кода Waitress	354
Тестирование кода	355
Взгляд на текущую архитектуру	357
Интеграция с java.util.Iterator	359
Определение паттерна Итератор	362
Структура паттерна Итератор	363
Принцип одной обязанности	364
Знакомьтесь: интерфейс Iterable языка Java	367
Расширенный цикл for в языке Java	368
Знакомство с классом CafeMenu	371
Итераторы и коллекции	377
Определение паттерна Компоновщик	384
Реализация MenuComponent	388
Реализация MenuItem	389
Реализация комбинационного меню	390
Новые инструменты	399
Ответы к упражнениям	400

# 10

## Паттерн Состояние Состояние дел

**Малоизвестный факт:** паттерны **Стратегия** и **Состояние** — близнецы, разлученные при рождении. Можно было бы ожидать, что их жизненные пути будут похожими, но паттерну Стратегия удалось построить невероятно успешный бизнес с взаимозаменяемыми алгоритмами, тогда как паттерн Состояние выбрал, пожалуй, более благородный путь — он помогает объектам управлять своим поведением за счет изменения своего внутреннего состояния. Но какими бы разными ни были их пути, в их внутренней реализации используются практически одинаковые архитектуры. Разве такое возможно? Как вы увидите, Стратегия и Состояние используются для совершенно разных целей. Сначала мы присмотримся к паттерну Состояние и разберемся в его сути, а в оставшейся части главы будем изучать отношения между этими паттернами.



Техника на грани фантастики	404
Краткий курс конечных автоматов	406
Программирование	408
Внутреннее тестирование	410
Кто бы сомневался... запрос на изменение!	412
Новая архитектура	416
Определение интерфейса State и классов	417
Реализация классов состояний	419
Переработка класса GumballMachine	420
А теперь полный код класса GumballMachine...	421
Реализация других состояний	422
Определение паттерна Состояние	428
Реализация игры «1 из 10»	431
Завершение реализации игры	432
Демонстрация для начальства	433
Проверка разумности	435
Новые инструменты	441
Ответы к упражнениям	442



# 11

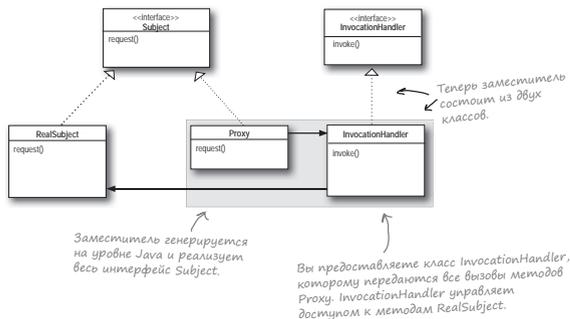
## Паттерн Заместитель

### 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 по каждому пустяку. Поэтому вы обзаводитесь «плохим полицейским», который управляет доступом к вам. Именно этим и занимаются заместители: они управляют доступом. Как вы вскоре увидите, существует множество способов взаимодействия заместителей с обслуживаемыми объектами. Иногда заместители пересылают по интернету целые вызовы методов, а иногда просто терпеливо стоят на месте, изображая временно отсутствующие объекты.



Программирование монитора	449
Тестирование монитора	450
Роль «удаленного заместителя»	452
Включение удаленного заместителя в код мониторинга GumballMachine	454
Регистрация в реестре RMI...	470
Определение паттерна Заместитель	477
Знакомьтесь: Виртуальный Заместитель	479
Отображение обложек альбомов	480
Класс ImageProху	482
Создание защитного заместителя средствами Java API	491
Знакомства для гиков	492
Реализация Person	493
Общая картина: динамический заместитель для Person	496
Тестирование службы знакомств	501
Запуск кода...	502
Разновидности заместителей	504
Новые инструменты	506
Код просмотра обложек альбомов	509



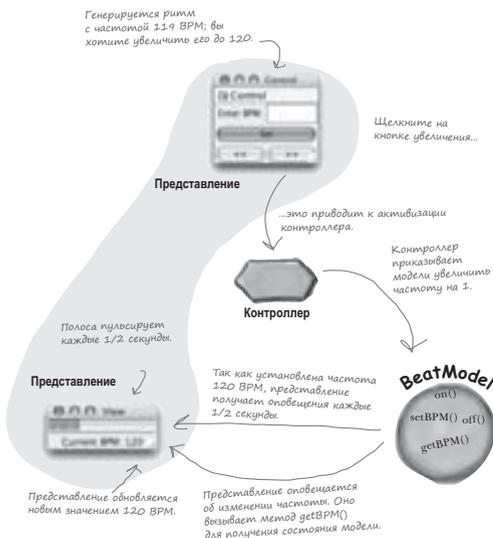
# 12

## 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.

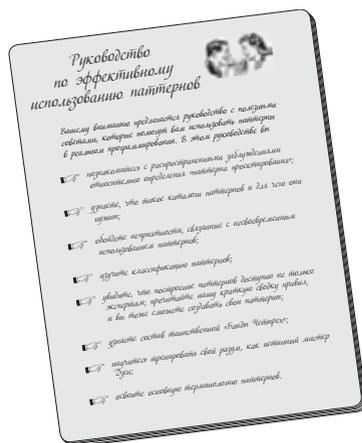
Совместная работа паттернов	514
И снова утки	515
Диаграмма классов с высоты утиноного полета	538
Паттерны проектирования — ключ к MVC	540
MVC как набор паттернов	544
Использование MVC для управления ритмом...	546
Построение компонентов	549
Представление	551
Анализ паттерна Стратегия	557
Адаптация модели	558
Можно переходить к HeartController	559
Новые инструменты	563
Ответы к упражнениям	564
Готовый код	567



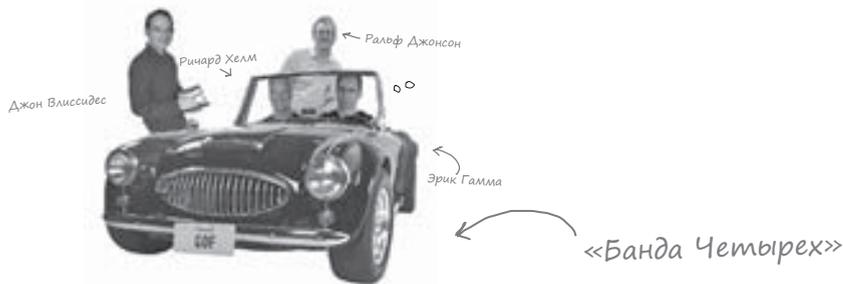
# 13 Паттерны для лучшей жизни

## 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 Объективиле. К счастью, у вас имеется хороший путеводитель, который упростит ваши первые шаги...



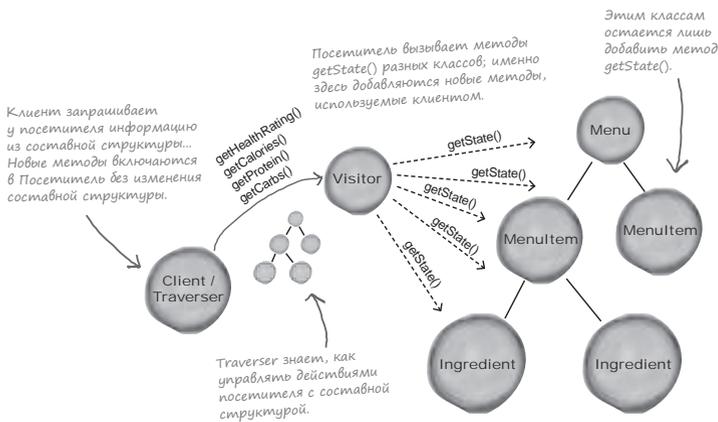
Определение паттерна проектирования	583
Подробнее об определении паттерна проектирования	585
Хотите создавать паттерны?	591
Классификация паттернов проектирования	593
Мыслить паттернами	598
Разум и паттерны	601
Разум новичка	601
Просветленный разум	601
Разум опытного разработчика	601
И не забудьте о единстве номенклатуры	603
Прогулка по Объективилу с «Бандой Четырех»	605
Наше путешествие только начинается...	606
Разновидности паттернов	608
Антипаттерны и борьба со злом	610
Новые инструменты	612
Покидая Объективиль...	613



# 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	616
Строитель	618
Цепочка Обязанностей	620
Приспособленец	622
Интерпретатор	624
Посредник	626
Хранитель	628
Прототип	630
Посетитель	632