Ильдар Хабибуллин

bhv®

Java 7

- Принципы объектно-ориентированного программирования
- Пакеты классов и интерфейсы Java SE 7
- Библиотеки Swing и Java 2D, NIO2
- Апплеты, графика, звук и печать
- Технологии Servlet, JSP, JSTL, JSF, XML
- Около 200 законченных программ

Наиболее полное руководство

В ПОДЛИННИКЕ®

Ильдар Хабибуллин

Java 7

Санкт-Петербург
«БХВ-Петербург»
2012

УДК 681.3.06
ББК 32.973.26-018.2
Х12

Хабибуллин И. Ш.

Х12 Java 7. — СПб.: БХВ-Петербург, 2012. — 768 с.: ил. — (В подлиннике)

ISBN 978-5-9775-0735-6

Рассмотрено все необходимое для разработки, компиляции, отладки и запуска приложений Java. Изложены практические приемы использования как традиционных, так и новейших конструкций объектно-ориентированного языка Java, графической библиотеки классов Swing, расширенной библиотеки Java 2D, работа со звуком, печать, способы русификации программ. Приведено полное описание нововведений Java SE 7: двоичная запись чисел, строковые варианты разветвлений, "ромбовидный оператор", NIO2, новые средства многопоточности и др. Дано подробное изложение последней версии сервлетов, технологии JSP и библиотек тегов JSTL. Около двухсот законченных программ иллюстрируют рассмотренные приемы программирования. Приведена подробная справочная информация о классах и методах Core Java API.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Екатерина Капалыгина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.08.11.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 61,92.
Тираж 1800 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0735-6

Оглавление

Введение

Книга, которую вы держите в руках, возникла из курса лекций, читаемых автором студентам младших курсов уже более десяти лет. Подобные книги рождаются после того, как студенты в очередной раз зададут вопрос, который лектор уже несколько раз разъяснял в разных вариациях. Возникает желание отослать их к... какой-нибудь литературе. Пересмотрев еще раз несколько десятков книг, использованных при подготовке лекций, порывшись в библиотеке и на прилавках книжных магазинов, лектор с удивлением обнаруживает, что не может предложить студентам ничего подходящего. Остается сесть за стол и написать книгу самому. Такое происхождение книги накладывает на нее определенные особенности. Она:

- представляет собой сгусток практического опыта, накопленного автором и его студентами с 1996 г.;
- содержит ответы на часто задаваемые вопросы, последних "компьютерщики" называют *FAQ* (Frequently Asked Questions);
- написана кратко и сжато, как конспект лекций, в ней нет лишних слов (за исключением, может быть, тех, что вы только что прочитали);
- рассчитана на читателей, стремящихся быстро и всерьез ознакомиться с новинками компьютерных технологий;
- содержит много примеров применения конструкций Java, которые можно использовать как фрагменты больших производственных разработок в качестве "How to...?";
- включает материал, являющийся обязательной частью подготовки специалиста по информационным технологиям;
- не предполагает знание какого-либо языка программирования, а для знатоков — выделяет особенности языка Java среди других языков;
- предлагает обсуждение вопросов русификации Java.

Прочитав эту книгу, вы вступите в ряды программистов на Java — разработчиков передовой технологии начала XXI века.

Если спустя несколько месяцев эта книга будет валяться на вашем столе с растрепанными страницами, залитыми кофе и засыпанными пеплом, с массой закладок и загнутых углов, а вы начнете сетовать на то, что книга недостаточно полна и слишком проста и ее содержание тривиально и широко известно, а примеры банальны, тогда автор будет считать, что его скромный труд не пропал даром.

Пошел второй десяток лет с того дня, когда были написаны эти строки. Все случилось так, как я и написал. Разошлись три издания книги "Самоучитель Java". Я видел много ее экземпляров в самом разном состоянии. Читатели высказали мне множество нелицеприятных соображений по поводу содержания книги, обнаруженных ошибок и опечаток. Студенты на зачетах и экзаменах пересказывали мне целые куски книги, что тоже наводило на размышления по поводу ее содержания и стиля изложения. У меня накопилось много дополнительного материала, который так и просился в книгу.

Технология Java развивается очень быстро. Сначала предназначавшаяся для небольших сетевых приложений, Java прочно утвердилась на Web-серверах, проникла в сотовые телефоны, планшеты и другие мобильные устройства. Популярная операционная система Android базируется на Java. Теперь Java — обязательная часть Web-программирования.

Развивается и сам язык. В него вводятся новые конструкции, появляются новые библиотеки классов. Графическая библиотека Swing стала частью стандартной поставки Java. В стандартную поставку теперь включены и средства работы с документами XML. Вышла уже седьмая версия Java.

Все это привело к необходимости сделать новое издание, дополнив книгу новым материалом и исправив, увы, неизбежные опечатки.

Ну что же, начнем!

Что такое Java?

Это остров Ява в Малайском архипелаге, территория Индонезии. Это сорт кофе, который любят пить создатели Java (произносится "джава", с ударением на первом слоге). А если серьезно, то ответить на этот вопрос трудно, потому что границы Java, и без того размытые, все время расширяются.

Сначала Java (официальный день рождения технологии Java — 23 мая 1995 г.) предназначалась для программирования бытовых электронных устройств, таких как сотовые телефоны и другие мобильные устройства.

Потом Java стала применяться для программирования браузеров — появились *апплеты*.

Затем оказалось, что на Java можно создавать полноценные приложения. Их графические элементы стали оформлять в виде компонентов — появились *JavaBeans*, с которыми Java вошла в мир распределенных систем и промежуточного программного обеспечения, тесно связавшись с технологией CORBA.

Остался один шаг до программирования серверов — этот шаг был сделан — появились *сервлеты* (servlets), страницы *JSP* (JavaServer Pages) и *EJB* (Enterprise JavaBeans). Серверы должны взаимодействовать с базами данных — появились драйверы *JDBC*. Взаимодействие оказалось удачным, и многие системы управления базами данных и даже операционные системы включили Java в свое ядро, например Oracle, Linux, MacOS X, AIX. Что еще не охвачено? Назовите и через полгода услышите, что Java уже вовсю применяется и там. Из-за этой размытости самого понятия его описывают таким же размытым словом — *технология*.

Такое быстрое и широкое распространение технологии Java не в последнюю очередь связано с тем, что она использует новый, специально созданный язык программирования, который так и называется — язык Java. Этот язык создан на базе языков Smalltalk,

Pascal, C++ и др., вобрав их лучшие, по мнению создателей, черты и отбросив худшие. На этот счет есть разные мнения, но бесспорно, что язык получился удобным для изучения, написанные на нем программы легко читаются и отлаживаются: первую программу можно написать уже через час после начала изучения языка. Язык Java становится языком обучения объектно-ориентированному программированию, так же как язык Pascal был языком обучения структурному программированию. Недаром на Java уже написано огромное количество программ, библиотек классов, а собственный апплет не написал только уж совсем ленивый.

Для полноты картины следует сказать, что создавать приложения для технологии Java можно не только на языке Java, есть и другие языки: Clojure, Scala, Jython, есть даже компиляторы с языков Pascal и C++, но лучше все-таки использовать язык Java: на нем все аспекты технологии излагаются проще и удобнее.

Язык Java часто используется для описания различных приемов объектно-ориентированного программирования, так же как для записи алгоритмов применялся вначале язык Algol, а затем язык Pascal.

Ясно, что всю технологию Java нельзя изложить в одной книге, полное описание ее возможностей составит целую библиотеку. Эта книга посвящена только языку Java. Прочитав ее, вы сможете создавать Java-приложения любой сложности, свободно разбираться в литературе и листингах программ, продолжать изучение аспектов технологии Java по специальной литературе и по исходным кодам свободно распространяемых программных продуктов.

Язык Java тоже очень бурно развивается, некоторые его методы объявляются устаревшими (deprecated), появляются новые конструкции, увеличивается встроенная библиотека классов, но есть устоявшееся ядро языка, сохраняется его дух и стиль. Вот это-то устоявшееся и излагается в книге.

Структура книги

Книга состоит из пяти частей.

Часть I содержит три главы, в которых рассматриваются базовые понятия языка. По прочтении ее вы сможете свободно разбираться в понятиях объектно-ориентированного программирования и их реализации на языке Java, создавать свои объектно-ориентированные программы, рассчитанные на консольный ввод/вывод.

В *главе 1* описываются типы исходных данных, операции с ними, выражения, массивы, операторы управления потоком информации, приводятся примеры записи часто встречающихся алгоритмов на Java. После знакомства с этой главой вы сможете писать программы на Java, реализующие любые вычислительные алгоритмы, встречающиеся в вашей практике.

В *главе 2* вводятся основные понятия объектно-ориентированного программирования: объект и метод, абстракция, инкапсуляция, наследование, полиморфизм, контракты методов и их поручения друг другу. Эта глава призвана привить вам "объектный" взгляд на реализацию сложных проектов, после ее прочтения вы научитесь описывать проект как совокупность взаимодействующих объектов. Здесь же предлагается реализация всех этих

понятий на языке Java. Тут вы, наконец, поймете, что же такое эти объекты и как они взаимодействуют.

В *главе 3* определяются пакеты классов и интерфейсы, ограничения доступа к классам и методам, на примерах подробно разбираются правила их использования. Объясняется структура встроенной библиотеки классов Java API.

В *части II* рассматриваются пакеты основных классов, составляющих неотъемлемую часть Java, разбираются приемы работы с ними и приводятся примеры практического использования основных классов. Здесь вы увидите, как идеи объектно-ориентированного программирования реализуются на практике в сложных производственных библиотеках классов. После изучения этой части вы сможете реализовывать наиболее часто встречающиеся ситуации объектно-ориентированного программирования с помощью стандартных классов.

Глава 4 прослеживает иерархию стандартных классов и интерфейсов Java, на этом примере показано, как в профессиональных системах программирования реализуются концепции абстракции, инкапсуляции и наследования.

В *главе 5* подробно излагаются приемы работы со строками символов, которые, как и всё в Java, являются объектами, приводятся примеры синтаксического анализа текстов, обсуждаются вопросы русификации.

В *главе 6* показано, как в языке Java реализованы коллекции, позволяющие работать с совокупностями объектов и создавать сложные структуры данных.

Глава 7 описывает различные классы-утилиты, полезные во многих ситуациях при работе с датами, случайными числами, словарями и другими необходимыми элементами программ.

В *части III* объясняется создание графического интерфейса пользователя (ГИП) с помощью стандартной библиотеки классов AWT (Abstract Window Toolkit) с компонентами Swing и даны многочисленные примеры построения интерфейса. Подробно разбирается принятый в Java метод обработки событий, основанный на идее делегирования. Здесь же появляются апплеты как программы Java, работающие в окне браузера. Подробно обсуждается система безопасности выполнения апплетов. После прочтения третьей части вы сможете создавать с помощью Swing полноценные приложения под графические платформы MS Windows, X Window System и др., а также программировать браузеры.

Глава 8 описывает иерархию классов библиотеки AWT, которую необходимо четко себе представлять для создания удобного интерфейса. Здесь же рассматривается библиотека графических компонентов Swing, ставшая стандартной наряду с AWT.

В *главе 9* демонстрируются приемы рисования с помощью графических примитивов, способы задания цвета и использование шрифтов, а также решается вопрос русификации приложений Java.

В *главе 10* обсуждается понятие графического компонента, рассматриваются готовые компоненты AWT и их применение, а также создание собственных компонентов AWT.

В *главе 11* рассматриваются графические компоненты общего назначения, относящиеся к библиотеке Swing.

В *главе 12* рассматриваются текстовые графические компоненты библиотеки Swing.

В *главе 13* подробно обсуждаются возможности создания таблиц средствами Swing.

В *главе 14* показано, какие способы размещения компонентов в графическом контейнере имеются в AWT и Swing и как их применять в разных ситуациях.

В *главе 15* вводятся способы реагирования компонентов на сигналы от клавиатуры и мыши, а именно модель делегирования, принятая в Java.

В *главе 16* описывается создание рамок, окружающих графические компоненты Swing.

В *главе 17* обсуждается интересная способность Swing изменять свой внешний вид, сливаясь с окружающей графической средой или, наоборот, выделяясь из нее.

В *главе 18*, наконец-то, появляются апплеты — Java-программы, предназначенные для выполнения в окне браузера, и обсуждаются особенности их создания.

В *главе 19* собраны сведения о библиотеке Swing, не вошедшие в предыдущие главы.

В *главе 20* рассматривается работа с изображениями и звуком средствами AWT.

В *части IV* изучаются конструкции языка Java, не связанные общей темой. Некоторые из них необходимы для создания надежных программ, учитывающих все нештатные ситуации, другие позволяют реализовывать сложное взаимодействие объектов. Здесь же рассматривается передача потоков данных от одной программы Java к другой. Внимательное изучение четвертой части позволит вам дополнить свои разработки гибкими средствами управления выполнением приложения, создавать сложные клиент-серверные системы.

Глава 21 описывает встроенные в Java средства обработки исключительных ситуаций, возникающих во время выполнения готовой программы.

Глава 22 рассказывает об интересном свойстве языка Java — способности создавать подпроцессы (threads) и управлять их взаимодействием прямо из программы.

В *главе 23* обсуждается концепция потока данных и ее реализация в Java для организации ввода/вывода на внешние устройства.

Глава 24 рассматривает сетевые средства языка Java, позволяющие скрыть все сложности протоколов Интернета и максимально облегчить написание клиент-серверных и распределенных приложений.

Часть V книги посвящена Web-технологии Java, точнее, тех ее разделов, которые касаются программирования серверов.

В *главе 25* описываются те аспекты технологии Java, которые необходимы для Web-программирования: архиватор JAR, компоненты JavaBeans, драйверы соединения с базами данных JDBC.

Глава 26 посвящена основному средству программирования серверов — сервлетам.

В *главе 27* разбираются страницы JSP, значительно облегчающие оформление ответов на запросы Web-клиентов.

Наконец, в *главе 28* рассматривается вездесущая технология XML и инструменты Java для обработки документов XML.

Выполнение Java-программы

Как вы знаете, программа, написанная на одном из языков высокого уровня, к которым относится и язык Java, так называемый *исходный модуль* ("исходник", или "сырец" на жаргоне от английского *source*), не может быть сразу же выполнена. Ее сначала надо скомпилировать, т. е. перевести в последовательность машинных команд — *объектный модуль*. Но и он, как правило, не может быть сразу же выполнен: объектный модуль надо еще скомпоновать с библиотеками использованных в модуле функций и разрешить перекрестные ссылки между секциями объектного модуля, получив в результате *загрузочный модуль* — полностью готовую к выполнению программу.

Исходный модуль, написанный на Java, не может избежать этих процедур, но здесь проявляется главная особенность технологии Java — программа компилируется сразу в машинные команды, но не команды какого-то конкретного процессора, а в команды так называемой виртуальной машины Java (Java Virtual Machine, JVM). *Виртуальная машина Java* — это совокупность команд вместе с системой их выполнения. Для специалистов скажем, что виртуальная машина Java полностью стековая, так что не требуется сложная адресация ячеек памяти и большое количество регистров. Поэтому команды JVM короткие, большинство из них имеет длину 1 байт, отчего команды JVM называют *байт-кодами* (bytecodes), хотя имеются команды длиной 2 и 3 байта. Согласно статистическим исследованиям средняя длина команды составляет 1,8 байта. Полное описание команд и всей архитектуры JVM содержится в *спецификации виртуальной машины Java* (Virtual Machine Specification, VMS). Ознакомьтесь с этой спецификацией, если вы хотите в точности узнать, как работает виртуальная машина Java.

Другая особенность Java — все стандартные функции, вызываемые в программе, подключаются к ней только на этапе выполнения, а не включаются в байт-коды. Как говорят специалисты, происходит *динамическая компоновка* (dynamic binding). Это тоже сильно уменьшает объем скомпилированной программы.

Итак, на первом этапе программа, написанная на языке Java, переводится компилятором в байт-коды. Эта компиляция не зависит от типа какого-либо конкретного процессора и архитектуры конкретного компьютера. Она может быть выполнена один раз сразу же после написания программы, программу не надо перекомпилировать под разные платформы. Байт-коды записываются в одном или нескольких файлах, могут храниться во внешней памяти или передаваться по сети. Это особенно удобно благодаря небольшому размеру файлов с байт-кодами. Затем полученные в результате компиляции байт-коды можно выполнять на любом компьютере, имеющем систему, реализующую JVM. При этом не важен ни тип процессора, ни архитектура компьютера. Так реализуется принцип Java "Write once, run anywhere" — "Написано однажды, выполняется где угодно".

Интерпретация байт-кодов и динамическая компоновка значительно замедляют выполнение программ. Это не имеет значения в тех ситуациях, когда байт-коды передаются по сети, сеть все равно медленнее любой интерпретации, но в других ситуациях требуется мощный и быстрый компьютер. Поэтому постоянно идет усовершенствование интерпретаторов в сторону увеличения скорости интерпретации. Разработаны *JIT-компиляторы* (Just-In-Time), запоминающие уже интерпретированные участки кода в машинных командах процессора и просто выполняющие эти участки при повторном обращении, например в циклах. Это значительно увеличивает скорость повторяющихся вычислений. Корпорация Sun Microsystems разработала целую технологию HotSpot и включает ее

в свою виртуальную машину Java. Но, конечно, наибольшую скорость может дать только специализированный процессор.

Компания Sun Microsystems выпустила микропроцессоры picoJava, работающие на системе команд JVM. Есть Java-процессоры и других фирм. Эти процессоры непосредственно выполняют байт-коды. Но при выполнении программ Java на других процессорах требуется еще интерпретация команд JVM в команды конкретного процессора, а значит, нужна программа-интерпретатор, причем для каждого типа процессоров и для каждой архитектуры компьютера следует написать свой интерпретатор.

Эта задача уже решена практически для всех компьютерных платформ. На них реализованы виртуальные машины Java, а для наиболее распространенных платформ имеется несколько реализаций JVM разных фирм. Все больше операционных систем и систем управления базами данных включают реализацию JVM в свое ядро. Создана и специальная операционная система JavaOS, применяемая в электронных устройствах. В большинство браузеров встроена виртуальная машина Java для выполнения апплетов. Операционная система Andriod содержит виртуальную машину Java, называемую Dalvik, которая работает на ядре Linux.

Программы, приведенные в этой книге, выполнялись в операционных средах программирования MS Windows 2000/XP/Server 2003, Red Hat Linux, Fedora Core Linux, SUSE Linux без перекомпиляции. Это видно по рисункам, приведенным во многих главах книги. Они "сняты" с экранов графических оболочек разных операционных систем.

Внимательный читатель уже заметил, что кроме реализации JVM для выполнения байт-кодов на компьютере еще нужно иметь набор функций, вызываемых из байт-кодов и динамически компонующихся с байт-кодами. Этот набор оформляется в виде библиотеки классов Java, состоящей из одного или нескольких *пакетов*. Каждая функция может быть записана байт-кодами, но, поскольку она будет храниться на конкретном компьютере, ее можно записать прямо в системе команд этого компьютера, избегнув тем самым интерпретации байт-кодов. Такие функции, написанные чаще всего на языке C/C++ и скомпилированные под определенную платформу, называют *"родными" методами* (native methods). Применение "родных" методов ускоряет выполнение программы.

Корпорация Oracle, купившая фирму Sun Microsystems — создателя технологии Java, — бесплатно распространяет набор необходимых программных инструментов для полного цикла работы с этим языком программирования: компиляции, интерпретации, отладки, включающий и богатую библиотеку классов. Называется этот набор JDK (Java Development Kit). Он весь содержится в одном файле. Есть наборы инструментальных программ и других фирм. Например, большой популярностью пользуется JDK корпорации IBM.

Что такое JDK?

Набор программ и классов JDK содержит:

- компилятор из исходного текста в байт-коды `javac`;
- интерпретатор `java`, содержащий реализацию JVM;
- облегченный интерпретатор `jre` (в последних версиях отсутствует);

- программу просмотра апплетов `appletviewer`, заменяющую браузер;
- отладчик `jdb`;
- дизассемблер `javap`;
- программу архивации и сжатия `jar`;
- программу сбора и генерирования документации `javadoc`;
- программу генерации заголовочных файлов языка C для создания "родных" методов `javah`;
- программу генерации электронных ключей `keytool`;
- программу `native2ascii`, преобразующую бинарные файлы в текстовые;
- программы `rmic` и `rmiregistry` для работы с удаленными объектами;
- программу `serialver`, определяющую номер версии класса;
- библиотеки и заголовочные файлы "родных" методов;
- библиотеку классов Java API (Application Programming Interface).

В прежние версии JDK включались и отладочные варианты исполнимых программ: `javac_g`, `java_g` и т. д.

Компания Sun Microsystems активно развивала и обновляла JDK, почти каждый год выходили новые версии.

В 1996 г. была выпущена первая версия — JDK 1.0, которая модифицировалась до версии с номером 1.0.2. В этой версии библиотека классов Java API содержала 8 пакетов. Весь набор JDK 1.0.2 поставлялся в упакованном виде в одном файле размером около 5 Мбайт, а после распаковки занимал на диске около 8 Мбайт.

В 1997 г. появилась версия JDK 1.1, последняя ее модификация, 1.1.8, выпущена в 1998 г. В этой версии было 23 пакета классов, занимала она 8,5 Мбайт в упакованном виде и около 30 Мбайт — в распакованном.

В первых версиях JDK все пакеты библиотеки Java API были упакованы в один архивный файл classes.zip и вызывались непосредственно из этого архива, его не нужно было распаковывать.

Затем набор инструментальных средств JDK был сильно переработан.

Версия JDK 1.2 вышла в декабре 1998 г. и содержала уже 57 пакетов классов. В архивном виде это файл размером почти 20 Мбайт и еще отдельный файл размером более 17 Мбайт с упакованной документацией. Полная версия располагается на 130 Мбайт дискового пространства, из них около 80 Мбайт занимает документация.

Начиная с этой версии, все продукты технологии Java собственного производства компания Sun стала называть *Java 2 Platform, Standard Edition*, сокращенно J2SE, а в литературе утвердилось название Java 2. Кроме 57 пакетов классов, обязательных на любой платформе и получивших название *Core API*, в Java 2 JDK 1.2 входят еще дополнительные пакеты классов, называемые Standard Extension API.

В версии J2SE JDK 1.5.0, вышедшей в конце 2004 г., было уже под сотню пакетов, составляющих Core API (Application Programming Interface). В упакованном виде — это файл размером около 46 Мбайт и необязательный файл с упакованной документацией такого же размера. В это же время произошло очередное переименование технологии

Java: из версии убрали первую цифру и стали писать *Java 2 Platform, Standard Edition 5.0*, сокращенно J2SE 5.0 и JDK 5.0, хотя во внутрифирменной документации сохраняется название JDK 1.5.0.

Последнее обновление J2SE 5.0, JDK 1.5.0_22, было выпущено 3 ноября 2009 года.

В шестой версии, вышедшей в начале 2007 г., из названия технологии убрали цифру 2 и стали писать *Java Platform, Standard Edition 6*, сокращенно — Java SE 6 и JDK 6. Впрочем, во внутрифирменной документации остается прежнее обозначение, например последнее на момент написания книги обновление обозначается JDK 1.6.0_26.

Летом 2011 года появилась седьмая версия Java SE 7 и распространяется JDK 1.7.0, описанию которой посвящена эта книга.

Java SE JDK создается для каждой платформы: MS Windows, Solaris, Linux, отдельно, а документация написана на языке HTML и одинакова на всех платформах. Поэтому она записана в отдельном файле. Например, для MS Windows файл с Java SE JDK 1.7.0 называется jdk-7-windows-i586.exe с добавлением номера обновления, а файл с документацией называется jdk-7-fcs-bin-b147-apidocs-27_jun_2011.zip.

Эти файлы можно совершенно свободно скачать со страницы **http://www.oracle.com/technetwork/java/javase/downloads/index.html**.

Для создания Web-программ в *части V* книги вам потребуется еще набор пакетов *Java Platform, Enterprise Edition* (Java EE). Так же как Java SE, он поставляется одним самораспаковывающимся архивом, в который входит SDK (Software Development Kit), Java EE API и сервер приложений. Архив можно скопировать с того же сайта. Набор Java EE SDK — это дополнение к Java SE и поэтому устанавливается после Java SE JDK. Впрочем, на том же сайте есть полная версия архива, содержащая в себе и Java EE SDK, и Java SE JDK.

Java EE входит в состав серверов приложений, поэтому если вы установили JBoss, GlassFish или другой сервер приложений, то у вас уже есть набор классов Java EE.

Кроме JDK компания Oracle отдельно распространяет еще и набор JRE (Java Runtime Environment).

Что такое JRE?

Набор программ и пакетов классов JRE содержит все необходимое для выполнения байт-кодов, в том числе интерпретатор `java` (в прежних версиях — облегченный интерпретатор `jre`) и библиотеку классов. Это часть JDK, не содержащая компиляторы, отладчики и другие средства разработки. Именно Oracle JRE или его аналог, созданный другими фирмами, присутствует в тех браузерах, которые умеют выполнять программы на Java, в операционных системах и системах управления базами данных.

Хотя JRE входит в состав JDK, корпорация Oracle распространяет этот набор и отдельным файлом.

Как установить JDK?

Напомню, что набор JDK упаковывается в самораспаковывающийся архив. Раздобыв каким-либо образом этот архив: скачав из Интернета, с сайта **http://www.oracle.com/**

technetwork/java/javase/downloads/index.html или какого-то другого адреса, вам остается только запустить файл с архивом на выполнение. Откроется окно установки, в котором среди всего прочего вам будет предложено выбрать каталог (directory) установки, например, /usr/java/jdk1.7.0. Каталог и его название можно поменять, место и название установки не имеют значения.

После установки вы получите каталог с названием, например, jdk1.7.0, а в нем подкаталоги:

- bin с исполнимыми файлами;
- db с небольшой базой данных;
- demo с примерами программ, присутствует не во всех версиях JDK;
- docs с документацией, если вы ее установили в этот каталог;
- include с заголовочными файлами "родных" методов;
- jre с набором JRE;
- lib с библиотеками классов и файлами свойств;
- sample с примерами программ, присутствует не во всех версиях JDK;
- src с исходными текстами программ JDK, получаемый после распаковки файла src.zip.

Да-да! Набор JDK содержит исходные тексты большинства своих программ, написанные на Java. Это очень удобно. Вы всегда можете в точности узнать, как работает тот или иной метод обработки информации из JDK, посмотрев исходный код данного метода. Это очень полезно и для изучения Java на "живых", работающих примерах.

> ***Предупреждение***
>
> Не следует распаковывать zip- и jar-архивы, кроме архива исходных текстов src.zip.

После установки надо дополнить значение системной переменной `PATH`, добавив в нее путь к каталогу bin, например `/usr/java/jdk1.7.0/bin`. Некоторые программы, использующие Java, требуют определить и специальную переменную окружения `JAVA_HOME`, содержащую путь к каталогу установки JDK, например `/usr/java/jdk1.7.0`.

Проверить правильность установки Java, а заодно и посмотреть ее версию можно, набрав в командной строке

```
java -version
```

Как использовать JDK?

Несмотря на то что набор JDK предназначен для создания программ, работающих в графических средах, таких как MS Windows или X Window System, он ориентирован на выполнение из командной строки окна **Command Prompt** в MS Windows. В системах UNIX, Linux, BSD можно работать и в текстовом режиме, и в окне **Xterm**.

Написать программу на Java можно в любом текстовом редакторе, например Notepad, WordPad в MS Windows, редакторах vi, emacs в UNIX. Надо только сохранить файл в текстовом, а не графическом формате и дать ему расширение java. Пусть, для примера, именем файла будет MyProgram.java, а сам файл сохранен в текущем каталоге.

После создания этого файла из командной строки вызывается компилятор `javac` и ему передается исходный файл как параметр:

```
javac MyProgram.java
```

Компилятор создает в том же каталоге по одному файлу на каждый класс, описанный в программе, называя каждый файл именем класса с расширением class. Допустим, в нашем примере имеется только один класс, названный `MyProgram`, тогда получаем файл с именем MyProgram.class, содержащий байт-коды.

Компилятор молчалив — если компиляция прошла успешно, он ничего не сообщит, на экране появится только приглашение операционной системы. Если же компилятор заметит ошибки, то он выведет на экран сообщения о них. Большое достоинство компилятора JDK в том, что он "отлавливает" много ошибок и выдает подробные и понятные сообщения.

Далее из командной строки вызывается интерпретатор байт-кодов `java`, которому передается файл с байт-кодами, причем его имя записывается без расширения (смысл этого вы узнаете позднее):

```
java MyProgram
```

На экране появится вывод результатов работы программы или сообщения об ошибках времени выполнения.

Работая в графических оболочках операционных систем, мы привыкли вызывать программу на исполнение двойным щелчком мыши по имени исполнимого файла (в MS Windows у имени исполнимого файла стандартное расширение exe) или щелчком по его ярлыку. В технологии Java тоже есть такая возможность. Надо только упаковать class-файлы с байт-кодами в архив специального вида JAR. Как это сделать, рассказано в *главе 25*. При установке JDK на MS Windows для файлов с расширением jar автоматически создается ассоциация с интерпретатором java, который будет вызван при двойном щелчке мыши на jar-архиве.

Кроме того, можно написать командный файл (файл с расширением bat в MS Windows или Shell-файл командной оболочки в UNIX), записав в нем строку вызова интерпретатора java со всеми нужными параметрами.

Еще один способ запустить Java-программу средствами операционной системы — написать загрузчик (launcher) виртуальной машины Java. Так и сделано в стандартной поставке JDK: исполнимый файл java.exe содержит программу, написанную на языке C, которая запускает виртуальную машину Java и передает ей на исполнение класс Java с методом `main()`. Исходный текст этой программы есть среди исходных текстов Java в каталоге src/launcher. Им можно воспользоваться для написания своего загрузчика. Есть много программ, облегчающих написание загрузчика, например программа Java Launcher фирмы SyncEdit, **http://www.syncedit.com/software/javalauncher/**, или Advanced Installer for Java фирмы Caphyon, **http://www.advancedinstaller.com/**.

Наконец, существуют компиляторы исходного текста, написанного на языке Java, непосредственно в исполнимый файл операционной системы, с которой вы работаете. Их общее название AOT (Ahead-Of-Time) compiler. Например, у знаменитого компилятора GCC (GNU Compiler Collection) есть вход с именем GCJ, с помощью которого можно сделать компиляцию как в байт-коды, так и в исполнимый файл, а также перекомпиляцию байт-кодов в исполнимый файл.