Введение

Что отличает профессионального разработчика? Короткий ответ на этот вопрос очевиден: профессиональный разработчик создает качественный код и делает это надежно. Но намного менее очевидно, каким образом профессиональный разработчик достигает этой цели. Для этого недостаточно знать все технические подробности языка программирования и его библиотек, поскольку это не помогает принимать стратегические решения и нормально общаться в команде разработчиков. Недостаточно также знать терминологию разработки программного обеспечения и проектирования его архитектуры, поскольку в этой терминологии отсутствуют указания на конкретную реализацию. Недостаточно также проштудировать каталоги проектных шаблонов, поскольку они нацелены на решение отдельных задач и неверно понимаются и применяются вне своего контекста. Напротив, профессиональный разработчик должен иметь твердые знания в этих и многих других областях. Он должен ясно видеть возникающие связи и уметь вовремя рассматривать их под разными углами зрения. Код, который он в конечном итоге создает, должен лишь отражать огромное количество исходных соображений по поводу самых разных подробностей, которые нередко взаимосвязаны едва уловимо.

В этой книге преследуется цель пройти малоисследованную местность на пути к профессионализму, который лежит перед разработчиком, только что окончившем вводный курс программирования, университетский курс по компьютерным наукам или поступившем на свою первую работу. В ней представлены основные темы, которые оказались наиболее уместными за последние тридцать лет с того момента, когда был повсеместно принят объектно-ориентированный подход к разработке программного обеспечения. Кроме того, в книге освещаются важные вопросы на основании моего пятнадцатилетнего опыта преподавания дисциплин программной инженерии на всех уровнях университетских курсов и работы в самых разных программных проектах.

Основная тема книги: код и принципы его разработки

Основная тема этой книги — объектно-ориентированная разработка в частности и разработка программного обеспечения вообще, поэтому для ее раскрытия необходимо рассмотреть принципы разработки вместе с конкретным кодом. Без соблюдения этих принципов код будет иметь произвольную, непредсказуемую структуру. Эти принципы позволяют анализировать код и поддерживать его ясность и сопровождаемость. На их основании

можно принимать важные проектные и реализационные решения. Короче говоря, они поясняют, почему код выглядит именно так, а не иначе.

В области объектно-ориентированной разработки существует достаточное количество проверенных временем принципов. Приведем лишь некоторые примеры. В простейшем случае принцип замены "вызовов методов" на "сообщения" помогает сохранять независимость объектов. Подход к проектированию объектов с учетом их "обязанностей" в более крупной сети объектов объясняет, каким образом даже небольшие объекты могут взаимодействовать для создания надежного приложения. И тогда оказывается, что сети объектов нередко следуют "шаблонам", позволяющим постоянно и надежно решать стандартные задачи. Принцип описания вызовов методов по "контрактам" служит надежным руководством к получению правильного кода а "каркасы" и "инверсия управления" стали важными понятиями и принципами для эффективного построения крупных приложений.

Несмотря на то что принципы полезны и даже необходимы для написания качественного объектно-ориентированного кода, для их правильного претворения в код требуется немало прилежания, проницательности и опыта. Опыт преподавания подсказывает, что принципы легко понять неверно, и даже незначительные отклонения в их истолковании могут иногда иметь катастрофические последствия. В действительности один и тот же урок годится для многих практических занятий и вводных пояснений. Например, пояснение известного шаблона МОДЕЛЬ–ПРЕДСТАВЛЕНИЕ–КОНТРОЛЛЕР нередко сопровождается "минимальным" примером реализации. В некоторых случаях применения этого шаблона можно было наблюдать, что модель содержит ссылки на конкретный класс представления, а также его отдельный экземпляр. Подобные просчеты нарушают весь шаблон и сводят на нет его преимущества. Даже если код работоспособен, этого еще недостаточно, чтобы считать его создателя профессиональным разработчиком.

Код и принципы его разработки очень важны и должны быть тесно взаимосвязаны, поэтому в этой книге они так и рассматриваются. Для рассмотрения каждой темы вводятся центральные понятия и на ряде примеров поясняется общее положение дел. После этого сразу же показывается, каким образом принципы претворяются в конкретном коде. Мы не ограничиваемся минимальными примерами, а исследуем также более сложные вопросы. Так, шаблон МОДЕЛЬ–ПРЕДСТАВЛЕНИЕ–КОНТРОЛЛЕР нетрудно правильно понять на небольших примерах его применения. Но как только модели усложняются, профессиональный разработчик добивается того, чтобы повторно воспроизводились только изменившиеся их части. Аналогично не составит особого труда присоединить слушатель событий к экранной кнопке, но профессиональный разработчик должен исключить "застывания" изображения при выполнении длительных операций. Это, в свою очередь, требует распараллеливания выполняемых операций.

Разумеется, "минимальные" примеры таят в себе опасность пропустить что-нибудь важное. Поэтому там, где это возможно, мы представляем в книге код, взятый из интегрированной среды разработки Eclipse, и освещаем

те его элементы, которые отражают рассматриваемый принцип. У такого подхода есть еще одно важное преимущество: он демонстрирует рассматриваемые принципы в действии и в нужном контексте. Ведь зачастую подлинная ценность отдельного подхода, а иногда и его оправданность, проявляется только в по-настоящему крупных приложениях. Например, очень важно поддерживать расширяемость программного обеспечения. Но убедительные примеры расширяемости можно найти только в таких модульных системах, как Eclipse. И наконец, если читателю необходимо рассмотреть интересующий его вопрос более подробно, он может обратиться непосредственно к источникам, упоминаемым в книге.

С результирующим кодом связана еще одна завершающая история, которая обычно умалчивается: как на самом деле написан код. Профессиональные разработчики могут существенно увеличить свою производительность, если будут не набирать исходный код вручную, а применять специальные приемы, принуждающие интегрированную среду разработки генерировать код автоматически. Например, очень полезно знать принцип "рефакторинга кода". Но профессиональные разработчики должны также владеть инструментальными средствами рефакторинга кода, доступными в интегрированной среде разработки, вплоть до способности добиться желаемых изменений в коде, применяя три отдельных инструментальных средства подряд. Поэтому по теме кода и принципов его разработки мы осветим также инструментальные средства Eclipse, в которых применяется каждый рассматриваемый принцип.

Структура книги

Книга разделена на четыре части. В каждой из них излагаемая тема объектно-ориентированной разработки рассматривается путем перехода от "небольших" особенностей отдельных языковых средств к более "крупным" вопросам разработки программного обеспечения и проектирования его архитектуры. В них даются также дополнительные ответы на один и тот же вопрос: как должен выглядеть профессионально спроектированный "объект"?

Часть I. Применение языка. Грамотно написанный код всегда начинается с профессионального применения языка программирования. Профессиональный разработчик применяет языковые средства, исходя из своих намерений, а не злоупотребляя ими ради ловких, на первый взгляд, приемов и трюков. Термин "применение" на самом деле означает то же самое, что и "применение словаря" естественных языков. Так, если код подчиняется идиомам, фразам и скрытому подтексту языковых конструкций, он становится более удобочитаемым, понятным и сопровождаемым.

Часть II. Контракты. Профессионально написанный код должен быть, прежде всего, надежным. Он должен работать во всех ситуациях, для которых создан, а сами эти ситуации должны быть ясными. Принцип проектирования по контракту служит прочным основанием для необходимых рассуждений. Он распространяется от описаний методов на высоком уровне

вплоть до подробностей формальной верификации программного обеспечения. В качестве дополнительной меры поведение объектов должно устанавливаться всесторонним тестированием.

Часть III. События. Программное обеспечение любого масштаба обычно управляется событиями. Функциональные возможности приложения активизируются некоторым каркасом, устанавливающим общую структуру и основополагающие механизмы. По существу, интерпретация методов изменяется в сравнении с тем, что изложено в части II. В частности, метод не реализует службу, выполняющую конкретный запрос из вызывающего кода, но для вызываемого кода важнее подходящая реакция на запрос. Мы рассмотрим этот принцип в конкретной области пользовательских интерфейсов, а также уделим особое внимание архитектурным соображениям касательно центрального понятия разделения модели и представления в данной области. Практически во всех приложениях приходится одновременно выполнять многие операции, и поэтому материал этой части дополняется кратким введением в многопоточную обработку.

Часть IV. Проектирование на основе обязанностей. Одна из целей объектно-ориентированной разработки заключается в том, чтобы сохранять отдельные объекты небольшими и управляемыми. Чтобы решить задачу любой сложности, необходимо организовать правильное взаимодействие многих объектов. Метафора присваивания "обязанностей" отдельным объектам в таких крупных сетях оказалась особенно удобной и теперь находит широкое применение в разработке программного обеспечения. После вводной главы по проектированию объектов и организации их взаимодействия мы исследуем последствия такого подхода для принятия стратегических и архитектурных решений.

Все четыре части этой книги предназначены для того, чтобы дать всестороннее представление об объектно-ориентированной разработке. В них поясняется роль отдельных объектов в общей структуре приложения, их реакции на входящие события, правильное выполнение ими запросов на отдельные услуги, а также их роль в более крупном контексте приложения в целом.

Как читать эту книгу

Как пояснялось ранее, тема объектно-ориентированной разработки программного обеспечения имеет немало граней и особенностей. Более того, отдельные ее вопросы тесно переплетаются, образуя единое целое. В первоначальных описаниях объектно-ориентированного программирования подчеркивалось, что разработчику среднего уровня потребуется больше года в рамках отдельного проекта, чтобы получить общее представление о том, что в действительности сулит такой подход к программированию. Очевидно, что это вряд ли могло удовлетворить потребности серьезных разработчиков.

В этой книге предпринята попытка упростить в как можно большей степени ее чтение. Общая цель книги — дать читателю возможность пользо-

ваться ею в качестве справочного руководства. Читатель может обращаться к книге в поисках ответов на конкретные вопросы, не читая ее от корки до корки. В то же время эта книга относится к категории обычной технической литературы, а следовательно, ее можно читать, постепенно переходя от одной части к другой. Удобству чтения могут способствовать следующие составляющие ее назначения.

- **Многоуровневое изложение.** Материал каждой главы, раздела и подраздела излагается от общих положений к отдельным подробностям, от важных заключений к дополнительным комментариям. В итоге читатель можете прервать чтение, как только почувствует, что в достаточной степени усвоил излагаемый материал, чтобы вернуться к нему впоследствии для дополнительной проработки.
- **Основные разделы.** Каждая глава начинается с самостоятельного раздела, где поясняются основные принципы, рассматриваемые в этой главе. Благодаря этому материал последующих глав становится более понятным после прочтения основных разделов предыдущих глав. Читая основные разделы всех глав, вы получаете "одну книгу в другой", т.е. общий обзор принципов объектно-ориентированной разработки. Сами основные разделы сохранены как можно более компактными, и поэтому их можно читать за один присест.
- **Краткие итоги.** Каждый вопрос, поясняемый и раскрываемый в книге, сопровождается кратким итогом в одном предложении, специально выделенном во врезке. Эти итоги дают общее представление о рассматриваемой теме и служат удобным основанием для непосредственного перехода в процессе непрерывного изложения материала книги.
- **Самостоятельные очерки.** Все разделы и подразделы написаны как самостоятельные единицы изложения отдельных тем. После прочтения основного раздела главы можете сразу же переходить к тем вопросам, которые вас интересуют больше всего.
- **Изложение, ориентированное на конкретные цели.** Содержание книги отражает конкретные цели разработки: как писать качественные методы, пользоваться наследованием и интерфейсами, структурировать приложение, как организовать многопоточную обработку, обращаться с графическими пользовательскими интерфейсами, добиться гибкости программного обеспечения. Все остальное подчинено этим целям. В частности, проектные шаблоны представлены в том контексте, где они приносят наибольшую пользу. Они поясняются очень кратко с целью быстро передать их основное назначение, но для полноты изложения в книге всегда делаются ссылки на первоисточники шаблонов.
- **Обширные перекрестные ссылки.** Непосредственный переход посредине изложения материала означает, что вы можете пропустить чтение каких-то основ, если они вам уже известны. Чтобы упростить

чтение книги, весь излагаемый в ней материал снабжен обратными ссылками на предварительные условия для чтения. Так, если вас озадачит неизвестное понятие, вы будете знать, где найти его объяснение по ссылке. Как правило, рекомендуется также прочитать основной раздел главы, упоминаемой в ссылке. С другой стороны, многие вводные темы снабжены указателями на более подробный материал, дающий дополнительные разъяснения по теме. В частности, в основных разделах указываются дополнительные сведения по отдельным вопросам.

Для перекрестных ссылок используются следующие знаки.

📖 Ссылка на литературу с дополнительными сведениями или основополагающими определениями по порядку их уместности.

↰ Ссылка на предыдущие пояснения (как правило, предварительные условия для чтения).

⏩ Ссылка на последующий материал, где рассматриваются дополнительные вопросы или подробности.

Кроме того, многие абзацы в тексте отделены от обычного изложения материала следующими знаками.

Важные подробности, нередко упускаемые из виду начинающими разработчиками. В таком случае они нарушают более крупные цели рассматриваемой темы

Здесь можно найти представление или связь с отдельным принципом. Подобные представления составляют ряд принципов, образующих область объектно-ориентированной разработки

Представление о предыдущей теме, приобретающее новое и полезное значение в свете текущего обсуждения

Дополнительный комментарий с подробностями о том, что могло (или не могло) бы озадачить читателя. Например, конкретная подробность во фрагменте кода может потребовать дополнительных пояснений, если присмотреться к ней внимательнее

Момент принятия решения. При разработке программного обеспечения нередко приходится принимать решения. Там, где обычное изложение материала обходит молчанием заслуживающие внимания альтернативы, мы делаем их явными

Изящное применение отдельных инструментальных средств — как правило, для повышения производительности или в целях экономии, но не срезая углы

Эффект небольшого обзора [259], который может возникнуть, если выбрать другой язык, кроме Java, или вообще отказаться от объектно-ориентированного программирования. Зачастую характерные особенности объектов в Java лучше познаются в сравнении

Рекомендации по обучению с помощью данной книги

Настоящая книга стала результатом ряда лекций, прочитанных автором по курсу компьютерных наук в Тюбингенском университете в 2005–2014 гг. Эти лекции имеют самую разную тематику: от вводных курсов по программированию на Java до объектно-ориентированного программирования, разработки программного обеспечения и проектирования его архитектуры. Для этой книги были выбраны те темы, которые вероятнее всего помогут студентам в их будущей карьере разработчиков программного обеспечения. В то же время я решил изложить темы в этой книге значительно глубже, чем предполагают университетские курсы. Но особенно сложные вопросы были оставлены для пояснения в последующих разделах каждой главы и поэтому могут быть пропущены.

Если же вам нужен обычный учебный курс, то, возможно, будет интересно узнать, что краткие итоги фактически стали результатом переработки слайдов и заметок на доске. Материал книги следует стилю чтения лекций. В частности, после пояснения принципиальные положения демонстрируются на примере конкретного кода. Фрагменты кода, приведенные в книге, взяты из Eclipse или доступны в Интернете в виде приложения по адресу `http://coding-concepts.org/code.jsf`.

Как пояснялось ранее, изложение проектных шаблонов в этой книге направлено на упрощение ее чтения, сосредоточено на главных особенностях шаблонов и тесно привязано к контексту, в котором они применяются. Разумеется, альтернативой этому могло бы быть традиционное изложение с формализованной структурой имен, описанием намерений, объяснением мотиваций, формулировкой выводов и перечислением взаимосвязанных проектных шаблонов [59, 100, 263]. В этой книге выбран сравнительно неформальный стиль изложения, поскольку он, на мой взгляд, помог моим студентам лучше пояснить назначение и применение проектных шаблонов на устных экзаменах или практических занятиях по проектированию. На более крупных курсах с письменными экзаменами я нередко выбирал более формализованное изложение материала, чтобы дать возможность студентам лучше подготовиться к экзаменам. Для подобных случаев в материале по каждому проектному шаблону в книге делается ссылка на первоначальную публикацию.

Многоуровневое изложение материала книги дает возможность выбрать любой ряд тем, наиболее подходящих для вас в конкретной ситуации.

Возможно, вам будет полезно также знать, какие именно разделы книги были использованы вместе на отдельных курсах.

- **CompSci2.** Это вводный курс по программированию, имеющий в основном отношение к синтаксису и возможностям Java, а также к объектно-ориентированному программированию (разделы 1.3–1.5). Для большей убедительности он дополнен материалом, посвященным событийному программированию пользовательских интерфейсов (раздел 7.1). В ходе этого курса было использовано представление объектов как взаимодействующих элементов, берущих на себя конкретные обязанности (раздел 11.1). Такое всестороннее пояснение дало студентам возможность написать небольшие визуальные игры в конце курса.
- **Программная инженерия.** Этот курс дает обширное представление о практической программной инженерии, чтобы подготовить студентов к расширенному проекту в последующих семестрах. Поэтому я уделил основное внимание принципам объектно-ориентированной разработки (разделы 11.1, 11.2.1 и 11.5.1). Чтобы положить хорошее начало, я изложил студентам технические особенности, с которыми им придется иметь дело в последующих проектах: графические пользовательские интерфейсы (раздел 7.1), включая принцип разделения модели и представления (раздел 7.3), а также вопросы применимости длительных заданий (раздел 7.10). Я также изложил основополагающие принципы проектирования, приводящие к созданию сопровождаемого кода (раздел 11.5), уделив внимание принципу единственной ответственности (раздел 11.2.1) для отдельных объектов и принципу подстановки Лисков (раздел 3.1.1). В ходе этого курса я рассмотрел известные шаблоны, включая НАБЛЮДАТЕЛЬ (раздел 2.1), КОМПОНОВЩИК (раздел 2.3.1), АДАПТЕР (раздел 2.4.1), ЗАМЕСТИТЕЛЬ (раздел 2.4.3), УРОВНИ (раздел 12.2.2) и КАНАЛЫ И ФИЛЬТРЫ (раздел 12.3.4).
- **Объектно-ориентированное программирование.** Этот бакалаврский курс основан на упоминавшемся выше курсе CompSci2 и посвящен дополнительным навыкам программирования. В нем подробно рассматриваются особенности объектно-ориентированной разработки (разделы 11.1, 11.2.1, 11.3.2 и 11.3.3) и реализации (разделы 1.2.1, 1.3–1.8). В силу их практической уместности в ходе этого курса рассматриваются пользовательские интерфейсы, включая специально воспроизводимые виджеты и шаблон МОДЕЛЬ–ПРЕДСТАВЛЕНИЕ–КОНТРОЛЛЕР (разделы 7.1, 7.2, 7.5, 7.8 и 9.2). Конечные автоматы послужили принципиальным основанием для событийно-ориентированного программирования (глава 10). Прочным основанием для этого курса стало обращение с контрактами и инвариантами, включая практически уместное понятие полей модели (раздел 4.1). На мой взгляд, практические примеры отлично подходят для изложения этих довольно абстракт-

ных тем (раздел 4.2), и заинтересовавшиеся студенты были рады последовать за мной в область формальной верификации (раздел 4.7.2).

- **Архитектура программного обеспечения, часть 1.** Этот курс посвящен основополагающим принципам структурирования программных продуктов. В начале делается краткий обзор объектно-ориентированного проектирования и разработки (разделы 11.1, 11.3.2 и 10.1). Затем рассматриваются основные архитектурные шаблоны согласно [59] и [218]: УРОВНИ, КАНАЛЫ И ФИЛЬТРЫ, МОДЕЛЬ–ПРЕДСТАВЛЕНИЕ–КОНТРОЛЛЕР, ПЕРЕХВАТЧИК (разделы 12.2.2, 9.2, 12.3.4 и 12.3.2). В силу их практической уместности я включил в этот курс рассмотрение команд Undo/Redo (раздел 9.5), а также общей структуры приложений с графическими интерфейсами (раздел 9.4). В завершение курса было вкратце рассмотрено проектирование гибкого, расширяемого и повторно используемого программного обеспечения (разделы 12.2–12.4).
- **Архитектура программного обеспечения, часть 2.** Этот курс охватывает вопросы параллельного программирования и распределенных систем. Из-за недостатка места в книгу включена лишь первая часть данного курса (раздел 7.10, глава 8).