

Содержание

Об авторе	15
Введение	17
Как работать с этой книгой	17
Соглашения, принятые в книге	18
Что можно не читать	19
Исходные предположения	19
Структура книги	21
Часть I. Основы Java	21
Часть II. Создание программы на Java	21
Часть III. Объектно-ориентированное программирование	21
Часть IV. Продвинутое методики программирования	22
Часть V. Великолепные десятки	22
Часть VI. Приложения	22
Пиктограммы, используемые в книге	22
Что дальше	23
Ждем ваших отзывов!	24
Часть I. Основы Java	25
Глава 1. Знакомство с Java	27
Что можно делать с помощью Java	28
Зачем писать программы	29
Немного истории	30
Объектно-ориентированное программирование (ООП)	33
Объектно-ориентированные языки	33
Объекты и классы	36
Преимущества объектно-ориентированного подхода	36
Наглядное представление классов и объектов	39
Что дальше	40
Глава 2. Разработка программного обеспечения	41
Быстрый старт	42
Что должно быть установлено на компьютере	44
Что такое компилятор	46
Что такое виртуальная машина Java	49
Процесс разработки	55
Интегрированная среда разработки	56

Глава 3. Базовые компоненты Java	59
Поговорим на языке Java	60
Грамматика и общие имена	60
Слова в программе на Java	62
Ваша первая программа на Java	64
Как работает ваша первая программа	65
Классы в Java	65
Методы в Java	67
Метод <code>main</code>	69
Как приказать компьютеру выполнить нужную операцию	70
Фигурные скобки	72
Добавление комментариев в код	76
Не будьте слишком строгими к старине Барри	79
Использование комментариев для экспериментирования с кодом	80
Часть II. Создание программы на Java	83
Глава 4. Переменные и значения	85
Изменение переменной	85
Инструкция присваивания	88
Типы переменных	88
Отображение текста	93
Числа без десятичной точки	94
Инициализация при объявлении	96
Эксперименты с JShell	97
А как же визуальные эффекты?	100
Примитивные типы в Java	101
Тип <code>char</code>	102
Тип <code>boolean</code>	104
Ссылочные типы	106
Объявление импорта	110
Создание новых значений с помощью операторов	112
Инициализация может быть только однократной, присваивание — многократным	116
Операторы инкремента и декремента	117
Операторы присваивания	122
Глава 5. Управляющие инструкции	125
Принятие решений с помощью инструкции <code>if</code>	126
Угадайте число	126
Ввод текста с клавиатуры	127
Генерация случайных чисел	130

Инструкция <code>if</code>	132
Двойной знак равенства	133
Блоки инструкций	133
Отступы в инструкции <code>if</code>	134
Сломанная вилка (<code>if</code> без <code>else</code>)	135
Использование блоков инструкций в JShell	137
Условия с операторами сравнения и логическими операторами	138
Сравнение чисел и символов	138
Сравнение объектов	139
Импортируем все за один раз	142
Логические операторы	143
Да здравствуют нули!	146
Условия в скобках и скобки в условиях	147
Вложение инструкций <code>if</code>	150
Переключатель <code>switch</code>	152
Выбор варианта	152
Не забывайте вставлять <code>break</code> !	155
Строковые аргументы	157
Глава 6. Циклы	159
Цикл <code>while</code>	160
Цикл <code>for</code>	164
Анатомия цикла <code>for</code>	166
Премьера моего хита “Эл под дождем”	168
Цикл <code>do</code>	171
Чтение одиночного символа с клавиатуры	174
Работа с файлами	175
Объявление переменной в блоке	176
Часть III. Объектно-ориентированное программирование	179
Глава 7. Классы и объекты	181
Определение класса	182
Объявление переменных и создание объектов	184
Инициализация переменной	187
Использование полей объектов	188
Одна программа, несколько классов	188
Открытые классы	189
Определение метода в классе	190
Счет, отображающий сам себя	192
Заголовок метода	193

Передача параметров методу и получение значения, возвращаемого методом	194
Передача значения методу	196
Значение, возвращаемое методом	198
Что необходимо сделать, чтобы числа выглядели красиво	201
Скрытие деталей с помощью методов доступа	206
Хороший стиль программирования	206
Публичная жизнь и личные мечты: как сделать поле недоступным	210
Проверка соблюдения ограничений с помощью методов доступа	212
Собственный GUI-класс Барри	213
Глава 8. Экономия времени и денег: повторное использование существующего кода	221
Определение класса	222
Что означает “быть служащим”	223
Использование класса Employee	224
Создание платежного чека	228
Работа с файлами (небольшое отступление)	229
Хранение данных в файле	230
Копирование и вставка кода	231
Чтение данных из файла	232
Куда подевался мой файл?	235
Добавление имен папок к именам файлов	236
Построчное чтение данных	237
Закрытие соединения с дисковым файлом	239
Определение подклассов	239
Создание подкласса	242
Это входит в привычку	244
Использование подклассов	245
Обеспечение соответствия типов	247
Вторая половина истории	248
Переопределение существующих методов	249
Аннотации Java	252
Вызов методов базовых и производных классов	252
Глава 9. Конструкторы	257
Определение конструктора	257
Что такое температура	258
Что такое температурная шкала	258
Так что же такое температура?	260
Что можно сделать с температурой	262
Поиск нужного конструктора	264
Некоторые вещи никогда не меняются	266

Конструктор базового класса в производном классе	269
Усовершенствованный класс <code>Temperature</code>	269
Конструкторы подклассов	271
Использование усовершенствованного класса <code>TemperatureNice</code>	272
Конструктор, выполняемый по умолчанию	273
Конструктор может не только заполнять поля	276
Классы и методы из Java API	279
Аннотация <code>@SuppressWarnings</code>	280
Часть IV. Продвинутое программирование	283
Глава 10. Размещение переменных и методов в нужных местах	285
Определение класса	285
Еще один способ красивого вывода чисел	287
Использование класса <code>Player</code>	287
Считаем до девяти	290
Графический интерфейс пользователя	291
Переброска исключения из одного метода в другой	293
Статические поля и методы	294
Зачем столько статистики	296
Статическая инициализация	297
Отображение общей статистики команды	298
Статический импорт	300
Осторожно, статика!	301
Поэкспериментируем с переменными	304
Переменная на своем месте	304
Переменные в разных местах	307
Передача параметров	312
Передача по значению	312
Возвращение результата	314
Передача по ссылке	315
Возвращение объекта из метода	316
Эпилог	318
Глава 11. Использование массивов для хранения значений	319
По порядку рассчитайсь!	319
Создание массива в два этапа	322
Сохранение значений	323
Табуляции и другие специальные символы	325
Инициализация массива	326
Расширенный цикл <code>for</code>	327
Поиск в массиве	329

Запись в файл	331
Закрытие файла	332
Массивы объектов	334
Использование класса <code>Room</code>	336
Еще один способ форматирования чисел	339
Тернарный условный оператор	340
Аргументы командной строки	342
Использование аргументов командной строки в коде	344
Проверка количества аргументов командной строки	346
Глава 12. Использование коллекций и потоков	349
Ограничения массивов	349
Классы коллекций	351
Класс <code>ArrayList</code>	351
Использование обобщенных типов	354
Классы-оболочки	356
Проверка наличия данных для чтения	359
Использование итераторов	359
Другие классы коллекций <code>Java</code>	361
Функциональное программирование	363
Решение задач программирования старым способом	365
Потоки	368
Лямбда-выражения	369
Таксономия лямбда-выражений	372
Использование потоков и лямбда-выражений	372
К чему все эти хлопоты?	378
Ссылки на методы	381
Глава 13. Как сохранить хорошую мину при плохой игре	383
Обработка исключений	385
Параметр блока <code>catch</code>	389
Типы исключений	390
Кто должен перехватить исключение	392
Блок <code>catch</code> с несколькими типами исключений	399
Не будем чрезмерно осторожничать	400
Восстановление работы программы после генерирования исключения	400
Наши друзья — хорошие исключения	401
Обработайте исключение или передайте его дальше	403
Блок <code>finally</code>	410
Использование инструкции <code>try</code> при работе с ресурсами	412

Глава 14. Область видимости переменных	415
Модификаторы доступа к членам классов	416
Классы, виды доступа и деление программ на части	417
Классы и члены классов	417
Правила доступа к членам класса	418
Помещение рисунка во фрейм	421
Структура папок	424
Создание фрейма	425
Как изменить программу, не изменяя классы	427
Доступ, применяемый по умолчанию	428
Как проникнуть в пакет	431
Защищенный доступ	433
Подклассы, находящиеся в разных пакетах	433
Включение производного класса в тот же пакет	435
Модификаторы доступа к классам	439
Открытые классы	439
Классы, не являющиеся открытыми	440
Глава 15. Ссылочные типы данных	443
Типы данных Java	443
Интерфейсы в Java	444
Два интерфейса	445
Реализация интерфейсов	447
Объединяем все вместе	449
Абстрактные классы	451
Уход за домашним любимцем	454
Использование всех ранее созданных классов	456
Расслабьтесь, это не галлюцинация	458
Глава 16. Реагирование на события клавиатуры и мыши	461
Реагирование на щелчок мышью	462
События и их обработка	464
Потоки выполнения	465
Ключевое слово <code>this</code>	467
Тело метода <code>actionPerformed</code>	468
Идентификатор версии	469
Реагирование на другие события	471
Внутренние классы	477

Глава 17. Соединение с базой данных	481
Создание базы данных	482
Что происходит при выполнении кода	483
Использование команд SQL	484
Подключение и отключение базы данных	485
Добавление данных в таблицу	486
Извлечение данных	487
Удаление данных	489
Часть V. Великолепные десятки	491
Глава 18. Десять способов избежать ошибок	493
Правильное использование регистра букв	493
Проход через блок <code>switch</code>	494
Сравнение двух значений	495
Добавление элемента в графический интерфейс	495
Добавление слушателей событий	496
Определение обязательных конструкторов	496
Исправление нестатических ссылок	496
Соблюдение границ массива	497
Указатели на <code>null</code>	497
Помогите виртуальной машине Java найти классы	498
Глава 19. Десять полезных сайтов, посвященных Java	501
Веб-сайт данной книги	501
Сайты Oracle	502
Ответы на технические вопросы	502
Часть VI. Приложения	505
Приложение А. Установка интегрированной среды разработки	507
Загрузка и установка JDK	507
Java на платформе Windows, Linux и Solaris	508
Java для Macintosh	510
Загрузка и установка Eclipse	512
Загрузка Eclipse	513
Установка Eclipse	513
Конфигурирование Eclipse	514
Приложение Б. Использование Eclipse	517
Работа с примерами книги	517
Создание собственного проекта	520

Приложение В. Ответы к упражнениям	523
Глава 3	523
Глава 4	524
Глава 5	528
Глава 6	533
Глава 7	537
Глава 8	547
Глава 9	557
Глава 10	567
Глава 11	574
Глава 12	581
Глава 13	584
Глава 14	590
Глава 15	599
Глава 16	604
Глава 17	609
Предметный указатель	613



Глава 2

Разработка программного обеспечения

В ЭТОЙ ГЛАВЕ...

- » Средства разработки программ
- » Выбор подходящей версии Java
- » Подготовка к написанию и выполнению программ на Java

Наилучший способ познакомиться с Java — создать программу на компьютере. Работая с Java, вы пишете, тестируете и выполняете программу. В этой главе мы рассмотрим общий процесс разработки программного обеспечения на произвольном компьютере и под управлением любой операционной системы, будь то Windows, Mac, Linux или Solaris. Специфические особенности работы в конкретной операционной системе в данной главе не рассматриваются.



В ИНТЕРНЕТЕ

Инструкции по установке программного обеспечения на компьютерах, работающих под управлением различных операционных систем, приведены в приложениях А и Б. Эти инструкции также доступны на сайте книги, адрес которого указан во введении.

Быстрый старт

Если вы опытный пользователь (не будем уточнять, что именно под этим подразумевается) и слишком нетерпеливы, чтобы читать подробные инструкции в приложениях А и Б, попробуйте установить необходимое программное обеспечение, следуя приведенным ниже общим указаниям. Они годятся для большинства компьютеров и операционных систем и не содержат никаких подробных сведений относительно повышения эффективности процесса установки.

Чтобы подготовить компьютер к созданию программ на Java, выполните следующие действия.

1. Установите Java Development Kit.

Чтобы установить этот пакет, перейдите по следующему адресу:

<http://www.oracle.com/technetwork/java/javase/downloads>

Следуйте инструкциям, приведенным на сайте, для загрузки и установки новейшей версии Java SE JDK.



ЗАПОМНИ!

Выбирайте стандартную версию (Standard Edition — SE). Не останавливайтесь на корпоративной версии (Enterprise Edition — EE) либо других подобных версиях. Также выбирайте пакет JDK, а не JRE. Если вы видите код, такой как 9u3, он будет означать “третье обновление Java 9”. Вообще говоря, для запуска примеров кода этой книги подходит версия Java 9 и выше.

2. Установите интегрированную среду разработки.

Интегрированная среда разработки (Integrated Development Environment — IDE) — это программа, которая применяется для облегчения создания и тестирования нового программного обеспечения. Для выполнения примеров книги можно использовать практически любую IDE, которая поддерживает Java. Ниже приводится список наиболее популярных сред разработки Java.

- *Eclipse*

В соответствии с данными обзоров, опубликованных на сайте <http://www.baeldung.com/java-ides-2016>, в середине 2016 года 48,2% программистов на Java использовали среду разработки Eclipse.

Чтобы загрузить и использовать Eclipse, следуйте инструкциям, опубликованным на сайте <http://eclipse.org/downloads>. На странице загрузки Eclipse доступны разные пакеты, включая Eclipse для Java EE, Eclipse для JavaScript, Eclipse для Java и пр. Чтобы выполнять примеры книги, достаточно загрузить сравнительно небольшой пакет: Eclipse IDE for Java Developers.

Среда разработки Eclipse доступна на бесплатной основе для коммерческого и некоммерческого использования.

- *IntelliJ IDEA*

В соответствии с данными обзоров, опубликованных на сайте <http://www.baeldung.com/java-ides-2016>, в 2016 году среду разработки IntelliJ IDEA использовали 43,6% программистов.

Посетите сайт <http://www.jetbrains.com/idea>, чтобы загрузить версию Community Edition (доступна на бесплатной основе) либо Ultimate Edition (доступна на платной основе). Для выполнения примеров книги лучше использовать версию Community Edition. Эту версию можно использовать даже для создания коммерческого программного обеспечения!

- *NetBeans*

В соответствии с данными обзоров, опубликованных на сайте <http://www.baeldung.com/java-ides-2016>, в 2016 году среду NetBeans использовали менее 5,9% программистов. Но при этом NetBeans является официальной интегрированной средой разработки Java от Oracle. Если можно выбрать загружаемые пакеты, остановитесь на Java SE.

Чтобы получить собственную копию NetBeans, посетите следующий сайт:

<https://netbeans.org/downloads>

Среда разработки NetBeans бесплатна для коммерческого и некоммерческого использования.

3. Протестируйте установленное программное обеспечение.

Действия, выполняемые на этом шаге, зависят от интегрированной среды разработки, выбранной в п. 2. В любом случае обратите внимание на следующие обобщенные инструкции.

- Запустите интегрированную среду разработки (Eclipse, IntelliJ IDEA, NetBeans либо какую-то другую).
- В окне интегрированной среды разработки создайте новый проект Java.
- В окне проекта Java создайте новый класс Java под названием `Displayer`. Для создания нового класса в большинстве случаев достаточно выполнить команду `File⇒New⇒Class` (Файл⇒Новый файл⇒Класс).
- Отредактируйте новый файл `Displayer.java`, введя код из листинга 3.1 (он будет приведен в следующей главе).

В большинстве случаев можно добавить код, используя большую (преимущественно пустую) панель редактора исходного кода. Попробуйте ввести код точно в таком виде, в каком он представлен в листинге 3.1. Если вы видите заглавную букву, вводите именно ее. Также не изменяйте регистр строчных букв.



В ИНТЕРНЕТЕ

Вряд ли вам захочется вводить огромный фрагмент кода вручную. Это и не нужно. Посетите веб-страницу книги на сайте издательства “Диалектика”, адрес которого указан во введении. Там доступны примеры кода.

Запустите файл `Displayer.java` на выполнение и убедитесь в том, что на консоли появилось сообщение Вам понравится Java!

Вот и все, что вы должны сделать! Однако имейте в виду, что даже опытным пользователям может оказаться недостаточно этих простых инструкций для того, чтобы весь процесс прошел гладко от начала до конца. В таком случае выберите один из следующих возможных вариантов ваших дальнейших действий.

» **Воспользуйтесь приложением А.**

Ничего не предпринимайте. Игнорируйте приведенные выше краткие инструкции. Следуйте более подробным инструкциям, которые вы найдете в приложении А.

» **Попытайтесь воспользоваться краткими инструкциями по загрузке, приведенными в этом разделе.**

В конце концов, в результате вашей попытки ничего непоправимого не произойдет. Даже ошибочно установив не то программное обеспечение, которое вам было нужно, вы, скорее всего, сможете безболезненно оставить его на своем компьютере (вам необязательно его удалять). Если вы не уверены в правильности выбора варианта загрузки, вы всегда сможете обратиться к подробным инструкциям, приведенным на сайте книги, адрес которого указан во введении.

» **Задавайте автору книги вопросы по электронной почте: JavaForDummies@allmycode.com.**

» **Найдите меня в Твиттере: [@allmycode](https://twitter.com/allmycode).**

» **Посетите мою страницу [/allmycode](https://www.facebook.com/allmycode) на Facebook.**

Не стесняйтесь задавать мне вопросы.

Что должно быть установлено на компьютере

Однажды я встретил слесаря, и он показал мне инструменты, с помощью которых создаются лекала для изготовления других инструментов. Я был в восторге от этой встречи и уже тогда решил для себя, что когда-нибудь обязательно проведу аналогию между компьютерными программистами и изготовителями лекал.

Программист использует существующие программы в качестве инструментов для создания новых программ. Существующие и вновь создаваемые программы решают совершенно разные задачи. Например, программа на Java (создаваемая вами) может быть предназначена для учета заказов. Для ее разработки вы используете (вернее, будете использовать в ближайшем будущем) уже существующую программу, которая помогает вам вводить код, находить ошибки, запускать создаваемую программу на выполнение и т.п. С ее помощью нельзя решить поставленную задачу, но можно создать программу, отслеживающую заказы.

Что нужно, чтобы начать создавать программы на Java? Как новичку вам понадобятся всего лишь три инструмента.

» **Компилятор**

Это программа, которая преобразует код на языке Java в формат, пригодный для выполнения на компьютере (*байт-код*).

Люди не могут легко создавать или декодировать инструкции в формате байт-кода. Для генерирования и интерпретации байт-кода используется определенное программное обеспечение, устанавливаемое на компьютере.

» **Виртуальная машина Java**

Это программа, которая выполняет на компьютере созданный вами код (а также Java-код, написанный другими людьми).

» **Интегрированная среда разработки**

Интегрированная среда разработки (IDE) упрощает работу с кодом и предоставляет удобные средства для его написания, компиляции и выполнения.

По правде говоря, устанавливать интегрированную среду разработки вовсе *не обязательно*. Некоторые разработчики принципиально используют самые обычные текстовые редакторы, такие как Блокнот (Windows), TextEdit (Macintosh) или vim (Unix). Однако начинающим программистам все же лучше пользоваться специальной рабочей средой, в которой все будет получаться намного легче и быстрее.



ТЕХНИЧЕСКИЕ
ПОДРОБНОСТИ

В Интернете предлагается множество бесплатных версий каждого из перечисленных выше инструментов.

- » После загрузки Java SE JDK с сайта Oracle в вашем распоряжении окажутся компилятор и виртуальная машина Java (JVM). Этот сайт доступен по следующему адресу:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

» Чтобы загрузить интегрированную среду разработки, посетите сайт Eclipse (<http://www.eclipse.org/downloads>), IntelliJ IDEA (www.jetbrains.com/idea) или NetBeans (<https://netbeans.org/downloads>).

Многие интегрированные среды разработки имеют собственные виртуальные машины Java, а на сайте Oracle можно загрузить комбинированный пакет JDK+NetBeans. Если вы будете следовать приведенным инструкциям, то получите в свое распоряжение две копии виртуальной машины Java либо две интегрированные среды разработки. И это хорошо, поскольку запасной вариант никогда не помешает.



В ИНТЕРНЕТЕ

В данной главе приводятся лишь минимальные сведения о программном обеспечении, которое должно быть установлено на вашем компьютере, без детального описания соответствующих процедур установки. С подробными инструкциями вы сможете ознакомиться в приложениях А и Б, а также на сайте книги, адрес которого указан во введении.

Оставшаяся часть главы посвящена компиляторам, интегрированным средам разработки и виртуальной машине Java.

Что такое компилятор

Обратимся к коду, представленному в листинге 2.1.

Листинг 2.1. Поиск свободного номера

```
// Это фрагмент программы на Java,  
// а не готовая программа  
roomNum = 1;  
while (roomNum < 100) {  
    if(guests[roomNum] == 0) {  
        out.println("Комната " + roomNum + " свободна.");  
        exit(0);  
    } else {  
        roomNum++;  
    }  
}  
out.println("Свободные комнаты отсутствуют.");
```

Этот код выполняет поиск свободных комнат в небольшой гостинице (с номерами комнат от 1 до 99). Чтобы приведенный код можно было выполнить, в него необходимо добавить несколько дополнительных строк. Однако на данном этапе для нас это несущественно. Наша задача состоит в том, чтобы понять

назначение кода. Для этого запишем все операции, которые он выполняет, в словесной формулировке.

Присвоить номеру текущей комнаты значение 1.
Посмотреть комнаты с номерами меньше 100.
 Посмотреть, сколько постояльцев сейчас проживает в текущей комнате.
 Если количество постояльцев равно 0, то сообщить о том, что комната свободна, и завершить выполнение программы.
 В противном случае увеличить номер комнаты на 1 и перейти в начало цикла.
Если достигнут несуществующий номер комнаты 100, то сообщить об отсутствии свободных комнат.

Если вы пока что с трудом улавливаете связь между листингом 2.1 и его словесным эквивалентом, не огорчайтесь: вскоре вы будете читать код Java почти так же легко, как и текст на родном языке. Код, приведенный в листинге 2.1, называется *исходным кодом Java*.

Здесь необходимо подчеркнуть следующее. Обычно компьютеры не способны выполнять инструкции в том виде, в каком они записаны в листинге 2.1, т.е. в виде исходного кода. Инструкции, непосредственно выполняемые компьютером, выглядят более таинственно (листинг 2.2).

Листинг 2.2. Байт-код Java, полученный трансляцией исходного кода из листинга 2.1

```
aload_0
iconst_1
putfield Hotel/roomNum I
goto 32
aload_0
getfield Hotel/guests [I
aload_0
getfield Hotel/roomNum I
iaload
ifne 26
getstatic java/lang/System/out Ljava/io/PrintStream;
new java/lang/StringBuilder
dup
ldc "Комната "
invokespecial java/lang/StringBuilder/<init>(Ljava/lang/String;)V
aload_0
getfield Hotel/roomNum I
invokevirtual
    java/lang/StringBuilder/append(I)Ljava/lang/StringBuilder;
```

```

ldc " свободна."
invokevirtual
  java/lang/StringBuilder/append(Ljava/lang/String;)Ljava/lang/
StringBuilder;
invokevirtual java/lang/StringBuilder/toString()Ljava/lang/String;
invokevirtual java/io/PrintStream/println(Ljava/lang/String;)V
iconst_0
invokestatic java/lang/System/exit(I)V
goto 32
aload_0
dup
getfield Hotel/roomNum I
iconst_1
iadd
putfield Hotel/roomNum I
aload_0
getfield Hotel/roomNum I
bipush 100
if_icmplt 5
getstatic java/lang/System/out Ljava/io/PrintStream;
ldc "Свободные комнаты отсутствуют."
invokevirtual java/io/PrintStream/println(Ljava/lang/String;)V
return

```

Инструкции в этом листинге представляют собой так называемый *байт-код Java*. Создавая программу на языке Java, вы пишете инструкции исходного кода (как в листинге 2.1), после чего обрабатываете этот код с помощью специальной программы, называемой *компилятором*, которая транслирует (преобразует) исходный код в байт-код, выполняемый компьютером. Иными словами, компилятор получает код, подготовленный на понятном для вас языке (см. листинг 2.1), и переводит его на язык, понятный компьютеру (см. листинг 2.2).



ТЕХНИЧЕСКИЕ
ПОДРОБНОСТИ

Предположим, вы сохранили исходный код, приведенный в листинге 2.1, в файле `Hotel.java`. Тогда компилятор, вероятнее всего, поместит байт-код в файл `Hotel.class`. Если вы любознательны и захотите просмотреть содержимое файла `Hotel.class` с помощью обычного текстового редактора, то вместо текста, приведенного в листинге 2.2, увидите странные символы: квадратики, точки, стрелочки и т.п. Дело в том, что компилятор не только преобразует исходный код в байт-код, но и специальным образом его кодирует. Чтобы увидеть байт-код в том виде, в каком он показан в листинге 2.2, откройте файл `Hotel.class` с помощью программы `Java Bytecode Editor`, доступной по следующему адресу:

<http://www.cs.ioc.ee/~ando/jbe>



ЗАПОМНИ!

Ни один разработчик программ на Java не занимается написанием байт-кода, который создается компилятором. Единственным побудительным мотивом к просмотру содержимого файлов `.class` может быть лишь ваша любознательность.

Что такое виртуальная машина Java

Как отмечалось в предыдущем разделе, компьютер следует инструкциям наподобие тех, которые представлены в листинге 2.2. В последней фразе слово “наподобие” употреблено не случайно. Дело в том, что, несмотря на сходство инструкций в листинге 2.2 с инструкциями, выполняемыми компьютерами, в каждом компьютере вместо байт-кода Java используется собственный набор выполняемых инструкций, зависящий от типа установленного процессора, причем между наборами инструкций одного и того же процессора, предназначенными для разных операционных систем, имеются незначительные различия.

Рассмотрим следующую гипотетическую ситуацию. Предположим, что на дворе 1992 год (в то время Java еще не было). Допустим, вы работаете на компьютере с устаревшим процессором Pentium, используя операционную систему Linux. Ваш друг также использует Linux, но на его компьютере установлен процессор PowerPC.

В листинге 2.3 представлена программа, выводящая на экран фразу `Hello, world!`¹. Приведенные инструкции предназначены для процессора Pentium, работающего под управлением операционной системы Linux.

Листинг 2.3. Простая программа для процессора Pentium

```
.data
msg:
    .ascii "Hello, world!\n"
    len = . - msg

.text
.global _start
_start:
    movl $len,%edx
    movl $msg,%ecx
    movl $1,%ebx
    movl $4,%eax
    int $0x80
```

¹ См. страницу Linux Assembly HOWTO (<http://tldp.org/HOWTO/Assembly-HOWTO/hello.html>).

```
movl $0,%ebx
movl $1,%eax
int $0x80
```

В листинге 2.4 представлен код этой же программы, но ориентированный на процессор PowerPC, также работающий под управлением операционной системы Linux.

Листинг 2.4. Простая программа для процессора PowerPC

```
.data

msg:
    .string "Hello, world!\n"
    len = . - msg

.text

    .global _start
_start:

    li 0,4
    li 3,1

    lis 4,msg@ha
    addi 4,4,msg@l
    li 5,len
    sc

    li 0,1
    li 3,1
    sc
```

Инструкции листинга 2.3 беспрепятственно выполняются на процессоре Pentium, но для процессора PowerPC не имеют никакого смысла. И наоборот, код листинга 2.4 работает на PowerPC, но не работает на Pentium.

Предположим далее, что в компьютере вашей сестры установлен процессор Pentium (как и в вашем), но в качестве операционной системы используется не Linux (как у вас), а Windows. В результате представленный в листинге 2.3 код, несмотря на то что он предназначен для процессора Pentium, на компьютере сестры не заработает.

Байт-код Java позволяет избежать этого хаоса. Он похож на код, представленный в листингах 2.3 и 2.4, но не зависит от особенностей процессоров и операционных систем. Набор инструкций байт-кода Java выполняется на любом компьютере. Если вы напишете программу на Java и скомпилируете ее в байт-код,

он выполнится даже на компьютере вашей бабушки, если на нем установлена виртуальная машина Java, а также на смартфоне и планшете (в том случае, если такие возможности предусмотрены в них).



ПЕРЕКРЕСТНАЯ
ССЫЛКА

Вы уже получили представление о том, что такое байт-код, когда мы рассматривали листинг 2.2. Однако вам никогда не придется писать и читать программы на байт-коде. Создание байт-кода — это работа компилятора, а его расшифровка — работа виртуальной машины Java.

Предположим, у вас есть файл с байт-кодом, созданный на компьютере Windows. Скопировав этот файл на компьютер, работающий под управлением любой другой операционной системы (Mac, Linux, Solaris), вы сможете выполнить программу, не внося в нее никаких изменений. Программы, обладающие этим замечательным свойством — способностью выполняться на многих компьютерах без каких-либо переделок, — называются *переносимыми*, или *портируемыми*.

Что делает байт-код Java таким универсальным? Виртуальная машина Java (Java Virtual Machine — JVM). Это один из трех инструментов, которые должны быть обязательно установлены на вашем компьютере.

Вообразите, будто вы представляете сообщество пользователей Windows на заседании Совета безопасности ООН (рис. 2.1). Справа от вас находится представитель пользователей Mac, а слева — представитель пользователей Linux. (Разумеется, ладить с ними вам нелегко. Вы всегда вежливы в общении с ними, но при этом никогда не бываете до конца искренни. Впрочем, чему тут удивляться? Ведь это же политика!) На трибуне выступает представитель сообщества пользователей Java. Он говорит на языке байт-кода, который не знаете ни вы, ни другие участники заседания (Mac и Linux).

Как всегда в подобных случаях, выручают переводчики. Во время выступления представителя Java ваш переводчик выполняет перевод с языка байт-кода на язык Windows. Точно так же другие переводчики помогают представителям Mac и Linux.

Подобно переводчикам, виртуальная машина Java играет роль промежуточного агента — посредника между универсальным байт-кодом Java и конкретной компьютерной системой. Поэтому программа, написанная на Java, может выполняться на любом компьютере, на котором установлена виртуальная машина Java.

Таким образом, для каждой платформы (наподобие операционной системы и модели процессора) требуется своя виртуальная машина Java. Она служит посредником между универсальным байт-кодом и конкретной платформой. Виртуальная машина читает байт-код, преобразует его в машинный код данной платформы и передает его операционной системе для немедленного выполнения.

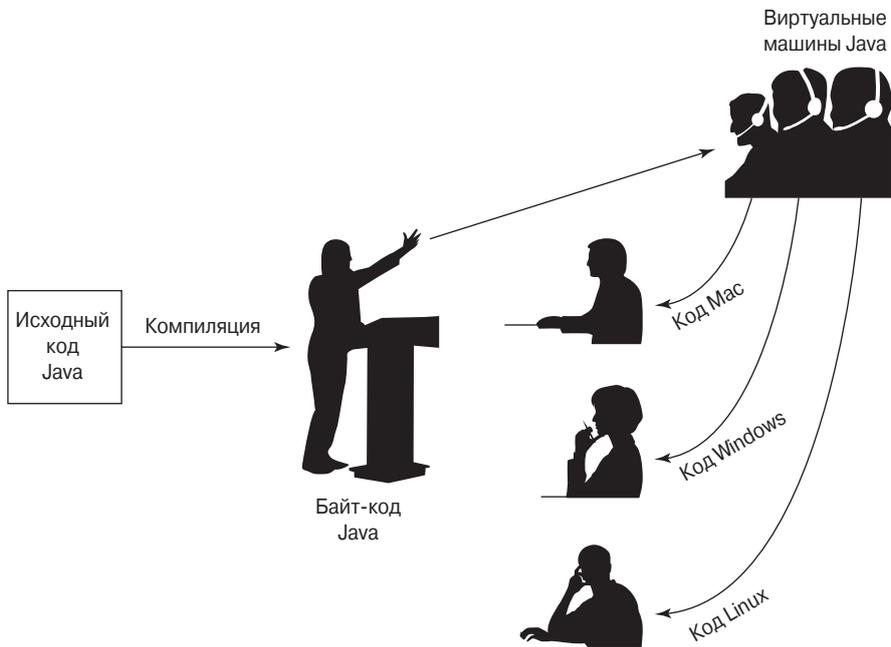


Рис. 2.1. Воображаемое заседание Совета безопасности ООН

ЧТО ТАКОЕ JAVA 2 STANDARD EDITION 1.2

Если вы попытаетесь получить в Интернете информацию о том, какие средства необходимы для работы с Java, то среди предложенных вам результатов поиска обязательно будут фигурировать такие инструменты, как Java Development Kit, Software Development Kit и Java Runtime Environment.

- Названия *Java Development Kit* (JDK — инструментальный комплект поддержки разработок в среде Java) и *Software Development Kit* (SDK — набор инструментальных средств разработки программного обеспечения) соответствуют двум различным версиям одного и того же инструментального набора, ключевым компонентом которого является компилятор Java.
- Название *Java Runtime Environment* (JRE — среда выполнения Java) относится к инструментальному набору, ключевым компонентом которого выступает виртуальная машина Java (JVM).

Для выполнения программ Java на компьютере должен быть установлен пакет JRE, а для разработки таких программ — пакет JDK. В процессе установки пакета JDK вместе с ним автоматически устанавливается пакет JRE, однако возможна также отдельная установка последнего. В действительности на компьютере могут мирно уживаться несколько комбинаций JDK и JRE.

Например, на моем компьютере, работающем под управлением Windows, в каталоге `c:\program files\Java` установлены пакеты JDK 1.6, JDK 1.8 и JRE 8, а в каталоге `c:\program files (x86)\Java` — пакет JDK 9, причем с конфликтами версий я практически не сталкивался. Если у вас возникли подозрения, что установленные на вашем компьютере версии конфликтуют между собой, удалите все версии JDK и JRE, кроме последней (например, JDK 9 или JRE 9).

Некоторую путаницу может вносить нумерация версий Java. Вместо стандартной нумерации “Java 1”, “Java 2” и “Java 3” применяется более сложная схема нумерации. В следующей таблице проиллюстрирован процесс появления новых версий Java. Каждая версия Java имеет несколько названий. *Версия продукта* — это официальное название, которое обычно используется всеми, а *версия разработки* — это номер, идентифицирующий версии продукта и применяемый программистами для отслеживания этих версий. (В обычной беседе программисты используют разные типы названий для разных версий Java.) *Кодовое имя* — это неформальное имя, которое идентифицирует версию программы на этапе разработки.

В приведенной ниже таблице звездочка означает смену названия продукта. В 1996 году версии Java носили названия *Java Development Kit 1.0* и *Java Development Kit 1.1*. В 1998 году кто-то решил переименовать продукт в *Java 2 Standard Edition 1.2*, внося в ряды пользователей сумятицу, которая наблюдается и по сей день. Тогда же последовало обращение с призывом ко всем использовать вместо термина *Java Development Kit (JDK)* термин *Software Development Kit (SDK)*.

В 2004 году исчез номер версии платформы 1, а в 2006 году из названия платформы исчезла цифра 2 и из номера версии — обозначение .0.

Самые серьезные изменения для программистов на Java произошли в 2004 году. В этом году увидела свет версия J2SE 5.0, в которую разработчики Java внесли такие изменения, как обобщенные типы, аннотации и расширенный цикл `for`. (Примеры использования аннотаций Java будут приведены в главах 8, 9 и 16. Применение расширенного цикла `for` и обобщенных типов данных будет продемонстрировано в главах 11 и 12.)

Большинство примеров, приведенных в книге, может выполняться только в версиях не ниже Java 5.0. В частности, примеры не будут выполняться в версиях Java 1.4 и Java 1.4.2. Есть и такие примеры, которые не будут выполняться в версии ниже Java 9. Однако не стоит особенно задумываться над номерами версий. Установленная на компьютере версия Java 6 или 7 — это в любом случае лучше, чем полное отсутствие Java. Вы сможете многому научиться, даже если у вас установлена не самая последняя версия Java.

Год выпуска	Платформа	Кодовое имя	Новинки
1995	Бета-версия		
1996	JDK* 1.0		
1997	JDK 1.1		Внутренние классы, Java Beans, Reflection
1998	J2SE* 1.2	Playground	Классы Swing, предназначенные для создания графических интерфейсов пользователя (GUI)
2000	J2SE 1.3	Kestrel	Java Naming and Directory Interface (JNDI)
2002	J2SE 1.4	Merlin	Новые средства ввода-вывода, регулярные выражения, XML-анализатор
2004	J2SE 5.0	Tiger	Обобщенные типы, аннотации, перечисления, переменное число аргументов, расширенный цикл <code>for</code> , статический оператор <code>import</code> , новые классы для параллельных вычислений
2006	Java SE* 6	Mustang	Поддержка языка сценариев, улучшенная производительность
2011	Java SE 7	Dolphin	Строки в операторе <code>switch</code> , перехват нескольких исключений, использование оператора <code>try</code> для работы с ресурсами, интеграция с JavaFX
2014	Java SE 8		Лямбда-выражения
2017	Java SE 9		Модульность, поддерживаемая в Project Jigsaw, интерактивное кодирование в JShell

Процесс разработки

При создании программы на Java вы многократно проходите одни и те же этапы разработки, представленные на рис. 2.2.

В первую очередь, вы пишете код будущей программы. Написав первый черновой вариант, вы многократно компилируете его, выполняете и вносите необходимые изменения. После того как вы приобретете определенный опыт, прохождение этих этапов не будет вызывать у вас никаких затруднений. Во многих случаях для компиляции или выполнения кода потребуется всего лишь один щелчок мышью.

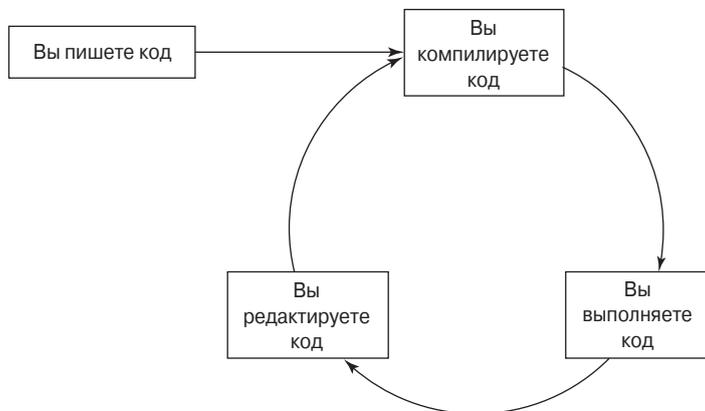


Рис. 2.2. Этапы разработки программного обеспечения

Однако собственно написание первого варианта кода и его последующее совершенствование — это сложный творческий процесс, требующий времени и концентрации внимания.



ЗАПОМНИ!

Никогда не падайте духом, если написанный вами код сразу не работает. Даже профессиональные программисты проходят цикл, показанный на рис. 2.2, сотни раз, пока код не будет работать так, как нужно. Постоянно экспериментируйте, пробуйте разные варианты, проверяйте работу кода с разными исходными данными. Научиться работать с кодом — ваша главная задача.

Для получения более подробных инструкций относительно компиляции и выполнения программ на Java обратитесь к приложению Б или посетите сайт книги, адрес которого указан во введении.

Формулировки на рис. 2.2 гласят: “Вы компилируете код”, “Вы выполняете код”... Однако на самом деле это делаете не вы, а компьютер по вашей команде. Ваша работа заключается в редактировании кода. Компьютер не сможет сделать это вместо вас, даже если вы ему прикажете. Уточненный цикл разработки проиллюстрирован на рис. 2.3.



Применительно к большинству случаев рис. 2.3 содержит избыточную информацию. Щелкая на кнопке Run (Выполнить), я не должен помнить о том, что компьютер компилирует исходный код, создает байт-код, запускает виртуальную машину Java и т.п. Все, что мне нужно, — выполнить код, который я вижу в окне редактирования. Поэтому подробности, показанные на рис. 2.3, не так уж важны. О них придется вспомнить лишь в том случае, если общие формулировки, использованные на рис. 2.2, не проясняют для вас ситуацию. Если рис. 2.2 вам понятен, можете проигнорировать рис. 2.3.

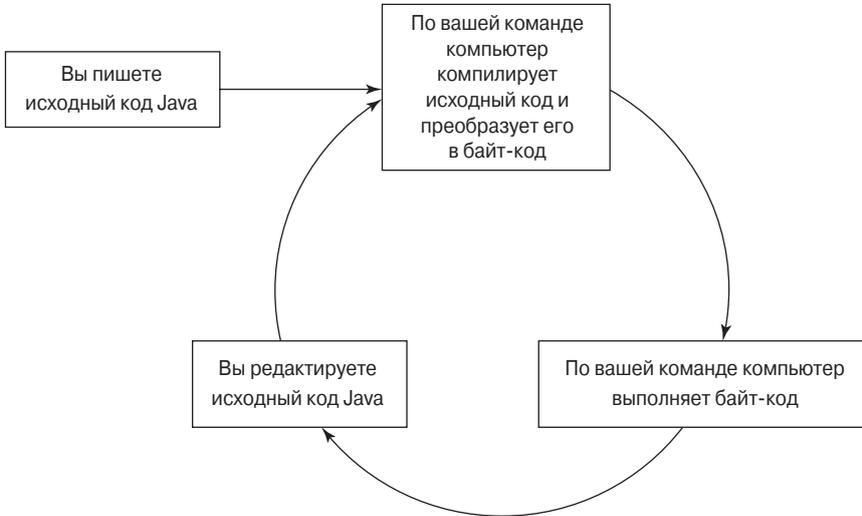


Рис. 2.3. Уточним, кто и что делает в цикле разработки

Интегрированная среда разработки

До появления интегрированных сред разработки для создания программы нужно было открыть несколько отдельных окон: одно — для редактирования программы, другое — для ее выполнения и третье — для просмотра списка имеющихся программ. Вы и сейчас можете отказаться от Eclipse и создать программу на Java с помощью проводника, блокнота и консоли (рис. 2.4).

Интегрированная среда разработки предоставляет все эти функции в одном хорошо организованном приложении (рис. 2.5).

Программа Eclipse — не единственная интегрированная среда разработки приложений Java. Другие наиболее популярные программы — IntelliJ IDEA и NetBeans. Большинство из них поддерживает операции перетаскивания и вставки компонентов, которыми вы сможете воспользоваться для разработки графического интерфейса своей программы (рис. 2.6).

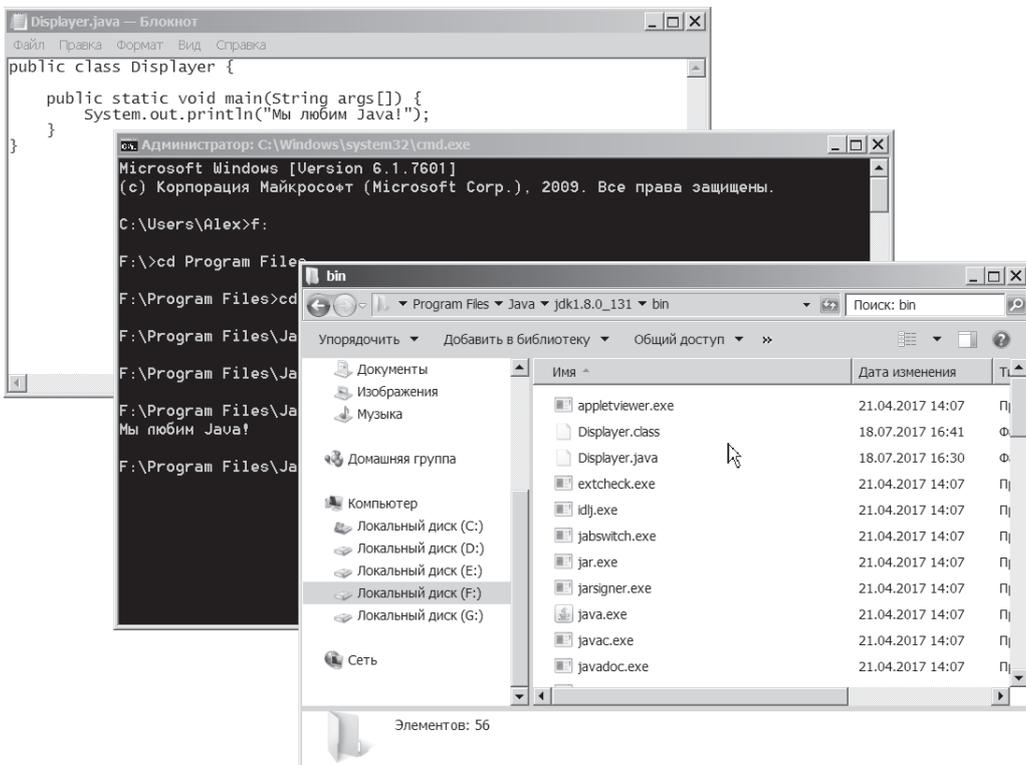


Рис. 2.4. Разработка кода Java “подручными средствами”

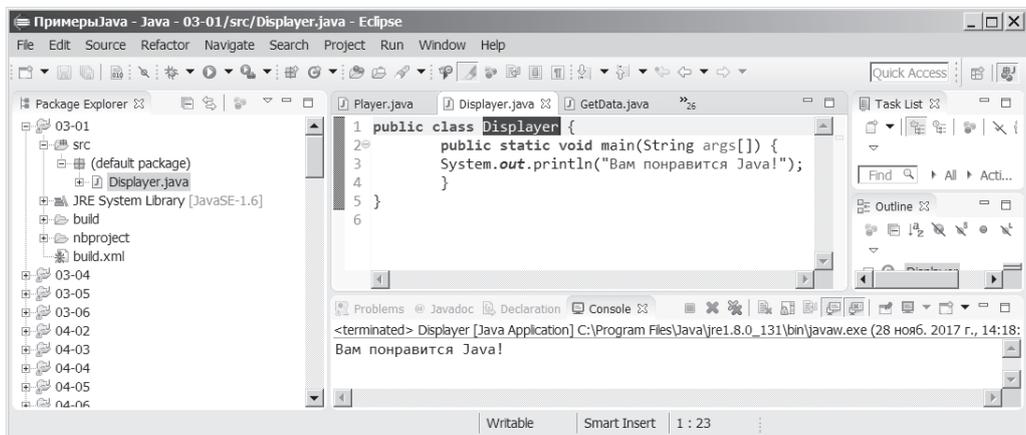


Рис. 2.5. Разработка кода Java в окне программы Eclipse

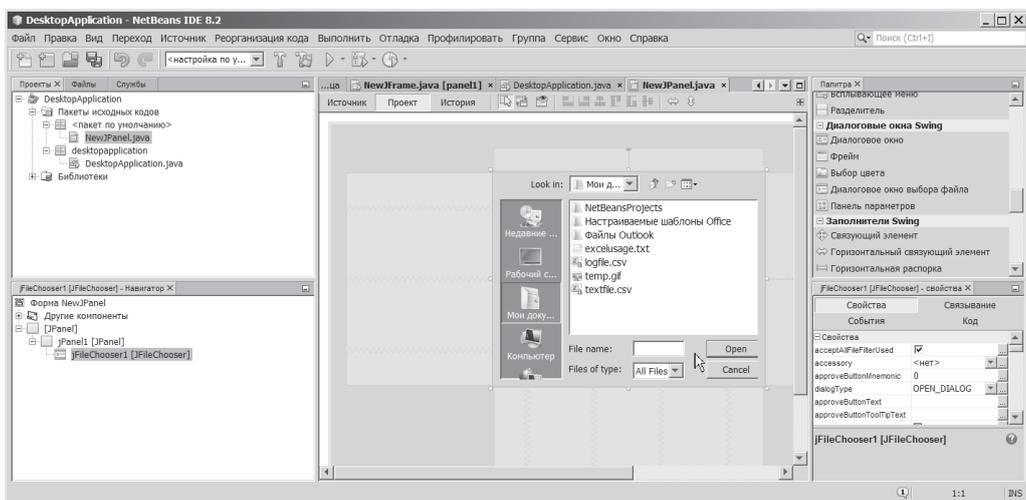


Рис. 2.6. Пример использования Swing GUI Builder в среде разработки NetBeans

Для запуска программы можно либо щелкнуть на соответствующей кнопке панели инструментов, либо выбрать пункт меню Run. Для компиляции программы от вас может вообще ничего не потребоваться. (Возможно, вам даже не придется использовать никакую команду. В некоторых средах разработки компиляция программы осуществляется автоматически по мере ввода исходного кода.)



В ИНТЕРНЕТЕ

Более подробное описание установки и использования интегрированных сред разработки приведено в приложении Б, а также на сайте книги, адрес которого указан во введении.