bhv
www.bhv.ru
www.bhv.kiev.ua

И. Хабибуллин

JAVA

САМОУЧИТЕЛЬ

Принципы объектно-ориентированного программирования

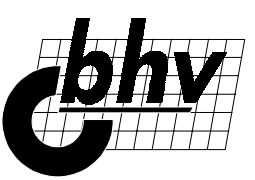
Пакеты классов и интерфейсы

Библиотеки AWT и Swing

Апплеты

Ильдар Хабибуллин

САМОУЧИТЕЛЬ JAVA



Дюссельдорф ◆ Киев ◆ Москва ◆ Санкт-Петербург

УДК 681.3.06

Книга посвящена объектно-ориентированному языку программирования Java 2. Последовательно излагаются практические приемы работы с новейшими конструкциями языка, графической библиотекой классов, расширенной библиотекой Java 2D, со звуком, печатью, способами русификации программ. Около двухсот законченных программ иллюстрируют приведенные приемы программирования. Подробные схемы и описания классов и методов J2SDK позволят использовать книгу как настольный справочник по технологии Java.

Для широкого круга программистов

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зав. редакцией	*Наталья Таркова*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Ангелины Лужиной*
Зав. производством	*Николай Тверских*

Хабибуллин И. Ш.
Самоучитель Java. — СПб.: БХВ-Петербург, 2001. — 464 с.: ил.

ISBN 5-94157-041-4

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 07.02.01.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 37,4.
Тираж 5000 экз. Заказ
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар, № 77.99.1.953.П.950.3.99
от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН.
199034, Санкт-Петербург, 9-я линия, 12.

Содержание

Введение

Книга, которую вы держите в руках, возникла из курса лекций, читаемых автором в течение последних лет для студентов младших курсов. Подобные книги рождаются после того, как студенты в сотый раз зададут один и тот же 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, последние "компьютерщики" называют *FAQ* (Frequency 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 выделяются особенности языка Java среди других языков;
- предлагает обсуждение вопросов русификации Java.

Прочитав эту книгу, вы вступите в ряды программистов на Java — разработчиков 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, тогда автор будет считать, что его скромный труд не пропал даром.

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 телефоны. 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* и *EJB* (Enterprise JavaBeans). Серверы должны взаимодействовать с базами данных — появились драйверы JDBC (Java DataBase Connection). 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, уже появились и другие языки, есть даже компиляторы с языков Pascal и C++, но лучше все-таки использовать язык Java: на нем все аспекты технологии излагаются проще и удобнее.

По скромному мнению автора, язык Java будет использоваться для описания различных приемов объектно-ориентированного программирования так же, как для реализации 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.

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 четырех частей и приложения.

Первая часть 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 друг с другом.

В *главе 3* определяются пакеты классов и интерфейсы, ограничения доступа к классам и методам, на примерах подробно разбираются правила их ис-

пользования. Объясняется структура встроенной библиотеки классов Java API.

Во *второй части* 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 все в Java, являются объектами, приводятся примеры синтаксического анализа текстов.

В *главе 6* показано, как в языке Java реализованы контейнеры, 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 *третьей части* объясняется создание графического интерфейса пользователя (ГИП) с помощью стандартной библиотеки классов AWT (Abstract Window Toolkit) 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 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 классов Swing, постепенно становящаяся 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 графической составляющей, рассматриваются готовые компоненты AWT и их применение, а также создание собственных компонентов.

В *главе 11* показано, какие способы размещения компонентов в графическом контейнере имеются в AWT, и как их применять в разных ситуациях.

В *главе 12* вводятся способы реагирования компонентов на сигналы от клавиатуры и мыши, а именно, модель делегирования, принятая в Java.

В *главе 13* описывается создание системы меню — необходимой составляющей графического интерфейса.

В *главе 14*, наконец-то, появляются апплеты — Java-программы, предназначенные для выполнения в окне браузера, и обсуждаются их особенности.

В *главе 15* рассматривается работа с изображениями и звуком средствами AWT.

В *четвертой части* 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 16 описывает средства обработки исключительных ситуаций, возникающих во время выполнения готовой программы, встроенные в Java.

Глава 17 рассказывает об уникальном свойстве языка Java — способности создавать подпроцессы (threads) и управлять их взаимодействием прямо из программы.

В *главе 18* обсуждается концепция потока данных и ее реализация в Java для организации ввода/вывода на внешние устройства.

Глава 19, последняя по счету, но не по важности, рассматривает сетевые средства языка Java, позволяющие скрыть все сложности протоколов Internet и максимально облегчить написание клиент-серверных приложений.

В *приложении* описываются дополнительные аспекты технологии Java: компоненты JavaBeans, сервлеты, драйверы соединения с базами данных JDBC, и прослеживаются пути дальнейшего развития технологии Java. Ознакомившись с этим приложением, вы сможете ориентироваться в информации о современном состоянии технологии Java и выбрать себе материал для дальнейшего изучения.

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 откомпилировать, т. е. перевести в по-

следовательность 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 (JVM, Java Virtual Machine). *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* (VMS, Virtual Machine Specification). Если вы хотите в точности узнать, как работает виртуальная машина Java, ознакомьтесь с этой спецификацией.

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 откомпилированной 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 некоего конкретного компьютера. Она может быть выполнена один раз сразу же после написания программы. 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. Фирма SUN разработала целую технологию HotSpot и включает ее в свою виртуальную машину Java. Но, конечно, наибольшую скорость может дать только специализированный процессор.

Фирма SUN Microsystems выпустила микропроцессоры PicoJava, работающие на системе команд JVM, и собирается выпускать целую линейку все более мощных Java-процессоров. Есть уже и Java-процессоры 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.

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 называют *"родными" методами* (native methods). Применение "родных" методов ускоряет выполнение программы.

Фирма SUN Microsystems — создатель технологии Java — бесплатно распространяет набор необходимых программных инструментов для полного цикла работы с этим языком программирования: компиляции, интерпретации, отладки, включающий и богатую библиотеку классов, под названием JDK (Java Development Kit). Есть наборы инструментальных программ и других фирм. Например, большой популярностью пользуется JDK фирмы IBM.

Что такое JDK

Набор программ и классов JDK содержит:

- компилятор `javac` из исходного текста в байт-коды;
- 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 документации `javadoc`;
- программу `javah` генерации заголовочных файлов языка C;
- программу `javakey` добавления электронной подписи;
- 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 постоянно развивает и обновляет JDK, каждый год появляются 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 около 8 Мбайт на диске.

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 на диске.

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 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 Мбайтах 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 JDK переименовала в *Java 2 SDK, Standard Edition* (Soft-

ware Development Kit), сокращенно J2SDK, поскольку выпускается еще *Java 2 SDK Enterprise Edition* и *Java 2 SDK Micro Edition*. Впрочем, сама компания SUN часто пользуется и старым названием, а в литературе утвердилось название Java 2. Кроме 57 пакетов классов, обязательных на любой платформе и получивших название *Core API*, в Java 2 SDK v1.2 входят еще дополнительные пакеты классов, называемые Standard Extension API.

В версии Java 2 SDK SE, v1.3, вышедшей в 2000 г., уже 76 пакетов классов, составляющих Core API. В упакованном виде это файл размером около 30 Мбайт, и еще файл с упакованной документацией размером 23 Мбайта. Все это распаковывается в 210 Мбайт дискового пространства. Эта версия требует процессор Pentium 166 и выше и не менее 32 Мбайт оперативной памяти.

В настоящее время версия JDK 1.0.2 уже не используется. Версия JDK 1.1.5 с графической библиотекой AWT встроена в популярные браузеры Internet Explorer 5.0 и Netscape Communicator 4.7, поэтому она применяется для создания апплетов. Технология Java 2 широко используется на серверах и в клиент-серверных системах.

Кроме JDK, компания SUN 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 облегченнный интерпретатор `jre`) и библиотеку классов. Это часть JDK, не содержащая компиляторы, отладчики и другие средства разработки. Именно JRE или его аналог других фирм содержится в браузерах, умеющих выполнять программы на Java, операционных системах и системах управления базами данных.

Хотя JRE входит в состав JDK, фирма SUN распространяет этот набор и отдельным файлом.

Версия JRE 1.3.0 — это архивный файл размером около 8 Мбайт, разворачивающийся в 20 Мбайт на диске.

Как установить JDK

Набор JDK упаковывается в самораспаковывающийся архив. Раздобыв каким-либо образом этот архив: "выкачав" из Internet, с **http://java.sun.com /products/jdk/** или какого-то другого адреса, получив компакт-диск, вам остается только запустить файл с архивом на выполнение. Откроется окно установки, в котором среди всего прочего вам будет предложено выбрать ката-

лог (directory) установки, например, C:\jdk1.3. Если вы согласитесь с предлагаемым каталогом, то вам больше не о чем беспокоиться. Если вы указали собственный каталог, то проверьте после установки значение переменной PATH, набрав в командной строке окна **MS-DOS Prompt** (или окна **Command Prompt** в Windows NT/2000, а тот, кто работает в UNIX, сами знают, что делать) команду set. Переменная PATH должна содержать полный путь к подкаталогу bin этого каталога. Если нет, то добавьте этот путь, например, C:\jdk1.3\bin. Надо определить и специальную переменную CLASSPATH, содержащую пути к архивным файлам и каталогам с библиотеками классов. Системные библиотеки Java 2 подключаются автоматически, без переменной CLASSPATH.

Еще одно предупреждение: не следует распаковывать zip- и jar-архивы.

После установки вы получите каталог с названием, например, jdk1.3, а в нем подкаталоги:

- bin, содержащий исполнимые файлы;
- demo, содержащий примеры программ;
- docs, содержащий документацию, если вы ее установили;
- include, содержащий заголовочные файлы "родных" методов;
- jre, содержащий набор JRE;
- old-include, для совместимости со старыми версиями;
- lib, содержащий библиотеки классов и файлы свойств;
- src, с исходными текстами программ JDK. В новых версиях вместо каталога имеется упакованный файл src.jar.

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.

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 **MS-DOS Prompt** в Windows 95/98/ME или окна **Command Prompt** в Windows NT/2000. В системах UNIX 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 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 о них.

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 появляется вывод результатов работы программы или сообщения об ошибках времени выполнения.

Если работа из командной строки, столь милая сердцу "юниксоидов", кажется вам несколько устаревшей, используйте для разработки интегрированную среду.

Интегрированные среды Java

Сразу же после создания Java, уже в 1996 г., появились интегрированные среды разработки программ для Java, и их число все время возрастает. Некоторые из них являются просто интегрированными оболочками над JDK, вызывающими из одного окна текстовый редактор, компилятор и интерпретатор. Эти интегрированные среды требуют предварительной установки JDK. Другие содержат JDK в себе или имеют собственный компилятор, например, Java Workshop фирмы SUN Microsystems, JBuilder фирмы Inprise, Visual Age for Java фирмы IBM и множество других программных продуктов. Их можно устанавливать, не имея под руками JDK. Надо заметить, что перечисленные продукты написаны полностью на Java.

Большинство интегрированных сред являются средствами визуального программирования и позволяют быстро создавать пользовательский интерфейс, т е. относятся к классу средств RAD (Rapid Application Development).

Выбор какого-либо средства разработки диктуется, во-первых, возможностями вашего компьютера, ведь визуальные среды требуют больших ресурсов, во-вторых, личным вкусом, в-третьих, уже после некоторой практики, достоинствами компилятора, встроенного в программный продукт.

В России по традиции, идущей от TurboPascal к Delphi, большой популярностью пользуется JBuilder, позволяющий подключать сразу несколько JDK разных версий и использовать их компиляторы кроме собственного. Многие профессионалы предпочитают Visual Age for Java, в котором можно графически установить связи между объектами.

К технологии Java подключились и разработчики CASE-средств. Например, популярный во всем мире продукт Rational Rose может сгенерировать код на Java.

Особая позиция Microsoft

Вы уже, наверное, почувствовали смутное беспокойство, не встречая название этой фирмы. Дело в том, что, имея свою операционную систему, огромное число приложений к ней и богатейшую библиотеку классов, компания Microsoft не имела нужды в Java. Но и пройти мимо технологии, распространившейся всюду, компания Microsoft не могла и создала свой компилятор Java, а также визуальное средство разработки, включив его в Visual Studio. Этот компилятор включает в байт-коды вызовы объектов ActiveX. Следовательно, выполнять эти байт-коды можно только на компьютерах, имеющих доступ к ActiveX. Эта "нечистая" Java резко ограничивает круг применения байт-кодов, созданных компилятором фирмы Microsoft. В результате судебных разбирательств с SUN Microsystems компания Microsoft назвала свой продукт Visual J++. Виртуальная машина Java фирмы Microsoft умеет выполнять байт-коды, созданные "чистым" компилятором, но не всякий интерпретатор выполнит байт-коды, написанные с помощью Visual J++.

Чтобы прекратить появление несовместимых версий Java, фирма SUN разработала концепцию "чистой" Java, назвав ее *Pure Java*, и систему проверочных тестов на "чистоту" байт-кодов. Появились байт-коды, успешно прошедшие тесты, и средства разработки, выдающие "чистый" код и помеченные как *"100% Pure Java"*.

Кроме того, фирма SUN распространяет пакет программ Java Plug-in, который можно подключить к браузеру, заменив тем самым встроенный в браузер JRE на "родной".

Java в Internet

Разработанная для применения в сетях, Java просто не могла не найти отражения на сайтах Internet. Действительно, масса сайтов полностью посвяще-

на или содержит информацию о технологии Java. Одна только фирма SUN содержит несколько сайтов с информацией о Java:

- **http://www.sun.com/** — здесь все ссылки, отсюда можно скопировать JDK;
- **http://java.sun.com/** — основной сайт Java, отсюда тоже можно скопировать JDK;
- **http://developer.java.sun.com/** — масса полезных вещей для разработчика;
- **http://industry.java.sun.com/** — новости технологии Java;
- **http://www.javasoft.com/** — сайт фирмы Javasoft, подразделения SUN;
- **http://www.gamelan.com/**.

На сайте фирмы IBM есть большой раздел **http://www.ibm.com/developer/java/**, где можно найти очень много полезного для программиста.

Компания Microsoft содержит информацию о Java на своем сайте: **http://www.microsoft.com/java/**.

Большой вклад в развитие технологии Java вносит корпорация Oracle: **http://www.oracle.com/**.

Существует множество специализированных сайтов:

- **http://java.iba.com.by/** — Java team IBA (Белоруссия);
- **http://www.artima.com/**;
- **http://www.freewarejava.com/**;
- **http://www.jars.com/** — Java Review Service;
- **http://www.javable.com** — русскоязычный сайт;
- **http://www.javaboutique.com/**;
- **http://www.javalobby.com/**;
- **http://www.javalogy.com/**;
- **http://www.javaranch.com/**;
- **http://www.javareport.com/** — независимый источник информации для разработчиков;
- **http://www.javaworld.com** — электронный журнал;
- **http://www.jfind.com/** — сборник программ и статей;
- **http://www.jguru.com/** — советы специалистов;
- **http://www.novocode.com/**;
- **http://www.sigs.com/jro/** — Java Report Online;
- **http://www.sys-con.com/java/**;
- **http://theserverside.com/** — вопросы создания серверных Java-приложений;

- **http://servlets.chat.ru/**;
- **http://javapower.da.ru/** — собрание FAQ на русском языке;
- **http://www.purejava.ru/**;
- **http:/java7.da.ru/**;
- **http://codeguru.earthweb.com/java/** — большой сборник апплетов и других программ;
- **http://securingjava.com/** — обсуждаются вопросы безопасности;
- **http://www.servlets.com/** — вопросы по написанию апплетов;
- **http://www.servletsource.com/**;
- **http://coolservlets.com/**;
- **http://www.servletforum.com/**;
- **http://www.javacats.com/**.

Персональные сайты:

- **http://www.bruceeckel.com/** — сайт Bruce Eckel;
- **http://www.davidreilly.com/java/**;
- **http://www.comita.spb.ru/users/sergeya/java/** — хозяин, Сергей Астахов, собрал здесь буквально все, касающееся русификации Java.

К сожалению, адреса сайтов часто меняются. Возможно, вы и не найдете некоторые из перечисленных сайтов, зато возникнет много других.

Литература по Java

Перечислим здесь только основные, официальные и почти официальные издания, более полное описание чрезвычайно многочисленной литературы дано в списке литературы в конце книги.

Полное и строгое описание языка изложено в книге *The Java Language Specification, Second Edition. James Gosling, Bill Joy, Guy Steele, Gilad Bracha.* Эта книга в электронном виде находится по адресу **http://java.sun.com/docs/books/jls/second_edition/html/j.title.doc.html** и занимает в упакованном виде около 400 Кбайт.

Столь же полное и строгое описание виртуальной машины Java изложено в книге *The Java Virtual Machine Specification, Second Edition. Tim Lindholm, Frank Yellin.* В электронном виде она находится по адресу **http://java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html**.

Здесь же необходимо отметить книгу "отца" технологии Java Джеймса Гослинга, написанную вместе с Кеном Арнольдом. Имеется русский перевод Гослинг Дж., Арнольд К. Язык программирования Java: Пер. с англ. — СПб.: Питер, 1997. — 304 с.: ил.

Компания SUN Microsystems содержит на своем сайте постоянно обновляемый электронный учебник Java Tutorial, размером уже более 14 Мбайт: **http://java.sun.com/docs/books/tutorial/**. Время от времени появляется его печатное издание *The Java Tutorial, Second Edition: Object-Oriented Programming for the Internet. Mary Campione, Kathy Walrath*.

Полное описание Java API содержится в документации, но есть печатное издание *The Java Application Programming Interface. James Gosling, Frank Yellin and the Java Team, Volume 1: Core Packages*; *Volume 2: Window Toolkit and Applets*.

Благодарности

Автор рад воспользоваться представившейся возможностью, чтобы поблагодарить всех, принявших участие в выпуске этой книги.

Отдельная благодарность Евгению Рыбакову, предложившему ее издать и так быстро оформившему договор, что автор не успел передумать; моим студентам с их бесконечными вопросами; всем "фидошникам" эхо-конференции RU.JAVA и, особенно, Вячеславу Педаку и Сергею Астахову — настоящим мастерам Java; своим друзьям — "сишникам", убежденным в том, что "Жаба — это отстой", и сыну, Камилю, для которого эта книга, собственно, и писалась.

Часть I

Базовые конструкции языка Java

ГЛАВА 1

Встроенные типы данных, операции над ними

Приступая к изучению нового языка, полезно поинтересоваться, какие исходные данные могут обрабатываться средствами этого языка, в каком виде их можно задавать, и какие стандартные средства обработки этих данных заложены в язык. Это довольно скучное занятие, поскольку в каждом развитом языке программирования множество типов данных и еще больше правил их использования. Однако несоблюдение этих правил приводит к появлению скрытых ошибок, обнаружить которые иногда бывает очень трудно. Ну что же, в каждом ремесле приходится сначала "играть гаммы", и мы не можем от этого уйти.

Все правила языка Java исчерпывающе изложены в его спецификации, сокращенно называемой JLS. Иногда, чтобы понять, как выполняется та или иная конструкция языка Java, приходится обращаться к спецификации, но, к счастью, это бывает редко, правила языка Java достаточно просты и естественны.

В этой главе перечислены примитивные типы данных, операции над ними, операторы управления, и показаны "подводные камни", которых следует избегать при их использовании. Но начнем, по традиции, с простейшей программы.

Первая программа на Java

По давней традиции, восходящей к языку C, учебники по языкам программирования начинаются с программы "Hello, World!". Не будем нарушать эту традицию. В листинге 1.1 эта программа в самом простом виде, записанная на языке Java.

Листинг 1.1. Первая программа на языке Java

```
class HelloWorld{
public static void main(String[] args){
 System.out.println("Hello, XXI Century World!");
}
}
```

Вот и все, всего пять строчек! Но даже на этом простом примере можно заметить целый ряд существенных особенностей языка Java.

- Всякая программа представляет собой один или несколько классов, в этом простейшем примере только один *класс* (class).
- Начало класса отмечается служебным словом `class`, за которым следует имя класса, выбираемое произвольно, в данном случае `HelloWorld`. Все, что содержится в классе, записывается в фигурных скобках и составляет *тело класса* (class body).
- Все действия производятся с помощью методов обработки информации, коротко говорят просто *метод* (method). Это название употребляется в языке Java вместо названия "функция", применяемого в других языках.
- Методы различаются по именам. Один из методов обязательно должен называться `main`, с него начинается выполнение программы. В нашей простейшей программе только один метод, а значит, имя ему `main`.
- Как и положено функции, метод всегда выдает в результате (чаще говорят, *возвращает* (returns)) только одно значение, тип которого обязательно указывается перед именем метода. Метод может и не возвращать никакого значения, играя роль процедуры, как в нашем случае. Тогда вместо типа возвращаемого значения записывается слово `void`, как это и сделано в примере.
- После имени метода в скобках, через запятую, перечисляются *аргументы* (arguments) или *параметры* метода. Для каждого аргумента указывается его тип и, через пробел, имя. В примере только один аргумент, его тип — массив, состоящий из строк символов. Строка символов — это встроенный в Java API тип `String`, а квадратные скобки — признак массива. Имя массива может быть произвольным, в примере выбрано имя `args`.
- Перед типом возвращаемого методом значения могут быть записаны *модификаторы* (modifiers). В примере их два: слово `public` означает, что этот метод доступен отовсюду; слово `static` обеспечивает возможность вызова метода `main()` в самом начале выполнения программы. Модификаторы вообще необязательны, но для метода `main()` они необходимы.

Замечание

В тексте этой книги после имени метода ставятся скобки, чтобы подчеркнуть, что это имя именно метода, а не простой переменной.

- Все, что содержит метод, *тело метода* (method body), записывается в фигурных скобках.

Единственное действие, которое выполняет метод `main()` в примере, заключается в вызове другого метода со сложным именем `System.out.println` и передаче ему на обработку одного аргумента, текстовой константы `"Hello, 21th Century World!"`. Текстовые константы записываются в кавычках, которые являются только ограничителями и не входят в состав текста.

Составное имя `System.out.println` означает, что в классе `System`, входящем в Java API, определяется переменная с именем `out`, содержащая экземпляры одного из классов Java API, класса `PrintStream`, в котором есть метод `println()`. Все это станет ясно позднее, а пока просто будем писать это длинное имя.

Действие метода `println()` заключается в выводе своего аргумента в выходной поток, связанный обычно с выводом на экран текстового терминала, в окно **MS-DOS Prompt** или **Command Prompt** или **Xterm**, в зависимости от вашей системы. После вывода курсор переходит на начало следующей строки экрана, на что указывает окончание `ln`, слово println — сокращение слов print line. В составе Java API есть и метод `print()`, оставляющий курсор в конце выведенной строки. Разумеется, это прямое влияние языка Pascal.

Сделаем сразу важное замечание. Язык Java различает строчные и прописные буквы, имена `main`, `Main`, `MAIN` различны с "точки зрения" компилятора Java. В примере важно писать `String`, `System` с заглавной буквы, а `main` с маленькой. Но внутри текстовой константы неважно, писать `Century` или `century`, компилятор вообще не "смотрит" на нее, разница будет видна только на экране.

Замечание

Язык Java различает прописные и строчные буквы.

Свои имена можно записывать как угодно, можно было бы дать классу имя `helloworld` или `helloWorld`, но между Java-программистами заключено соглашение, называемое "Code Conventions for the Java Programming Language", хранящееся по адресу **http://java.sun.com/docs/codeconv/index.html**. Вот несколько пунктов этого соглашения:

- имена классов начинаются с прописной буквы; если имя содержит несколько слов, то каждое слово начинается с прописной буквы;
- имена методов и переменных начинаются со строчной буквы; если имя содержит несколько слов, то каждое следующее слово начинается со строчной буквы;