

МИРОВОЙ КОМПЬЮТЕРНЫЙ БЕСТСЕЛЛЕР

Head First Java

2nd Edition

Kathy Sierra,
Bert Bates

O'REILLY®

Beijing • Cambridge • Köln • Sebastopol • Taipei • Tokyo

МИРОВОЙ КОМПЬЮТЕРНЫЙ БЕСТСЕЛЛЕР

Изучаем Java

2-е издание

Кэти Сьерра
и Берт Бейтс



“Изучаем Java” — это не просто книга. Она не только научит вас теории языка Java и объектно-ориентированного программирования, она сделает вас программистом. В ее основу положен уникальный метод обучения на практике. В отличие от классических учебников информация дается не в текстовом, а в визуальном представлении. Из нее вы узнаете все самое нужное: синтаксис и концепции языка, работа с потоками, работа в сети, распределенное программирование. Вся теория закрепляется интереснейшими примерами и тестами.



Москва
2017

Нашим мозгам. За то, что всегда были с нами
(хотя многие бы поспорили)

Создатели серии Head First

Кэти ↘



Кэти Сьерра (Kathy Sierra) заинтересовалась теорией обучения, еще когда была разработчиком игровых приложений (для студий Virgin, MGM и Amblin). Основную концепцию оформления серии Head First она придумала, когда преподавала предмет «Разработки новых подходов в сфере распространения информации» в рамках курса «Развлекательные СМИ» в Калифорнийском университете. Позже Кэти стала главным тренером компании Sun Microsystems, где обучала других тренеров технике преподавания современных технологий, связанных с Java. В то же время она была ведущим разработчиком нескольких сертификационных экзаменов для Java-программистов. Вместе с Бертом Бейтсом активно использовала методику изложения материала, представленную в этой книге, чтобы обучить сотни преподавателей, разработчиков и обычных людей, которые не имеют отношения к программированию. Кроме того, Кэти — основатель одного из крупнейших в мире сайтов для Java-сообщества javaranch.com и блога headrush.typepad.com.

Кэти также соавтор изданий из серии Head First, посвященных сервлетам, технологии EJB и шаблонам проектирования.

В свободное от работы время Кэти занимается верховой ездой на своей исландской лошади, катается на лыжах и бегает, а также пытается преодолеть скорость света.

kathy@wickedlysmart.com

Кэти и Берт пытаются не оставлять ни одного письма без ответа, но, учитывая количество посланий и напряженный график командировок, сделать это довольно сложно. Самый лучший (и быстрый) способ получить техническую помощь по этой книге — обратиться на очень оживленный форум для новичков в Java: javaranch.com.



← Берт

Берт Бейтс (Bert Bates) — разработчик программного обеспечения. Когда он трудился над проблемами искусственного интеллекта (около десяти лет), то заинтересовался теорией обучения и современными методиками преподавания. С тех пор он сам преподает программирование своим клиентам. Недавно Берт стал членом команды, разрабатывающей сертификационные экзамены для компании Sun.

Первые десять лет своей карьеры в качестве программиста Берт провел в разъездах, сотрудничал с такими вещательными компаниями, как Radio New Zealand, Weather Channel и Arts & Entertainment Network (A & E). Один из его самых любимых проектов заключался в построении полноценного симулятора сети железных дорог для компании Union Pacific Railroad.

Берт — заядлый игрок в го, и уже долгое время работает над соответствующей программой. Он прекрасный гитарист и сейчас пробует свои силы в игре на банджо. Любит кататься на лыжах, занимается бегом и тренирует свою исландскую лошадь Энди (хотя еще неизвестно, кто кого тренирует).

Берт пишет книги в соавторстве с Кэти. Они уже корпят над новой серией изданий (следите за обновлениями в блоге).

Иногда его можно застать на сервере IGC для игры в го (под псевдонимом *jackStraw*).

terrapin@wickedlysmart.com

Содержание (краткое)

Введение	19
1. Погружаемся	31
2. Путешествие в Объектвилль	57
3. Свои переменные нужно знать в лицо	79
4. Как себя ведут объекты	101
5. Особо мощные методы	125
6. Использование библиотеки Java	155
7. Прекрасная жизнь в Объектвилле	195
8. Серьезный полиморфизм	227
9. Жизнь и смерть объектов	265
10. Числа имеют значение	303
11. Опасное поведение	345
12. Очень графическая история	383
13. Улучшай свои навыки	429
14. Сохранение объектов	459
15. Устанавливаем соединение	501
16. Структуры данных	559
17. Выпусти свой код	611
18. Распределенные вычисления	637
Приложение А. Итоговая кухня кода	679
Приложение Б. Десять самых важных тем, которым не хватило самой малости, чтобы попасть в основную часть книги...	689

Содержание (подробное)

В Введение

Ваш мозг по отношению к Java. Когда *вы* стараетесь что-то изучить, ваш мозг пытается оказать вам услугу, убеждая, что все это не имеет никакого значения. Он думает: «Лучше сосредоточиться на том, как избежать встречи со свирепым хищником или что обнаженный сноубордист — это плохая идея». Как же *убедить* мозг в том, что от знания Java зависит ваша жизнь?

Для кого эта книга	20
Мы знаем, о чем вы подумали	21
Мы знаем, о чем подумал ваш мозг	21
Метапознание: размышления о мышлении	23
Вот как вы можете подчинить себе свой мозг	25
Что необходимо для чтения этой книги	26
Технические редакторы	28

1 Погружаемся

Java открывает новые возможности. Еще во времена первой (и довольно скромной) открытой версии 1.02 этот язык покорила разработчиков своим дружелюбным синтаксисом, объектно ориентированной направленностью, управлением памятью и, что важнее всего, перспективой переносимости на разные платформы. Сблудн написать код один раз и запускать его везде оказался слишком велик. Однако, по мере того как программистам приходилось бороться с ошибками, ограничениями и чрезвычайной медлительностью Java, первоначальный интерес к языку остывал. Но это было давно. Сегодня Java достаточно мощный и работает быстрее.



Как работает Java	32
Краткая история Java	34
Структура кода в Java	37
Структура класса	38
Создание класса с методом main	39
Что можно разместить внутри главного метода	40
Защита, защита и...	41
Условное ветвление	43
Создание серьезного бизнес-приложения	44
Генератор фраз	47
Упражнения	50

2 Путешествие в Объектвилль

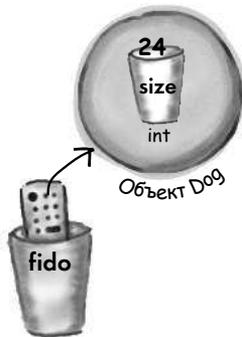
Мне говорили, что там будут объекты. В главе 1 вы размещали весь свой код в методе main(), но это не совсем объектно ориентированный подход. По сути, он не имеет ничего общего с объектами. Да, вы использовали некоторые объекты, например строковые массивы для генератора фраз, но не создавали свои собственные типы объектов. Теперь вы оставите позади мир процедур, выберетесь из тесного метода main() и сами начнете разрабатывать объекты. Вы узнаете, чем же так удобно объектно ориентированное программирование (ООП) на языке Java, и почувствуете разницу между классом и объектом.



Война за кресло, или Как объекты могут изменить вашу жизнь	58
Создаем первый объект	66
Создание и тестирование объектов Movie	67
Скорее! Выбирайтесь из главного метода!	68
Запускаем нашу игру	70
Упражнения	72

3 Свои переменные нужно знать в лицо

Переменные делятся на два вида: примитивы (простые типы) и ссылки. В жизни должно быть что-то помимо чисел, строк и массивов. Как быть с объектом PetOwner с переменной экземпляра типа Dog? Или Cat с переменной экземпляра Engine? В этой главе мы приоткроем завесу тайны над типами в языке Java и вы узнаете, что именно можно *объявлять* в качестве переменных, какие значения *присваивать* им и как вообще с ними *работать*.



Ссылка типа Dog

Объявление переменной	80
Простые типы	81
Ключевые слова в Java	83
Управление объектом	84
Управление объектом с помощью ссылки	85
Массивы	89
Упражнения	93

4 Как себя ведут объекты

Состояние влияет на поведение, а поведение — на состояние. Вы знаете, что объекты характеризуются **состоянием** и **поведением**, которые представлены **переменными экземпляра** и **методами**. До этого момента вопрос о связи между состоянием и поведением не поднимался. Вам уже известно, что каждый экземпляр класса (каждый объект определенного типа) может иметь уникальные значения для своих переменных экземпляра. Суть объектов заключается в том, что их поведение зависит от состояния. Иными словами, **методы используют значения переменных экземпляра**. В этой главе вы узнаете, как изменить состояние объекта.

Передавать по значению означает

Копировать при передаче



```
foo.go(x); void go(int z) { }
```

Класс описывает, что объект знает и делает	102
Передаем методу разные значения	104
Получаем значения обратно из метода	105
Передаем в метод сразу несколько значений	106
Трюки с параметрами и возвращаемыми значениями	109
Инкапсуляция	110
Объекты внутри массива	113
Объявление и инициализация переменных экземпляра	114
Разница между переменными экземпляра и локальными переменными	115
Сравниваем переменные (примитивы или ссылки)	116
Упражнения	118

5

Особо мощные методы

Сделаем методы еще более мощными. Вы уже поработали с переменными, несколькими объектами и написали небольшой код. Но для полноценной работы требуется гораздо больше инструментов, например **операторы** и **циклы**. Почему бы вам во время учебы не *создать* что-нибудь настоящее, чтобы собственными глазами увидеть, как с нуля пишутся (и тестируются) программы. **Может быть, это будет игра** вроде «Морского боя».

Мы создадим игру «Потопи сайт»

A							
B	Go2.com						
C							
D			Pets.com				
E							
F							
G				AskMe.com			
	0	1	2	3	4	5	6

Создадим аналог «Морского боя»: игра «Потопи сайт»	126
Плавное введение в игру «Потопи сайт»	128
Разработка класса	129
Записываем реализацию метода	131
Тестовый код для класса SimpleDotCom	132
Метод checkYourself()	134
Метод main() в игре	140
Поговорим о циклах for	144
Путешествия сквозь цикл	145
Улучшенный цикл for	146
Приведение простых типов	147
Упражнения	148

6

Использование библиотеки Java

Вместе с Java поставляются сотни готовых классов. Можете не тратить время на изобретение собственного велосипеда, если знаете, как отыскать нужное в библиотеке Java, называемой еще **Java API**. Думаем, у вас найдутся дела поважнее. При написании кода сосредоточьтесь на той части, которая уникальна для вашего приложения. Стандартная библиотека Java представляет собой гигантский набор классов, готовых к применению в качестве строительных блоков.

Хорошо, когда известно, что ArrayList находится в пакете java.util. Но как бы я смогла додуматься до этого самостоятельно?

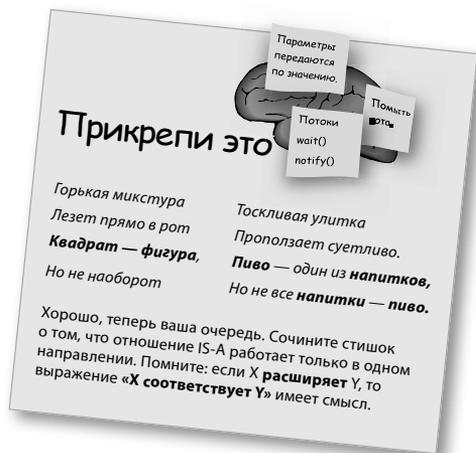


Джулия, 31 год, модель

Небольшая интрига из предыдущей главы — ошибка	156
Примеры использования ArrayList	163
Сравнение ArrayList с обычным массивом	166
Исправляем код класса DotCom	168
Новый и улучшенный класс DotCom	169
Создаем настоящую игру «Потопи сайт»	170
Что делает каждый элемент в игре DotComBust (и в какой момент)	172
Псевдокод для настоящего класса DotComBust	174
Окончательная версия класса DotCom	180
Супермощные булевы выражения	181
Использование библиотеки (Java API)	184
Как работать с API	188
Упражнения	191

7 Прекрасная жизнь в Объектвилле

Планируйте свои программы с прицелом на будущее. Как вы оцените способ создания Java-приложений, при котором у вас останется много свободного времени? Заинтересует ли вас создание гибкого кода, которому не страшны досадные изменения в техническом задании, возникающие в последний момент? Поверьте, вы можете получить все это лишь за три простых подхода по 60 минут каждый. Изучив принципы полиморфизма, вы узнаете о пяти шагах грамотного проектирования классов, трех приемах для достижения полиморфизма и восьми способах создания гибкого кода.



Принципы наследования	198
Использование наследования для предотвращения дублирования кода в дочерних классах	201
Ищем новые возможности, которые дает наследование	203
Проектирование иерархии наследования	206
Отношения IS-A и HAS-A	207
Как узнать, что наследование оформлено правильно	209
Настоящая ценность наследования	212
Перегрузка метода	221
Упражнения	222

8 Серьезный полиморфизм

Наследование — это только начало. Чтобы задействовать полиморфизм, нужны интерфейсы. Новый уровень гибкости и масштабируемости может обеспечить только архитектура, основанная на интерфейсах. Мы уверены, что вы захотите их использовать. Вы удивитесь, как могли жить без них раньше. Что такое интерфейс? Это на 100 % абстрактный класс. Что такое абстрактный класс? Это класс, для которого нельзя создать экземпляр. Зачем это нужно? Читайте главу...

```
Object o = al.get(id);
Dog d = (Dog) o;
d.bark();
```



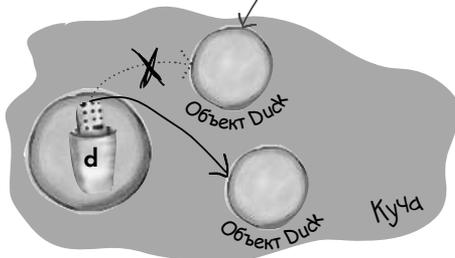
Абстрактный против Конкретного	232
Абстрактные методы	233
Полиморфизм в действии	236
Класс Object	238
Использование полиморфических ссылок типа Object	241
Интерфейс спешит на помощь!	254
Упражнения	260



9 Жизнь и смерть объектов

Объекты рождаются и умирают. Вы управляете их жизненным циклом. Вы решаете, когда и как *создавать* их. И вы решаете, когда их *уничтожить*. Но на самом деле вы не уничтожаете их, а просто делаете брошенными и недоступными. Уже после этого безжалостный **сборщик мусора** может аннулировать объекты, освобождая используемую память. В этой главе мы рассмотрим, как создаются объекты, где размещаются, как эффективно использовать их и делать недоступными.

Когда кто-нибудь вызовет метод `do()`, `Duck` станет недоступным. Его единственная ссылка перенастроится на другой объект `Duck`.



Переменной `d` присвоен новый объект `Duck`, что делает первый объект недоступным (то есть практически мертвым).

Стек и куча: где все хранится	266
Методы размещаются в стеке	267
Как работают локальные переменные, являющиеся объектами	268
Создание объекта	270
Создаем объект <code>Duck</code>	272
Инициализация состояния нового объекта <code>Duck</code>	273
Роль конструкторов родительского класса в жизни объекта	274
Как вызвать конструктор родительского класса	283
Конструкторы родительских классов с аргументами	285
Вызов одного перегруженного конструктора из другого	286
Сколько живет объект	288
Упражнения	296

10 Числа имеют значение

Поговорим о математике. В Java API есть множество удобных и простых в использовании методов для работы с числами. Поскольку большинство из них статические, то сначала разберемся, какими особенностями обладают статические переменные и методы, включая константы, которые в Java считаются *статическими* финализированными переменными.

Статические переменные универсальны для всех экземпляров класса.



Переменные экземпляра (поля): по одной на экземпляр.

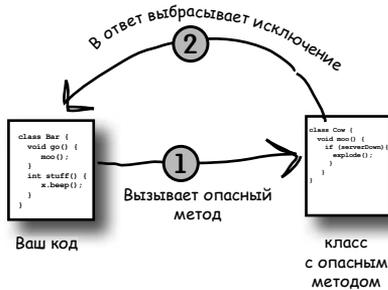
Статические переменные: по одной на класс.

Математические методы — наиболее близкие к глобальным	304
Разница между обычными (не статическими) и статическими методами	305
Что значит иметь класс со статическими методами	306
Инициализация статической переменной	311
Автоматическая упаковка: стираем границы между примитивом и объектом	319
Форматирование чисел	324
Спецификатор форматирования	328
Работа с датами	332
Работа с объектами <code>Calendar</code>	335
Основные методы класса <code>Calendar</code>	336
Статический импорт	337
Упражнения	340

11

Опасное поведение

Случается всякое. То файл пропадает, то сервер падает. Не важно, насколько хорошо вы программируете, ведь невозможно контролировать *все*. Что-то может пойти не так. При создании опасного метода вам понадобится код, который будет обрабатывать возможные нестандартные ситуации. Но как узнать, опасен ли метод? И куда *поместить* код для обработки **непредвиденной** ситуации? В *этой* главе мы разработаем музыкальный MIDI-проигрыватель, использующий опасный JavaSound API.



Что происходит, если вы вызываете опасный метод	349
Методы в Java используют исключения, чтобы сообщить вызывающему коду: «Случилось нечто плохое. Я потерпел неудачу».	350
Исключение — объект типа Exception	352
Управление программным потоком в блоках try/catch	356
Finally: для действий, которые нужно выполнить несмотря ни на что	357
Исключения поддерживают полиморфизм	360
Приложение для проигрывания звуков	372
Создание MIDI-событий (данных о композиции)	373
MIDI-сообщение: сердце MidiEvent	374
Как изменить сообщение	375
Кухня кода	369
Упражнения	378

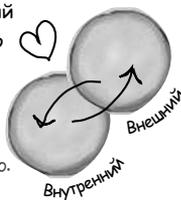
12

Очень графическая история

Соберитесь, вам предстоит придумать GUI (graphical user interface — графический пользовательский интерфейс). Даже если вы уверены, что всю оставшуюся жизнь будете писать код для серверных программ, где работающий на клиентской стороне пользовательский интерфейс представляет собой веб-страницу, рано или поздно вам придется создавать инструменты и вы захотите применить графический интерфейс. Работе над GUI посвящены две главы, из которых вы узнаете ключевые особенности языка Java, в том числе такие, как **обработка событий** и **внутренние классы**. В этой главе мы расскажем, как поместить на экран кнопку и заставить ее реагировать на нажатие. Кроме того, вы научитесь рисовать на экране, добавлять изображение в формате JPEG и даже создавать анимацию.

```
class MyOuter {
    class MyInner {
        void go() {
        }
    }
}
```

Внешний и внутренний объекты теперь тесно связаны.



Эти два объекта в куче обладают особой связью. Внутренний может использовать переменные внешнего (и наоборот).

Ваш первый графический интерфейс: кнопка во фрейме	385
Пользовательское событие	387
Слушатели, источники и события	391
Вернемся к графике...	393
Личный виджет для рисования	394
Что интересного можно сделать в paintComponent()	395
За каждой хорошей графической ссылкой стоит объект Graphics2D	396
Компоновка графических элементов: помещаем несколько виджетов во фрейм	400
Попробуем сделать это с двумя кнопками	402
Как создать экземпляр внутреннего класса	408
Используем внутренний класс для анимации	412
Легкий способ создания сообщений/событий	418
Кухня кода	416
Упражнения	424

13 Улучшай свои навыки

Swing — это просто. Код для работы со Swing выглядит просто, но, скомпилировав его, запустив и посмотрев на экран, вы подумаете: «Эй, этот объект должен быть в другом месте». Инструмент, который *упрощает* создание кода, одновременно усложняет управление им — это **диспетчер компоновки**. Но, приложив небольшие усилия, вы можете подчинить диспетчер компоновки. В этой главе мы будем работать со Swing и ближе познакомимся с виджетами.

Компоненты в областях east и west вытягиваются по ширине.
Компоненты в областях north и south вытягиваются в длину.



Компоненты Swing	430
Диспетчеры компоновки	431
Как диспетчер компоновки принимает решения	432
Три главных диспетчера компоновки: border, flow и box	433
Играем со Swing-компонентами	443
Кухня кода	448
Упражнения	454

14 Сохранение объектов

Объекты могут быть сплюснутыми и восстановленными. Они характеризуются состоянием и поведением. Поведение содержится в классе, а *состояние* определяется каждым *объектом* в отдельности. Что же происходит при сохранении состояния объекта? Если вы создаете игру, вам понадобится функция сохранения/восстановления игрового процесса. Если вы пишете приложение, которое рисует графики, вам также необходима функция сохранения/восстановления. **Вы можете сделать это легким объектно ориентированным способом** — нужно просто сублимировать/сплющить/сохранить/опустошить сам объект, а затем реконструировать/надуть/восстановить/наполнить его, чтобы получить снова.

Есть вопрос?

Сериализованный



Десериализованный



Сохранение состояния	461
Запись сериализованного объекта в файл	462
Что на самом деле происходит с объектом при сериализации	464
Десериализация: восстановление объекта	471
Сохранение и восстановление игровых персонажей	474
Запись строки в текстовый файл	477
Пример текстового файла: электронные флеш-карты	478
Quiz Card Builder: структура кода	479
Класс java.io.File	482
Чтение из текстового файла	484
Quiz Card Player: структура кода	485
Разбор текста с помощью метода split() из класса String	488
Использование serialVersionUID	491
Сохранение схемы BeatBox	493
Восстановление схемы BeatBox	494
Кухня кода	492
Упражнения	496

15

Устанавливаем соединение

Свяжитесь с внешним миром. Это просто. Обо всех низкоуровневых сетевых компонентах заботятся классы из библиотеки `java.net`. В Java отправка и получение данных по Сети — это обычный ввод/вывод, разве что со слегка измененным соединительным потоком в конце цепочки. Получив класс `BufferedReader`, вы можете считывать данные. Ему все равно, откуда они приходят — из файла или по сетевому кабелю. В этой главе вы установите связь с внешним миром с помощью сокетов. Вы создадите клиентские и серверные сокет. В итоге у вас будут клиенты и серверы. И вы сделаете так, чтобы они смогли общаться между собой. К концу этой главы вы получите полноценный многопоточный клиент для чатов. Ой, мы только что сказали «*многопоточный*»?



Чат в BeatBox в режиме реального времени	502
Подключение, отправка и прием	504
Устанавливаем сетевое соединение с помощью сокета	505
<code>BufferedReader</code> для считывания данных из сокета	508
<code>PrintWriter</code> для записи данных в сокет	509
<code>DailyAdviceClient</code>	510
Код программы <code>DailyAdviceClient</code>	511
Создание простого сервера	513
Код приложения <code>DailyAdviceServer</code>	514
Создание чат-клиента	516
Несколько стеков вызовов	521
Планировщик потоков	527
Приостановление потока	531
Метод <code>sleep()</code>	532
Создание и запуск двух потоков	533
Использование блокировки объектов	541
Новая улучшенная версия <code>SimpleChatClient</code>	548
Очень-очень простой чат-сервер	550
Упражнения	554

16 Структуры данных

Сортировка в Java — проще простого. У вас уже есть все необходимые инструменты для сбора данных и управления ими, поэтому не нужно писать собственные алгоритмы для сортировки. Фреймворк для работы с коллекциями в Java (Java collections framework) содержит структуры данных, которые должны подойти практически для всего, что вам может понадобиться. Хотите получить список, в который можно добавлять элементы? Вам необходимо найти что-нибудь по имени? Хотите создать список, который автоматически убирает все дубликаты? Желаете отсортировать своих сослуживцев по количеству ударов, которые они нанесли вам в спину, или домашних любимцев по количеству трюков, которые они выучили? Читайте главу...

<p>Список</p>	Коллекции	561
	Сортировка ArrayList методом Collections.sort()	564
	Обобщения и безопасность типов	570
	Коллекция API — List (Список), Set (Множество) и Map (Отображение, или Ассоциативный массив)	587
<p>Множество</p>	Переопределение hashCode() и equals()	591
	Использование полиморфических аргументов и обобщений	599
<p>Отображение</p>	Упражнения	606

17 Выпусти свой код

Пришло время его отпустить. Вы написали код. Вы его протестировали и откорректировали. Вы рассказали всем знакомым, что больше не желаете видеть ни единой его строки. По большому счету вы создали произведение искусства. Это то, что действительно работает! Но что дальше? В последних двух главах мы расскажем, как организовывать, упаковывать и развертывать (внедрять или доставлять) код на языке Java. Мы рассмотрим локальный, полулокальный и удаленный варианты развертывания, включая исполняемые Java-архивы (JAR), Java Web Start, RMI и сервлеты. Большую часть этой главы мы посвятим организации и упаковке вашего кода — это то, что нужно знать вне зависимости от выбранного варианта доставки.

	Варианты развертывания	612
	Отделение исходных файлов от скомпилированных	614
	Помещаем программу в JAR-архив	615
	Помещаем классы в пакеты	617
	Предотвращение конфликтов при именовании пакетов	618
	Компилируем и запускаем, используя пакеты	620
	Флаг -d	621
	Создание исполняемых Java-архивов с пакетами внутри	622
	Java Web Start	627
	JNLP-файл	629
Упражнения	631	