

# Java 7



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

**Наиболее  
полное  
руководство**

**В ПОДЛИННИКЕ®**

**Ильдар Хабибуллин**

# **Java 7**

Санкт-Петербург  
«БХВ-Петербург»  
2012

УДК 681.3.06  
ББК 32.973.26-018.2  
X12

**Хабибуллин И. Ш.**

X12 Java 7. — СПб.: БХВ-Петербург, 2012. — 768 с.: ил. — (В подлиннике)  
ISBN 978-5-9775-0735-6

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

*Для программистов*

УДК 681.3.06  
ББК 32.973.26-018.2

**Группа подготовки издания:**

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

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.08.11.  
Формат 70×100<sup>1/16</sup>. Печать офсетная. Усл. печ. л. 61,92.  
Тираж 1800 экз. Заказ №  
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

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

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

# Оглавление

<b>Введение</b> .....	<b>19</b>
Что такое Java? .....	20
Структура книги.....	21
Выполнение Java-программы .....	24
Что такое JDK? .....	25
Что такое JRE? .....	27
Как установить JDK? .....	27
Как использовать JDK? .....	28
Интегрированные среды Java.....	30
Особая позиция Microsoft.....	30
Java в Интернете .....	31
Литература по Java.....	32
Благодарности .....	33
<b>ЧАСТЬ I. БАЗОВЫЕ КОНСТРУКЦИИ ЯЗЫКА JAVA</b> .....	<b>35</b>
<b>Глава 1. Встроенные типы данных, операции над ними</b> .....	<b>37</b>
Первая программа на Java .....	37
Комментарии .....	40
Аннотации .....	42
Константы.....	42
Целые .....	42
Действительные.....	43
Символы.....	43
Строки .....	44
Имена .....	45
Примитивные типы данных и операции .....	45
Логический тип .....	47
Логические операции.....	47
Упражнения .....	48
Целые типы.....	48
Операции над целыми типами .....	49
Арифметические операции .....	49
Приведение типов .....	50

Операции сравнения .....	52
Побитовые операции .....	52
Сдвиги .....	53
Упражнения .....	54
Вещественные типы .....	54
Операции присваивания .....	55
Упражнения .....	56
Условная операция .....	56
Упражнения .....	56
Выражения .....	56
Приоритет операций .....	57
Операторы .....	58
Блок .....	59
Операторы присваивания .....	59
Условный оператор .....	59
Упражнения .....	61
Операторы цикла .....	62
Оператор <i>continue</i> и метки .....	64
Оператор <i>break</i> .....	65
Упражнения .....	65
Оператор варианта .....	65
Массивы .....	67
Многомерные массивы .....	69
Заключение .....	71
Вопросы для самопроверки .....	71
<b>Глава 2. Объектно-ориентированное программирование в Java .....</b>	<b>73</b>
Парадигмы программирования .....	73
Принципы объектно-ориентированного программирования .....	76
Абстракция .....	76
Иерархия .....	79
Ответственность .....	80
Модульность .....	81
Принцип KISS .....	83
Упражнения .....	84
Как описать класс и подкласс? .....	84
Передача аргументов в метод .....	86
Перегрузка методов .....	87
Переопределение методов .....	88
Реализация полиморфизма в Java .....	89
Упражнения .....	90
Абстрактные методы и классы .....	90
Окончательные члены и классы .....	91
Класс <i>Object</i> .....	92
Конструкторы класса .....	93
Операция <i>new</i> .....	94
Упражнение .....	94
Статические члены класса .....	94
Класс <i>Complex</i> .....	96

Метод <i>main()</i> .....	99
Методы с переменным числом аргументов.....	100
Где видны переменные.....	101
Вложенные классы.....	103
Отношения "быть частью" и "являться".....	107
Заключение.....	108
Вопросы для самопроверки.....	108
<b>Глава 3. Пакеты, интерфейсы и перечисления.....</b>	<b>109</b>
Пакет и подпакет.....	110
Права доступа к членам класса.....	111
Размещение пакетов по файлам.....	113
Импорт классов и пакетов.....	115
Java-файлы.....	116
Интерфейсы.....	117
Перечисления.....	121
Объявление аннотаций.....	124
Design patterns.....	126
Схема проектирования MVC.....	126
Шаблон Singleton.....	127
Заключение.....	129
Вопросы для самопроверки.....	129
<b>ЧАСТЬ II. ИСПОЛЬЗОВАНИЕ КЛАССОВ ИЗ JAVA API.....</b>	<b>131</b>
<b>Глава 4. Классы-оболочки и generics.....</b>	<b>133</b>
Числовые классы.....	134
Автоматическая упаковка и распаковка типов.....	136
Настраиваемые типы (generics).....	137
Шаблон типа (wildcard type).....	140
Настраиваемые методы.....	141
Класс <i>Boolean</i> .....	142
Класс <i>Character</i> .....	143
Класс <i>BigInteger</i> .....	146
Класс <i>BigDecimal</i> .....	148
Класс <i>Class</i> .....	152
Вопросы для самопроверки.....	155
<b>Глава 5. Работа со строками.....</b>	<b>156</b>
Класс <i>String</i> .....	157
Как создать строку.....	157
Упражнение.....	162
Сцепление строк.....	162
Как узнать длину строки.....	162
Как выбрать символы из строки.....	163
Как выбрать подстроку.....	163
Как разбить строку на подстроки.....	164
Как сравнить строки.....	164
Как найти символ в строке.....	166

Как найти подстроку .....	167
Как изменить регистр букв.....	167
Как заменить отдельный символ .....	168
Как заменить подстроку .....	168
Как убрать пробелы в начале и конце строки.....	168
Как преобразовать в строку данные другого типа .....	168
Упражнения .....	169
Класс <i>StringBuilder</i> .....	169
Конструкторы .....	169
Как добавить подстроку .....	170
Как вставить подстроку .....	170
Как удалить подстроку .....	171
Как удалить символ.....	171
Как заменить подстроку .....	171
Как перевернуть строку.....	171
Синтаксический разбор строки.....	172
Класс <i>StringTokenizer</i> .....	172
Заключение .....	173
Вопросы для самопроверки .....	173
<b>Глава 6. Классы-коллекции.....</b>	<b>174</b>
Класс <i>Vector</i> .....	174
Как создать вектор .....	175
Как добавить элемент в вектор .....	175
Как заменить элемент .....	176
Как узнать размер вектора.....	176
Как обратиться к элементу вектора .....	176
Как узнать, есть ли элемент в векторе.....	176
Как узнать индекс элемента .....	177
Как удалить элементы.....	177
Класс <i>Stack</i> .....	178
Класс <i>Hashtable</i> .....	179
Как создать таблицу <i>Hashtable</i> .....	180
Как заполнить таблицу <i>Hashtable</i> .....	180
Как получить значение по ключу .....	180
Как узнать наличие ключа или значения .....	181
Как получить все элементы таблицы <i>Hashtable</i> .....	181
Как удалить элементы.....	181
Класс <i>Properties</i> .....	182
Интерфейс <i>Collection</i> .....	185
Интерфейс <i>List</i> .....	185
Интерфейс <i>Set</i> .....	186
Интерфейс <i>SortedSet</i> .....	186
Интерфейс <i>NavigableSet</i> .....	187
Интерфейс <i>Queue</i> .....	188
Интерфейс <i>BlockingQueue</i> .....	188
Интерфейс <i>Deque</i> .....	188
Интерфейс <i>BlockingDeque</i> .....	189

Интерфейс <i>Map</i> .....	190
Вложенный интерфейс <i>Map.Entry</i> .....	191
Интерфейс <i>SortedMap</i> .....	191
Интерфейс <i>NavigableMap</i> .....	191
Абстрактные классы-коллекции .....	192
Интерфейс <i>Iterator</i> .....	193
Интерфейс <i>ListIterator</i> .....	194
Классы, создающие списки .....	195
Двунаправленный список .....	196
Дек .....	196
Упражнение .....	197
Классы, создающие отображения .....	197
Связанные отображения .....	197
Упорядоченные отображения .....	197
Сравнение элементов коллекций .....	198
Упражнение .....	199
Классы, создающие множества .....	199
Связанные множества .....	199
Упорядоченные множества .....	200
Действия с коллекциями .....	200
Методы класса <i>Collections</i> .....	200
Упражнение .....	201
Заключение .....	202
Вопросы для самопроверки .....	202
<b>Глава 7. Классы-утилиты .....</b>	<b>203</b>
Работа с массивами .....	203
Сортировка массива .....	203
Бинарный поиск в массиве .....	203
Заполнение массива .....	204
Копирование массива .....	204
Сравнение массивов .....	205
Представление массива строкой .....	205
Получение хеш-кода массива .....	206
Локальные установки .....	206
Работа с датами и временем .....	208
Часовой пояс и летнее время .....	208
Класс <i>Calendar</i> .....	209
Подкласс <i>GregorianCalendar</i> .....	209
Представление даты и времени .....	210
Получение случайных чисел .....	211
Копирование массивов .....	211
Взаимодействие с системой .....	212

## **ЧАСТЬ III. СОЗДАНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ И АППЛЕТОВ .....**

<b>Глава 8. Принципы построения графического интерфейса .....</b>	<b>215</b>
Компонент и контейнер .....	217
Иерархия классов AWT .....	220

Окно библиотеки Swing.....	221
Использование системных приложений .....	222
System Tray .....	223
Splash Screen .....	224
Заключение.....	224
Вопросы для самопроверки .....	224
<b>Глава 9. Графические примитивы .....</b>	<b>226</b>
Методы класса <i>Graphics</i> .....	226
Как задать цвет .....	226
Упражнение .....	228
Как нарисовать чертеж .....	228
Класс <i>Polygon</i> .....	229
Упражнение .....	230
Прочие методы класса <i>Graphics</i> .....	230
Как вывести текст .....	231
Как установить шрифт.....	231
Как задать шрифт .....	231
Класс <i>FontMetrics</i> .....	235
Упражнение .....	238
Возможности Java 2D .....	238
Преобразование координат .....	240
Класс <i>AffineTransform</i> .....	240
Упражнение .....	243
Рисование фигур средствами Java 2D .....	243
Класс <i>BasicStroke</i> .....	243
Класс <i>GeneralPath</i> .....	246
Классы <i>GradientPaint</i> и <i>TexturePaint</i> .....	247
Классы <i>LinearGradientPaint</i> и <i>RadialGradientPaint</i> .....	249
Вывод текста средствами Java 2D .....	250
Методы улучшения визуализации .....	254
Упражнение .....	256
Заклучение.....	256
Вопросы для самопроверки .....	256
<b>Глава 10. Основные компоненты AWT.....</b>	<b>257</b>
Класс <i>Component</i> .....	257
Класс <i>Cursor</i> .....	259
Как создать свой курсор .....	259
Упражнение .....	260
События .....	260
Класс <i>Container</i> .....	261
События .....	262
Текстовая метка <i>Label</i> .....	262
События .....	262
Кнопка <i>Button</i> .....	262
События .....	263
Кнопка выбора <i>Checkbox</i> .....	263
События .....	263

Класс <i>CheckboxGroup</i> .....	263
Как создать группу радиокнопок .....	264
Раскрывающийся список <i>Choice</i> .....	265
События .....	266
Список <i>List</i> .....	266
События .....	267
Компоненты для ввода текста .....	268
Класс <i>TextComponent</i> .....	268
События .....	269
Строка ввода <i>TextField</i> .....	269
События .....	269
Поле ввода <i>TextArea</i> .....	269
События .....	270
Линейка прокрутки <i>Scrollbar</i> .....	272
События .....	272
Контейнер <i>Panel</i> .....	274
Контейнер <i>ScrollPane</i> .....	275
Контейнер <i>Window</i> .....	276
События .....	276
Контейнер <i>Frame</i> .....	277
События .....	277
Контейнер <i>Dialog</i> .....	279
События .....	280
Контейнер <i>FileDialog</i> .....	282
События .....	282
Создание собственных компонентов .....	283
Компонент <i>Canvas</i> .....	283
Создание "легкого" компонента .....	285
Упражнение .....	287
Создание меню .....	287
Всплывающее меню .....	292
Вопросы для самопроверки .....	295
<b>Глава 11. Оформление GUI компонентами Swing .....</b>	<b>296</b>
Состав библиотеки Swing .....	297
Основные компоненты Swing .....	299
Компонент <i>JComponent</i> .....	299
Схема MVC в компонентах Swing .....	300
Надпись <i>JLabel</i> .....	302
Кнопки .....	304
Кнопка <i>JButton</i> .....	306
Кнопка выбора <i>JToggleButton</i> .....	306
Кнопка выбора <i>JCheckBox</i> .....	308
Радиокнопка <i>JRadioButton</i> .....	308
Упражнение .....	309
Раскрывающийся список <i>JComboBox</i> .....	310
Список выбора <i>JList</i> .....	311
Визуализация элементов списков .....	312

Упражнение .....	314
Счетчик <i>JSpinner</i> .....	314
Полосы прокрутки <i>JScrollBar</i> .....	316
Ползунок <i>JSlider</i> .....	316
Упражнение .....	318
Индикатор <i>JProgressBar</i> .....	318
Дерево объектов <i>JTree</i> .....	318
Построение меню средствами Swing .....	322
Строка меню <i>JMenuBar</i> .....	322
Меню <i>JMenu</i> .....	323
Пункт меню <i>JMenuItem</i> .....	323
Всплывающее меню <i>JPopupMenu</i> .....	325
Панель выбора цвета <i>JColorChooser</i> .....	326
Упражнение .....	328
Окно выбора файла <i>JFileChooser</i> .....	328
Фильтр файлов <i>FileFilter</i> .....	328
Как получить выбранный файл .....	330
Дополнительный компонент .....	330
Замена изображений .....	331
Русификация Swing .....	333
Вопросы для самопроверки .....	333
<b>Глава 12. Текстовые компоненты .....</b>	<b>334</b>
Компонент <i>JTextComponent</i> .....	334
Модель данных — документ .....	334
Строка символов <i>Segment</i> .....	335
Запись текста в документ .....	336
Атрибуты текста .....	336
Удаление текста из документа .....	337
Фильтрация документа .....	337
Внесение структуры в документ .....	337
События в документе .....	338
Реализации документа .....	338
Установка модели данных .....	339
Вид .....	339
Контроллер — редактор текста .....	341
Курсор .....	341
Ограничение перемещения курсора .....	342
Реализации редактора .....	343
Раскладка клавиатуры .....	343
Печать текста документа .....	344
Поле ввода <i>JTextField</i> .....	344
Поле ввода пароля <i>JPasswordField</i> .....	347
Редактор объектов <i>JFormattedTextField</i> .....	347
Область ввода <i>JTextArea</i> .....	348
Текстовый редактор <i>JEditorPane</i> .....	349
Редактор <i>JTextPane</i> .....	350
Вопросы для самопроверки .....	350

<b>Глава 13. Таблицы</b> .....	<b>351</b>
Класс <i>JTable</i> .....	351
Модель данных таблицы .....	353
Модель ячеек таблицы .....	353
Свойства столбца таблицы <i>TableColumn</i> .....	358
Модель столбцов таблицы .....	358
Заголовки столбцов таблицы <i>JTableHeader</i> .....	358
Модель выделения ячеек .....	360
Визуализация ячеек таблицы .....	361
Редактор ячеек таблицы .....	364
Сортировка строк таблицы .....	367
Фильтрация строк таблицы .....	369
Печать таблицы .....	370
Вопросы для самопроверки .....	371
<b>Глава 14. Размещение компонентов и контейнеры Swing</b> .....	<b>372</b>
Менеджер <i>FlowLayout</i> .....	372
Менеджер <i>BorderLayout</i> .....	374
Менеджер <i>GridLayout</i> .....	376
Менеджер <i>CardLayout</i> .....	377
Менеджер <i>GridBagLayout</i> .....	379
Контейнеры Swing .....	381
Панель <i>JPanel</i> .....	381
Панель прокрутки <i>JScrollPane</i> .....	382
Двойная панель <i>JSplitPane</i> .....	384
Панель с вкладками <i>JTabbedPane</i> .....	385
Линейная панель <i>Box</i> .....	387
Менеджер размещения <i>BoxLayout</i> .....	387
Компоненты-заполнители .....	388
Менеджер размещения <i>SpringLayout</i> .....	389
Размеры <i>Spring</i> .....	390
Промежутки <i>Constraints</i> .....	391
Размещение компонентов .....	392
Панель инструментальных кнопок <i>JToolBar</i> .....	393
Интерфейс <i>Action</i> .....	395
Слоеная панель <i>JLayeredPane</i> .....	396
Корневая панель <i>JRootPane</i> .....	397
Окно <i>JWindow</i> .....	399
Диалоговое окно <i>JDialog</i> .....	400
Окно верхнего уровня <i>JFrame</i> .....	401
Внутреннее окно <i>JInternalFrame</i> .....	402
Рабочий стол <i>JDesktopPane</i> .....	404
Стандартные диалоги <i>JOptionPane</i> .....	405
Окно с индикатором <i>ProgressMonitor</i> .....	409
Заключение .....	410
Вопросы для самопроверки .....	411
<b>Глава 15. Обработка событий</b> .....	<b>412</b>
Самообработка событий .....	416
Обработка вложенным классом .....	417
Упражнение .....	418

Событие <i>ActionEvent</i> .....	418
Обработка действий мыши .....	419
Упражнение .....	422
Классы-адаптеры.....	422
Управление колесиком мыши.....	423
Обработка действий клавиатуры .....	424
Упражнение .....	425
Событие <i>TextEvent</i> .....	425
Событие изменения <i>ChangeEvent</i> .....	426
Обработка действий с окном .....	426
Событие <i>ComponentEvent</i> .....	427
Событие <i>ContainerEvent</i> .....	428
Событие <i>FocusEvent</i> .....	428
Событие <i>ItemEvent</i> .....	428
Событие <i>AdjustmentEvent</i> .....	429
Несколько слушателей одного источника .....	431
Диспетчеризация событий .....	432
Создание собственного события.....	434
Вопросы для самопроверки .....	435
<b>Глава 16. Оформление рамок .....</b>	<b>436</b>
Пустая рамка <i>EmptyBorder</i> .....	438
Прямолинейная рамка <i>LineBorder</i> .....	438
Объемная рамка <i>BevelBorder</i> .....	439
Закругленная объемная рамка <i>SoftBevelBorder</i> .....	439
Врезанная рамка <i>EtchedBorder</i> .....	440
Рамка с изображением <i>MatteBorder</i> .....	440
Рамки с надписями <i>TitledBorder</i> .....	441
Сдвоенные рамки <i>CompoundBorder</i> .....	444
Создание собственных рамок .....	445
Вопросы для самопроверки .....	450
<b>Глава 17. Изменение внешнего вида компонента .....</b>	<b>451</b>
Получение свойств L&F .....	453
Задание стандартного L&F.....	455
Дополнительные L&F .....	457
Смена всего L&F.....	457
Замена отдельных свойств L&F.....	459
Темы Java L&F .....	462
Вопросы для самопроверки .....	465
<b>Глава 18. Апплеты .....</b>	<b>466</b>
Упражнения .....	472
Передача параметров в апплет.....	472
Атрибуты тега <i>&lt;applet&gt;</i> .....	475
Сведения об окружении апплета .....	476
Упражнение .....	477
Изображение и звук в апплетах .....	477
Слежение за процессом загрузки.....	477
Класс <i>MediaTracker</i> .....	478

Упражнения .....	480
Защита от апплета .....	480
Апплеты в библиотеке Swing .....	481
Апплет <i>JApplet</i> .....	482
Упражнение .....	483
Заключение .....	484
Вопросы для самопроверки .....	484
<b>Глава 19. Прочие свойства Swing .....</b>	<b>485</b>
Свойства экземпляра компонента .....	485
Прокрутка содержимого компонента .....	486
Передача фокуса ввода .....	486
Перенос данных Drag and Drop .....	491
Временная задержка <i>Timer</i> .....	492
<b>Глава 20. Изображения и звук .....</b>	<b>494</b>
Модель "поставщик-потребитель" .....	494
Классы-фильтры .....	497
Как выделить фрагмент изображения .....	498
Как изменить цвет изображения .....	499
Как переставить пиксели изображения .....	500
Упражнения .....	501
Модель обработки прямым доступом .....	501
Преобразование изображения в Java 2D .....	504
Аффинное преобразование изображения .....	504
Изменение интенсивности изображения .....	507
Изменение составляющих цвета .....	508
Создание различных эффектов .....	509
Упражнения .....	510
Анимация .....	510
Улучшение изображения двойной буферизацией .....	512
Упражнения .....	516
Звук .....	516
Проигрывание звука в Java .....	517
Синтез и запись звука в Java .....	522
Упражнение .....	524
Вопросы для самопроверки .....	525

## **ЧАСТЬ IV. НЕОБХОДИМЫЕ КОНСТРУКЦИИ JAVA .....**

<b>Глава 21. Обработка исключительных ситуаций .....</b>	<b>529</b>
Блоки перехвата исключения .....	530
Упражнения .....	533
Часть заголовка метода <i>throws</i> .....	533
Оператор <i>throw</i> .....	536
Обработка нескольких типов исключений с помощью иерархии .....	536
Иерархия классов-исключений .....	537
Порядок обработки исключений .....	538
Упражнение .....	538

Обработка нескольких типов исключений с помощью перечисления .....	539
Создание собственных исключений .....	539
Заключение .....	541
Вопросы для самопроверки .....	541
<b>Глава 22. Подпроцессы .....</b>	<b>542</b>
Класс <i>Thread</i> .....	545
Синхронизация подпроцессов .....	550
Согласование работы нескольких подпроцессов .....	552
Приоритеты подпроцессов .....	557
Подпроцессы-демоны .....	558
Группы подпроцессов .....	559
Заключение .....	559
Вопросы для самопроверки .....	559
<b>Глава 23. Потоки ввода/вывода и печать .....</b>	<b>560</b>
Консольный ввод/вывод .....	565
Форматированный вывод .....	568
Спецификации вывода целых чисел .....	569
Спецификации вывода вещественных чисел .....	570
Спецификация вывода символов .....	570
Спецификации вывода строк .....	570
Спецификации вывода логических значений .....	570
Спецификации вывода хеш-кода объекта .....	570
Спецификации вывода даты и времени .....	570
Класс <i>Console</i> .....	571
Упражнения .....	572
Файловый ввод/вывод .....	572
Получение свойств файла .....	574
Работа с файлом средствами NIO2 .....	576
Буферизованный ввод/вывод .....	578
Каналы буферизованного ввода/вывода .....	579
Упражнения .....	581
Поток простых типов Java .....	582
Кодировка UTF-8 .....	582
Класс <i>DataOutputStream</i> .....	582
Прямой доступ к файлу .....	584
Упражнение .....	585
Каналы обмена информацией .....	585
Сериализация объектов .....	587
Печать в Java .....	590
Печать средствами Java 2D .....	592
Печать файла .....	596
Печать страниц с разными параметрами .....	598
Вопросы для самопроверки .....	599
<b>Глава 24. Сетевые средства Java .....</b>	<b>601</b>
Работа в WWW .....	604
Упражнения .....	607

Работа по протоколу TCP .....	608
Работа с проху-сервером .....	611
Упражнения .....	612
Работа по протоколу UDP .....	612
Упражнение .....	614
Вопросы для самопроверки .....	614
<b>ЧАСТЬ V. WEB-ТЕХНОЛОГИИ JAVA .....</b>	<b>617</b>
<b>Глава 25. Web-инструменты Java.....</b>	<b>619</b>
Архиватор <i>jar</i> .....	619
Создание архива .....	620
Файл описания MANIFEST.MF .....	622
Файл INDEX.LIST.....	623
Компоненты JavaBeans.....	624
Связь с базами данных через JDBC.....	625
Вопросы для самопроверки .....	629
<b>Глава 26. Сервлеты .....</b>	<b>631</b>
Web-приложение.....	632
Интерфейс <i>Servlet</i> .....	633
Конфигурационный файл .....	634
Интерфейс <i>ServletConfig</i> .....	637
Контекст сервлета .....	639
Метод <i>Service</i> .....	639
Интерфейс <i>ServletRequest</i> .....	640
Интерфейс <i>ServletResponse</i> .....	641
Цикл работы сервлета.....	641
Класс <i>GenericServlet</i> .....	642
Работа по протоколу HTTP .....	643
Интерфейс <i>HttpServletRequest</i> .....	643
Интерфейс <i>HttpServletResponse</i> .....	645
Класс <i>HttpServlet</i> .....	646
Аннотации сервлета .....	646
Пример сервлета класса <i>HttpServlet</i> .....	647
Сеанс связи с сервлетом .....	652
Фильтры.....	655
Обращение к другим ресурсам .....	660
Асинхронное выполнение запросов .....	661
Вопросы для самопроверки .....	664
<b>Глава 27. Страницы JSP.....</b>	<b>665</b>
Стандартные действия (теги) JSP .....	668
Язык записи выражений EL .....	671
Встроенные объекты JSP.....	672
Обращение к компоненту <i>JavaBean</i> .....	674
Выполнение апплета в браузере клиента .....	675
Передача управления .....	676

Пользовательские теги .....	677
Класс-обработчик пользовательского тега .....	679
Пользовательский тег с атрибутами .....	681
Пользовательский тег с телом .....	682
Обработка тела пользовательского тега .....	684
Обработка взаимодействующих тегов .....	686
Обработка исключений в пользовательских тегах .....	690
Обработка тегов средствами JSP .....	690
Стандартные библиотеки тегов JSTL .....	692
Библиотека core .....	693
Библиотека xml .....	696
Библиотека fmt .....	696
Библиотека sql .....	697
Библиотека fn .....	697
Frameworks .....	697
JavaServer Faces .....	698
Вопросы для самопроверки .....	703
<b>Глава 28. Связь Java с технологией XML .....</b>	<b>704</b>
Описание DTD .....	709
Пространства имен XML .....	711
Схема XML .....	713
Встроенные простые типы XSD .....	714
Вещественные числа .....	714
Целые числа .....	714
Строки символов .....	714
Дата и время .....	715
Двоичные типы .....	715
Прочие встроенные простые типы .....	715
Определение простых типов .....	716
Сужение .....	716
Список .....	717
Объединение .....	718
Описание элементов и их атрибутов .....	719
Определение сложных типов .....	719
Определение типа пустого элемента .....	720
Определение типа элемента с простым телом .....	720
Определение типа вложенных элементов .....	721
Определение типа со сложным телом .....	723
Пример: схема адресной книги .....	724
Безымянные типы .....	726
Пространства имен языка XSD .....	728
Включение файлов схемы в другую схему .....	730
Связь документа XML со своей схемой .....	731
Другие языки описания схем .....	732
Инструкции по обработке .....	732
Анализ документа XML .....	733
Анализ документов XML с помощью SAX2 .....	734

---

Анализ документов XML с помощью StAX .....	741
Связывание данных XML с объектами Java .....	743
Объекты данных JDO .....	744
Анализ документов XML с помощью DOM API.....	745
Интерфейс <i>Node</i> .....	746
Интерфейс <i>Document</i> .....	747
Интерфейс <i>Element</i> .....	748
Другие DOM-парсеры.....	751
Преобразование дерева объектов в XML.....	752
Таблицы стилей XSL .....	754
Преобразование документа XML в HTML .....	756
Вопросы для самопроверки .....	757
<b>Список литературы .....</b>	<b>758</b>
<b>Предметный указатель .....</b>	<b>760</b>



# Введение

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

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

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

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

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

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

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

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

Ну что же, начнем!

## Что такое Java?

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

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

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

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

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

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

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

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

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

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

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

## Структура книги

Книга состоит из пяти частей.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*Часть V* книги посвящена Web-технологии Java, точнее, тем ее разделам, которые касаются программирования серверов.

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

*Глава 26* посвящена основному средству программирования серверов — сервлетам.

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

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

## Выполнение Java-программы

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

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

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

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

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

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

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

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

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

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

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

## Что такое JDK?

Набор программ и классов JDK содержит:

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

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

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

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

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

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

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

Затем набор инструментальных средств JDK был сильно переработан.

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

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

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

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

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

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

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

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

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

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

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

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

## Что такое JRE?

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

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

## Как установить JDK?

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

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

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

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

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

### **ПРЕДУПРЕЖДЕНИЕ**

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

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

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

```
java -version
```

## **Как использовать JDK?**

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

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

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

```
javac MyProgram.java
```

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

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

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

```
java MyProgram
```

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

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

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

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

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

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

## Интегрированные среды Java

Сразу же после создания Java, уже в 1996 г., появились интегрированные среды разработки программ IDE (Integrated Development Environment) для Java, и их число все время возрастает. Некоторые из них, такие как Eclipse, IntelliJ IDEA, NetBeans, являются просто интегрированными оболочками над JDK, вызывающими из одного окна текстовый редактор, компилятор и интерпретатор. Эти интегрированные среды требуют предварительной установки JDK. Впрочем, Eclipse содержит собственный компилятор.

Другие интегрированные среды содержат JDK в себе или имеют собственный компилятор, например JBuilder фирмы Embarcadero или IBM Rational Application Developer. Их можно устанавливать, не имея под руками JDK. Надо заметить, что перечисленные продукты сами написаны полностью на Java.

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

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

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

Для изучения Java, пожалуй, удобнее всего интегрированная среда NetBeans IDE, которую можно свободно скопировать с сайта <http://netbeans.org/>. Она содержит много примеров, статей и учебников по различным разделам Java.

## Особая позиция Microsoft

Вы уже, наверное, почувствовали смутное беспокойство, не встречая название этой корпорации. Дело в том, что, имея свою операционную систему, огромное число приложений к ней и богатейшую библиотеку классов, Microsoft не имела нужды в Java. Но и пройти мимо технологии, распространившейся всюду, компания Microsoft не могла и создала свой компилятор Java, а также визуальное средство разработки, входящее в Visual Studio. Данный компилятор включает в байт-коды вызовы объектов ActiveX. Следовательно, выполнять эти байт-коды можно только на компьютерах, имеющих доступ к ActiveX. Эта "нечистая" Java резко ограничивает круг применения байт-кодов, созданных компилятором корпорации Microsoft. В результате судебных разбирательств с Sun Microsystems компания Microsoft назвала свой продукт Visual J++. Виртуальная машина Java корпорации Microsoft умеет выполнять байт-коды, созданные "чистым" компилятором, но не всякий интерпретатор выполнит байт-коды, написанные с помощью Visual J++. Этот продукт вошел в состав Visual Studio .NET 2005 под названием

J# (J sharp), но он генерирует не байт-коды JVM, а код .NET Framework CLR. Язык J# не получил распространения и был исключен из дальнейших версий Visual Studio .NET.

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

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

## Java в Интернете

Разработанная для применения в компьютерных сетях, Java просто не могла не найти отражения на сайтах Интернета. Действительно, масса сайтов полностью посвящена технологии Java или содержит информацию о ней. Одна только компания Oracle имеет несколько сайтов с информацией о Java:

- ☐ <http://www.oracle.com/technetwork/java/index.html> — основной сайт Java, отсюда можно скопировать JDK;
- ☐ <http://forums.oracle.com/forums/category.jspa?categoryID=285> — форумы для разработчиков Java;
- ☐ <http://www.java.net/> — сайт для разработчиков, знакомящихся с технологией Java.

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

Корпорация Microsoft содержит информацию о Java на сайте <http://www.microsoft.com/mscorp/java/default.mspx>.

Существует множество специализированных сайтов:

- ☐ <http://www.artima.com/forums/> — форумы для разработчиков, в том числе Java;
- ☐ <http://www.developer.com/java/> — большой сборник статей по Java;
- ☐ <http://www.freewarejava.com/> — советы разработчикам Java и готовые программы;
- ☐ <http://www.jars.com/> — Java Review Service;
- ☐ <http://www.javable.com/> — новостной сайт с русскими статьями, посвященный Java;
- ☐ <http://javaboutique.internet.com/> — еще один новостной сайт;
- ☐ <http://www.javalobby.com/> — новости, статьи и советы по Java;
- ☐ <http://www.javaranch.com/> — дружественный сайт и форум для разработчиков Java;
- ☐ <http://www.javaworld.com/> — электронный журнал;
- ☐ <http://www.jfind.com/> — сборник программ и статей;
- ☐ <http://www.jguru.com/> — советы специалистов;
- ☐ <http://java.sys-con.com/> — новинки технологии Java;
- ☐ <http://www.theserverside.com/> — вопросы создания серверных Java-приложений;
- ☐ <http://www.codeguru.com/Java/> — большой сборник статей, апплетов и других программ;

- <http://securingjava.com/> — здесь обсуждаются вопросы безопасности;
- <http://www.servlets.com/> — здесь обсуждаются вопросы написания сервлетов;
- <http://www.javacats.com/> — общая информация о Java и не только о Java.

Персональные сайты:

- <http://www.mindviewinc.com/Index.php> / — сайт Брюса Эккеля, автора популярных книг и статей;
- <http://www.davidreilly.com/> — сайт Дэвида Рейли, автора многих статей и книг о Java.

К сожалению, адреса сайтов часто меняются, некоторые сайты перестают существовать, возникают другие сайты. Возможно, вы и не найдете некоторые из перечисленных сайтов, зато появится много других.

## Литература по Java

Перечислим здесь только основные, официальные и почти официальные издания. Более полное описание чрезвычайно многочисленной литературы приведено в конце книги.

Полное и строгое описание языка изложено в книге *James Gosling, Bill Joy, Guy Steele, Gilad Bracha, "The Java Language Specification, Third Edition"*. В электронном виде она находится по адресу <http://java.sun.com/docs/books/jls/>, занимает в упакованном виде около 400 Кбайт.

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

Здесь же необходимо отметить книгу "отца" технологии Java Джеймса Гослинга, написанную вместе с Кеном Арнольдом и Дэвидом Холмсом. Имеется русский перевод: Арнольд К., Гослинг Дж., Холмс Д. Язык программирования Java. 3-е изд.: Пер. с англ. — М.: Издательский дом "Вильямс", 2001. — 624 с.: ил.

Официальным учебником хорошего стиля программирования на языке Java стала книга Блоха Д., *Java. Эффективное программирование*. Пер. с англ. — М.: Лори, 2008. — 223 с. На английском языке вышло второе издание этой книги, значительно расширенное и обновленное.

Компания Oracle содержит на своем сайте постоянно обновляемый электронный учебник Java Tutorial, размером уже в несколько десятков мегабайт: <http://download.oracle.com/javase/tutorial/> / . Время от времени появляется его печатное издание: *Mary Campione, Kathy Walrath, "The Java Tutorial, Second Edition: Object-Oriented Programming for the Internet"*.

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

## Благодарности

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

Отдельная благодарность Игорю Шишигину, предложившему ее издать и так быстро оформившему договор, что автор не успел передумать; моим студентам с их бесконечными вопросами; своим "сплюснутым" друзьям, убежденным в том, что "Жаба — это отстой", и сыну, Камиллю, для которого эта книга, собственно, и писалась.





# ЧАСТЬ I

## Базовые конструкции языка Java

<b>Глава 1.</b>	Встроенные типы данных, операции над ними
<b>Глава 2.</b>	Объектно-ориентированное программирование в Java
<b>Глава 3.</b>	Пакеты, интерфейсы и перечисления



# ГЛАВА 1



## Встроенные типы данных, операции над ними

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

Все правила языка Java исчерпывающе изложены в его спецификации, сокращенно называемой JLS (Java Language Specification), местоположение которой указано во *введении*. Иногда, чтобы понять, как выполняется та или иная конструкция языка Java, приходится обращаться к спецификации, но, к счастью, это бывает редко: правила языка Java достаточно просты и естественны.

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

## Первая программа на Java

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

### Листинг 1.1. Первая программа на языке Java

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

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

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

### ЗАМЕЧАНИЕ

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

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

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

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

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

Сильное влияние языка C привело к появлению в Java SE 5 (Java Standard Edition) метода `System.out.printf()`, очень похожего на одноименную функцию языка C. Мы подробно опишем этот метод в *главе 23*, но желающие могут ознакомиться с ним прямо сейчас.

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

#### **ЗАМЕЧАНИЕ**

Язык Java различает прописные и строчные буквы.

В именах нельзя оставлять пробелы. Свои имена можно записывать как угодно, можно было бы дать классу имя `helloworld` или `helloWorld`, но между Java-программистами заключено соглашение, называемое "Code Conventions for the Java Programming Language", хранящееся по адресу <http://www.oracle.com/technetwork/java/codeconv-138413.html>. Вот несколько пунктов этого соглашения:

- имена классов начинаются с прописной (заглавной) буквы; если имя содержит несколько слов, то каждое слово начинается с прописной буквы;
- имена методов и переменных начинаются со строчной буквы; если имя содержит несколько слов, то каждое следующее слово начинается с прописной буквы;
- имена констант записываются полностью прописными буквами; если имя состоит из нескольких слов, то между ними ставится знак подчеркивания.

Конечно, эти правила необязательны, хотя они и входят в JLS, п. 6.8, но сильно облегчают понимание кода и придают программе характерный для Java стиль.

Стиль определяют не только имена, но и размещение текста программы по строкам, например расположение фигурных скобок: оставлять ли открывающую фигурную скобку в конце строки с заголовком класса или метода или переносить на следующую строку? Почему-то этот пустячный вопрос вызывает ожесточенные споры, некоторые средства разработки даже предлагают выбрать определенный стиль расстановки фигурных скобок. Многие фирмы устанавливают свой внутрифирменный стиль. В книге мы постараемся следовать стилю "Code Conventions" и в том, что касается разбиения текста программы на строки (компилятор же рассматривает всю программу как одну длинную строку, для него программа — это просто последовательность символов), и в том, что касается отступов (`indent`) в тексте.

Итак, программа написана в каком-либо текстовом редакторе, например в Блокноте (`Notepad`), `emacs` или `vi`. Теперь ее надо сохранить в файле в текстовом, но не в графич-

ческом формате. Имя файла должно в точности совпадать с именем класса, содержащего метод `main()`. Данное правило очень желательно выполнять. При этом система исполнения Java будет быстро находить метод `main()` для начала работы, просто отыскивая класс, совпадающий с именем файла. Расширение имени файла должно быть `java`.

### СОВЕТ

Называйте файл с программой именем класса, содержащего метод `main()`, соблюдая регистр букв.

В нашем примере сохраним программу в файле с именем `HelloWorld.java` в текущем каталоге. Затем вызовем компилятор, передавая ему имя файла в качестве аргумента:

```
javac HelloWorld.java
```

Компилятор создаст файл с байт-кодами, даст ему имя `HelloWorld.class` и запишет этот файл в текущий каталог.

Осталось вызвать интерпретатор байт-кодов, передав ему в качестве аргумента имя класса (а не файла!):

```
java HelloWorld
```

На экране появится строка:

```
Hello, XXI Century World!
```

### ЗАМЕЧАНИЕ

Не указывайте расширение `class` при вызове интерпретатора.

На рис. 1.1 показано, как все это выглядит в окне **Command Prompt** операционной системы MS Windows 2003.

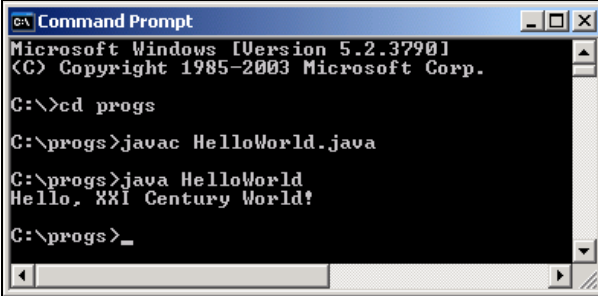
The image shows a screenshot of a Windows Command Prompt window. The title bar reads "Command Prompt". The window content shows the following text: "Microsoft Windows [Version 5.2.3790] (C) Copyright 1985-2003 Microsoft Corp." followed by a series of commands and their outputs: "C:\>cd progs", "C:\progs>javac HelloWorld.java", "C:\progs>java HelloWorld", and the output "Hello, XXI Century World!". The prompt "C:\progs>\_" is visible at the bottom.

Рис. 1.1. Окно **Command Prompt**

При работе в какой-либо интегрированной среде, например Eclipse или NetBeans, все эти действия вызываются выбором соответствующих пунктов меню или "горячими" клавишами — единых правил здесь нет.

## Комментарии

В текст программы можно вставить комментарии, которые компилятор не будет учитывать. Они очень полезны для пояснений по ходу программы. В период отладки можно выключать из действий один или несколько операторов, пометив их символами

комментария, как говорят программисты, "закомментировав" их. Кроме того, некоторые программы, работающие с Java, извлекают из комментариев полезные для себя сведения.

Комментарии вводятся таким образом:

- за двумя наклонными чертами, написанными подряд `//`, без пробела между ними, начинается комментарий, продолжающийся до конца строки;
- за наклонной чертой и звездочкой `/*` начинается комментарий, который может занимать несколько строк, до звездочки и наклонной черты `*/` (без пробелов между этими знаками);
- за наклонной чертой и двумя звездочками `/**` начинается комментарий, который может занимать несколько строк, до звездочки и наклонной черты `*/`. Из таких комментариев формируется документация.

Комментарии очень удобны для чтения и понимания кода, они превращают программу в документ, описывающий ее действия. Программу с хорошими комментариями называют *самодокументированной*. Поэтому в Java и введены комментарии третьего типа, а в состав JDK включена утилита — программа `javadoc`, извлекающая эти комментарии в отдельные файлы формата HTML и создающая гиперссылки между ними. В такой комментарий кроме собственно комментария можно вставить указания программе `javadoc`, которые начинаются с символа `@`.

Именно так создается документация к JDK.

Добавим комментарий к нашему примеру (листинг 1.2).

### Листинг 1.2. Первая программа с комментариями

```
/**
 * Разъяснение содержания и особенностей программы...
 * @author Имя Фамилия (автора)
 * @version 1.0 (это версия программы)
 */
class HelloWorld{ // HelloWorld — это только имя
 // Следующий метод начинает выполнение программы
 public static void main(String[] args){ // args не используются
 /* Следующий метод просто выводит свой аргумент
  * на экран дисплея */
 System.out.println("Hello, XXI Century World!");
 // Следующий вызов закомментирован,
 // метод не будет выполняться
 // System.out.println("Farewell, XX Century!");
 }
}
```

Звездочки в начале строк не имеют никакого значения, они написаны просто для выделения комментария. Пример, конечно, перегружен пояснениями (это плохой стиль), здесь просто показаны разные формы комментариев.

## Аннотации

Обратите внимание на комментарий, приведенный в начале листинга 1.2. В него вставлены указания-теги `@author` и `@version` утилите `javadoc`. Просматривая текст этого комментария и встретив какой-либо из тегов, утилита `javadoc` выполнит предписанные тем действием. Например, тег `@see` предписывает сформировать гиперссылку на другой документ HTML, а тег `@deprecated`, записанный в комментарий перед методом, вызовет пометку этого метода в документации как устаревшего.

Идея давать утилите предписания с помощью тегов оказалась весьма плодотворной. Кроме `javadoc` были написаны другие утилиты и целые программные продукты, которые вводят новые теги и используют их для своих целей. Например, программа `XDoclet` может автоматически создавать различные конфигурационные файлы, необходимые для работы сложных приложений. Разработчику достаточно вставить в свою программу комментарии вида `/**...*/` с тегами специального вида и запустить утилиту `Xdoclet`, которая сгенерирует все необходимые файлы.

Использование таких утилит стало общепризнанной практикой, и, начиная с пятой версии Java SE, было решено ввести прямо в компилятор возможность обрабатывать теги, которые получили название *аннотаций*. Аннотации записываются не внутри комментариев вида `/**...*/`, а непосредственно в том месте, где они нужны. Например, после того как мы запишем непосредственно перед заголовком какого-либо метода аннотацию `@Deprecated`, компилятор будет выводить на консоль предупреждение о том, что этот метод устарел и следует воспользоваться другим методом. Обычно замена указывается тут же, в этом же комментарии.

Несколько аннотаций, количество которых увеличивается с каждой новой версией JDK, объявлено прямо в компиляторе. Ими можно пользоваться без дополнительных усилий. Мы будем вводить их по мере надобности. Кроме них разработчик может объявить и использовать в своем приложении свои аннотации. Как это делается, рассказано в *главе 3*.

## Константы

В языке Java можно записывать константы различных типов в разных видах. Форма записи констант почти полностью заимствована из языка C. Перечислим все разновидности констант.

### Целые

Целые константы можно записывать в четырех системах счисления:

- в привычной для нас десятичной форме: `+5`, `-7`, `12345678`;
- в двоичной форме, начиная с нуля и латинской буквы `b` или `B`: `0b1001`, `0B11011`;
- в восьмеричной форме, начиная с нуля: `027`, `-0326`, `0777` (в записи таких констант недопустимы цифры 8 и 9);

#### ЗАМЕЧАНИЕ

Целое число, начинающееся с нуля, трактуется как записанное в восьмеричной форме, а не в десятичной.

- в шестнадцатеричной форме, начиная с нуля и латинской буквы `x` или `X`: `0xff0a`, `0xFC2D`, `0X45a8`, `0X77FF` (здесь строчные и прописные буквы не различаются).

Для улучшения читаемости группы цифр в числе можно разделять знаком подчеркивания: `1_001_234`, `0xFC_2D`.

Целые константы хранятся в оперативной памяти в формате типа `int` (см. далее).

В конце целой константы можно записать латинскую букву `"L"` (прописную `L` или строчную `l`), тогда константа будет сохраняться в длинном формате типа `long` (см. далее): `+25L`, `-037l`, `0xffL`, `0XDFDFL`.

### **СОВЕТ**

Не используйте при записи длинных целых констант строчную латинскую букву `l`, ее легко спутать с единицей.

## **Действительные**

Действительные константы записываются только в десятичной системе счисления в двух формах:

- с фиксированной точкой: `37.25`, `-128.678967`, `+27.035`;
- с плавающей точкой: `2.5e34`, `-0.345e-25`, `37.2E+4`; можно писать строчную или прописную латинскую букву `e`; пробелы и скобки недопустимы.

В конце действительной константы можно поставить букву `F` или `f`, тогда константа будет сохраняться в оперативной памяти в формате типа `float` (см. далее): `3.5f`, `-45.67F`, `4.7e-5f`. Можно приписать и букву `D` (или `d`): `0.045D`, `-456.77889d`, означающую тип `double`, но это излишне, поскольку действительные константы и так хранятся в формате типа `double`.

## **Символы**

Одиночные символы записываются в апострофах, чтобы отличить их от имен переменных. Для записи символов используются следующие формы:

- печатные символы, записанные на клавиатуре, просто записываются в апострофах (одинарных кавычках): `'a'`, `'N'`, `'?'`;
- управляющие и специальные символы записываются в апострофах с обратной наклонной чертой, чтобы отличить их от обычных символов:
  - `'\n'` — символ перевода строки LF (Line Feed) с кодом ASCII 10;
  - `'\r'` — символ возврата каретки CR (Carriage Return) с кодом 13;
  - `'\f'` — символ перевода страницы FF (Form Feed) с кодом 12;
  - `'\b'` — символ возврата на шаг BS (Backspace) с кодом 8;
  - `'\t'` — символ горизонтальной табуляции HT (Horizontal Tabulation) с кодом 9;
  - `'\'` — обратная наклонная черта;
  - `'\"'` — кавычка;
  - `'\''` — апостроф;

- код любого символа с десятичной кодировкой от 0 до 255 можно задать, записав его не более чем тремя цифрами в восьмеричной системе счисления в апострофах после обратной наклонной черты: '\123' — буква s, '\346' — буква ж в кодировке CP1251. Нет смысла использовать эту форму записи для печатных и управляющих символов, перечисленных в предыдущем пункте, поскольку компилятор сразу же переведет восьмеричную запись в указанную ранее форму. Наибольший восьмеричный код '\377' — десятичное число 255;
- код любого символа в кодировке Unicode набирается в апострофах после обратной наклонной черты и латинской буквы u четырьмя шестнадцатеричными цифрами: '\u0053' — буква s, '\u0416' — буква ж.

Символы хранятся в формате типа `char` (см. далее).

### **ПРИМЕЧАНИЕ**

Прописные русские буквы в кодировке Unicode занимают диапазон от '\u0410' — заглавная буква А, до '\u042F' — заглавная Я, строчные буквы от '\u0430' — а, до '\u044F' — я.

В какой бы форме ни записывались символы, компилятор переводит их в Unicode, включая и исходный текст программы.

### **ЗАМЕЧАНИЕ**

Компилятор и исполняющая система Java работают только с кодировкой Unicode.

## Строки

Строки символов заключаются в кавычки. Управляющие символы и коды записываются в строках точно так же, с обратной наклонной чертой, но, разумеется, без апострофов, и оказывают то же действие. Строки могут располагаться только на одной строке исходного кода, нельзя открывающую кавычку поставить на одной строке, а закрывающую — на следующей.

Вот некоторые примеры:

```
"Это строка\nс переносом"
"\Зубило\" — Чемпион!"
```

### **ЗАМЕЧАНИЕ**

Строки символов нельзя начинать на одной строке исходного кода, а заканчивать на другой.

Для строковых констант определена операция сцепления, обозначаемая плюсом. Запись

```
"Сцепление " + "строк"
```

дает в результате строку "Сцепление строк". Обратите внимание на то, что между сцепляемыми строками не вставлены никакие дополнительные символы. Пробел между ними принадлежал первой строке.

Чтобы записать длинную строку в виде одной строковой константы, надо после закрывающей кавычки на первой и следующих строках поставить плюс (+); тогда компилятор соберет две (или более) строки в одну строковую константу, например:

```
"Одна строковая константа, записанная " +
"на двух строках исходного текста"
```

Тот, кто попытается вывести символы в кодировке Unicode, например слово "Россия":

```
System.out.println("\u04229\u043e\u0441\u0441\u0441\u0438\u0444");
```

должен знать, что MS Windows использует для вывода в окно **Command Prompt** шрифт Terminal, в котором буквы кириллицы расположены в начальных кодах Unicode (почему-то в кодировке CP866) и разбросаны по другим сегментам Unicode.

Не все шрифты Unicode содержат начертания (glyphs) всех символов, поэтому будьте осторожны при выводе строк в кодировке Unicode.

### **СОВЕТ**

Используйте Unicode напрямую только в крайних случаях.

## Имена

*Имена* (names) переменных, классов, методов и других объектов могут быть простыми (общее название — *идентификаторы* (identifiers)) и *составными* (qualified names). Идентификаторы в Java состоят из так называемых *букв Java* (Java letters) и арабских цифр 0—9, причем первым символом идентификатора не может быть цифра. (Действительно, как понять запись 2e3: как число 2000,0 или как имя переменной?) В набор букв Java обязательно входят прописные и строчные латинские буквы, знак доллара (\$) и знак подчеркивания (\_), а также символы национальных алфавитов.

### **ЗАМЕЧАНИЕ**

Не указывайте в именах знак доллара. Компилятор Java использует его для записи имен вложенных классов.

Вот примеры правильных идентификаторов:

```
a1      my_var      var3_5      _var      veryLongVarName
aName   theName     a2Vh36kBnMt456dX
```

В именах лучше не использовать строчную букву l, которую легко спутать с единицей, и букву o, которую легко принять за нуль.

Придумывая имена, не забывайте о рекомендациях "Code Conventions".

В классе Character, входящем в состав Java API, есть два метода, проверяющие, пригоден ли данный символ для использования в идентификаторе: метод isJavaIdentifierStart(), проверяющий, является ли символ буквой Java, и метод isJavaIdentifierPart(), выясняющий, является ли символ буквой, цифрой, знаком подчеркивания (\_) или знаком доллара (\$).

Служебные слова Java, такие как class, void, static, зарезервированы, их нельзя использовать в качестве идентификаторов своих объектов.

*Составное имя* (qualified name) — это несколько идентификаторов, разделенных точками, без пробелов, например уже встречавшееся нам имя System.out.println.

## Примитивные типы данных и операции

Все типы исходных данных, встроенные в язык Java, делятся на две группы: *примитивные типы* (primitive types) и *ссылочные типы* (reference types).

Ссылочные типы включают *массивы* (arrays), *классы* (classes) и *интерфейсы* (interfaces). Начиная с Java SE 5 появился *перечислимый тип* (enum).

Примитивных типов всего восемь. К ним относятся *логический* (иногда говорят *булев*) тип, называемый `boolean`, и семь *числовых* (numeric) типов.

Числовые типы делятся на *целые* (integral<sup>1</sup>) и *вещественные* (floating-point).

Целых типов пять: `byte`, `short`, `int`, `long`, `char`.

Символы можно применять везде, где используется тип `int`, поэтому JLS причисляет тип `char` к целым типам. Например, символы можно использовать в арифметических вычислениях, скажем, можно написать `2 + 'ж'`, к двойке будет прибавляться кодировка Unicode `'\u0416'` буквы 'ж'. В десятичной форме это число 1046, и в результате сложения получим 1048.

Напомним, что в записи `2 + "ж"`, где буква Ж записана как строка, в кавычках, плюс понимается как сцепление строк, двойка будет преобразована в строку, в результате получится строка "2ж".

Вещественных типов всего два: `float` и `double`.

На рис. 1.2 показана иерархия типов данных Java.

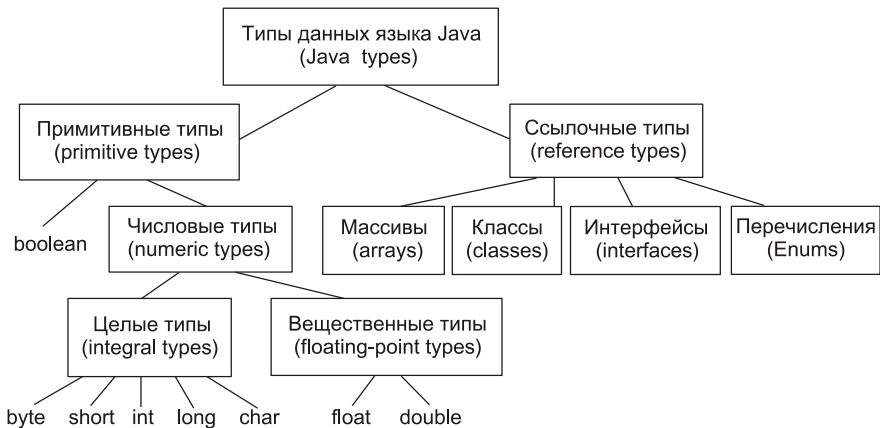


Рис. 1.2. Типы данных языка Java

Поскольку по имени переменной невозможно определить ее тип, все переменные обязательно должны быть описаны перед их использованием. Описание заключается в том, что записывается имя типа, затем через пробел список имен переменных, относящихся к этому типу. Имена в списке разделяются запятой. Для всех или некоторых переменных можно указать начальные значения после знака равенства, которыми могут служить любые константные выражения того же типа. Описание каждого типа завершается точкой с запятой. В программе может быть сколько угодно описаний каждого типа.

<sup>1</sup> Название "integral" не является устоявшимся термином. Так названа категория целых типов данных в книге *The Java Language Specification, Third Edition*. James Gosling, Bill Joy, Guy Steele, Gilad Bracha (см. введение). — Ред.

**ЗАМЕЧАНИЕ ДЛЯ СПЕЦИАЛИСТОВ**

Java — язык со строгой типизацией (strongly typed language).

Разберем каждый тип подробнее.

**Логический тип**

Значения логического типа `boolean` возникают в результате различных сравнений, вроде  $2 > 3$ , и используются главным образом в условных операторах и операторах циклов. Логических значений всего два: `true` (истина) и `false` (ложь). Это служебные слова Java. Описание переменных данного типа выглядит так:

```
boolean b = true, bb = false, bool2;
```

Над логическими данными можно выполнять операции присваивания, например `bool2 = true`, в том числе и составные с логическими операциями; сравнение на равенство `b == bb` и на неравенство `b != bb`, а также логические операции.

**Логические операции**

В языке Java реализованы четыре логические операции:

- отрицание (NOT) — `!` (обозначается восклицательным знаком);
- конъюнкция (AND) — `&` (амперсанд);
- дизъюнкция (OR) — `|` (вертикальная черта);
- исключающее ИЛИ (XOR) — `^` (каре).

Они выполняются над логическими данными типа `boolean`, их результатом будет тоже логическое значение — `true` или `false`. Про эти операции можно ничего не знать, кроме того, что представлено в табл. 1.1.

**Таблица 1.1.** Логические операции

<code>b1</code>	<code>b2</code>	<code>!b1</code>	<code>b1 &amp; b2</code>	<code>b1   b2</code>	<code>b1 ^ b2</code>
<code>true</code>	<code>true</code>	<code>false</code>	<code>true</code>	<code>true</code>	<code>false</code>
<code>true</code>	<code>false</code>	<code>false</code>	<code>false</code>	<code>true</code>	<code>true</code>
<code>false</code>	<code>true</code>	<code>true</code>	<code>false</code>	<code>true</code>	<code>true</code>
<code>false</code>	<code>false</code>	<code>true</code>	<code>false</code>	<code>false</code>	<code>false</code>

Словами эти правила можно выразить так:

- отрицание меняет значение истинности;
- конъюнкция истинна, только если оба операнда истинны;
- дизъюнкция ложна, только если оба операнда ложны;
- исключающее ИЛИ истинно, только если значения операндов различны.

**ЗАМЕЧАНИЕ**

Если бы Шекспир был программистом, фразу "To be or not to be" он написал бы так:  
`2b | ! 2b.`

Кроме перечисленных четырех логических операций есть еще две логические операции сокращенного вычисления:

- сокращенная конъюнкция (conditional-AND) — `&&`;
- сокращенная дизъюнкция (conditional-OR) — `||`.

Удвоенные знаки амперсанда и вертикальной черты следует записывать без пробелов.

Правый операнд сокращенных операций вычисляется только в том случае, если от него зависит результат операции, т. е. если левый операнд конъюнкции имеет значение `true` или левый операнд дизъюнкции имеет значение `false`.

Это правило очень удобно и довольно ловко используется программистами, например можно записывать выражения `(n != 0) && (m/n > 0.001)` или `(n == 0) || (m/n > 0.001)`, не опасаясь деления на нуль.

### ЗАМЕЧАНИЕ

Практически всегда в Java используются именно сокращенные логические операции.

## Упражнения

- Для переменных `b` и `bb`, определенных в разд. "Логический тип" данной главы, найдите значение выражения `b & bb && !bb | b`.
- При тех же определениях вычислите выражение `(!b || bb) && (bb ^ b)`.

## Целые типы

Спецификация языка Java, JLS, определяет разрядность (количество байтов, выделяемых для хранения значений типа в оперативной памяти) каждого типа. Для целых типов она приведена в табл. 1.2. В таблице указан также диапазон значений каждого типа, получаемый на процессорах архитектуры Pentium.

Таблица 1.2. Целые типы

Тип	Разрядность (байт)	Диапазон
<code>byte</code>	1	От -128 до 127
<code>short</code>	2	От -32 768 до 32 767
<code>int</code>	4	От -2 147 483 648 до 2 147 483 647
<code>long</code>	8	От -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
<code>char</code>	2	От '\u0000' до '\uFFFF', в десятичной форме от 0 до 65 535

Хотя тип `char` занимает два байта, в арифметических вычислениях он участвует как тип `int`, ему выделяется 4 байта, два старших байта заполняются нулями.

Вот примеры определения переменных целых типов:

```
byte b1 = 50, b2 = -99, b3;
short det = 0, ind = 1, sh = 'd';
```

```
int i = -100, j = 100, k = 9999;  
long big = 50, veryBig = 2147483648L;  
char c1 = 'A', c2 = '?', c3 = 36, newLine = '\n';
```

Целые типы, кроме `char`, хранятся в двоичном виде с дополнительным кодом. Последнее означает, что для отрицательных чисел хранится не их двоичное представление, а *дополнительный код* этого двоичного представления.

Дополнительный код получается так: в двоичном представлении числа все нули меняются на единицы, а единицы на нули, после чего к результату прибавляется единица, разумеется, в двоичной арифметике.

Например, значение 50 переменной `b1`, определенной ранее, будет храниться в одном байте с содержимым 00110010, а значение `-99` переменной `b2` — в байте с содержимым, которое вычисляется так: число 99 переводится в двоичную форму, получая 01100011, меняются единицы и нули, получая 10011100, и прибавляется единица, получая окончательно байт с содержимым 10011101.

Смысл всех этих преобразований в том, что сложение числа с его дополнительным кодом в двоичной арифметике даст в результате нуль; старший бит, равный 1, просто теряется, поскольку выходит за разрядную сетку. Это означает, что в такой странной арифметике дополнительный код числа является противоположным к нему числом, числом с обратным знаком. А это, в свою очередь, означает, что вместо того, чтобы вычесть из числа `A` число `B`, можно к `A` прибавить дополнительный код числа `B`. Таким образом, операция вычитания исключается из набора машинных операций.

Над целыми типами можно производить массу операций. Их набор восходит к языку `C`, он оказался удобным и кочует из языка в язык почти без изменений. Особенности применения этих операций в языке `Java` показаны на примерах.

## Операции над целыми типами

Все операции, которые производятся над целыми числами, можно разделить на следующие группы.

### Арифметические операции

К арифметическим операциям относятся:

- сложение — + (плюс);
- вычитание — - (дефис);
- умножение — \* (звездочка);
- деление — / (наклонная черта, слэш);
- взятие остатка от деления (деление по модулю) — % (процент);
- инкремент (увеличение на единицу) — ++;
- декремент (уменьшение на единицу) — --.

Между сдвоенными плюсами и минусами нельзя оставлять пробелы.

Сложение, вычитание и умножение целых значений выполняются как обычно, а вот деление целых значений в результате дает опять целое (так называемое *целочисленное*