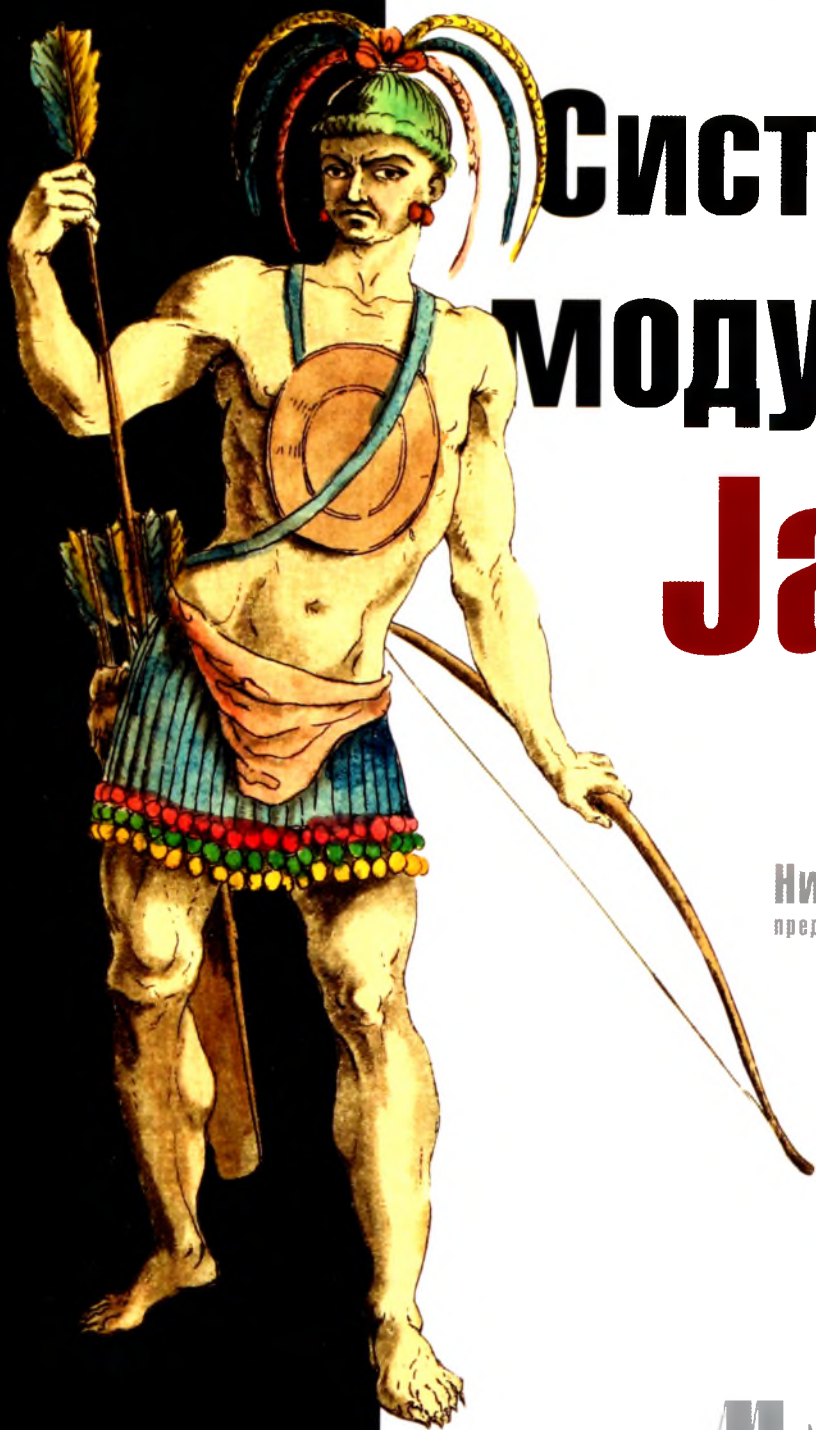


адаптировано для JAVA 11

Система модулей Java

Николай Парлог
предисловие Кевлина Хенни



 MANNING



Краткое содержание

Предисловие.....	16
Вступление.....	18
Благодарности.....	20
О книге.....	22
Об авторе.....	31
Об обложке.....	32

Часть I. Привет, модули

Глава 1. Первый элемент головоломки.....	35
Глава 2. Структура модульного приложения.....	67
Глава 3. Определение модулей и их свойств.....	87
Глава 4. Построение модулей от исходного кода до JAR.....	124
Глава 5. Запуск и отладка модульных приложений.....	143

Часть II. Адаптация под реальные проекты

Глава 6. Проблемы совместимости при переходе на Java 9 и выше.....	167
Глава 7. Повторяющиеся проблемы при переходе на Java 9 и выше.....	186
Глава 8. Постепенная модуляризация существующих проектов.....	213
Глава 9. Стратегии миграции и модуляризации.....	243

Часть III. Расширенные функции системы модулей

Глава 10. Использование сервисов для разделения модулей.....	271
Глава 11. Уточнение API и зависимостей.....	300
Глава 12. Рефлексия в модульном мире.....	329
Глава 13. Версии и модули: возможное и невозможное.....	362
Глава 14. Настройка образа среды выполнения с помощью jlink.....	378
Глава 15. Собираем все вместе.....	406

Приложения

Приложение А. Резюмируем путь класса.....	436
Приложение Б. Обзор API рефлексии.....	439
Приложение В. Наблюдение за JVM с унифицированным ведением журнала.....	443
Приложение Г. Анализ зависимостей проекта с помощью JDeps.....	450
Приложение Д. Ориентация на несколько версий Java с мультиверсионными JAR-файлами.....	458

Оглавление

Предисловие	16
Вступление	18
Благодарности	20
О книге	22
Кому стоит прочитать эту книгу.....	22
Структура книги.....	23
Части и главы.....	23
Выберите собственный путь	24
Предостережение.....	26
О коде	26
О версии Java	27
Оформление листингов	28
Оформление имен методов	28
Заместители во фрагментах кода	29
Команды и их вывод.....	29
Дискуссионный форум liveBook.....	29
От издательства	30
Об авторе	31
Об обложке	32

Часть I. Привет, модули

Глава 1. Первый элемент головоломки.....	35
1.1. Что такое модульность	36
1.1.1. Визуализация ПО в виде графов	37
1.1.2. Влияние принципов проектирования.....	39
1.1.3. Что такое модульность.....	40
1.2. Уничтожение модулей до Java 9	41
1.3. Трудности до Java 9	44
1.3.1. Невыраженные зависимости между JAR-файлами	45

1.3.2.	Затенение классов с одинаковыми именами	46
1.3.3.	Конфликты разных версий на одном проекте	48
1.3.4.	Сложная загрузка класса	49
1.3.5.	Слабая инкапсуляция в JAR-файлах	49
1.3.6.	Проверки безопасности должны быть выполнены вручную	51
1.3.7.	Низкая производительность при загрузке	51
1.3.8.	Негибкая среда выполнения в Java	52
1.4.	Взгляд на модули с высоты птичьего полета	52
1.4.1.	Все на свете — это модули	52
1.4.2.	Ваш первый модуль	55
1.4.3.	Система модулей в действии	55
1.4.4.	Ваш немодульный проект в целом будет в порядке	60
1.5.	Цели системы модулей	61
1.5.1.	Надежная настройка: не оставлять после себя JAR	62
1.5.2.	Надежная инкапсуляция: делаем внутренний код модуля недоступным	62
1.5.3.	Автоматическая безопасность и улучшенное удобство сопровождения	63
1.5.4.	Улучшенная производительность при запуске	63
1.5.5.	Масштабируемая платформа Java	63
1.5.6.	Не цели	64
1.6.	Навыки, старые и новые	64
1.6.1.	Чему вы научитесь	65
1.6.2.	Что нужно знать	65
	Резюме	66
Глава 2.	Структура модульного приложения	67
2.1.	Знакомство с ServiceMonitor	68
2.2.	Модуляризация ServiceMonitor	72
2.3.	Разделение ServiceMonitor на модули	72
2.4.	Организация файлов в структуре каталогов	73
2.5.	Объявление и описание модулей	74
2.5.1.	Объявление зависимостей от других модулей	76
2.5.2.	Объявление публичных API модуля	76
2.5.3.	Визуализация ServiceMonitor с помощью модульного графа	77
2.6.	Компиляция и упаковка модулей	79
2.7.	Запуск ServiceMonitor	81
2.8.	Расширение модульной кодовой базы	81
2.9.	Разбор полетов: эффекты системы модулей	82
2.9.1.	Что система модулей делает для вас	82
2.9.2.	Что еще модульная система может сделать для вас	84
2.9.3.	Добавление необязательных зависимостей	86
	Резюме	86

Глава 3. Определение модулей и их свойств	87
3.1. Модули: строительные блоки модульных приложений	88
3.1.1. Модули Java (JMOD), поставляемые с JDK	88
3.1.2. Модульные файлы JAR: доморощенные модули	89
3.1.3. Декларация модулей: определение свойств модулей	90
3.1.4. Множество типов модулей	95
3.2. Читательность: соединяем части вместе	97
3.2.1. Достижение надежной настройки	98
3.2.2. Эксперименты с ненадежными настройками	99
3.3. Доступность: объявление публичных API	105
3.3.1. Обеспечение надежной инкапсуляции	107
3.3.2. Инкапсуляция переходных зависимостей	109
3.3.3. Столкновения инкапсуляций	110
3.4. Путь модуля: позвольте Java узнать о модулях	114
3.4.1. Регулирование модулей: анализ и верификация структуры приложения ...	115
3.4.2. Граф модуля: представление архитектуры приложения	117
3.4.3. Добавление модулей в граф	120
3.4.4. Добавление ребер в граф	121
3.4.5. Доступность — постоянная цель	122
Резюме	122
Глава 4. Построение модулей от исходного кода до JAR	124
4.1. Структурирование проекта с помощью каталогов	125
4.1.1. Новое предложение — новое соглашение?	125
4.1.2. Созданная структура каталогов	126
4.1.3. Место для деклараций модулей	127
4.2. Компиляция отдельного модуля	128
4.2.1. Компиляция модульного кода	128
4.2.2. Модульное или немодульное?	129
4.3. Компиляция нескольких модулей	131
4.3.1. Безыскусный подход	132
4.3.2. Путь к источнику модуля: информирование компилятора о структуре проекта	132
4.3.3. Звездочка как токен для имени модуля	133
4.3.4. Несколько путей к источникам модулей	135
4.3.5. Настройка начального модуля	136
4.3.6. Стоит ли оно того?	137
4.4. Параметры компилятора	138
4.5. Упаковка модульного JAR-файла	139
4.5.1. Быстрый обзор jar	139
4.5.2. Анализ JAR	140

4.5.3. Объявление точки входа	141
4.5.4. Параметры архиватора	141
Резюме	142
Глава 5. Запуск и отладка модульных приложений	143
5.1. Запуск JVM с модулями	144
5.1.1. Определение основного класса	144
5.1.2. Что, если начальный модуль и основной модуль — не один и тот же	145
5.1.3. Передача параметров приложению	146
5.2. Загрузка ресурсов из модулей	147
5.2.1. Загрузка ресурсов до Java 9	148
5.2.2. Загрузка ресурсов, начиная с Java 9 и позже	148
5.2.3. Загрузка ресурсов за пределами модуля	150
5.3. Отладка модулей и модульных приложений	151
5.3.1. Анализ отдельных модулей	152
5.3.2. Проверка набора модулей	152
5.3.3. Проверка модульного графа	153
5.3.4. Перечисление обозреваемых модулей и зависимостей	154
5.3.5. Исключение модулей во время разрешения	156
5.3.6. Наблюдение за системой модулей с помощью ведения журнала	159
5.4. Параметры виртуальной машины Java	162
Резюме	164

Часть II. Адаптация под реальные проекты

Глава 6. Проблемы совместимости при переходе на Java 9 и выше	167
6.1. Работа с модулями JEE	169
6.1.1. Что особенного в модулях JEE?	170
6.1.2. Разрешение модулей JEE вручную	171
6.1.3. Вмешиваемся в сторонние реализации модулей JEE	172
6.2. Приведение к URLClassLoader	173
6.2.1. Загрузчики классов приложения, тогда и сейчас	173
6.2.2. Обходимся без URLClassLoader	175
6.2.3. Поиск проблемных приведений	176
6.3. Обновленный макет каталога образов среды выполнения	176
6.4. Выбор, замена и расширение платформы	179
6.4.1. Компактных профилей больше нет	179
6.4.2. Удален механизм расширения	180
6.4.3. Удален механизм переопределения утвержденных стандартов	180
6.4.4. Удалены некоторые параметры пути к загрузочному классу	180

6.4.5. Компиляции для Java 5 больше нет.....	180
6.4.6. Удален выбор версии JRE.....	181
6.5. Мелочи, которые приводят к большим неприятностям	181
6.5.1. Новый формат версий.....	182
6.5.2. Исчезновение инструментов	183
6.5.3. Всякие мелочи	183
6.5.4. Новые устаревшие элементы в Java 9, 10 и 11	184
Резюме	184
Глава 7. Повторяющиеся проблемы при переходе на Java 9 и выше.....	186
7.1. Инкапсуляция внутренних API	187
7.1.1. Внутренние API под микроскопом	188
7.1.2. Анализ зависимостей с помощью JDepс.....	192
7.1.3. Компиляция с помощью внутренних API.....	194
7.1.4. Выполнение с помощью внутренних API	195
7.1.5. Параметры компилятора и JVM для доступа к внутренним API	200
7.2. Улучшение разделенных пакетов	202
7.2.1. Что не так с разделенными пакетами.....	203
7.2.2. Эффекты разделенных пакетов.....	204
7.2.3. Множество способов справиться с разделенными пакетами	208
7.2.4. Исправление модулей: последнее средство для обработки разделенных пакетов.....	209
7.2.5. Поиск разделенных пакетов с помощью JDepс	210
7.2.6. Примечание о конфликте версий зависимостей	211
Резюме	211
Глава 8. Постепенная модуляризация существующих проектов.....	213
8.1. Почему наращивание модульности необязательно	214
8.1.1. Если бы требовалось, чтобы каждый JAR был модульным	215
8.1.2. Смешивание и соединение простых JAR-файлов с модулями	215
8.1.3. Технические основы наращиваемой модульности	217
8.2. Безымянный модуль, он же путь к классу	218
8.2.1. Хаос пути к классам, захваченного безымянным модулем	220
8.2.2. Разрешение модулей для безымянного модуля	221
8.2.3. Зависимость от безымянного модуля	223
8.3. Автоматические модули: простые JAR в пути модуля.....	225
8.3.1. Имена автоматических модулей: мелочь, имеющая большое значение	227
8.3.2. Разрешение модулей для автоматических модулей	230
8.3.3. Все в автоматических модулях?	238
8.3.4. Зависимость от автоматических модулей	239
Резюме	242

Глава 9. Стратегии миграции и модуляризации	243
9.1. Стратегии миграции	244
9.1.1. Подготовительные обновления	244
9.1.2. Оценка усилий	245
9.1.3. Непрерывная интеграция в Java 9+	246
9.1.4. Мысли о параметрах командной строки	250
9.2. Стратегии модуляризации	253
9.2.1. Восходящая модуляризация: когда все зависимости — модульные	255
9.2.2. Нисходящая модуляризация: когда нет времени дожидаться всех зависимостей	256
9.2.3. Модуляризация изнутри наружу: если проект находится в середине стека	257
9.2.4. Применение данных стратегий к проекту	258
9.3. Делаем JAR-файлы модульными	259
9.3.1. Открытые модули как промежуточный шаг	259
9.3.2. Генерация модульных деклараций с помощью JDepends	260
9.3.3. Взлом сторонних JAR-файлов	263
9.3.4. Публикация модульных JAR-файлов для Java 8 и выше	265
Резюме	267

Часть III. Расширенные функции системы модулей

Глава 10. Использование сервисов для разделения модулей	271
10.1. Изучение потребности в сервисах	272
10.2. Сервисы в модульной системе платформы Java	274
10.2.1. Использование, предоставление и потребление услуг	274
10.2.2. Разрешение модулей для сервисов	279
10.3. Эффективное проектирование сервисов	282
10.3.1. Типы, которые могут стать сервисами	283
10.3.2. Использование фабрик в качестве сервисов	283
10.3.3. Изоляция потребителей от глобального состояния	285
10.3.4. Организация сервисов, потребителей и поставщиков в модули	288
10.3.5. Использование сервисов для разбиения циклических зависимостей	289
10.3.6. Объявление сервисов в разных версиях Java	291
10.4. Доступ к сервисам с помощью ServiceLoader API	294
10.4.1. Загрузка и доступ к сервисам	294
10.4.2. Особенности загрузки сервисов	296
Резюме	297
Глава 11. Уточнение API и зависимостей	300
11.1. Подразумеваемая читабельность: передача зависимостей	301
11.1.1. Выявление зависимостей модуля	302

11.1.2. Модификатор transitive: подразумеваемая читабельность зависимости.....	304
11.1.3. Когда использовать подразумеваемую читабельность.....	306
11.1.4. Когда стоит полагаться на подразумеваемую читабельность	307
11.1.5. Рефакторинг модулей с подразумеваемой читабельностью.....	309
11.1.6. Рефакторинг модулей путем их слияния	313
11.2. Необязательные зависимости	314
11.2.1. Загадка надежной конфигурации	315
11.2.2. Модификатор static: маркировка зависимостей как необязательных	316
11.2.3. Разрешение модулей для необязательных зависимостей	318
11.2.4. Написание кода с необязательной зависимостью	319
11.3. Квалифицированный экспорт: ограничение доступа к конкретным модулям	321
11.3.1. Предоставление внутренних API	322
11.3.2. Экспорт пакетов в модули.....	323
11.3.3. Когда использовать квалифицированный экспорт	325
11.3.4. Экспорт пакетов в командной строке	327
Резюме	327
Глава 12. Рефлексия в модульном мире	329
12.1. Почему директивы exports не подходят для рефлексии	331
12.1.1. Врываемся в немодульный код.....	332
12.1.2. Принудительная публикация внутренних типов	332
12.1.3. Квалифицированный экспорт создает связь с определенными модулями	333
12.1.4. Нет поддержки глубокой рефлексии	333
12.2. Открытые пакеты и модули: предназначены для использования с рефлексией	334
12.2.1. Открытие пакетов для доступа в режиме выполнения	335
12.2.2. Открытие пакетов для определенных модулей.....	336
12.2.3. Экспорт и открытие пакетов	337
12.2.4. Открытие модулей: закрытие рефлексии	338
12.3. Рефлексия модулей	339
12.3.1. Обновление рефлексивного кода для модулей (или нет)	340
12.3.2. Использование обработчика переменных вместо рефлексии.....	342
12.3.3. Анализ свойств модуля с помощью рефлексии.....	344
12.3.4. Изменение свойств модуля с помощью рефлексии	347
12.3.5. Пересылка открытых пакетов.....	348
12.4. Динамическое создание диаграмм модулей со слоями.....	349
12.4.1. Что такое слои	350
12.4.2. Анализ слоев	352
12.4.3. Создание модульных слоев	355
Резюме	359

Глава 13. Версии и модули: возможное и невозможное	362
13.1. Отсутствие поддержки версий в JPMS	363
13.1.1. Нет поддержки нескольких версий.....	363
13.1.2. Нет поддержки выбора версии.....	366
13.1.3. Что может принести будущее.....	368
13.2. Запись информации о версии	369
13.2.1. Запись версий при сборке модулей.....	369
13.2.2. Доступ к версиям модуля.....	370
13.3. Запуск нескольких версий модуля в отдельных слоях.....	372
13.3.1. Зачем нужен стартер для раскрутки дополнительных слоев.....	373
13.3.2. Раскручивание слоев для вашего приложения, Apache Twill и Cassandra Java Driver.....	374
Резюме	377
Глава 14. Настройка образа среды выполнения с помощью jlink	378
14.1. Создание пользовательских образов среды выполнения	379
14.1.1. Начало работы с jlink	380
14.1.2. Содержание и структура образа.....	381
14.1.3. Включение сервисов в образы среды выполнения	382
14.1.4. Правильно подобранные образы с помощью jlink и jdeps	385
14.2. Создание автономных образов приложений	387
14.2.1. Включение модулей приложений в образы	388
14.2.2. Создание собственного средства запуска приложения	391
14.2.3. Безопасность, производительность и стабильность.....	393
14.3. Генерация образов в операционных системах	393
14.4. Использование плагинов jlink для оптимизации образов	395
14.4.1. Плагины для jlink	395
14.4.2. Уменьшение размера образа.....	398
14.4.3. Улучшение производительности во время выполнения	402
14.5. Параметры для jlink.....	403
Резюме	404
Глава 15. Собираем все вместе	406
15.1. Добавляем навороты в ServiceMonitor.....	406
15.1.1. Разнообразные зависимости	410
15.1.2. Снижение видимости	411
15.1.3. Отделение от сервисов	411
15.1.4. Загрузка кода со слоями во время выполнения	412
15.1.5. Обработка зависимостей от простых JAR-файлов	412
15.2. Советы для разработчиков модульных приложений	413
15.2.1. Модуль или нет?	413
15.2.2. Идеальный модуль.....	414

15.2.3. Позаботьтесь о декларациях модуля	419
15.2.4. Взлом кода путем редактирования модульных деклараций	422
15.3. Технологический ландшафт	424
15.3.1. Maven, Gradle и другие инструменты сборки	425
15.3.2. OSGi.....	427
15.3.3. Микросервисы.....	431
15.4. Размышления о модульной экосистеме	433
Резюме	434

Приложения

Приложение А. Резюмируем путь класса.....	436
А.1. Использование пути к классам для загрузки JAR-файлов приложений	436
А.2. Путь к классам, начиная с Java 9	437
Приложение Б. Обзор API рефлексии	439
Б.1. Основные типы и методы	441
Б.2. Взлом API с помощью setAccessible	442
Б.3. Аннотации помечают код для рефлексии	442
Приложение В. Наблюдение за JVM с унифицированным ведением журнала	443
В.1. Что такое унифицированное ведение журнала	444
В.2. Определяем, какие сообщения должны отображаться	444
В.3. Определяем, куда вывести сообщения	447
В.4. Определяем, что выводить в сообщениях	447
В.5. Настройка всего конвейера регистрации	448
Приложение Г. Анализ зависимостей проекта с помощью JDeps	450
Г.1. Знакомство с JDeps	451
Г.2. Добавление зависимостей в анализ	452
Г.3. Настройка вывода JDeps	453
Г.4. Детализация в зависимости от вашего проекта	454
Г.5. JDeps понимает модули	456
Приложение Д. Ориентация на несколько версий Java с мультиверсионными JAR-файлами	458
Д.1. Создание мультиверсионного JAR.....	459
Д.2. Внутренняя работа MB-JAR	461
Д.3. Рекомендации по использованию	461
Д.3.1. Организация исходного кода.....	461
Д.3.2. Организация байт-кода	462
Д.3.3. Когда использовать MB-JAR.....	462