Глава 1

Введение в язык Java

В этой главе...

- Язык Java как средство программирования
- Преимущества языка Java
- Характерные особенности языка Java
- Язык Java и Интернет
- Краткий курс истории языка Java
- Распространенные заблуждения относительно языка Java

Долгое время трудно было представить себе компьютерный журнал без статьи, посвященной языку Java. О нем писали даже такие популярные газеты и журналы, как *The New York Times*, *The Washington Post* и *Business Week*. Невозможно припомнить, чтобы национальное общественное радио (National Public Radio) когда-либо посвящало языку программирования десятиминутную передачу. Хорошо это или плохо, зависит от точки зрения. А инвестиции объемом 100 миллионов долларов, вложенные в производство программного обеспечения, создаваемого с помощью *конкретного* языка программирования?! Телекомпании CNN, CNBC и другие средства массовой информации только и говорили, да и сейчас говорят, о том, как язык Java и то сможет, и это сделает.

Однако эта книга предназначена для серьезных программистов, а поскольку язык Java – это серьезный язык программирования, нам есть о чем рассказать. Итак, мы не станем погружаться в анализ рекламных обещаний и пытаться выяснить, что в них правда, а что вымысел. Вместо этого мы достаточно подробно опишем язык Java именно как язык программирования (включая, разумеется, особенности, позволяю-

щие использовать его для работы в Интернет, которые, собственно, и вызвали столько рекламной шумихи). После этого мы попытаемся отделить реальность от фантазий, объяснив, что язык Java действительно может, а что — нет.

На первых порах между рекламными обещаниями и реальными возможностями языка Java лежала пропасть. По мере его созревания технология становилась все более стабильной и надежной, а ожидания снизились до разумного уровня. Сейчас язык Java все шире используется для создания "промежуточного программного обеспечения" (middleware), поддерживающего связь между клиентами и ресурсами серверов (например, базами данных). Несмотря на то что эти важные приложения не поражают воображение, именно в этой области язык Java оказался наиболее полезным благодаря своей машинной независимости, многопоточности и возможностям сетевого программирования. Кроме того, язык Java захватил лидерство в области встроенных систем (embedded systems), став фактическим стандартом портативных устройств, виртуальных киосков, бортовых автомобильных компьютеров и т.п. Однако первые попытки переписать на языке Java широко распространенные программы для персональных компьютеров не увенчались успехом — полученные приложения оказались маломощными и медленными. С появлением новой версии некоторые из этих проблем удалось решить, и все же нужно признать, что пользователям, в общем-то, совершенно безразлично, на каком языке написаны купленные ими программы. Мы полагаем, что основные преимущества языка Java проявятся при создании новых видов устройств и приложений, а не при переписывании уже существующих программ.

Язык Java как средство программирования

Как язык программирования Java перевыполнил свои рекламные обещания. Несомненно, это один из лучших языков, доступных серьезным программистам. *Потенциально* Java имеет все предпосылки, чтобы стать великим языком программирования, однако, вероятно, сейчас уже слишком поздно. Когда появляется новый язык программирования, немедленно возникает неприятная проблема его совместимости с программным обеспечением, созданным ранее. Более того, даже если изменения в эти программы можно внести без вмешательства в их текст, создателям языка, который так горячо приветствовался публикой, как, например, язык Java, сложно прямо сказать: "Да, возможно мы ошиблись при разработке версии X, но версия Y будет лучше". В итоге, ожидая появления дальнейших улучшений, мы должны констатировать, что структура языка Java в ближайшем будущем существенно не изменится.

Возникает очевидный вопрос: "Как удалось улучшить язык Java?". Оказывается, это сделано не за счет усовершенствования собственно языка программирования, а путем *коренного изменения библиотек программ, написанных на языке Java.* Компания Sun Microsystems изменила все: начиная с имен отдельных библиотечных функций (сделав их более осмысленными) и методов работы графических модулей (изменив способ обработки событий и частично переписав рабочие программы), и заканчивая созданием новых свойств языка, например, средств вывода информации на печать, которых не было в версии Java 1.0. В результате получилась гораздо более полезная программная платформа, чем все предыдущие версии языка Java.

Компания Microsoft выпустила в свет свой собственный продукт под названием J++, имеющий отношение к языку Java. Язык J++ интерпретируется виртуальной машиной, совместимой с виртуальной машиной языка Java (Java Virtual Machine) при выполнении байт-кода, но интерфейсы с внешними кодами у этих языков значительно различаются. Языки J++ и Java имеют практически одинаковый синтаксис. Однако компания Microsoft создала дополнительные языковые конструкции. Все они имеют довольно сомнительную ценность, за исключением интерфейса Windows API. Помимо того, что у этих языков одинаковый синтаксис, их основные библиотеки (строки, утилиты, средства сетевого программирования, средства поддержки многопоточности, математические библиотеки и т.п.), по существу, также совпадают. Однако графические библиотеки, пользовательский интерфейс и доступ к удаленным объектам у этих языков совершенно разные. В настоящее время компания Microsoft больше не поддерживает язык J++, разработав новый язык C#, имеющий много общего с Java, но использующий другую виртуальную машину. В этой книге ни язык J++, ни язык C# не описываются.

Преимущества языка Java

Одно из основных преимуществ языка Java — независимость от платформы, на которой выполняются программы: один и тот же код можно запускать под управлением операционных систем Windows, Solaris, Linux, Machintosh и др. Это действительно необходимо, когда программы загружаются через Интернет для последующего выполнения под управлением разных операционных систем.

Другое преимущество заключается в том, что синтаксис языка Java похож на синтаксис языка C++, и программистам, знающим языки C и C++, его изучение не составляет труда. Правда, для программистов, владеющих языком Visual Basic, этот синтаксис, возможно, будет непривычен.

Если вы никогда не программировали на языке C++, некоторые термины, использованные в этом разделе, будут вам непонятны. В этом случае можете пропустить его. Пока вы доберетесь до конца главы 6, эти термины станут для вас привычными.

Кроме того, Java — полностью объектно-ориентированный язык, даже в большей степени, чем C++. Все сущности в языке Java являются объектами, за исключением немногих основных типов (primitive types), например чисел. (Поскольку с помощью объектно-ориентированного программирования легко разрабатывать сложные проекты, оно заменило собой более древнее структурное программирование. Если вы не знакомы с объектно-ориентированным программированием, главы 3–6 предоставят вам все необходимые сведения о нем.)

Однако разработать еще один, слегка улучшенный, диалект языка C++ недостаточно. Принципиально важно, что *разрабатывать программы, не содержащие ошибок, на языке Java легче, чем на языке C++*.

Почему? Разработчики языка Java долго размышляли о том, отчего программы, написанные на языке C++, так подвержены ошибкам. Они снабдили язык Java средст-

вами, позволяющими исключить саму возможность создавать программы, в которых были бы скрыты наиболее распространенные ошибки. Для этого в языке Java сделано следующее.

- Исключена возможность явного выделения и освобождения памяти.

  Память в языке Java освобождается автоматически с помощью механизма сборки мусора. Программист *гарантирован* от ошибок, связанных с неправильным использованием памяти.
- Введены истинные массивы и запрещена арифметика указателей.

  Теперь программисты *в принципе* не могут стереть данные из памяти вследствие неправильного использования указателей.
- Исключена возможность перепутать оператор присваивания с оператором сравнения на равенство.

  Теперь нельзя даже скомпилировать выражение `if(ntries=3)...` (программисты на языке Visual Basic могут вообще не заметить здесь никакой проблемы, поскольку эта ошибка — источник большинства недоразумений в языках C и C++).
- Исключено множественное наследование. Оно заменено новым понятием — *интерфейсом*, позаимствованным из языка Objective C.

  Интерфейс дает программисту почти все, что тот может получить от множественного наследования, избегая при этом сложностей, возникающих при управлении иерархиями классов. (Если понятие множественного наследования вам незнакомо, обратитесь к главе 5.)

Спецификации языка Java открыты. Их можно найти на Web-странице `http://java.sun.com.docs/books/jls/html/index.html`.

Характерные особенности языка Java

Авторы языка Java написали руководство, в котором объяснялись цели его разработки и достоинства языка. В этом документе приведено одиннадцать характерных особенностей языка Java.

Простой

Объектно-ориентированный

Распределенный

Надежный

Безопасный

Не зависящий от архитектуры компьютера

Машинонезависимый

Интерпретируемый

Высокопроизводительный

Многопоточный

Динамичный

В последнем разделе мы уже коснулись некоторых из этих пунктов. В этом разделе мы:

- приведем цитаты из руководства по языку Java, раскрывающие особенности языка;
- поделимся с читателями соображениями по поводу отдельных свойств языка, основываясь на собственном опыте работы с его последней версией.

Руководство по языку Java можно найти по адресу:
`http://java.sun.com/doc/language_environment`.

Простой

> *Мы хотели создать систему, которая легко программируется, не требует дополнительного обучения и учитывает сложившуюся практику и стандарты программирования. Поэтому, несмотря на то, что мы считали язык C++ неподходящим для этих целей, язык Java был разработан максимально похожим на него, чтобы сделать систему более доступной. В языке Java нет многих редко используемых, малопонятных и невразумительных средств языка C++, которые, по нашему мнению, приносят больше вреда, чем пользы.*

Синтаксис языка Java, по существу, представляет собой очищенный вариант синтаксиса языка C++. В этом языке нет заголовочных файлов, арифметики указателей (и самих указателей), структур, объединений, перегрузки операторов, виртуальных базовых классов и т.п. (Различия между языками Java и C++ описываются в замечаниях о языке C++, разбросанных по всей книге.) Однако разработчики не стремились исправить все недостатки языка C++. Например, синтаксис оператора `switch` в языке Java остался неизменным. Зная язык C++, перейти к синтаксису языка Java будет легко.

Если обычно вы используете визуальную среду программирования (например Visual Basic), язык Java покажется вам сложным. Его синтаксис часто выглядит довольно странным (хотя понять смысл выражения не составляет труда). Важнее то, что при работе на языке Java приходится намного больше программировать. Прелесть языка Visual Basic заключается в том, что его визуальная среда программирования позволяет почти автоматически создавать инфраструктуру приложения. Чтобы достичь того же результата с помощью языка Java, необходимо программировать вручную, но при этом получаются намного более короткие программы. Существует, однако, и третья разновидность сред программирования, позволяющих создавать программы с помощью технологии “перетащить-и-опустить” (“drag-and-drop”).

> *Другой аспект простоты – краткость. Одна из целей языка Java – обеспечить разработку программ, которые можно было бы совершенно самостоятельно выполнять на небольших машинах. Размер основного интерпретатора и средств поддержки классов составляет около 40 Кбайт; стандартные библиотеки и средства поддержки потоков (особенно автономное микроядро (self-contained microkernel)) занимают еще 175 Кбайт.*

Это огромный успех. Заметим, однако, что библиотеки средств поддержки графического пользовательского интерфейса значительно крупнее.

Объектно-ориентированный

> *Попросту говоря, объектно-ориентированное программирование – это метод программирования, в центре внимания которого находятся данные (т.е. объекты) и средства доступа к ним. Проводя аналогию со столярным делом, можно сказать, что "объектно-ориентированный" мастер в основном сосредоточен на стуле, который он изготавливает, и лишь во вторую очередь его интересуют инструменты, необходимые для этого; в то же время "не объектно-ориентированный" столяр думает лишь о своих инструментах. Объектно-ориентированные свойства языков Java и C++, по существу, совпадают.*

Объектная ориентация за прошедшие 30 лет уже доказала свою ценность, и без нее невозможно представить себе современный язык программирования. Действительно, объектно-ориентированные особенности языка Java сравнимы с языком C++. Основное различие между ними заключается в механизме множественного наследования, для которого в языке Java найдено лучшее решение, а также в модели метаклассов языка Java. Механизмы отражения (глава 5) и сериализации объектов (глава 12) позволяют реализовать устойчивые объекты и средства для создания графических пользовательских интерфейсов на основе готовых компонентов.

Если вы никогда не программировали на объектно-ориентированных языках, внимательно изучите главы 4–6. В этих главах излагаются основы объектно-ориентированного программирования и показываются его преимущества при разработке сложных проектов над такими традиционными, процедурно-ориентированными языками, как язык C или Basic.

Распределенный

> *Язык Java обладает большой библиотекой программ для передачи данных на основе таких протоколов TCP/IP (Transmission Control Protocol/Internet Protocol – протокол управления передачей/Интернет-протокол), как HTTP (Hypertext Transfer Protocol – протокол передачи гипертекстов) или FTP (File Transfer Protocol – протокол передачи файлов). Приложения, написанные на языке Java, могут открывать объекты и получать к ним доступ через сеть с помощью URL-адресов (Uniform Resource Location – универсальный адрес ресурса) так же легко, как и в локальной сети.*

Язык Java предоставляет мощные и удобные средства для работы в сети. Каждый, кто когда-либо пытался писать программы для работы в Интернет на других языках, будет приятно удивлен тем, как легко решаются на языке Java самые трудные задачи, например, открытие сетевых соединений (sockets connection). Элегантный механизм, состоящий из так называемых сервлетов (servlets), делает работу на сервере чрезвычайно эффективной. Сервлеты поддерживаются многими популярными Web-

серверами. (Работа в сети будет описана во втором томе.) Связь между распределенными объектами в языке Java обеспечивается механизмом вызова удаленных методов (эта тема также раскрывается во втором томе).

Надежный

> *Язык Java предназначен для создания программ, которые должны надежно работать в любых ситуациях. Основное внимание в языке Java уделяется раннему обнаружению возможных ошибок, динамической проверке (во время выполнения программы), а также исключению ситуаций, подверженных ошибкам... Единственное значительное отличие языка Java от языка C++ заключается в модели указателей, принятой в языке Java, которая исключает возможность перезаписи участка памяти и повреждения данных.*

Это свойство также очень полезно. Компилятор языка Java выявляет такие ошибки, которые в других языках обнаруживаются только на этапе выполнения программы. Кроме того, программисты, потратившие многие часы на поиски ошибки, вызвавшей повреждение памяти из-за неверного указателя, будут очень рады тому, что в языке Java такие проблемы возникнуть в принципе не могут.

Если раньше вы программировали на языках Visual Basic или COBOL, в которых указатели явно не используются, возможно, вам непонятно, почему это настолько важно. Программистам на языке C повезло намного меньше. Им нужны указатели для доступа к строкам, массивам, объектам и даже файлам. При программировании на языке Visual Basic ничего этого не требуется, и программист может не беспокоиться о распределении памяти для этих сущностей. С другой стороны, многие структуры данных в языке, не имеющем указателей, реализовать очень трудно. Для обычных структур, вроде строк и массивов, указатели не нужны. Вся мощь указателей проявляется лишь там, где без них нельзя обойтись, например, при создании связанных списков. Программист на языке Java навсегда избавлен от неверных указателей, неправильного распределения и утечки памяти.

Безопасный

> *Язык Java предназначен для использования в сетевой или распределенной среде. По этой причине большое внимание было уделено безопасности. Язык Java позволяет создавать системы, защищенные от вирусов и постороннего вмешательства.*

В первом издании мы написали: "Никогда не говори *никогда*", — и оказались правы. Группа экспертов по вопросам безопасности из Принстонского университета обнаружила первые ошибки в системе защиты версии Java 1.0 вскоре после появления в продаже первой версии набора инструментальных средств JDK. Более того, и они, и другие специалисты продолжали и впоследствии находить все новые и новые ошибки в механизмах безопасности всех последующих версий языка Java. Мнения независимых экспертов о современных механизмах безопасности, предусмотренных в языке Java, можно найти по URL-адресу Принстонской группы (`http://www.cs.princeton.edu/sip/`), а также на Web-странице `comp.risks`. По-

ложительной стороной этой ситуации является то, что группа разработчиков языка Java заявила о своей полной нетерпимости к любым ошибкам в системе защиты и немедленно приступила к исправлению всех проблем, обнаруженных в механизме безопасности аплетов. В частности, опубликовав внутренние спецификации интерпретатора языка Java, компания Sun намного облегчила поиск скрытых ошибок в системе безопасности и привлекла к их поиску независимых специалистов. Это повысило вероятность того, что все ошибки в системе защиты будут вскоре обнаружены. В любом случае обмануть систему защиты языка Java чрезвычайно трудно. Обнаруженные до сих пор ошибки были почти неуловимыми, к тому же их количество (относительно) невелико.

Web-страница компании Sun, посвященная вопросам безопасности, имеет следующий URL-адрес: `http://java.sun.com/sfaq/`.

Перечислим некоторые ситуации, возникновение которых предотвращает система безопасности языка Java.

1. Переполнение стека выполняемой программы, к которому приводил печально известный "червь", распространявшийся в Интернет.
2. Повреждение участков памяти, находящихся за пределами пространства, выделенного процессу.
3. Считывание и запись локальных файлов при использовании безопасного загрузчика классов, например Web-броузера, который запрещает такой доступ к файлам.

Все эти меры безопасности вполне уместны и обычно работают безупречно, однако осмотрительность никогда не повредит. Хотя обнаруженные к данному моменту ошибки были далеко не тривиальными, и все детали их поиска часто хранятся в секрете, следует признать, что *доказать* безопасность языка Java, вероятно, все же невозможно.

Со временем в язык были добавлены новые средства защиты. Начиная с версии 1.1, в языке Java появилось понятие классов с цифровой подписью (см. том 2). Пользуясь классом с цифровой подписью, вы можете быть уверенным в его авторе. Если вы ему доверяете, то можете предоставить этому классу все привилегии, доступные на вашей машине.

Альтернативный механизм доставки кода, предложенный компанией Microsoft, опирается на технологию ActiveX и для безопасности использует только цифровые подписи. Очевидно, что этого недостаточно — любой пользователь программного обеспечения фирмы Microsoft может подтвердить, что программы широко известных производителей часто завершаются аварийно, создавая тем самым опасность повреждения данных. Система безопасности в языке Java намного надежнее, чем технология ActiveX, поскольку она контролирует приложение с момента его запуска и не позволяет ему причинять ущерб.

Не зависящий от архитектуры

Компилятор генерирует объектный файл, формат которого не зависит от архитектуры компьютера, — скомпилированная программа может выполняться на любых процессорах под управлением системы выполнения программ языка Java. Для этого компилятор языка Java генерирует команды байт-кода, не зависящие от конкретной архитектуры компьютера. Байт-код разработан таким образом, чтобы на любой машине его можно было легко интерпретировать либо на лету перевести в машинозависимый код.

Это не новая идея. Более 20 лет назад и в системе реализации языка Pascal, разработанной Никлаусом Виртом (Niclaus Wirth), и в системе UCSD Pascal применялась та же самая технология. Использование байт-кодов дает большой выигрыш при выполнении программы (правда, синхронная компиляция во многих случаях его компенсирует). Разработчики языка Java прекрасно справились с разработкой набора команд байт-кода, которые отлично работают на большинстве современных компьютеров, легко транслируясь в реальные машинные команды.

Машинонезависимый

В отличие от языков C и C++, в спецификации Java нет аспектов, зависящих от системы реализации. И размер основных типов данных, и арифметические операции над ними точно определены.

Например, тип `int` в языке Java всегда означает 32-разрядное целое число. В языках C и C++ тип `int` может означать как 16-разрядное, так и 32-разрядное целое число, а также целое число произвольного размера, по выбору разработчика конкретного компилятора. Единственное ограничение заключается в том, что размер типа `int` не может быть меньше размера типа `short int` и больше размера типа `long int`. Фиксированный размер числовых типов позволяет избежать многих неприятностей, связанных с выполнением программ на разных компьютерах. Бинарные данные хранятся и передаются в фиксированном формате, что также позволяет избежать недоразумений, связанных с разным порядком записи байтов на разных платформах (конфликт "big endian/little endian"). Строки сохраняются в стандартном формате Unicode.

Библиотеки, представляющие собой часть системы, определяют машинонезависимый интерфейс. Например, в языке предусмотрен абстрактный класс Window и его реализации для операционных систем Unix, Windows и Macintosh.

Каждый, кто когда-либо пытался написать программу, которая одинаково хорошо работала бы под управлением операционных систем Windows, Macintosh и десяти разновидностей системы Unix, знает, что это очень трудная задача. Версия Java 1.0 предприняла героическую попытку решить эту проблему, предоставив простой инструментальный набор, адаптирующий обычные элементы пользовательского интерфейса к большому количеству программных платформ. К сожалению, библиотека, на

которую было затрачено немало труда, не позволила достичь приемлемых результатов на разных платформах. (При этом на разных платформах в графических программах проявлялись *разные* ошибки.) Однако это было лишь началом. Во многих приложениях машинная независимость намного важнее изысканности графического пользовательского интерфейса. Именно эти приложения выиграли от появления версии Java 1.0. Однако теперь инструментальный набор для создания графического пользовательского интерфейса полностью переработан и больше не зависит от интерфейса пользователя на главном компьютере. Новая версия более осмысленна и, по нашему мнению, более привлекательна для пользователя, чем предыдущие.

Интерпретируемый

> *Интерпретатор языка Java может пересылаться на любую машину и выполнять байт-код непосредственно на ней. Поскольку редактирование связей – более легкий процесс, разработка программ может стать намного быстрее и эффективнее.*

Возможно, это дает преимущество при разработке приложений, однако приведенная цитата — явное преувеличение. В любом случае компилятор языка Java, входящий в набор инструментальных средств JSDK (Java Software Development Kit), работает довольно медленно. (Некоторые компиляторы, принадлежащие к третьей разновидности, например, компиляторы компании IBM, работают намного быстрее.) Скорость перекомпиляции — это всего лишь один из факторов, характеризующих эффективность среды программирования. Сравнив скорость работы сред программирования на языке Java и языке Visual Basic, вы, возможно, будете разочарованы.

Высокопроизводительный

> *Хотя обычно интерпретируемые байт-коды имеют более чем достаточную производительность, бывают ситуации, в которых требуется еще более высокая эффективность. Байт-коды можно "на лету" (во время выполнения) транслировать в машинные коды для конкретного процессора, на котором выполняется данное приложение.*

Если для выполнения байт-кодов применяется интерпретатор, не следует употреблять словосочетание "высокая производительность". Однако на многих платформах возможен другой вид компиляции, обеспечиваемый синхронными компиляторами (just-in-time compilers — JIT). Они транслируют байт-код в машинозависимый код, сохраняют результат в памяти, а затем вызывают его при необходимости. Поскольку при этом интерпретация выполняется только один раз, этот подход во много раз увеличивает скорость работы. Несмотря на то что синхронные компиляторы все равно медлительнее, чем машинозависимые компиляторы, они во всяком случае работают намного быстрее интерпретаторов, обеспечивая для некоторых программ 10- и даже 20-кратное ускорение. Эта технология постоянно совершенствуется и в конце концов может достичь той скорости, которую никогда не превзойдут традиционные компиляторы. Например, синхронный компилятор может определить, какой фрагмент кода выполняется чаще, и оптимизировать его по скорости выполнения.

Многопоточный

Многопоточность обеспечивает лучшую интерактивность и выполнение программы.

Если вы когда-либо пытались организовать многопоточные вычисления на каком-нибудь еще языке программирования, вы будете приятно удивлены тем, как это легко сделать на языке Java. Потоки в языке Java могут использовать преимущества многопроцессорных систем, если операционная система позволяет это сделать. К сожалению, реализации потоков на большинстве платформ сильно отличаются друг от друга, а разработчики языка Java не предпринимают никаких попыток достичь единообразия. Только код для вызова потоков остается одинаковым для всех машин; язык Java перекладывает реализацию многопоточности на базовую операционную систему или библиотеку потоков. (Потоки описываются во втором томе.) Несмотря на это, именно легкость организации многопоточных вычислений делает язык Java таким привлекательным для разработки программного обеспечения серверов.

Динамичный

Во многих отношениях язык Java является более динамичным, чем языки C или C++. Он был разработан так, чтобы легко адаптироваться к постоянно изменяющейся среде. В библиотеки можно свободно добавлять новые методы и объекты, не причиняя никакого вреда. Язык Java позволяет легко получать информацию о ходе выполнения программы.

Это очень важно в тех случаях, когда требуется добавить код в уже выполняемую программу. Ярким примером этого является код, который загружается из Интернет для выполнения броузером. В версии Java 1.0 получить информацию о ходе выполняемой программы было совсем не легко, однако нынешняя версия языка Java раскрывает перед программистом как структуру, так и поведение объектов выполняемой программы. Это весьма ценно для систем, которые должны анализировать объекты в ходе выполнения программы. К таким системам относятся средства создания графического пользовательского интерфейса, интеллектуальные отладчики, сменные компоненты и объектные базы данных.

Язык Java и Интернет

Идея проста — пользователи загружают байт-коды языка Java из Интернет и выполняют их на своих машинах. Программы Java, работающие под управлением Web-броузеров, называются *аплетами*. Для использования аплета нужен Web-броузер, поддерживающий язык Java и способный интерпретировать байт-коды. Лицензия на исходные коды языка Java принадлежит компании Sun, настаивающей на неизменности как самого языка, так и структуры его основных библиотек. К сожалению, в реальности все не так. Разные версии броузеров Netscape и Internet Explorer поддерживают разные версии языка Java, причем некоторые из этих версий значительно устарели. Эта достойная сожаления ситуация создает все больше препятствий при разработке аплетов, позволяющих использовать преимущества последней версии языка Java. Чтобы решить эту проблему, компания Sun разработала программу *Java Plug-in*, позво-

ляющую создавать наиболее современную среду для запуска программ на языке Java на основе броузеров Netscape и Internet Explorer (глава 10).

Загрузка аплета напоминает внедрение изображения в Web-страницу. Аплет становится частью страницы, а текст обтекает занятое им пространство. Однако отличие заключается в том, что изображение теперь является *живым* (alive). Оно реагирует на команды пользователя, изменяет свой внешний вид и обеспечивает пересылку данных между компьютером, на котором просматривается аплет, и компьютером, управляющим этим аплетом.

Загрузка аплета напоминает вставку рисунка в Web-страницу. Аплет становится частью страницы, а текст обтекает занимаемое им место. Дело в том, что изображение является "*живым*". Оно реагирует на команды пользователя, изменяет свой внешний вид и выполняет пересылку данных между компьютером, на котором выполняется аплет, и компьютером, управляющим этим аплетом.

На рис. 1.1 показан хороший пример динамической Web-страницы, выполняющей сложные вычисления и применяющей аплет для изображения молекул. Чтобы лучше понять структуру молекулы, можно вращать ее либо изменять масштаб изображения, используя мышь. Такие манипуляции нельзя реализовать на статических Web-страницах, однако аплеты делают это возможным. (Этот аплет можно найти по адресу `http://jmol.sourceforge.net`.)

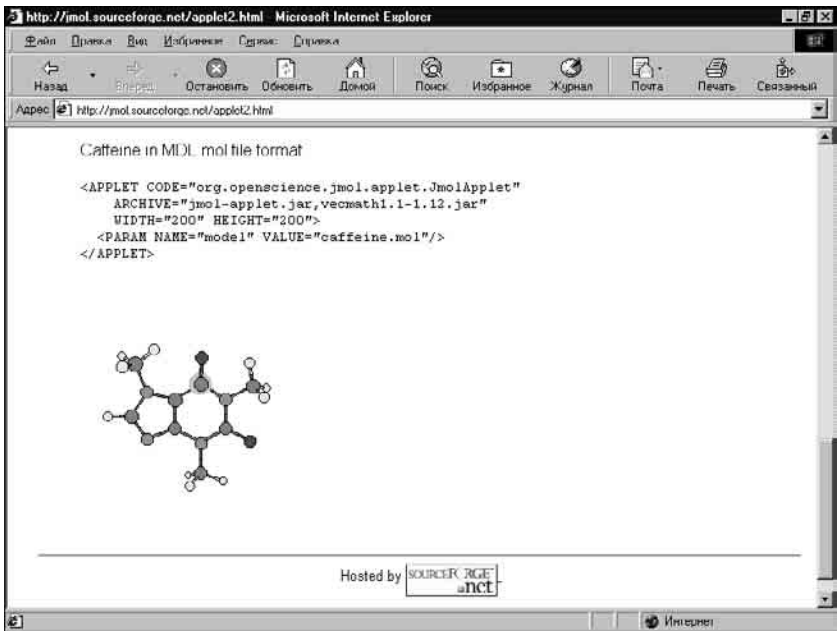

Рис. 1.1. Аплет Jmol

С помощью аплетов на Web-страницу можно добавлять новые кнопки и текстовые поля. Однако такие аплеты медленно загружаются по телефонной линии. Почти то же самое можно сделать с помощью языка Dynamic HTML, форм языка HTML (Hypertext Markup Language – язык разметки гипертекстов) или языка сценариев, например, языка JavaScript. Разумеется, первые аплеты предназначались для анимации: вращающиеся глобусы, танцующие персонажи мультфильмов, вычурные тексты и т.п.

Однако большинство из перечисленного могут делать и анимированные GIF-файлы, а язык Dynamic HTML в сочетании с разработкой сценариев делает намного больше, чем аплеты.

В результате несовместимости броузеров и несогласованности процесса загрузки через медленные сетевые соединения аплеты, предназначенные для Web-страниц, не стали огромным достижением. В *локальных сетях* (intranets) ситуация совершенно иная. В них обычно не возникают проблемы, связанные с пропускной способностью канала, поэтому время загрузки аплетов несущественно. В локальной сети можно выбирать нужный броузер или применять программу Java Plug-In. Сотрудники не могут переместить программу, доставленную через сеть, в неправильное место или неверно ее установить, а системному администратору не нужно обходить все клиентские машины и обновлять на них программы. Большое количество программ, предназначенных для учета товаров, планирования отпуска, возмещения транспортных расходов и тому подобного, были разработаны многими корпорациями в виде аплетов, использующих броузеры.

Пока мы писали книгу, маятник вновь качнулся от клиентских программ в сторону *программирования серверных приложений* (server-side programming). В частности, *прикладные серверы* (application servers) могут использовать мониторинговые возможности виртуальной машины языка Java для автоматического выравнивания нагрузки, объединения связей с базой данных, синхронизации объектов, безопасного завершения работы и повторной загрузки, а также для выполнения других процессов, необходимых для масштабируемых серверных приложений, которые почти невозможно корректно реализовать. Таким образом, программисты, создающие приложения, получили возможность купить эти сложные механизмы, вместо того, чтобы разрабатывать их самостоятельно. Это повысило производительность труда программистов — они сосредоточились на логике своих программ, не отвлекаясь на детали, связанные с работой серверов.

Краткий курс истории языка Java

В этом разделе кратко описывается история языка Java. В основу этого раздела положены различные опубликованные материалы (в основном, интервью с создателями языка Java в июльском выпуске электронного журнала *SunWorld* за 1995 год).

История Java восходит к 1991 году, когда группа инженеров из компании Sun под руководством Патрика Нотона (Patrick Naughton) и члена Совета директоров (и разностороннего компьютерного волшебника) Джеймса Гослинга (James Gosling) занялась разработкой небольшого языка, который можно было бы использовать для программирования бытовых устройств, например, контроллеров для переключения каналов кабельного телевидения (cable TV switchboxes). Поскольку такие устройства не потребляют много энергии и не имеют больших микросхем памяти, язык должен был быть маленьким и генерировать очень компактные программы. Кроме того, поскольку разные производители могут выбирать разные центральные процессоры (Central Processor Unit — CPU), было важно не завязнуть в какой-то одной архитектуре компьютеров. Проект получил кодовое название “Green”.

Стремясь изобрести небольшой, компактный и машинонезависимый код, разработчики возродили модель, использованную при реализации первых версий языка Pascal на заре эры персональных компьютеров. Никлаус Вирт, создатель языка Pascal, в свое время разработал машинонезависимый язык, генерирующий промежуточный

код для некоей гипотетической машины. Этот язык стал коммерческим продуктом под названием UCSD Pascal. (Такие гипотетические машины часто называются *виртуальными* — например, виртуальная машина языка Java, или JVM.) Этот промежуточный код можно выполнять на любой машине, имеющей соответствующий интерпретатор. Инженеры, работавшие над проектом "Green", также использовали виртуальную машину, что решило их основную проблему.

Однако большинство сотрудников компании Sun имели опыт работы с операционной системой UNIX, поэтому в основу разрабатываемого ими языка был положен язык C++, а не Pascal. В частности, они сделали язык объектно-, а не процедурно-ориентированным. Как сказал Гослинг в своем интервью: "Язык — это всегда средство, а не цель". Сначала Гослинг решил назвать его "Oak" ("Дуб"). (Возможно потому, что он любил смотреть на дуб, растущий прямо под окнами его офиса в компании Sun.) Потом сотрудники компании Sun узнали, что слово Oak уже используется в качестве имени ранее созданного языка программирования, и изменили название на Java.

В 1992 году в рамках проекта Green была выпущена первая продукция, названная "*7". Это было средство для чрезвычайно интеллектуального дистанционного управления. (Оно имело мощность рабочей станции SPARK, помещаясь в коробочке размером 6x4x4 дюйма.) К сожалению, ни одна из компаний — производителей электронной техники не заинтересовалась этим изобретением. Затем группа стала заниматься разработкой устройства для кабельного телевидения, которое могло бы осуществлять новые виды услуг, например, включать видеосистему по требованию. И снова они не получили ни одного контракта. (Забавно, что одной из компаний, отказавшихся подписать с ними контракт, руководил Джим Кларк (Jim Clark) — основатель компании Netscape, впоследствии сделавшей очень много для успеха языка Java.)

Весь 1993 год и половину 1994 года продолжались безрезультатные поиски покупателей продукции, разработанной в рамках проекта "Green" (под новым названием "First Person, Inc."). (Патрик Нотон, один из основателей группы, впоследствии в основном занимавшийся маркетингом, налетал в общей сложности более 300 тысяч миль, пытаясь продать разработанную технологию.) Проект "First Person, Inc." был прекращен в 1994 году.

Тем временем в рамках Интернет разрасталась сеть World Wide Web. Ключом к этой сети является броузер, превращающий гипертекст в изображение на экране. В 1994 году большинство людей пользовалось броузером Mosaic, некоммерческим Web-броузером, разработанным в суперкомпьютерном центре Университета штата Иллинойс (University of Illinois) в 1993 году. (Частично этот броузер был написан Марком Андреессеном (Mark Andreessen) за 6,85 доллара в час. В то время Марк заканчивал университет и броузер был его дипломной работой. Затем он стал одним из основателей и главным программистом компании Netscape, и к нему пришли слава и богатство.)

В своем интервью журналу *Sun World* Гослинг сказал, что в середине 1994 года разработчики языка поняли: "Нам нужно создать действительно крутой броузер. Такой броузер должен представлять собой одно из немногих приложений модной клиент-серверной технологии, в которой жизненно важным было бы именно то, что мы сделали: архитектурная независимость, выполнение в реальном времени, надежность, безопасность — вопросы, не являвшиеся чрезвычайно важными для рабочих станций. И мы создали такой броузер".

На самом деле броузер был разработан Патриком Нотоном и Джонатаном Пэйном (Johnatan Payne). Позднее он превратился в современный броузер HotJava. Этот бро-

узер был написан на языке Java, чтобы продемонстрировать всю его мощь. Однако разработчики не забывали о мощных средствах, которые теперь называются аплетами, наделив свой броузер способностью выполнять код внутри Web-страниц. "Демонстрация технологии" была представлена на выставке SunWorld '95 23 мая 1995 года и вызвала всеобщее помешательство на почве языка Java, продолжающееся и поныне.

Компания Sun выпустила первую версию языка Java в начале 1996 года. Через несколько месяцев после нее появилась версия Java 1.02. Люди быстро поняли, что версия Java 1.02 не подходит для разработки серьезных приложений. Конечно, эту версию можно применять для разработки Web-страниц с пляшущими человечками, однако в версии Java 1.02 ничего нельзя даже *напечатать*. Честно говоря, версия Java 1.02 была еще сырой. Ее преемница, версия Java 1.1, заполнила большинство зияющих провалов, намного улучшив возможность отражения и добавив новую модель событий для программирования графического пользовательского интерфейса. Несмотря на это, она все еще была довольно ограниченной.

Выпуск версии Java 1.2 стал основной новостью конференции JavaOne в 1998 году. В новой версии слабые средства для создания графического пользовательского интерфейса и графических приложений были заменены сложным и масштабным инструментарием. Это был шаг вперед, к реализации лозунга "Write Once, Run Anywhere"™ ("Один раз напиши — и везде выполняй"), выдвинутого при разработке предыдущих версий. В декабре 1998 года через три дня (!) после выхода в свет название новой версии было изменено на громоздкое словосочетание *Java 2 Standart Edition Software Development Kit Version 1.2* (Стандартное издание пакета инструментальных средств для разработки программного обеспечения на языке Java 2, версия 1.2).

Кроме стандартного издания пакета ("Standart Edition") были предложены еще два варианта: "микроиздание" ("Micro Edition") для портативных устройств, например, для мобильных телефонов, и "промышленное издание" ("Enterprise Edition") для создания серверных приложений. В нашей книге в центре внимания находится стандартное издание.

Версии 1.3 и 1.4 стандартного издания пакета инструментальных средств намного совершеннее первоначального выпуска языка Java 2. Они обладают новыми возможностями и, разумеется, содержат намного меньше ошибок. В табл. 1.1 показан стремительный рост объема библиотеки API по мере появления новых версий стандартного издания пакета SDK.

Таблица 1.1. Рост объема библиотеки API из пакета Java Standart Edition

Версия	*Количество классов и интерфейсов*	*Количество методов и полей*
1.0	212	2125
1.1	504	5478
1.2	1781	20935
1.3	2130	23901
1.4	3020	32138

Распространенные заблуждения относительно языка Java

В этом разделе перечислены и прокомментированы некоторые из распространенных заблуждений, касающихся языка Java.

Язык Java – это расширение языка HTML.

Java — это язык программирования, а язык HTML — это способ описания структуры Web-страниц. Между ними нет ничего общего, за исключением расширений языка HTML, позволяющих размещать на Web-страницах аплеты, написанные на языке Java.

Я работаю на языке XML, поэтому мне не нужен язык Java.

Язык Java — это язык программирования, а язык XML — просто способ описания данных. Данные, описанные с помощью языка XML, можно обрабатывать с помощью программ, написанных на любом языке программирования, но лишь библиотека API языка Java содержит превосходные средства поддержки для обработки таких данных. Кроме того, в языке Java реализованы многие возможности языка XML. Более подробно они описаны во втором томе.

Язык Java легко выучить.

Нет ни одного языка программирования, столь же мощного, как язык Java, который можно было бы легко выучить. Игрушечные программы писать довольно просто, однако выполнять серьезную работу всегда трудно. Кроме того, обратите внимание на то, что в этой книге обсуждению собственно языка Java посвящено только четыре главы. В остальных главах обьясняется, как работать с его библиотеками, содержащими тысячи классов и интерфейсов, а также многие тысячи функций. К счастью, знать каждую из них не обязательно, однако для выполнения реального проекта от пользователя требуется на удивление много знаний.

На языке Java легко программировать.

Набор инструментальных средств Java SDK использовать нелегко всем, за исключением программистов, привыкших к режиму командной строки. Существуют интегрированные среды программирования, включающие текстовые редакторы, компиляторы, средства для создания форм с помощью технологии “drag-and-drop”, однако для новичка они выглядят слишком сложными и устрашающими. Кроме того, они часто генерируют программы, состоящие из сотен строк. Нам кажется, что начинать изучение языка Java по длинным программам, сгенерированным автоматически и заполненным комментариями типа “РУКАМИ НЕ ТРОГАТЬ!”, — не слишком удачная идея. Мы полагаем, что лучше всего изучать язык Java, пользуясь привычным текстовым редактором. Именно так мы и поступим.

Язык Java со временем станет универсальным языком программирования для всех платформ.

Теоретически это возможно. Именно об этом мечтают все производители программного обеспечения, кроме компании Microsoft. Однако есть много приложений, прекрасно работающих на персональных компьютерах, которые не будут так же хорошо работать на других устройствах или под управлением броузера. Кроме того, эти приложения написаны так, чтобы максимально использовать возможности процессоров и машинозависимых библиотек. В любом случае они уже перенесены на все важные платформы. К таким приложениям относятся текстовые и графические редак-

торы, а также Web-броузеры. Обычно эти приложения пишутся на языках C или C++, и пользователь ничего не выиграет, если переписать их на языке Java. Помимо всего прочего, после переписывания на языке Java эти программы станут более медленными и менее эффективными, по крайней мере, в ближайшем будущем.

Язык Java — это просто еще один язык программирования.

Java — прекрасный язык программирования. Многие программисты отдают предпочтение именно ему, а не языкам C или C++. Однако в мире существуют сотни прекрасных языков программирования, так никогда и не получивших широкого распространения, в то время как языки с очевидными недостатками, такие как C++ и Visual Basic, достигли ошеломительных успехов.

Почему? Успех любого языка программирования в основном определяется практичностью его *системы поддержки* (support system), а не элегантностью его синтаксиса. Существуют ли полезные, удобные и стандартные библиотеки, позволяющие сделать именно то, что необходимо программисту? Разработана ли удобная среда для создания и отладки программ? Интегрирован ли язык и его инструментарий в компьютерную инфраструктуру? Язык Java достиг успехов в области серверных приложений, поскольку его библиотеки классов позволяют легко сделать то, что раньше было трудным, например, поддерживать работу в сети и многопоточность. Тот факт, что язык Java уменьшил количество ошибок, связанных с указателями, также говорит в его пользу. Благодаря этому производительность труда программистов повысилась. Однако не в этом кроется причина его успеха.

Программы на языке Java интерпретируются, значит, серьезные приложения будут слишком медленно выполняться на конкретной платформе.

Многие программы затрачивают большую часть времени на взаимодействие с пользовательским интерфейсом или ожидание данных из сети. Все программы, независимо от того, на каком языке они написаны, должны реагировать на щелчок мыши за определенное время. Разумеется, не следует выполнять задачи, требующие высокой производительности центрального процессора, с помощью интерпретатора языка Java. Однако на платформах, допускающих синхронную компиляцию, нужно лишь запустить байт-коды на выполнение, и большинство вопросов просто отпадут. В конце концов язык Java отлично подходит для разработки сетевых программ. Опыт показывает, что он легко поддерживает высокую скорость передачи данных в сети даже во время таких интенсивных вычислений, как шифрование. Поскольку скорость работы на языке Java выше скорости передачи данных в сети, становится совершенно неважным, что программы на языке C++ могут работать еще быстрее. На языке Java легче программировать, а программы, написанные на нем, машинонезависимы.

Все программы на языке Java выполняются под управлением Web-броузеров.

Все *аплеты*, написанные на языке Java, действительно выполняются под управлением Web-броузеров. Собственно, это и есть определение аплета — программа, выполняемая Web-броузером. Однако вполне возможно и уместно создавать самостоятельные программы на языке Java, которые выполняются независимо от Web-броузера. Эти программы (обычно называемые *приложениями*) являются полностью машинонезависимыми. Просто берите программу и выполняйте ее на другой машине! Поскольку язык Java более удобен и менее подвержен ошибкам, чем язык C++, он представляет собой хороший выбор. В сочетании со средствами доступа к базам данных, например, пакетом Java Database Connectivity (см. том 2), язык Java становится

просто неотразимым. Особенно удобно его изучать в качестве первого языка программирования.

Большинство программ в этой книге являются абсолютно самостоятельными. Конечно, аплеты очень забавны. Однако приложения на языке Java на практике более важны и полезны.

Аплеты на языке Java представляют собой большую опасность для системы защиты.

Было опубликовано несколько отчетов об ошибках в системе безопасности языка Java. Большинство из них касалось реализации языка Java с помощью конкретного броузера. Исследователи восприняли это как вызов и принялись искать лазейки в системе защиты языка Java, чтобы преодолеть силу и сложность модели безопасности аплетов. Обнаруженные технические ошибки вскоре были исправлены, и, насколько мы знаем, ни одна реальная система никогда не была скомпрометирована. Чтобы оценить важность этого факта, вспомните о миллионах вирусных атак на выполняемые файлы операционной системы Windows и макросы текстового процессора Word, действительно вызвавшие немало бед, и удивительно беззубую критику слабостей атакованной платформы. Кроме того, механизм ActiveX в броузере Internet Explorer мог бы вызвать много нареканий, однако способы его взлома настолько очевидны, что лишь немногие специалисты потрудились опубликовать свои изыскания.

Некоторые системные администраторы даже стали выключать системы защиты языка Java в своих броузерах, чтобы пользователи могли, как и прежде, загружать исполняемые файлы, элементы управления ActiveX и документы, созданные с помощью текстового процессора Word. Забавно, что в настоящее время вероятность преодоления системы защиты аплетов на языке Java сравнима с вероятностью погибнуть в авиационной катастрофе, в то время как риск заразить компьютер, открыв документ, созданный текстовым процессором Word, сравним с риском погибнуть под колесами автомобиля, перебегая дорогу в час пик.

Язык JavaScript — упрощенная версия языка Java.

Язык JavaScript — это язык разработки сценариев, который можно *использовать на Web-страницах*. Он был разработан компанией Netscape и сначала назывался LiveScript. Синтаксис языка JavaScript напоминает синтаксис языка Java, однако на этом их сходство заканчивается (за исключением имени, конечно). Подмножество языка JavaScript было стандартизовано под именем ECMA-262, однако его расширения, необходимые для реальной работы, стандартизованы не были. В результате программа на языке JavaScript, которую можно было бы выполнять как под управлением броузеров компании Netscape, так и с помощью броузера Internet Explorer, осталась несбыточной мечтой.

Для разработки сценариев графического пользовательского интерфейса нужно использовать язык Java, а не Perl.

Это правда лишь наполовину. Для разработки серверных приложений больше не следует применять не только язык Perl, но и сами сценарии графических пользовательских интерфейсов (CGI-сценарии). Сервлеты языка Java превосходно справляются с решением этой задачи. Они работают намного эффективнее, чем CGI-сценарии, и для их реализации можно использовать реальный язык программирования — язык Java.

Язык Java совершит революцию в области клиент-серверных приложений.

Это возможно. Именно в этой области язык Java особенно полезен. Некоторые серверные приложения, например BEA Weblogic, полностью встроены в язык Java.

Язык Java отменит компонентную модель вычислений.

Вряд ли найдутся хотя бы два человека, у которых совпали бы мнения о компонентах. В качестве визуальных компонентов управления в версии Java 1.1 были предложены функциональные компоненты JavaBeans (см. том 2), аналогичные компонентам ActiveX, использующимся для создания графического пользовательского интерфейса. Функциональные компоненты JavaBeans позволяют делать то же самое, что и компоненты ActiveX, *за исключением* того, что они автоматически являются кросс-платформенными. С точки зрения серверных приложений *промышленные компоненты*, предназначенные для повторного использования, легко применять при разработке широкого круга серверных приложений. Возможно, в будущем возникнет рынок этих компонентов, аналогичный рынку компонентов ActiveX в мире Windows.

Пользуясь языком Java, можно заменить компьютер "Интернет-устройством" стоимостью 500 долларов.

Некоторые специалисты уверены, что это непременно случится. Нам кажется абсурдным думать, что домашние пользователи могут заменить мощные и удобные персональные компьютеры ограниченными машинами без жестких дисков. Однако сетевой компьютер, оснащенный средствами языка Java, предоставляет вполне реальную возможность реализовать "нулевую администраторскую инициативу", направленную на уменьшение стоимости компьютеров, занятых в бизнесе.

Мы рассматриваем Интернет-устройство как *дополнение* к персональному компьютеру. Если цена позволяет, почему бы не приобрести устройство, поддерживающее работу в Интернет, для просмотра на экране электронной почты и новостей? Ядро языка Java настолько мало, что он является очевидным выбором для такого рода телефонов и других "Интернет-устройств".

Глава 2

Среда программирования на языке Java

В этой главе...

- Инсталляция набора инструментальных средств Java Software Development Kit
- Среда разработки программ
- Использование инструментов в режиме командной строки
- Использование интегрированной среды разработки программ
- Компилирование и запуск программ из текстового редактора
- Графические приложения
- Аплеты

В этой главе мы научимся инсталлировать набор инструментальных средств Java Software Development Kit (SDK), а также компилировать и запускать на выполнение программы различных видов: консольные программы, графические приложения и аплеты. Мы будем использовать средства пакета SDK, набирая команды в окне оболочки. Однако многие программисты предпочитают удобство интегрированной среды разработки программ. Мы покажем читателю, как использовать широко распространенную интегрированную среду разработки программ для компиляции и выполнения программ, написанных на языке Java. Изучать интегрированные среды разработки программ и пользоваться ими довольно легко, однако они долго загружаются и предъявляют высокие требования к ресурсам компьютера. В качестве ком-

промисса можно использовать текстовый процессор, из которого вызывается компилятор и интерпретатор языка Java. В главе описано несколько таких текстовых редакторов. Овладев приемами, рассмотренными в этой главе, и выбрав инструмент для разработки программ, читатель может переходить к главе 3, где и начинается изучение языка программирования Java.

Инсталляция набора инструментальных средств Java Software Development Kit

Наиболее полная и современная версия пакета Java 2 Standart Edition (J2SE) предназначена для операционных систем SOLARIS, Linux и Windows. Существуют также разные версии языка Java для операционных систем Linux, OS/2, Macintosh и многих других платформ.

Читатель, работающий под управлением операционных систем Solaris, Linux или Windows, может загрузить пакет Java Software Development Kit с Web-страницы `http://java.sun.com/j2se`. Способы инсталляции на разных платформах отличаются друг от друга. На момент написания этой книги инсталляционные пакеты были доступны на следующих Web-страницах:

- `http://java.sun.com/j2se/1.4/install-solaris.html`
- `http://java.sun.com/j2se/1.4/install-linux.html`
- `http://java.sun.com/j2se/1.4/install-windows.html`

От системы зависят только команды инсталляции и компиляции. Установив набор инструментальных средств, можно делать с ним все, что описано в этой книге. Независимость от операционной системы — основное преимущество языка Java.

При инсталляции пользователю по умолчанию предлагается использовать каталог `j2sdk1.4`, название которого повторяет название версии пакета JSDK. При желании можно изменить это имя на `jdk`. Однако если вы приверженец языка Java и коллекционируете разные версии пакета Java SDK, принимайте предложение по умолчанию. В этой книге мы предполагаем, что каталог называется `jdk`. Например, ссылаясь на каталог `jdk/bin`, мы имеем в виду каталог `bin` внутри каталога `jdk`. Заметим также, что для имен каталогов мы используем стиль операционной системы UNIX. При работе на платформе Windows нужно использовать обратную косую черту, например: `c:\jdk\bin`.

Задание путей к выполняемым файлам

После инсталляции пакета Java SDK вам нужно сделать еще один шаг: добавить имя каталога `jdk/bin` в список путей, по которым операционная система может найти выполняемые файлы. Этот шаг в разных системах выполняется по-разному.

- На платформе UNIX (включая операционные системы Solaris или Linux) процедура редактирования путей к выполняемым файлам зависит от используемой

оболочки. Если используется оболочка C (принимаемая на платформе Solaris по умолчанию), добавьте строку, приведенную ниже, в конец файла `~/.cshrc`:

```
set path=(/usr/local/jdk/bin $path)
```

Если вы используете оболочку Bourne Again (принимаемую по умолчанию на платформе Linux), добавьте строку, приведенную ниже, в конец файла `~/.bashrc` или `~/.bash_profile`:

```
export PATH=/usr/local/jdk/bin:$PATH
```

Для других оболочек операционной системы UNIX нужно найти способ выполнить аналогичную процедуру.

- На платформе Windows 95/98 добавьте строку, приведенную ниже, в конец файла `AUTOEXEC.BAT`:

  ```
  SET PATH=c:\jdk\bin;%PATH%
  ```

  Заметим, что вокруг знака = *нет никаких пробелов*. Чтобы эта строка подействовала, нужно перезагрузить компьютер.

- На платформе Windows NT/2000 откройте панель управления, выберите пиктограмму System, а затем — вкладку Environment. Прокрутите на экране окно User Variables, пока не найдете переменную `PATH`. Добавьте имя каталога `jdk/bin` в начало списка, используя точку с запятой для того, чтобы отделить новый элемент списка от уже существующих, например:

  ```
  c:\jdk\bin; остальное
  ```

  Сохраните свои установки. Любое новое консольное окно, которое вы создадите, будет использовать правильный путь к выполняемым файлам.

Вот как можно проверить, правильно ли вы все сделали.

Откройте окно оболочки. Как вы это сделаете, зависит от операционной системы. Наберите следующую строку:

```
java -version
```

и нажмите клавишу <ENTER>. На экране должно появиться следующее сообщение.

```
java version "1.4.0"
Java (TM) 2 Runtime Environment, Standard Edition
Java HotSpot (TM) Client VM
```

Если вместо этого сообщения появляется строка наподобие “java:command not found”, “Bad command or file name” или “The name specified is nor recognized as an internal or external command, operable program or batch file”, нужно вернуться и еще раз проверить, правильно ли выполнена инсталляция.

Инсталляция библиотек и документации

Библиотечные файлы поставляются в пакете Java SDK в виде архива `src.jar`. Вам следует распаковать этот файл, чтобы получить доступ к исходным текстам программ. Мы настоятельно рекомендуем вам сделать это. Выполните следующие инструкции.

1. Убедитесь, что пакет Java SDK установлен, а имя каталога `jdk/bin` находится в списке путей к выполняемым файлам.
2. Откройте командную оболочку.
3. Перейдите в каталог `jdk` (т.е. в каталог `/usr/local/jdk` или `C:/jdk`).
4. Выполните команду:
   ```
   jar xvf src.jar
   ```

Файл `src.jar` содержит исходные тексты всех открытых библиотек. Чтобы получить еще больше исходных текстов (для компилятора, виртуальной машины, машинозависимых методов и закрытых вспомогательных классов), зайдите на Web-страницу `http://www.sun.com/software/communitysource/java2`.

Документация содержится в отдельном архиве, который может иметь различные расширения (`.zip`, `.gz` и `.Z`). Ее можно загрузить с Web-страницы `http://java.sun.com/docs`. Выбирайте тот формат, который вам подходит. Если сомневаетесь, выбирайте формат `.zip`, поскольку такой файл можно распаковать с помощью программы `jar`, входящей в пакет Java SDK. Если вы решили использовать программу `jar`, выполните следующие действия.

1. Убедитесь, что пакет Java SDK установлен правильно, а имя каталога `jdk/bin` находится в списке путей к выполняемым файлам.
2. Загрузите файл с расширением `.zip`, содержащий документацию, в каталог, содержащий подкаталог `jdk` (например, каталог `/user/local` или `C:\`). Этот файл называется `j2sdk`*`version`*`-doc.zip`, где слово *`version`* может означать что-нибудь, вроде `1_4_0`.
3. Откройте командную оболочку.
4. Зайдите в каталог, содержащий подкаталог `jdk` и архив с документацией.
5. Выполните следующую команду:
   ```
   jar xvf j2sdkversion-doc.zip
   ```
   Здесь слово *`version`* означает соответствующий номер версии.

Установка примеров программ

Вам понадобятся примеры программ, приведенные в книге. Вы можете найти их на Web-странице `http://www.phptr.com.corejava`. Программы хранятся в архиве `corejava.zip`. Его нужно распаковать в отдельный каталог — мы рекомендуем назвать его `CoreJavaBook`. Для распаковки можно применять любую утилиту для работы с `zip`-файлами, например, программу `WinZip` (которую также можно найти по адресу `http://www.winzip.com`), либо использовать утилиту `jar`, входящую в пакет Java SDK. Если вы решили использовать программу `jar`, выполните следующие действия.

1. Убедитесь, что пакет Java SDK установлен правильно, а имя каталога `jdk/bin` находится в списке путей к выполняемым файлам.
2. Создайте каталог `CoreJavaBook`.

3. Скопируйте файл `corejava.zip` в этот каталог.
4. Откройте командную оболочку.
5. Зайдите в каталог `CoreJavaBook`.
6. Выполните следующую команду:

```
jar xvf corejava.zip
```

Навигация по каталогам

Изучая язык Java, вероятно, вы захотите заглянуть в его исходные тексты программ. Кроме того, вам, без сомнения, придется интенсивно работать с библиотечной документацией. В табл. 2.1 показано дерево каталогов, относящихся к языку Java. Схема этих каталогов может измениться, если вы работаете в интегрированной среде разработки программ, а имя корневого каталога зависит от номера инсталлированной версии.

Таблица 2.1. Дерево каталогов, относящихся к языку Java

`jdk`	(имя может быть другим, например `jdk1.4.0`)	
	`docs`	Библиотечная документация в формате HTML (после распаковки архива `j2sdkversion-doc.zip`)
	`bin`	Компилятор и другие инструментальные средства
	`demo`	Демонстрационные программы
	`include`	Файлы с машинозависимыми методами (см. том 2)
	`lib`	Библиотечные файлы
	`src`	Разные подкаталоги с исходными текстами библиотечных программ (после распаковки файла `src.jar`)
	`jre`	Файлы среды выполнения программ на языке Java

Наиболее важными среди подкаталогов являются docs и src. Каталог docs содержит документацию о библиотеках языка Java в формате HTML. Их можно просматривать с помощью Web-броузера, например, броузера Netscape.

Установите в вашем броузере закладку на локальную версию файла `docs/api/index.html`. При изучении языка Java вам придется не раз ссылаться на эту страницу.

Каталог src содержит исходные тексты программ, находящихся в открытой части библиотек языка Java. Освоившись с языком, вы можете оказаться в ситуации, когда ни информация из Интернет, ни эта книга не помогут вам решить вашу проблему. В этом случае воспользуйтесь исходными кодами языка Java. По крайней мере, они гарантируют, что вы всегда сможете по исходным текстам понять, что именно делает

та или иная библиотечная функция. Например, если вы интересуетесь, как работает класс System, загляните в файл `src/java/lang/System.java`.

Среда разработки программ

Если раньше вы программировали на языках Visual Basic или Visual C++, значит, вы уже знакомы со средой разработки, содержащей встроенный текстовый редактор, меню для компиляции и запуска программ, а также отладчик. Пакет Java SDK не имеет никаких средств, даже отдаленно напоминающих интегрированную среду разработки программ. *Все* команды выполняются путем их набора в окне оболочки. Мы рассказываем, как инсталлировать, а затем использовать пакет Java SDK, поскольку считаем, что полномасштабная среда разработки программ не обязательно облегчает процесс изучения языка Java — она может оказаться слишком сложной, скрывая некоторые важные детали, представляющие интерес для программиста.

Интегрированные среды разработки программ становятся все более неудобными для создания простых программ, поскольку они работают медленнее, требуют более мощных компьютеров и выполнения скучной процедуры установки переменных окружения для каждой вновь создаваемой программы. Интегрированные среды полезны, если вы разрабатываете большой проект, состоящий из многих файлов. Кроме того, в состав этих сред входит отладчик (debugger), крайне необходимый при разработке серьезных программ (отладчик, работающий в режиме командной строки, поставляемый в пакете Java SDK, крайне неудобно использовать). Мы покажем, как работать с программой Sun ONE Studio Community Edition, представляющей собой свободно распространяемую среду разработки, которая сама написана на языке Java. (Пока в дело не вмешались энтузиасты из отдела маркетинга компании Sun, эта программа называлась Forte.) Разумеется, если у вас уже есть среда разработки программ, например, JBuilder, Kawa, CodeWarrior или Café, поддерживающая новую версию языка Java, вы можете смело использовать ее вместе с этой книгой.

Для простых программ удобно использовать нечто среднее между работой в режиме командной строки и интегрированной средой разработки, а именно: текстовый редактор, интегрированный с пакетом Java SDK. На платформе Linux предпочтительнее использовать текстовый редактор Emacs. Под управлением операционной системы Windows удобна программа TextPad — прекрасный, свободно распространяемый редактор программ для среды Windows, хорошо интегрированный с языком Java. Подобными свойствами обладают и многие другие текстовые редакторы. Используя текстовый редактор, интегрированный с пакетом Java SDK, можно легко и быстро создавать программы на языке Java. Большинство программ, приведенных в этой книге, разработаны именно так. Поскольку компилировать и выполнять программы можно, не выходя из текстового редактора, фактически он образовывает интегрированную среду, используемую на протяжении всей книги.

Итак, у вас есть три возможных варианта выбора.

1. Использовать набор инструментальных средств Java SDK вместе со своим любимым текстовым редактором. Компиляция и запуск программы на выполнение в этом случае производится в режиме командной строки.
2. Использовать набор Java SDK и текстовый редактор, интегрированный вместе с этим пакетом. Эту и многие другие возможности предоставляют программы

TextPad и Emacs. В этом случае компиляция и запуск на выполнение программ производятся внутри редактора.

3. Использовать интегрированную среду разработки программ, например, свободно распространяемую программу Sun ONE Studio Community Edition или одну из многих других как бесплатных, так и коммерческих сред.

Использование инструментов в режиме командной строки

Есть два способа компиляции и запуска на выполнение программ, написанных на языке Java: из командной строки или из другой программы, например, интегрированной среды разработки или текстового редактора.

Откройте окно оболочки. Перейдите в каталог `CoreJavaBook/v1ch2/Welcome`. Затем введите несколько команд:

```
javac Welcome.java
java Welcome
```

На экране должно появиться сообщение, показанное на рис. 2.1.

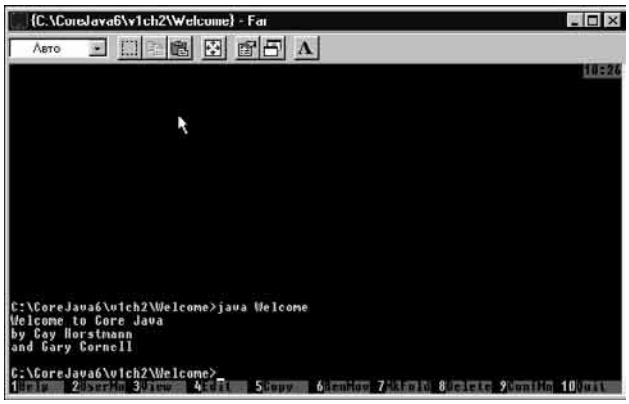

Рис. 2.1. Компиляция и выполнение программы Welcome.java

Примите наши поздравления! Вы только что скомпилировали и выполнили свою первую программу на языке Java.

Что же произошло? Программа `javac` — это компилятор языка Java. Она скомпилировала файл `Welcome.java` в файл `Welcome.class`. Программа `java` — это интерпретатор языка Java. Он интерпретирует байт-коды, которые компилятор поместил в файле с расширением `.class`.

Программа `Welcome` чрезвычайно проста. Она просто выводит сообщение на экран. Можете посмотреть на текст этой программы, представленный в листинге 2.1.

Листинг 2.1. Программа Welcome.java

```
1. public class Welcome
2. {
3.   public static void main(String[] args)
4.   {
5.     String[] greeting = new String[3];
6.     greeting[0] = "Welcome to Core Java";
7.     greeting[1] = "by Cay Hortsmann";
8.     greeting[2] = "and Gary Cornell";
9.
10.    for (int i = 0; i < greeting.length; i++)
11.       System.out.println(greeting[i]);
12.  }
13. }
```

Возможные ошибки

В эпоху визуальных сред разработки программ многие программисты не знают методов работы в режиме командной строки. По этой причине они могут сделать много ошибок и получить неверные результаты.

Обратите внимание на следующие моменты.

- Если вы набираете программу вручную, внимательно следите за употреблением прописных и строчных букв. В частности, имя класса — `Welcome`, а не `welcome` или `WELCOME`.
- Компилятор требует, чтобы *файл назывался* `Welcome.java`. Интерпретатору нужно, чтобы *класс назывался* `Welcome`, без расширения `.java` или `.class`.
- Если вы получили сообщение "Bad command or file name" или "javac:command not found", вернитесь назад и вновь проверьте, правильно ли выполнена инсталляция, в частности, верно ли указаны пути к выполняемым файлам.
- Если компилятор `javac` выдал сообщение "cannot read: Welcome.java", нужно проверить, находится ли файл в соответствующем каталоге.

  При работе под управлением операционной системы UNIX проверьте, правильно ли использованы прописные буквы в имени файла `Welcome.java`.

  При работе в среде Windows используйте команду `dir`, а не графические средства. Некоторые текстовые редакторы (в частности, Notepad) сохраняют текст в файлах с расширением `.txt`. Если вы используете эту программу для редактирования файла `Welcome.java`, она сохранит его под именем `Welcome.java.txt`. По умолчанию в среде Windows программа Explorer скрывает расширение `.txt`, поскольку такой тип является "неопределенным". В этом случае нужно переименовать файл, пользуясь командой `ren`.
- Если компилятор `java` выдал сообщение об ошибке `java.lang.NoClassDefFoundError`, внимательно проверьте имя соответствующего файла.

Если компилятор недоволен именем `welcome` (в котором буква `w` является строчной), повторно выполните команду `java Welcome`, теперь с прописной буквой `W`. В языке Java регистр буквы имеет очень большое значение.

Если интерпретатор выдает сообщение об ошибке `Welcome/java`, значит, вы случайно ввели команду `java Welcome.java`. Повторите команду `java Welcome`.

Если интерпретатор недоволен именем `Welcome`, значит, у вас неверно установлены *пути к классам*. Вам следует либо удалить текущие настройки переменных окружения, либо добавить текущий каталог (обозначенный точкой) в множество путей к классам. Детали описываются в главе 4.

- Если в вашей программе слишком много ошибок, то все сообщения о них мелькают на экране очень быстро. Интерпретатор `java` выводит сообщения об ошибках в стандартный поток, который позволяет просматривать список ошибок, занимающий больше одного экрана.

  При работе на платформах UNIX или Windows NT/2000/XP это не представляет большой сложности. Чтобы перенаправить сообщения об ошибках в файл, достаточно применить оператор `2>`:

  ```
  javac MyProg.java 2> errors.txt
  ```

  При работе под управлением операционной системы Windows 95 невозможно перенаправить сообщения об ошибках в стандартный поток ошибок, находясь в режиме командной строки. Вы можете загрузить программу `errout` с Web-страницы `http://www.horstmann.com/corejava/faq.html` и выполнить команду

  ```
  errout javac Myprog.java > errors.txt
  ```

Вы можете найти превосходное учебное пособие, которое гораздо подробнее описывает ловушки, в которые могут попасть новички, на Web-странице `http://java.sun.com.docs/books/tutorial/getStarted/cupojava`.

Использование интегрированной среды разработки программ

В этом разделе мы покажем, как скомпилировать программу в интегрированной среде Sun ONE Studio Community Edition, которая свободно распространяется компанией Sun Microsystems. Вы можете загрузить ее с Web-страницы `http://www.sun.com/software/sundev/jde`. Программа Sun ONE Studio написана на языке Java и должна работать на любой платформе, в которой есть среды выполнения программ, написанных на языке Java 2. Существуют версии этой среды разработки, специально предназначенные для операционных систем Soilaris, Linux и Windows. Открытый код этой программы можно также загрузить с Web-страницы `http://netbeans.org`.

После запуска программы Sun ONE Studio на экране появляются разнообразные панели инструментов и окна (рис. 2.2).

Выберите в меню опцию File -> Open File, а затем загрузите файл `CoreJavaBook/v1ch2/Welcome/Welcome.java`. Вас спросят, должен ли этот файл принадлежать "стандартному пакету" ("default package"). Щелкните на кнопке "Accept". (В главе 4 содержится больше информации о пакетах. Пока будем считать, что все наши программы находятся в стандартном пакете.) Теперь посмотрите на окно с текстом программы (рис. 2.3).

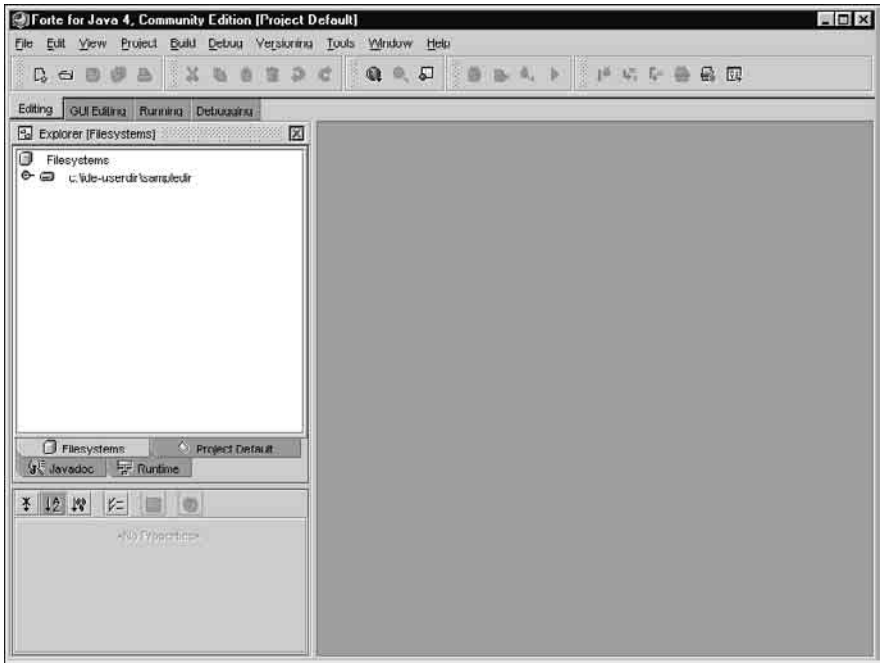

Рис. 2.2. Запуск среды Sun ONE Studio