Содержание (сводка)

Содержание (настоящее)

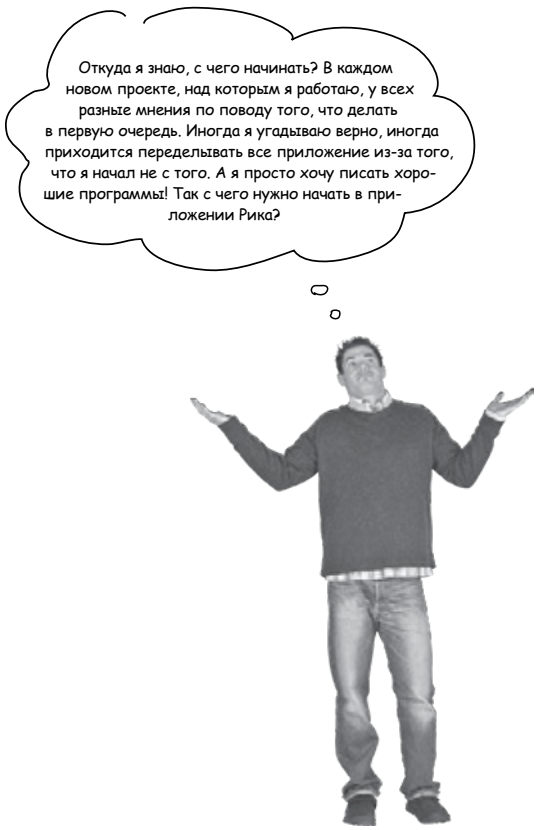
Введение

Ваш мозг думает об ООАП. Вы сидите за книгой и пытаетесь что-нибудь выучить, но ваш мозг считает, что вся эта писанина не нужна. Ваш мозг говорит: «Выгляни в окно! На свете есть более важные вещи, например сноуборд». Как заставить мозг думать, что ваша жизнь действительно зависит от объектно-ориентированного анализа и проектирования?

1 О пользе качественного проектирования

С чего начинаются хорошие программы

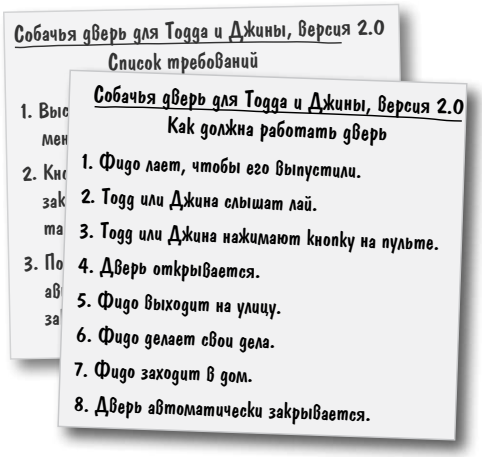
Так как же на самом деле пишутся хорошие программы? Всегда сложно понять, с чего следует начинать. Делает ли приложение то, что ему положено делать? И как насчет таких тонкостей, как дублирование кода, — ведь это всегда плохо, верно? Да, определить, **над чем следует работать в первую очередь**, бывает нелегко, к тому же еще нужно не запороть все остальное по ходу дела. Но пусть вас это не тревожит! К тому моменту, когда вы дочитаете эту главу, вы **будете знать**, **как пишутся хорошие программы**, а ваш подход к разработке приложений навсегда изменится. И наконец-то вы поймете, почему ООАП — это такое слово, которое не стыдно произносить в приличном обществе.



2 Сбор требований

Дайте им то, что они хотят

Хорошо, когда заказчик доволен. Вы ведь уже знаете, что написание хороших программ начинается с того, что вы проверяете, работает ли программа так, как хочет заказчик. Но как узнать, чего на самом деле хочет заказчик? И как убедиться в том, что заказчик вообще знает, чего он хочет? На помощь приходят хорошо составленные требования. В этой главе вы узнаете, как выполнить требования заказчика и предоставить ему именно то, чего он хотел. После этого на всех ваших проектах можно будет ставить «знак качества», а вы сделаете еще один большой шаг к заветной цели — написанию хороших программ... раз за разом.

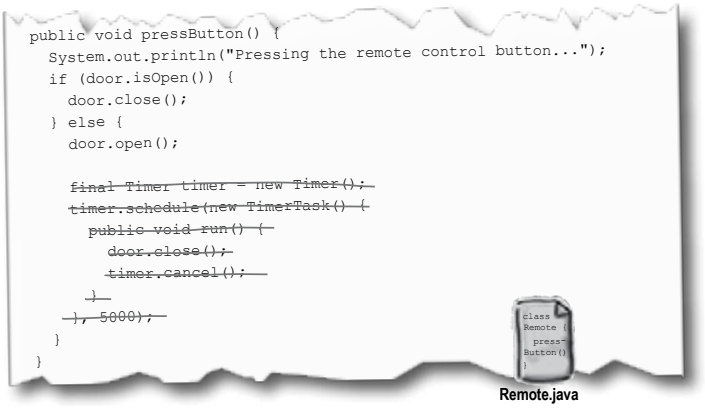


Дверь и пульт являются частью системы, то есть находятся в системе.

3 Изменение требований

Я тебя люблю, ты мой идеал... Теперь изменись

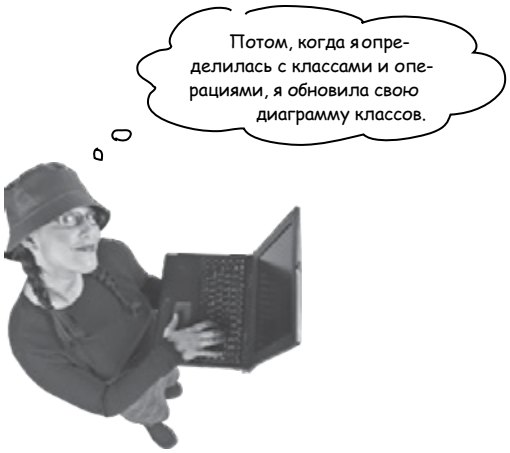
Думаете, вы сделали то, что нужно заказчику? Как бы не так... Вы поговорили с заказчиком, собрали требования, записали варианты использования и выдали убойное приложение. Можно расслабиться, верно? Верно... Пока заказчик не решит, что ему нужно **что-то отличное от того, о чем он говорил вам**. То, что вы сделали, ему понравилось, честно, но **теперь это не совсем то, что нужно**. В реальном мире **требования всегда изменяются**; вы должны адаптироваться к этим изменениям и сделать так, чтобы заказчик был доволен.



Анализ

4 Программы для реального мира

Пора переходить к реальным приложениям. Даже если ваше приложение идеально работает на машине разработки, этого недостаточно: приложения должны работать тогда, когда их используют реальные люди. В этой главе говорится о том, как заставить вашу программу работать в реальном контексте. Вы узнаете, как текстологический анализ преобразовывает вариант использования, над которым вы работали, в классы и методы, которые делают то, что нужно вашим заказчикам. А когда работа будет завершена, вы сможете сказать: «Да, у меня получилось! Моя программа готова к жизни в реальном мире!»



Реальный мир

Качественное проектирование = гибкость программного продукта

5 (часть 1)

Все течет, все меняется

Изменения неизбежны. Какой бы замечательной ни была ваша программа сегодня, завтра ее, скорее всего, придется изменять. И чем труднее вносить поправки в программу, тем сложнее реагировать на изменяющиеся потребности заказчика. В этой главе мы вернемся к старому знакомому, попробуем улучшить существующую программу и увидим, как маленькие изменения создают большие трудности. А при этом мы обнаружим проблему настолько значительную, что для ее решения потребуется глава ИЗ ДВУХ ЧАСТЕЙ!

5 (вставка)

5 (часть 2)

Хорошая структура = гибкость программы

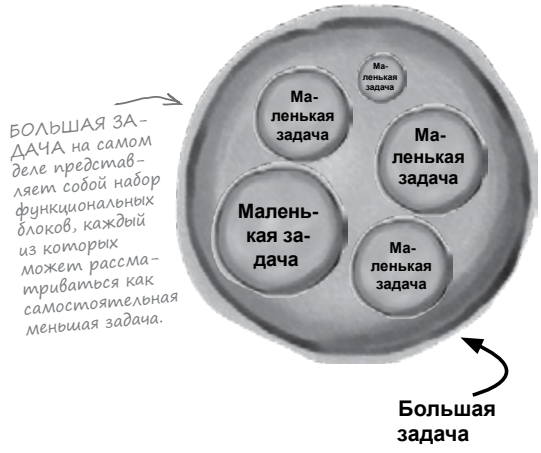
Зарядка для программ

А вам когда-нибудь хотелось обрести большую гибкость? Если внесение изменений в приложение создает проблемы, скорее всего, это означает, что вашей программе не хватает гибкости и пластичности. Чтобы помочь приложению прийти в хорошую форму, необходимо провести анализ, основательно проработать структуру кода приложения и узнать, как ОО-принципы способствуют ослаблению связности. А в завершение вы увидите, как высокое сцепление способствует ослаблению связности. Звучит заманчиво? Переверните страницу, и мы продолжим наводить порядок в нашем негибком приложении.

6 Решение больших задач

«Меня зовут Арт... И я архитектор»

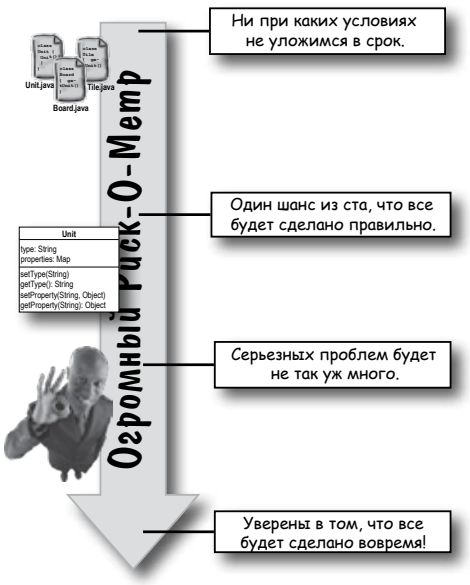
Пора построить нечто ДЕЙСТВИТЕЛЬНО БОЛЬШОЕ. Готовы? В вашем инструментарии ООАП скопилось множество инструментов, но как использовать эти инструменты при построении большого проекта? Возможно, вы этого и не осознали, но у вас имеется все необходимое для решения больших задач. В этой главе мы рассмотрим новые инструменты — анализ предметной области и диаграммы вариантов использования. Но даже эти инструменты основаны на том, что вам уже известно, — на необходимости прислушиваться к мнению заказчика и необходимости понимания того, что вы строите, до начала работы над кодом. Приготовьтесь... Пора поиграть в архитектора.



7 Архитектура

Навести порядок в хаосе

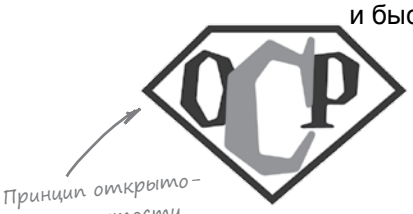
Любая работа с чего-то начинается, но вам стоит выбрать правильное что-то! Вы уже знаете, как разбить приложение на множество мелких задач, но это означает, что мелких задач будет действительно МНОГО. В этой главе мы поможем вам разобраться, с чего начать и как избежать потерь времени на движение в ошибочном направлении. Пришло время взять все эти мелкие детали, разложенные на вашем рабочем месте, и понять, как собрать из них четко структурированное, хорошо спроектированное приложение. Попутно вы узнаете о трех главных вопросах архитектуры и поймете, что «Риск» — это не только интересная настольная игра из 1980-х годов.



8 Принципы проектирования

Не стремитесь к оригинальности

Имитация — самая искренняя форма проявления ума. Ничто не доставляет столько удовлетворения, как создание совершенно нового, оригинального решения задачи, которая не давала вам покоя несколько дней. Пока не выяснится, что кто-то уже решил ту же задачу задолго до вас, да еще и сделал это лучше вас! В этой главе рассматриваются принципы проектирования, которые были созданы за прошедшие годы, и то, как они помогают вам стать более квалифицированным программистом. Откажитесь от мысли «сделать по-своему», в этой главе вы узнаете, как сделать умнее и быстрее.



Принцип открытости/закрытости

Принцип «не повторяйтесь»

Принцип единственной обязанности

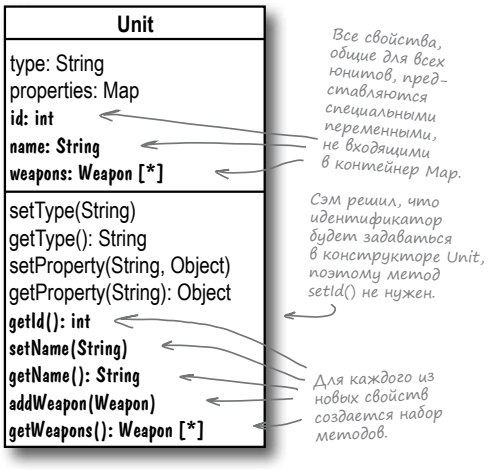
Принцип подстановки Лисков

9

Итерации и тестирование

Программы пишутся для заказчика

Пришло время показать заказчику, что вы действительно беспокоитесь о его интересах. Назойливое начальство? Обеспокоенные заказчики? Люди, которые постоянно спрашивают: «Ну что, успеете к сроку?» Никакой хорошо структурированный код не порадует вашего заказчика; вы должны показать ему что-то работающее. А теперь, когда вы овладели солидным инструментарием ОО-программирования, пора узнать, как доказать заказчику, что его программа действительно работает. В этой главе мы рассмотрим два способа углубленного анализа функциональности вашего продукта, после которого заказчик с чувством скажет: *«Да, вы определенно идеально подходите для этой работы!»*



10 Жизненный цикл ООАП

Все вместе

Далеко ли до цели? Мы изучили множество способов совершенствования программных продуктов, но пришло время собрать все воедино. Да, это именно тот момент, которого вы так долго ждали: мы возьмем все, о чем вы узнали из книги, и покажем, что в действительности это были лишь части общего процесса, который можно использовать вновь и вновь для написания хороших программ.

I

Приложение I. Остатки

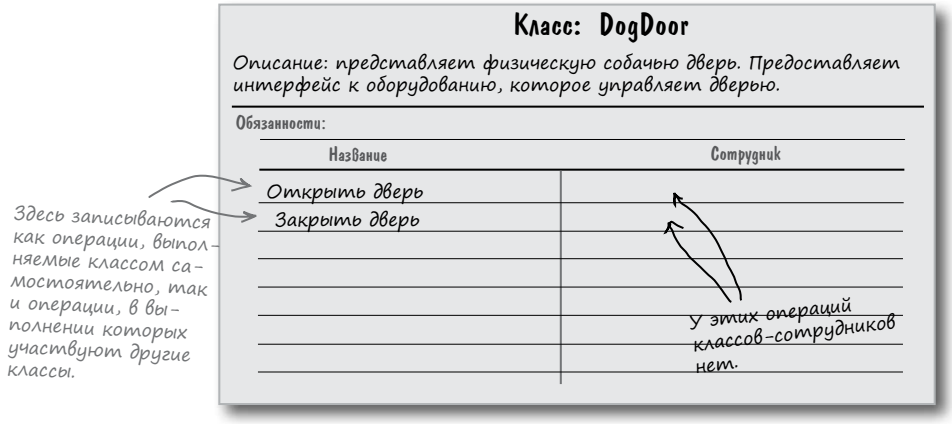
Десять главных тем (не рассмотренных в книге)

Хотите верьте, хотите нет, но это еще не все. Да, позади осталось более 550 страниц, и кое-что в них не уместилось. И хотя каждая из последних днсяти тем заслуживают разве что краткого упоминания, мы решили выдать вам на дорогу чуть больше информации по каждой из них. Теперь вам будет о чем поговорить во время рекламных пауз... да и кто не любит время от времени поговорить о ООАП?

А когда закончите читать это приложение, останется еще одно... И немного рекламы... А потом — действительно всё. Честное слово!

Антипаттерны

Антипаттерны являются противоположностью паттеров проектирования: это распространенные ПЛОХИЕ решения. Старайтесь своевременно распознавать ловушки и избегать их.



Приложение II. Добро пожаловать в Объективиль

Говорим на языке ООП

Приготовьтесь к путешествию в другую страну. Пришло время навестить Объективиль — место, где объекты делают то, что им положено, все приложения хорошо инкапсулированы (вскоре вы узнаете, что это означает), а программные структуры обеспечивают простоту расширения и повторного использования. Но прежде чем браться за дело, необходимо кое-что знать заранее — в частности, овладеть кое-какими языковыми навыками. Не беспокойтесь, это не займет много времени. Вы и не заметите, как начнете говорить на языке ООП так, словно вы уже давно живете в элитном районе Объективиля.