Содержание (сводка)

Содержание (настоящее)

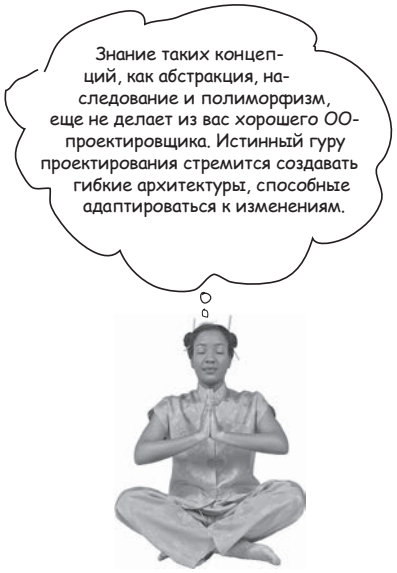
Введение

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?

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*.

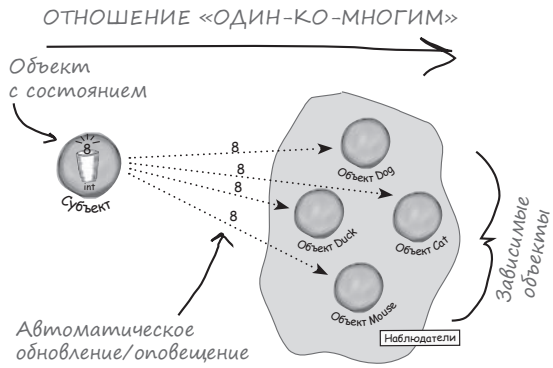


2

Паттерн Наблюдатель

Объекты в курсе событий

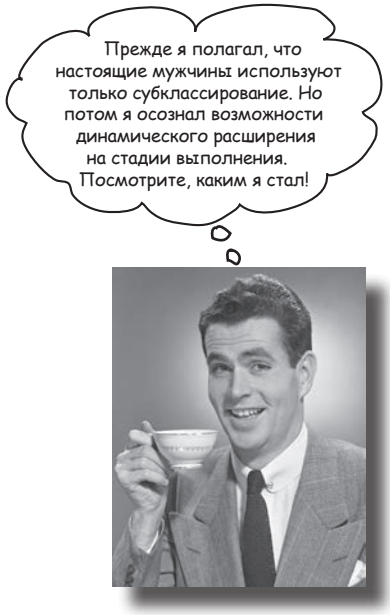
Не упустите, когда происходит что-то интересное! Наш следующий паттерн оповещает объекты о наступлении неких событий, которые могут представлять для них интерес, — причем объекты даже могут решать во время выполнения, желают ли они и дальше получать информацию. Паттерн Наблюдатель чрезвычайно полезен 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.



3 Паттерн Декоратор

Украшение объектов

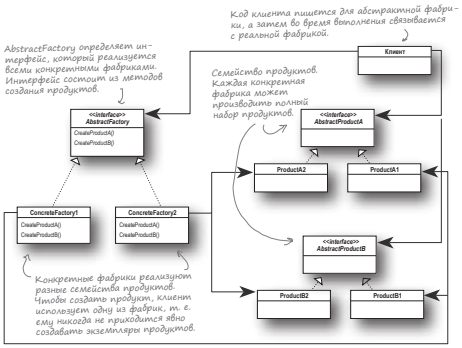
Эту главу можно назвать «Взгляд на архитектуру для любителей наследования». Мы проанализируем типичные злоупотребления из области наследования, и вы научитесь декорировать свои классы во время выполнения с использованием разновидности композиции. Зачем? Затем, что этот прием позволяет вам наделить свои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



4 Паттерн Фабрика

Домашняя ОО-выпечка

Приготовьтесь заняться 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.



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!

6

Паттерн Команда

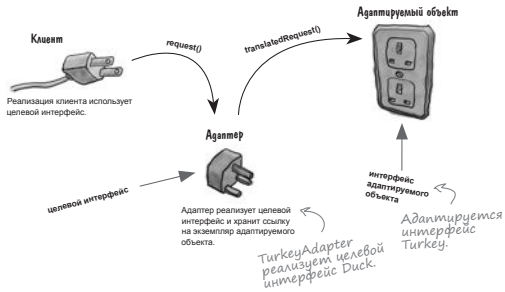
Инкапсуляция вызова

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.

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.



8 Паттерн Шаблонный Метод

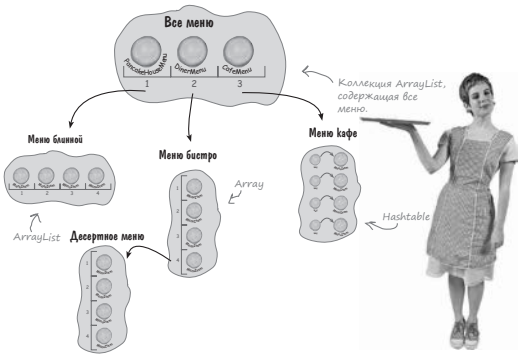
Инкапсуляция алгоритмов

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

9 Паттерны Итератор и Компоновщик

Управляемые коллекции

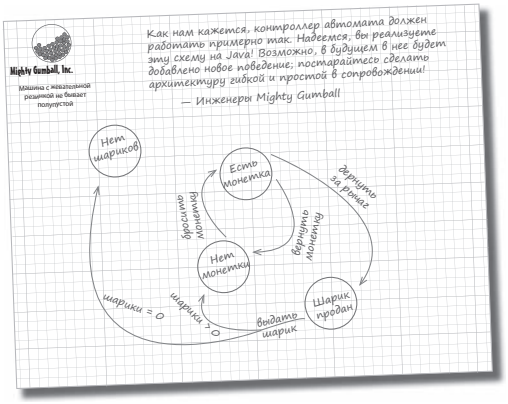
Существует много способов создания коллекций. Объекты можно разместить в контейнере Array, Stack, List, Hashtable — выбирайте сами. Каждый способ обладает своими достоинствами и недостатками. Но в какой-то момент клиенту потребуется перебрать все эти объекты, и, когда это произойдет, собираетесь ли вы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



10 Паттерн Состояние

Состояние дел

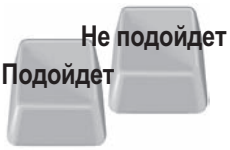
Малоизвестный факт: паттерны Стратегия и Состояние — близнецы, разлученные при рождении. Как известно, паттерн Стратегия организовал успешный бизнес в области взаимозаменяемых алгоритмов. Паттерн Состояние выбрал, пожалуй, более благородный путь: он помогает объектам управлять своим поведением посредством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

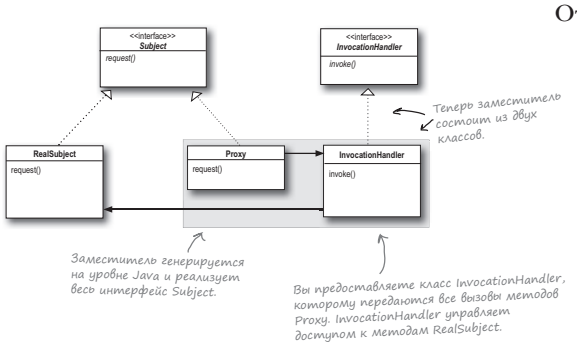


11 Паттерн Заместитель

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.

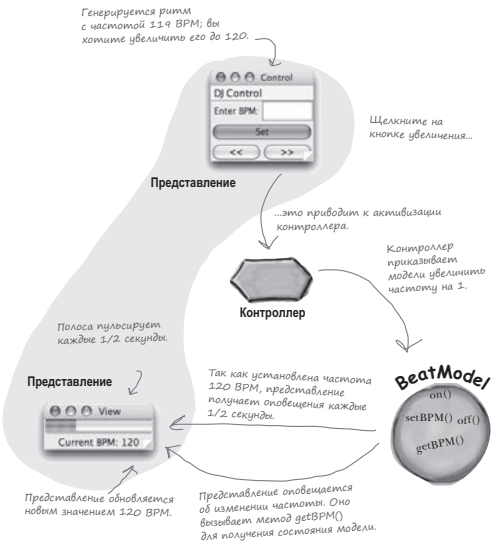




12 Составные паттерны

Паттерны паттернов

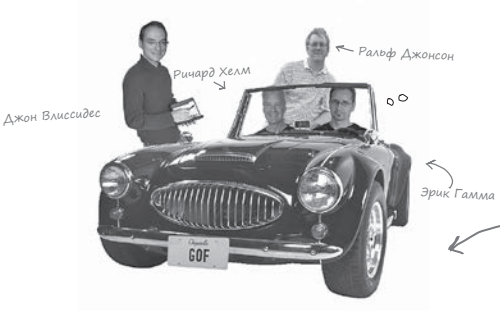
Кто бы мог предположить, что паттерны порой работают рука об руку? Вы уже были свидетелями ожесточенных перепалок в «Беседах у камина» (причем вы не видели «Смертельные поединки» паттернов — редактор заставил нас исключить их из книги!). И после этого оказывается, что мирное сосуществование все же возможно.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



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...



«Банда Четырех»

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.