Глава 2
ТИПЫ ДАННЫХ И ОПЕРАТОРЫ

Любая программа манипулирует информацией (простыми данными и объектами) с помощью операторов. Каждый оператор производит результат из значений своих операндов или изменяет непосредственно значение операнда.

Базовые типы данных и литералы

В языке Java используются базовые типы данных, значения которых размещаются в стековой памяти (stack). Эти типы обеспечивают более высокую производительность вычислений по сравнению с объектами. Кроме этого, для каждого базового типа имеются классы-оболочки, которые инкапсулируют данные базовых типов в объекты, располагаемые в динамической памяти (heap).

Определено восемь базовых типов данных, размер каждого из которых остается неизменным независимо от платформы. Беззнаковых типов в Java не существует. Каждый тип данных определяет множество значений и их представление в памяти. Для каждого типа определен набор операций над его значениями.

Тип	Размер (бит)	По умолчанию	Значения (диапазон или максимум)
`boolean`	8	`false`	`true, false`
`byte`	8	0	`-128..127`
`char`	16	`'\u0000'`	`0..65535`
`short`	16	0	`-32768..32767`
`int`	32	0	`-2147483648..2147483647`
`long`	64	0	`9223720368547758O7L`
`float`	32	0.0	`3.40282347E+38`
`double`	64	0.0	`1.797693134486231570E+308`

В Java используются целочисленные литералы, например: **`35`** – целое десятичное число, **`071`** – восьмеричное значение, **`0x51`** – шестнадцатеричное значение. Целочисленные литералы по умолчанию относятся к типу **`int`**. Если необходимо определить длинный литерал типа **`long`**, в конце указывается символ **`L`** (например: **`0xffffL`**). Если значение числа больше значения, помещающегося в **`int`** (**`2147483647`**), то Java автоматически предполагает, что оно типа **`long`**. Литералы с плавающей точкой записываются в виде **`1.618`** или в экспоненциальной форме **`0.112E-05`** и относятся к типу **`double`**, таким образом, действительные числа относятся к типу **`double`**. Если необходимо определить литерал типа **`float`**, то в конце литерала следует добавить символ **`F`**. Символьные лите-

ралы определяются в апострофах (**'a'**, **'\n'**, **'\141'**, **'\u005a'**). Для размещения символов используется формат Unicode, в соответствии с которым для каждого символа отводится два байта. В формате Unicode первый байт содержит код управляющего символа или национального алфавита, а второй байт соответствует стандартному ASCII коду, как в C++. Любой символ можно представить в виде **'\u***code***'**, где *code* представляет двухбайтовый шестнадцатеричный код символа. Java поддерживает управляющие символы, не имеющие графического изображения;

'\n'– новая строка, **'\r'** – переход к началу, **'\f'** – новая страница, **'\t'**– табуляция, **'\b'** – возврат на один символ, **'\uxxxx'** – шестнадцатеричный символ Unicode, **'\ddd'**– восьмеричный символ и др. Начиная с J2SE 5.0 используется формат Unicode 4.0. Поддержку четырехбайтным символам обеспечивает наличие специальных методов в классе **Character**.

К литералам относятся булевские значения **true** и **false**, а также **null** – значение по умолчанию для ссылки на объект. При инициализации строки всегда создается объект класса **String** – это не массив символов и не строка. Строки, заключенные в двойные апострофы, считаются литералами и размещаются в пуле литералов, но в то же время такие строки представляют собой объекты.

В арифметических выражениях автоматически выполняются расширяющие преобразования типа **byte → short → int → long → float → double**. Java автоматически расширяет тип каждого **byte** или **short** операнда до **int** в выражениях. Для сужающих преобразований необходимо производить явное преобразование вида **(тип)значение**. Например:

byte b = (byte)128; *//преобразование int в byte*

Указанное в данном примере преобразование необязательно, так как в операциях присваивания литералов при инициализации преобразования выполняются автоматически. При инициализации полей класса и локальных переменных с использованием арифметических операторов автоматически выполняется приведение литералов к объявленному типу без необходимости его явного указания, если только их значения находятся в допустимых пределах, кроме инициализации объектов классов-оболочек. Java не позволяет присваивать переменной значение более длинного типа, в этом случае необходимо явное преобразование типа. Исключение составляют операторы инкремента (++), декремента (--) и сокращенные операторы (+=, /= и т.д.). При явном преобразовании **(тип)значение** возможно усечение значения.

Имена переменных не могут начинаться с цифры, в именах не могут использоваться символы арифметических и логических операторов, а также символ '**#**'. Применение символов '**$**' и '**_**' допустимо, в том числе и в первой позиции имени.

```
/* пример # 1 : типы данных, литералы и операции над ними :TypeByte.java */
package chapt02;

public class TypeByte {
      public static void main(String[] args) {
            byte b = 1, b1 = 1 + 2;
            final byte B = 1 + 2;
            //b = b1 + 1; //ошибка приведения типов
```

```
        /* b1 – переменная, и на момент выполнения кода b = b1 + 1;
        может измениться, и выражение b1 + 1 может превысить до-
        пустимый размер byte- типа */
        b = (byte)(b1 + 1);
        b = B + 1; // работает
        /* B - константа, ее значение определено, компилятор вычисля-
        ет значение выражения B + 1, и если его размер не превышает
        допустимого для byte типа, то ошибка не возникает */
        //b = -b; //ошибка приведения типов
        b = (byte) -b;
        //b = +b; //ошибка приведения типов
        b = (byte) +b;
        int i = 3;
        //b = i; //ошибка приведения типов, int больше чем byte
        b = (byte) i;
        final int I = 3;
        b = I;        // работает
        /*I –константа. Компилятор проверяет, не превышает ли ее
        значение допустимый размер для типа byte, если не превышает,
        то ошибка не возникает */
        final int I2 = 129;
        //b=I2; //ошибка приведения типов, т.к. 129 больше, чем 127
        b = (byte) I2;

        b += i++;    // работает
        b += 1000;   // работает
        b1 *= 2;     // работает
        float f = 1.1f;
        b /= f;      // работает
        /* все сокращенные операторы автоматически преобразуют ре-
        зультат выражения к типу переменной, которой присваивается
        это значение. Например, b /= f; равносильно b = (byte)(b / f); */
    }
}
```

Переменная базового типа, объявленная как член класса, хранит нулевое значение, соответствующее своему типу. Если переменная объявлена как локальная переменная в методе, то перед использованием она обязательно должна быть проинициализирована, так как она не инициализируется по умолчанию нулем. Область действия и время жизни такой переменной ограничена блоком **{}**, в котором она объявлена.

Документирование кода

В языке Java используются блочные и однострочные комментарии **/* */** и **//**, аналогичные комментариям, применяемым в C++. Введен также новый вид комментария **/** */**, который может содержать дескрипторы вида:

@author – задает сведения об авторе;

@version – задает номер версии класса;