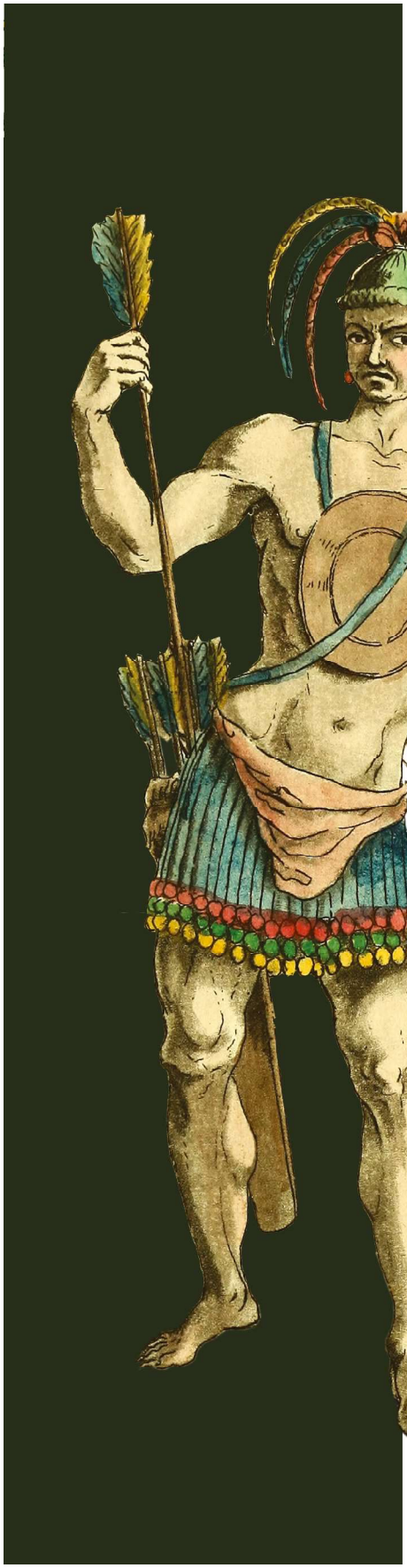
адаптировано для JAVA 11



Система модулей Java

Николай Парлог
предисловие Кевлина Хенни

MANNING

The Java Module System

NICOLAI PARLOG

Foreword by Kevlin Henney

MANNING

SHELTER ISLAND

Николай Парлог
предисловие Кевлина Хенни

Система модулей Java

ПИТЕР®
Санкт-Петербург • Москва • Екатеринбург • Воронеж
Нижний Новгород • Ростов-на-Дону • Самара • Минск

2021

ББК 32.973.2-018.1
УДК 004.438
П18

Парлог Николай

П18 Система модулей Java. — СПб.: Питер, 2021. — 464 с.: ил. — (Серия «Для профессионалов»).

ISBN 978-5-4461-1620-1

Создать надежное и безопасное приложение гораздо проще, если упаковать код в аккуратные блоки. Система модулей в Java представляет собой языковой стандарт для создания таких блоков. Теперь вы можете контролировать взаимодействия различных JAR и легко обнаруживать недостающие зависимости. Фундаментальные изменения архитектуры затронули ядро Java, начиная с версии 9. Все API ядра распространяются в виде модулей, а для библиотек, фреймворков и приложений аналогичный подход можно считать хорошей практикой и рекомендацией.

Вы освоите наилучшие практики модульного проектирования, отладки приложения и его развертывания перед сдачей в продакшен.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.973.2-018.1
УДК 004.438

ISBN 978-1617294280 англ.
ISBN 978-5-4461-1620-1

Краткое содержание

Оглавление

Часть II. Адаптация под реальные проекты

Часть III. Расширенные функции системы модулей

Приложения

Разве это не прекрасно, что все происходит по его желанию?[1]

Посвящается Габи и Майе

[1] Перевод отрывка из песни No Leaf Clover группы Metallica.

Предисловие

Стремление к модульности не ново. В 1968 году на знаковой конференции НАТО, посвященной программной инженерии, сыгравшей ключевую роль в популяризации видения программных компонентов и самого термина *«разработка ПО»* (software engineering), Э. Э. Дэвид (E. E. David) изложил подход к разработке больших систем:

> Определите подмножества системы, достаточно малые, чтобы ими можно было управлять, затем постройте из них подсистему. Эта стратегия подразумевает создание системы на основе модулей, которые могут быть написаны, протестированы и изменены без учета их взаимодействия друг с другом.

На этой же конференции Х. Р. Джиллетт (H. R. Gillette) рассказал, как модульность стала предпосылкой эволюции систем:

> Модульность помогает изолировать функциональные элементы системы. Один модуль можно отладить, улучшить или расширить с минимальным взаимодействием или разрывом системы.

Ничего нового. Просто понадобилось немного практики и несколько языков программирования, прежде чем эта тема была поднята и изучена.

История модульности Java, подобно фрагментам головоломки, разбросана во времени и пространстве и закодирована. Первой и существенной реализацией модулей в Java стал *класс*. В других языках он являл собой смесь модульности и типов данных, предоставляя типам данных, их свойствам и методам некоторую приватность и связность. Язык Java сделал следующий шаг, превратив класс из артефакта в самостоятельный двоичный компонент.

Увы, если попытаться ответить на вопрос: «Что означает *быть достаточно малым, чтобы им можно было управлять?*» — класс оказывается не просто, а чересчур малым. Как заметили Маркс и Энгельс в 1848 году:

> История всех доселе существующих обществ — это история классовой борьбы.
>
> *Манифест Коммунистической партии*

За исключением совсем небольших кодовых баз и очень загроможденных классов, класс — не самая большая часть компонентной архитектуры.

История Java также связана с пустыми обещаниями насчет *пакета* — во всех смыслах данного слова — чего-то, что запечатано и отправлено, целостно, неделимо и готово к использованию. Вот только это переросло в кое-что не имеющее ничего общего с первоначальным замыслом: пространства имен для организации кода в папках — в открытый, но нескромный и обреченный на провал план, запятнанный и отягощенный модными в свое время, но в итоге непрактичными доменными именами.

Затем наступил момент Пандоры — появление мифа. Греческий миф о Пандоре, девушке, приносящей дары, обычно искажают: якобы она открывает ящик, содержимое которого наводит болезни на все человечество. Так вот, это был не ящик, а сосуд. То, что открыла Пандора, — *pithos* (сосуд), впоследствии неверно переведенный как *pyxis* (ящик). Как и при написании кода, названия имеют значение.

Появление JAR-файлов стало первым шагом в сторону компонентной модели, но за ним не последовало ничего больше чем архивирование файлов классов. Для борьбы с возникающим JAR-адом многие средства (возможно, самые наглядные — это инструменты сборки и OSGi-контейнеры) расширили JAR-модель и ее манифест, что позволило продвинуться дальше на пути к модульности.

Но все это в прошлом. А что же сейчас? Что ждет нас в будущем?

Ответ прямо перед вами. Именно поэтому вы и читаете данную книгу.

Java 9 объединил множество фрагментов головоломки в один с помощью модулей — системы, вплетенной прямо в ядро платформы, а не в расширение за ее пределами. Система модулей Java не имеет никаких отсылок к прошлому. Она обязана не только сохранить богатство уже существующего кода, чтобы не превращать в хаос действующие экосистемы, но и одновременно предложить нечто для проектов, которым еще только предстоит быть написанными в этом постоянно меняющемся мире.

Природу модулей и зависимостей, детали синтаксиса и компонентизации необходимо понять на механическом уровне. С точки зрения проектирования стоит знать все за и против работы с модулями. Как и в случае с любой другой концепцией, модульность — это не волшебный соус, который можно просто добавить в разработку; она требует осторожности, навыков и внимания. Вам нужно знать ответы на вопросы о том, что произойдет с уже существующим кодом в модульном мире; о том, как это повлияет на развертывание и разработку; и даже о том, о чем вы еще не задумывались.

Вопросов очень много. Вот почему вы сейчас читаете данную книгу.

Николай способен ответить на эти и другие вопросы. Он начал отслеживать модули с того момента, как они появились на горизонте. Он нырнул прямо в глубину и преодолел мелководье JSR и реализаций. Он углубился в такие детали, куда вам, может быть, и не придется заглянуть. Его тщательность и внимание к мелочам позволят вам овладеть всей полнотой знаний — от теории к практике, от уровня новичка до гуру.

Это книга-дар. Открывайте, читайте, наслаждайтесь.

Кевлин Хенни (Kevlin Henney), Curbralan

Вступление

Я познакомился с системой модулей ранним утром в апреле 2015 года. Перед работой, проверяя входящие сообщения в OpenJFX, я наткнулся на сообщение от пользователя JavaFX, который был обеспокоен тем, что приватные API стали недоступными «из-за ограничений модульности». Я помню, как подумал: «Не может быть. Java ни за что не пошел бы на такое несовместимое изменение». Списав это на недоразумение, я отправился на работу.

Затем, после обеда, мы немного поспорили с коллегой. Ничего особенного, но из-за появившегося раздражения я решил вернуться домой пораньше и насладиться солнечным весенним днем. Выйдя на балкон со стаканом прохладного пива, я решил почитать что-нибудь. Но что именно? Из любопытства я начал просматривать ответы на сообщение, которое я пролистал сегодня утром, — и это меня очень увлекло!

Всю следующую неделю я буквально пожирал любой обрывок информации, какой только смог найти, о *Project Jigsaw* — базе, на основе которой развивалась система модулей. Оказалось, что опасения пользователя JavaFX вовсе не были беспочвенными.

Сначала я сосредоточился только на том, что могло выйти из строя в Java 9. Намечающаяся выгода пока еще казалась несколько туманной. К счастью, в то время я работал над крупным Java-проектом и постепенно начал понимать, как можно использовать систему модулей для улучшения и сопровождения его общей структуры. Все больше и больше частей складывалось в единую картину, и спустя пару недель я загорелся идеей ввести модули в экосистему — даже если бы это что-нибудь нарушило.

Переход от опасений по поводу совместимости к пониманию ценности, которую модули могут внести в проект, — самый распространенный вариант. Но это еще не все! Помимо беспокойства об уже существующих кодовых базах, вы можете захотеть ввести систему модулей в только-только стартовавший проект либо заинтересоваться изучением растущего влияния модулей на экосистему. Куда бы вы ни направились, данная книга станет вашим путеводителем.

Если вам интересно, куда приведет это путешествие, то вспомните Java 8. Введенные в нем лямбда-выражения стали чем-то гораздо более важным, чем просто новая особенность языка, — они глубоко повлияли на сообщество и экосистему. Они позна-

комили миллионы Java-разработчиков с основами функционального программирования и стали началом путешествия, которое открыло нам глаза на новые концепции и сделало нас более сильными программистами. Они также повлияли на создание новых библиотек и даже кое-чему научили уже существующие фреймворки.

Имейте это в виду, размышляя о системе модулей. Это больше, чем просто термин, — он приглашает вас в путешествие, в котором вы узнаете о модульности во всех ее проявлениях, научитесь правильно проектировать и сопровождать большие программные проекты, а также более активно использовать модульность в библиотеках, фреймворках и инструментах.

Благодарности

Прежде всего я хочу поблагодарить Марину Майклс (Marina Michaels), моего главного редактора в Manning. Без ее доброты, настойчивости, профессионализма и чувства юмора эта книга не вышла бы никогда. Марина постоянно направляла меня на моем писательском пути. Что еще более важно, она снова и снова помогала мне сконцентрироваться и написать пару новых глав.

По этой же причине я хочу отказаться от благодарностей в адрес создателей *Civilization* и *Stellaris*, *«Во все тяжкие»* и *«Экспансии»*, авторов многих выдающихся научно-фантастических книг, прочитанных мной за последние годы, а также любого человека на американском ночном телевидении: было бы здорово наслаждаться плодами вашего труда, но мне пришлось потратить бо́льшую часть этого времени на то, чтобы трудиться самому.

Хотелось бы выделить еще трех человек из Manning: Джеанну Боярски (Jeanne Boyarsky), которая наладила со мной важную техническую обратную связь и занималась данной книгой вплоть до самого ее выпуска, когда я уже не мог этого делать; Вишеслава Радовича (Viseslav Radovic), проявившего бесконечное терпение при создании иллюстраций и подгонке их под мой вкус; а также Жан-Франсуа Морина (Jean-François Morin), который безустанно просматривал книгу и примеры кода в поисках ошибок. Вот еще несколько человек, вовлеченных в превращение почти миллиона писем, разбросанного по куче Asciidoc-файлов, в настоящую книгу: Шерил Вейсман (Cheryl Weisman), Рэйчел Герберт (Rachael Herbert), Дэвид Новак (David Novak), Тиффани Тейлор (Tiffany Taylor), Мелоди Долаб (Melody Dolab), Happenstance Type-O-Rama, Александар Драгосавлевич (Aleksandar Dragosavljević), Мэри Пирджис (Mary Piergies) и Мария Тюдор (Marija Tudor). Наконец, я хочу поблагодарить всех рецензентов, которые нашли время, чтобы прочитать мою книгу и оставить к ней комментарии: Анто Аравинта (Anto Aravinth), Бориса Василе (Boris Vasile), Кристиана Кройцер-Бека (Christian Kreutzer-Beck), Конора Редмонда (Conor Redmond), Гаурава Тули (Gaurav Tuli), Джанкарло Массари (Giancarlo Massari), Гвидо Пио Мариотти (Guido Pio Mariotti), Ивана Милосавлевича (Ivan Milosavljević), Джеймса Райта (James Wright), Джереми Брайана (Jeremy Bryan), Кэтлин Эстрада (Kathleen Estrada), Марию Джемини (Maria Gemini), Марка Дечампса (Mark Dechamps), Миккеля Арентофта (Mikkel Arentoft), Рамбабу Поса (Rambabu Posa), Себастьяна Чеха (Sebastian Czech), Шобху Айера (Shobha Iyer), Стива Доусонн-

Андоха (Steve Dawsonn-Andoh), Татьяну Фесенко (Tatiana Fesenko), Томаша Борека (Tomasz Borek) и Тони Свитса (Tony Sweets). Спасибо всем вам!

Кроме того, я благодарен всем людям из сообщества и Oracle, которые приняли участие в разработке системы модулей. Не только за то, что без их усердного труда просто не было бы ничего, о чем стоило бы писать, но и за качественную документацию и интересные разговоры. Я хочу отдельно упомянуть Марка Рейнхолда (Mark Reinhold), Алекса Бакли (Alex Buckley) и Алана Бейтмана (Alan Bateman) за распространение наших совместных усилий, а также за ответы на многочисленные вопросы, которыми я засыпал почтовый ящик Project Jigsaw. Спасибо вам!

Среди тех, кто, возможно, не ждет благодарности, я хотел бы поблагодарить Роберта Крюгера (Robert Krüger), который, сам того не ведая, вызвал во мне интерес к системе модулей тем самым роковым письмом от 8 апреля 2015 года; Кристиана Глёклера (Christian Glökler) за спор, впервые приведший меня к Project Jigsaw в тот день; и Бориса Терзиса (Boris Terzic) за то, что всегда вдохновлял меня на следующий шаг (и за разрешение поиграться с каждой новой версией Java на работе). Затем — всех тех прекрасных людей, которые обратились ко мне за последний год и оставили отзывы обо всем, что я уже написал, — ваш всепоглощающий позитив подарил мне кучу энергии. Спасибо вам!

Всем моим друзьям: спасибо, что были рядом, несмотря на то что у меня было так мало времени, и спасибо за вдохновение и поддержку на протяжении всего пути. Моей семье: я очень старался, чтобы работа над книгой не отнимала то драгоценное время, которое мы проводим вместе, — спасибо, что прощали меня, когда мне это не удавалось. Без вашего бесконечного терпения, любви и поддержки всего этого не было бы. Я бы не стал тем, кто я есть, без вас. Я люблю вас.

О книге

Java 9 ввел систему модулей для платформы Java в язык и экосистему, сделав примитивы модульности легкодоступными всем Java-разработчикам. Для многих, включая меня, такая концепция нова, поэтому в данной книге обучение начинается с нуля. Мы пройдем путь от самых основ к постижению расширенных функций языка. Более того, книга поможет обновить ваши существующие проекты до Java 9+, постепенно наращивая их модульность.

Обратите внимание: мы не собираемся лишь изучать модульность как таковую. Это сложная тема, на которую написаны отдельные книги (к примеру, *Java Application Architecture* Кирка Кнорншильда (Kirk Knoernschild, Prentice Hall, 2012). Однако в процессе введения модульности в действие вы просто не сможете избежать изучения причин, по которым это вообще стоит делать.

Кому стоит прочитать эту книгу

Система модулей — интересный зверь. Насколько просты ее основные принципы и концепции — настолько же сложно ее влияние на экосистему. Она не удивит сию же секунду, подобно лямбда-выражениям, но ее воздействие на экосистему невозможно недооценить. Однако сейчас все это вряд ли имеет значение. В настоящее время модульная система — такая же часть Java, как компилятор, модификатор `private` и оператор `if`, и каждый разработчик обязан знать и понимать ее так же, как все перечисленное.

К счастью, начинать всегда просто. В основу системы модулей заложено всего несколько простых концепций, которые поймет любой разработчик, хоть немного знающий Java. В целом вы поймете эту систему, если уже знаете принципы работы модификаторов доступа, имеете представление, как использовать `javac`, `jar` и `java`, и знаете, что JVM загружает классы из JAR-файлов.

Если все описанное — о вас и вы любите новые испытания, то я приглашаю прочитать эту книгу. Возможно, вы не сразу расставите все точки над i, но по крайней мере, приобретете четкое понимание системы модулей и многих других фактов, которые сложатся в ваше собственное видение экосистемы Java.

С другой стороны, чтобы расставить все точки над i, нужно иметь многолетний опыт участия в разработке Java-проектов. Если коротко, то чем обширнее эти про-

екты и чем больше вы вовлечены в развитие их архитектуры, выбор верных зависимостей и борьбу с ошибками, тем больше вы оцените вклад системы модулей в них. И тогда же проще будет определить влияние модульности на ваш проект и экосистему в целом.

Структура книги

Данная книга разбита на несколько уровней. Прежде всего, она разделена на главы (и три большие части), но вам не обязательно изучать их от начала до конца. У меня есть пара предложений, что читать и в каком порядке.

Части и главы

Книга состоит из 15 глав, разделенных на три части.

Часть I, *«Привет, модули»*, показывает недостатки Java, для устранения которых и была придумана система модулей, а также объясняет ее базовые механизмы вкупе с принципами создания, построения и запуска модульных приложений.

- Глава 1, *«Первый элемент головоломки»*, рассказывает о недостатках поддержки Java модульности на уровне JAR, их негативных последствиях и о том, как система модулей намерена устранить эти недостатки.
- Глава 2, *«Структура модульного приложения»*, показывает, как спроектировать и запустить модульное приложение, и приводит примеры программы, которая будет использоваться на протяжении всей книги. Данная глава обрисует цельную картину, но не будет вдаваться в детали — за нее это сделают остальные главы.
- Глава 3, *«Определение модулей и их свойств»*, описывает модули и основные блоки, из которых и строятся модули, а также то, как система модулей обрабатывает их для достижения приоритетной цели — построения надежного и удобного в сопровождении проекта.
- Глава 4, *«Построение модулей от исходного кода до JAR»*, показывает, как скомпилировать и собрать модульный проект с помощью команд `javac` и `jar`.
- Глава 5, *«Запуск и отладка модульных приложений»*, исследует множество новых опций команды `java`. Запустить модульное приложение достаточно легко, поэтому большая часть данной главы посвящена инструментам, которые понадобятся вам для поиска и устранения неполадок.

Часть II, *«Адаптация под реальные проекты»*, отклоняется от использования полностью модульных приложений и предназначена для перевода существующих проектов на Java 9+ и их постепенной модуляризации.

- Глава 6, *«Проблемы совместимости при переходе на Java 9 и выше»*, исследует наиболее распространенные препятствия, с которыми вы можете столкнуться при переводе действующей кодовой базы на Java 9 (и это еще до создания модулей).
- Глава 7, *«Повторяющиеся проблемы при переходе на Java 9 и выше»*, обсуждает еще два препятствия, вынесенные отдельно, — они не ограничены миграцией,

и потому вы можете столкнуться с ними даже после того, как обновите и модуляризуете проект.

- Глава 8, *«Постепенная модуляризация существующих проектов»*, показывает, как взять большой проект, запущенный на Java 9, и перевести его в модули. Хорошая новость — вам не придется делать это за раз.
- Глава 9, *«Стратегии миграции и модуляризации»*, основывается на предыдущих главах и рассматривает стратегии, которые помогут вам перенести и модуляризовать существующую кодовую базу.

Часть III, *«Расширенные функции системы модулей»*, демонстрирует возможности, основанные на базисе, полученном в части I.

- Глава 10, *«Использование сервисов для разделения модулей»*, показывает, как система модулей поддерживает разделение потребителей и разработчиков API.
- Глава 11, *«Уточнение API и зависимостей»*, расширяет основные механизмы зависимости и доступности, рассмотренные в главе 3, предоставляя гибкость, необходимую для разработки под хаотичные реальные сценарии.
- Глава 12, *«Рефлексия в модульном мире»*, обсуждает, как рефлексия потеряла суперсилу; какими должны быть приложения, библиотеки и фреймворки для того, чтобы она заработала; а также новые эффективные возможности, которые были добавлены в рефлексию.
- Глава 13, *«Версии и модули: возможное и невозможное»*, объясняет причины, по которым система модулей в большинстве своем игнорирует информацию о версиях, минимальную поддержку версионности в модулях, а также то, как сложно (но можно) запустить разные версии одного и того же модуля.
- Глава 14, *«Настройка образа среды выполнения с помощью jlink»*, показывает, как можно извлечь выгоду из модуляризованного JDK, если создать собственный образ среды выполнения, включающий только необходимые модули, а также демонстрирует плюсы модуляризованного приложения, включенного в этот образ, — самостоятельной единицы развертывания.
- Глава 15, *«Собираем все вместе»*, показывает, как выглядит приложение, представленное в главе 2, со всеми наворотами из части III. Кроме того, включает советы по оптимальному использованию системы модулей.

Выберите собственный путь

Пусть данная книга станет для вас не просто одноразовым инструментом, который научит пользоваться системой модулей, когда вы прочтете ее от корки до корки. Не то чтобы в этом было нечто плохое, но мне хочется чего-то большего. Я хочу, чтобы она стала вашим путеводителем и вы изучили по ней максимально интересные вам темы. И пусть затем она останется у вас на столе, готовая в любое время служить в качестве справочника.

Я предлагаю вам прочитать книгу от корки до корки, однако вы можете этого не делать. Я уже упомянул, что для каждого механизма и инструмента отведена отдельная глава и все подробности собраны в одном месте.

Чтобы упростить погружение в главу, я часто повторяю и сопоставляю факты, представленные в других частях книги, чтобы вы могли прочитать о них, даже пропустив соответствующий материал. Если я слишком часто повторяюсь или переборщил со ссылками, — надеюсь, вы меня простите.

На случай, если вы не книжный червь, ниже я добавил несколько путей на выбор.

У меня всего два часа — покажите, что у вас есть:

- *«Цели системы модулей»*, раздел 1.5;
- *«Структура модульного приложения»*, глава 2;
- *«Определение модулей и их свойств»*, глава 3;
- *«Советы для разработчиков модульных приложений»*, раздел 15.2.

Хочу, чтобы мое приложение работало на Java 9:

- *«Первый элемент головоломки»*, глава 1;
- *«Определение модулей и их свойств»*, глава 3;
- *«Проблемы совместимости при переходе на Java 9 и выше»*, глава 6;
- *«Повторяющиеся проблемы при переходе на Java 9 и выше»*, глава 7;
- *«Безымянный модуль, он же путь к классу»*, раздел 8.2;
- *«Стратегии миграции»*, раздел 9.1.

Я думаю о запуске нового проекта с модулями:

- *«Привет, модули»*, часть I;
- *«Использование сервисов для разделения модулей»*, глава 10;
- *«Уточнение API и зависимостей»*, глава 11;
- *«Собираем все вместе»*, глава 15.

Как система модулей изменила экосистему Java:

- *«Первый элемент головоломки»*, глава 1;
- *«Структура модульного приложения»*, глава 2;
- *«Определение модулей и их свойств»*, глава 3;
- *«Проблемы совместимости при переходе на Java 9 и выше»*, глава 6;
- *«Повторяющиеся проблемы при переходе на Java 9 и выше»*, глава 7;
- *«Расширенные функции системы модулей»*, часть III (возможно, кроме глав 10 и 11).

Меня пригласили на вечеринку. Хочу узнать пару фишек системы модулей для поддержания разговора:

- *«Взгляд на модули с высоты птичьего полета»*, раздел 1.4;
- *«Цели системы модулей»*, раздел 1.5;
- *«Структурирование проекта с помощью каталогов»*, раздел 4.1;

- *«Загрузка ресурсов из модулей»*, раздел 5.2;
- *«Отладка модулей и модульных приложений»*, раздел 5.3;
- что-нибудь из *«Проблем совместимости при переходе на Java 9 и выше»*, глава 6; и *«Повторяющихся проблем при переходе на Java 9 и выше»*, глава 7, поможет начать разговор;
- *«Версии и модули: возможное и невозможное»*, глава 13;
- *«Настройка образа среды выполнения с помощью jlink»*, глава 14.

Это круто. Хочу знать все!

- Прочитайте все подряд. Можете оставить часть II, *«Адаптация под реальные проекты»*, на потом, если у вас еще нет своего проекта.

Какой бы путь вы ни выбрали, обращайтесь к указателям, в частности, в начале и конце главы, чтобы решить, куда пойти дальше.

Предостережение

Эта книга полна новых терминов, примеров, советов и сведений, которые нужно запомнить. Чтобы упростить для вас поиск необходимого материала, я специально выделил два вида информации.

Определения *новой концепции*, *термина*, *свойства модуля* или *параметра командной строки* выделены курсивом. Особенно значимые даны в рамке с заголовком. Эти абзацы в книге — самые важные, так что к ним стоит обращаться, чтобы уяснить, как работает тот или иной механизм.

ВАЖНАЯ ИНФОРМАЦИЯ

Абзацы, отмеченные таким значком, предоставляют самую актуальную информацию о концепции, обсуждаемой в текущий момент, или указывают на некий неочевидный факт, который стоит запомнить, — имейте это в виду!

О коде

Во всей книге используется приложение *ServiceMonitor* для демонстрации поведения и особенностей системы модулей. Его можно найти по ссылке github.com/CodeFX-org/demo-jpms-monitor.

С небольшими различиями приложение используется почти во всех главах. В Git-репозитории расположено несколько веток, которые четко демонстрируют особенности, описанные в части I (в основном `master` и некоторые из веток `break-…`) и части III (те или иные ветки `feature-…` и `break-…`).

Часть II, посвященная проблемам с миграцией и модуляризацией, также иногда задействует *ServiceMonitor* как пример, но для нее нет отдельной ветки. Еще один

вариант приложения показывает пару других проблем с миграцией: https://github.com/CodeFX-org/demo-java-9-migration.

Все, что нужно для написания кода по мере чтения книги или экспериментов с примерами, — Java версии 9 или выше (см. следующий подраздел), текстовый редактор и минимальные знания командной строки. Если вы решили использовать среду разработки, то выбирайте те, что поддерживают Java 9 (*как минимум* IntelliJ IDEA 2017.2, Eclipse Oxygen.1a или NetBeans 9). Еще я рекомендую набирать команды вручную или запускать сценарии `.sh`- или `.bat`, однако в ряде случаев вам может пригодиться Maven — для создания проектов необходима как минимум версия 3.5.0.

Больше подробностей по каждому из проектов можно найти в их файле `README`.

О версии Java

Java EE становится Jakarta EE

Система модулей — часть *Java Standard Edition 9 (Java SE 9)*. Помимо Java SE, есть еще Java EE — *Java Enterprise Edition (Java EE)*; его текущая версия — Java EE 8. Когда-то Java SE и EE управлялись одним и тем же процессом под крылом одного и того же хранителя — сначала Sun, затем Oracle.

Все изменилось в 2017 году. Oracle передала технологии Java EE в Eclipse Foundation, который основал проект *Eclipse Enterprise для Java (EE4J)* в целях управления ими. Платформа Java EE отныне будет называться *Jakarta EE*, а ее первый выпуск — Jakarta EE 8.

Периодически в данной книге я буду ссылаться на Java EE и Jakarta EE, особенно в разделе 6.1. Во избежание путаницы в названиях двух проектов и вследствие того, что технология все еще формально называется Java EE (или уже Jakarta EE), я буду использовать аббревиатуру *JEE*.

На момент написания этой книги версия Java 9 активно использовалась и весь код гарантированно работал в ней — если быть более точным, в версии 9.0.4. Он также был протестирован и обновлен для версий Java 10 и 11. Когда книга отправилась в печать, версия 11 еще была в раннем доступе, и, возможно, сейчас в ней есть паратройка изменений, на которые нет ссылок в данной книге.

Java 9 не только представил систему модулей — он также стал начальной точкой шестимесячного цикла выпуска. Уже сейчас вышли Java 10 и 11 и даже Java 12 на подходе (вполне вероятно, когда вы читаете это, 12-ю версию уже выпустили). Означает ли вышесказанное, что книга устарела?

К счастью, не совсем. Кроме пары небольших деталей, Java 10 и 11 ничего не изменили в системе модулей; и, если заглядывать в будущее, существенных изменений пока не запланировано. Поэтому, несмотря на то что здесь упоминается только Java 9, книга подходит и для Java 10 и 11, и, скорее всего, для нескольких будущих выпусков.

Это особенно касается проблем совместимости, описанных в части II. У вас не получится списать их со счетов, перепрыгнув на 8-ю или 10-ю версию. К тому же после освоения Java 9 другие версии покажутся гораздо более простыми, поскольку 10-я и 11-я содержат только небольшие изменения без проблем совместимости.

Оформление листингов

Книга содержит множество примеров исходного кода как в пронумерованных листингах, так и вперемешку с обычным текстом. В обоих случаях исходный код оформлен моноширинным шрифтом, подобно этому, чтобы не смешивать его с остальным текстом. (Хотя названия модулей выделены *курсивом* — см. ниже.)

Во многих случаях оригинальный исходный код и выходные данные компилятора или виртуальной машины были переформатированы, чтобы поместиться на странице книги:

- добавлены разрывы строк и переработаны отступы;
- урезаны выходные данные, в частности, убраны названия пакетов;
- сокращены сообщения об ошибках.

В редких случаях даже этого было недостаточно, и потому листинги включают в себя маркеры продолжения строки (➥). Вдобавок комментарии к исходному коду часто убраны из листинга, если код описан в тексте книги. Многие листинги сопровождаются примечаниями, чтобы подчеркнуть важные концепции.

Начиная с Java 8, для ссылки на метод класса часто используется синтаксис ссылки на метод (например, метод `add` из класса `List` обозначается как `List::add`). В отличие от `List.add` он не похож на обычный вызов метода (хотя куда делись круглые скобки?) и не рождает вопросов о количестве параметров. Фактически `List::add` обращается ко всем перегрузкам метода `add`, а не к одной из них. Я использую этот синтаксис на протяжении всей книги.

Оформление имен методов

ВАЖНАЯ ИНФОРМАЦИЯ

Имена модулей и пакетов почти одинаковы по длине, вследствие чего фрагменты кода и диаграммы могут увеличиться. Я отказался от этого, так что все мои собственноручно созданные модули имеют короткие имена. Не поступайте так в реальных проектах! Вместо этого обратитесь к руководству в подразделе 3.1.3 «Декларация модулей: определение свойств модулей».

Поскольку имена пакетов и модулей весьма похожи, я решил выделить имена модулей *курсивом*, а имена пакетов — моноширинным шрифтом. Это позволяет с легкостью различать их, и я призываю вас делать то же самое, если вы пишете о модулях.

Заместители во фрагментах кода

Новые возможности, к примеру флаги командной строки и содержание `module-info.java`, определяются в общих чертах. Поэтому в книге используются `${`*`заместители`*`}`, чтобы выделить специфические значения. Их легко распознать по знаку доллара с последующими фигурными скобками.

Этот синтаксис применяется исключительно в данном контексте, и его сходство с тем, как некоторые операционные системы и языки программирования обращаются к аргументам и переменным, не случайно. Однако он никогда не указывает на какой-либо определенный рабочий механизм, и такие заместители не должны использоваться в операционной системе или JVM сами по себе. Вам придется заполнить их самостоятельно, но обычно где-то рядом в тексте можно найти пояснение, что именно поместить в `${`*`заместитель`*`}`.

Пример. Из подраздела 4.5.3: *«Когда* `jar` *используется для упаковки файлов классов в архив, можно определить основной класс с помощью* `--main-class ${`*`класс`*`}`*, где* `${`*`класс`*`}` *— полное имя (то есть имя пакета, за которым следуют точка и название класса) класса с методом* `main`*».*

Не сложно, правда?

Команды и их вывод

Лучший способ понять систему модулей — использовать ее напрямую, вызывая команды `javac`, `java` и т. д. и читая сообщения, которые Java выведет в командной строке. В данной книге содержится множество переходов от ввода к выводу, от команд к сообщениям. В фрагментах кода перед командами всегда ставится префикс `$`, перед сообщениями — `>`, перед моими комментариями — `#`.

Пример. Вот команда из подраздела 5.3.2:

```
$ java
   --module-path mods
   --validate-modules

# урезанные стандартизированные модули Java
# урезанные нестандартизированные модули JDK
> file:.../monitor.rest.jar monitor.rest
> file:.../monitor.observer.beta.jar monitor.observer.beta
```

Дискуссионный форум liveBook

Приобретая «Систему модулей Java», вы одновременно получаете доступ к приватному форуму от издательства Manning Publications, где можете оставить отзыв о книге, задать технический вопрос и получить помощь от автора или других пользователей. Чтобы попасть на форум, перейдите по ссылке livebook.manning.com/#!/book/the-java-module-system/discussion. Чтобы узнать больше о форумах Manning и их правилах, откройте livebook.manning.com/#!/discussion.

Обязательство издательства Manning по отношению к читателям состоит в обеспечении площадки, где между автором и отдельными читателями может состояться содержательный диалог. Это не значит, что участие автора в форуме является обязательным, — его вклад остается добровольным (и неоплачиваемым). Мы советуем вам задавать автору интересные и сложные вопросы, чтобы его интерес не угас!

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

Об авторе

Николай Парлог — 30-летний парень (как выразился бы рассказчик, прищурясь), который нашел свое призвание в разработке программного обеспечения. Он постоянно читает, думает и пишет об этом, а также создает код не только ради денег, но и в свое удовольствие.

Профессионально программируя на Java с 2011 года, Николай впоследствии стал внештатным разработчиком, консультантом и спонсором нескольких проектов с открытым исходным кодом. Он также занимается блогом и новостными рассылками, выступает, записывает видео и общается в чатах — и все это на тему ПО (не одновременно, но весьма эффективно — с высокой пропускной способностью, хотя и с задержками). Его дом — codefx.org, где можно найти ссылки на все его проекты.

Об обложке

Иллюстрация на обложке книги называется «Житель Флориды» и изображает коренного американца из Флориды. Она взята из коллекции нарядных костюмов разных стран из французской книги *Costumes civils actuels de tous les peuples connus* Жака Грассе де Сен-Совера (Jacques Grasset de Saint-Sauveur) (1775–1810), опубликованной в 1788 году. Каждая иллюстрация создана и раскрашена вручную. Богатство коллекции Грассе де Сен-Совера напоминает о том, как в культурном плане взаимно далеки были города и поселения мира всего 200 лет назад. Изолированные друг от друга, люди говорили на разных языках и диалектах. На улицах города или в селах — везде легко было определить, основываясь лишь на одежде, откуда человек родом, каковы его профессия и материальное положение.

С тех пор манера одеваться значительно изменилась и разнообразие регионов, столь богатое в прошлом, сгладилось. Сегодня сложно различить даже жителей разных континентов, не говоря уже о городах, регионах или странах. Возможно, мы обменяли культурное разнообразие на жизнь, более богатую в личном плане и, конечно, быстро развивающуюся в технологическом.

Сейчас, когда одну техническую книгу сложно отличить от другой, Manning является образцом оригинальности и изобретательности в компьютерном бизнесе, выпуская книги с обложками, которые представляют богатое разнообразие жизни в регионах, имевшееся более двух столетий назад и возрожденное на картинах Грассе де Сен-Совера.

Часть I
Привет, модули

Java 9 считает модули первоклассной концепцией. Но что *означают* модули? Какие проблемы они решают и какую выгоду можно из них извлечь? И что же подразумевается под словом *«первоклассный»*?

Книга, которую вы читаете, отвечает на все эти вопросы и на множество других. Она учит, как объявить, спроектировать и запустить модули, а также тому, как они повлияют на проект и какие плюсы привнесут.

Но не все сразу. Эта часть книги начинается с объяснения, что же такое *модульность*, для чего она необходима и какие цели преследует система модулей (глава 1). Глава 2 сразу бросает вас в бой и показывает примеры кода, объявляющего, проектирующего и запускающего модули, после чего главы 3–5 объясняют три этих шага более подробно. Глава 3 особенно важна, поскольку в ней описываются базовые концепции и механизмы системы модулей.

Часть II книги обсуждает проблемы, с которыми Java 9 сталкивается в существующих приложениях, а часть III представляет отдельные расширенные функции.