

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®

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 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 61,92.

Тираж 1800 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

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

Отпечатано с готовых диапозитивов

в ГУП "Типография "Наука"

199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0735-6

© Хабибуллин И. Ш., 2011

© Оформление, издательство "БХВ-Петербург", 2011

Оглавление

Введение.....	19
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
ЧАСТЬ I. БАЗОВЫЕ КОНСТРУКЦИИ ЯЗЫКА JAVA	35
Глава 1. Встроенные типы данных, операции над ними	37
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
Глава 2. Объектно-ориентированное программирование в Java	73
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

Глава 3. Пакеты, интерфейсы и перечисления 109

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

ЧАСТЬ II. ИСПОЛЬЗОВАНИЕ КЛАССОВ ИЗ JAVA API..... 131

Глава 4. Классы-оболочки и generics 133

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

Глава 5. Работа со строками 156

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
Глава 6. Классы-коллекции	174
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
Глава 7. Классы-утилиты	203
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	213
Глава 8. Принципы построения графического интерфейса	215
Компонент и контейнер	217
Иерархия классов АWT	220

Окно библиотеки Swing.....	221
Использование системных приложений	222
System Tray	223
Splash Screen	224
Заключение	224
Вопросы для самопроверки	224
Глава 9. Графические примитивы	226
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
Заключение	256
Вопросы для самопроверки	256
Глава 10. Основные компоненты AWT.....	257
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
Глава 11. Оформление GUI компонентами Swing	296
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
Глава 12. Текстовые компоненты	334
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

Глава 13. Таблицы	351
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
Глава 14. Размещение компонентов и контейнеры Swing	372
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
Глава 15. Обработка событий	412
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
Глава 16. Оформление рамок	436
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
Глава 17. Изменение внешнего вида компонента	451
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
Глава 18. Апплеты	466
Упражнения	472
Передача параметров в апплет	472
Атрибуты тега <i><applet></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
Глава 19. Прочие свойства Swing.....	485
Свойства экземпляра компонента.....	485
Прокрутка содержимого компонента.....	486
Передача фокуса ввода.....	486
Перенос данных Drag and Drop.....	491
Временная задержка <i>Timer</i>	492
Глава 20. Изображения и звук.....	494
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.....	527
Глава 21. Обработка исключительных ситуаций.....	529
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
Глава 22. Подпроцессы	542
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
Глава 23. Потоки ввода/вывода и печать	560
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
Глава 24. Сетевые средства Java	601
Работа в WWW	604
Упражнения	607

Работа по протоколу TCP.....	608
Работа с прокси-сервером.....	611
Упражнения.....	612
Работа по протоколу UDP.....	612
Упражнение.....	614
Вопросы для самопроверки.....	614

ЧАСТЬ V. WEB-ТЕХНОЛОГИИ JAVA..... 617

Глава 25. Web-инструменты Java..... 619

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

Глава 26. Сервлеты..... 631

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

Глава 27. Страницы JSP..... 665

Стандартные действия (теги) JSP.....	668
Язык записи выражений EL.....	671
Встроенные объекты JSP.....	672
Обращение к компоненту JavaBean.....	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
Глава 28. Связь Java с технологией XML	704
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
Список литературы	758
Предметный указатель	760

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 `\n`, 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.

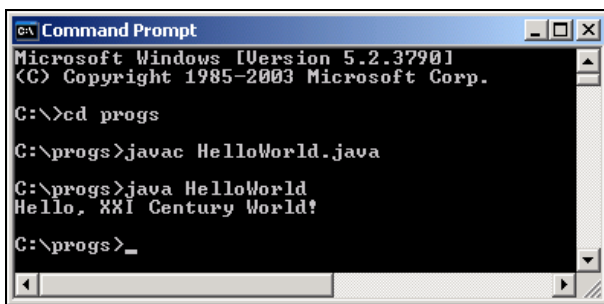


Рис. 1.1. Окно **Command Prompt**

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 комментарии 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 тегом действия. 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 " + "строка"
```

дает в результате строку "Сцепление строка". 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"
```