

Мультимедийный курс

Программирование на Java

Часть 1

Лекция 2.1

1. Примитивные (простые) типы данных
2. Операции, комментарии

Примитивные типы данных

Тип	Значение	Размер	Константы
boolean	true или false	1 бит	true, false
char	символ Unicode	2 байта	'X', \u00ff, 88,
byte	целое со знаком	1 байт	50 (определяется как int, но при присваивании не дает ошибки)
short	целое со знаком	2 байта	50 (- " -)
int	целое со знаком	4 байта	50
long	целое со знаком	8 байтов	50L, 50l
float	число с п.т. в формате IEEE 754	4 байта	50F, 50f, 0.443f, 6.015E 21F, 3445e -05f
double	число с п.т. в формате IEEE 754	8 байтов	6.015E 21, 3445e -05fD, 5780d

Размер каждого типа определен в самом языке и **не зависит от реализации**

Логические значения

- **boolean** не являются целыми и не могут быть преобразованы в числовой тип или из него
- Величины типа **boolean** принимают значения **true** или **false**
- Объявление булевских переменных:
boolean a;
boolean b;

*Встроенные примитивные **целые типы** `byte`, `short`, `int`, `long` и символьный тип `char`, в некотором смысле также являющийся целочисленным. При этом только тип `char` беззнаковый, все остальные – знаковые*

Символьный тип (*char*)

- Тип *char* в Java, как и в C/C++, является числовым, хотя и предназначен для хранения отдельных символов
- Переменной символьного типа можно присваивать один символ, заключенный в одинарные кавычки, либо кодирующую символ управляющую последовательность Unicode

Кодировка символов

Символы в Java являются **16-разрядными символами Unicode**. Первые 256 символов Unicode совместимы с кодировкой ISO8859-1.

Escape-последовательности

`\uxxxx` – 16-ричный код символа Unicode (xxxx);

В стиле C:

`\ddd` - 8-ричный код символа (ddd)

`\"` - двойная кавычка

`'` - одинарная кавычка кавычка

`\\` - обратный слэш

`\t` - символ табуляции

`\n` - перевод строки

Escape-последовательности (продолжение)

`\f` – перевод строки

`\b` – возврат на один символ (Backspace)

`\r` - возврат каретки

Замечание. Escape-последовательности *Unicode* могут находиться в любом месте программы, а не только в строковых константах:

```
int \u0069;           // означает то же, что int i;
```

Escape-последовательности (продолжение)

Применение Escape-последовательности для инициализации полей

```
int n = (short) '\uf000';
```

Преобразование символов в числа может давать отрицательные значения (в данном случае значение -4096)

Escape-последовательности (продолжение)

Применение Escape-последовательности для инициализации полей

```
int n = (short) '\uf000';
```

Преобразование символов в числа может давать отрицательные значения (в данном случае значение -4096)

Целочисленные типы

byte – однобайтовое целое число
(8-битное целое со знаком)

short – двухбайтовое целое число
(16-битное целое со знаком)

int – четырехбайтовое целое число
(32-битное целое со знаком)

long – восьмибайтовое целое число
(64-битное целое со знаком)

Целочисленные типы

- Для задания в тексте программы численных констант типа **long**, выходящих за пределы диапазона чисел типа **int**, после написания числа следует ставить постфикс – букву **L**
- Например, 6000000000000000L. Можно ставить и строчную **l**, но ее хуже видно, особенно – на распечатках программы (можно перепутать с единицей)
- В остальных случаях для всех целочисленных типов значение указывается в обычном виде, и оно считается имеющим тип **int** – но при присваивании число типа **int** автоматически преобразуется в значение типа **long**

Целочисленные типы

Примеры задания переменных в классе.

```
int i,j,k;
```

```
int j1;byte i1=0,i2=-5;short i3=-15600;
```

```
long m1=1,m2,m3=-100;
```

После указанных объявлений переменные *i,j,k,j1,i1,m2* имеют значение *0*

Использование в выражениях:

```
i=5;j=i*i + 1
```

```
m1=jm2=255;
```

```
m1=m1 + m2*2;
```

Вещественные типы

Формат - IEEE 754 (другой вариант названия IEC 60559:1989)

float (4 байта):

- ✓ знак - 1 бит, мантисса - 23 бита
- ✓ порядок - 8 битов (смещен на -127 т.е. нуль означает 2 в -127-й степени)
- ✓ диапазон чисел $3.4E-38 \div 3.4E+38$

double (8 байтов): :

- ✓ знак - 1 бит, мантисса - 52 бита
- ✓ порядок - 11 бит
- ✓ диапазон чисел $1.7E-308 \div 1.7E+308$

Вещественный тип (продолжение)

Особые значения, получаемые при выполнении операций:

1) значения бесконечности различаются как по знаку, так и по типу: `Float.NEGATIVE_INFINITY`, `Double.NEGATIVE_INFINITY` ($-\infty$)
– `Float.POSITIVE_INFINITY`, `Double.POSITIVE_INFINITY` ($+\infty$)

2) не число (NaN – Not a Number)

- значение NaN может получаться при преобразовании строки в число, взятии логарифма от отрицательного числа, тригонометрической функции от бесконечности и т.п
- не сравнимо ни с чем, даже с самим собой
- `Float.NaN` и `Double.NaN` различаются

При делении на ноль целых чисел генерируется исключение **`ArithmeticException`**

Характеристики операций

Приоритет - определяет порядок выполнения операций в выражении, когда нет скобок

Ассоциативность — задает порядок выполнения двух рядом стоящих операций с равным приоритетом (левая или правая выполняется раньше?)

Арифметические операции

Приоритет	Оператор	Ассоциативность	Название операции
1	++	П	пре- и постинкремент
1	--	П	пре- и постдекремент
1	+, -	П	унарные + и -
2	*, /, %	Л	умножение, деление, остаток от деления по модулю (применима и к целым, и к вещественным)
3	+, -	Л	сложение, вычитание
	+=, -=, *=, /=, %=		специальное присваивание со сложением, вычитанием, умножением, делением, модулем

Арифметические операции (продолжение)

□ Специальные операции присваивания

x=10;

x=x%3;

x%=3;

} альтернатива

реализуется эффективнее

Преобразование числовых типов

Неявное, если два типа совместимы (все числовые)

- целевой тип $>$ исходного (расширяющее преобразование)
- при сохранении литеральной целочисленной константы в поля типа *byte*, *short*, *long*
- в промежуточных выражениях (расширяющее преобразование), если
 - операнды типа *byte* или *short* - до *int*
 - один операнд *long* и другие целые – до *long*
 - один операнд типа *float*, а другие целые - до *float*
 - один операнд типа *double* - до *double*

```
byte c = 50;
```

```
c=c*2; // ошибка, т.к. результат выражения имеет тип int
```

Преобразование числовых типов

Явное приведение - сужающее
(**<целевой_тип>**) **<значение>**

```
byte c = 50;  
c = (byte) (c*2);
```

- ❖ При назначении вещественного целому типу дробная часть теряется !
- ❖ При превышении значением диапазона целевого типа оно редуцируется по модулю этого типа до остатка от деления на модуль диапазона!

Арифметические операции (продолжение)

- Различие между постинкрементом и преинкрементом

x=15;

x++; // значение x - 16

y=++x; // значение y - 16, x - 15

y=x++; // значение y - 15, x - 16

- Деление целых

int c = 7/2; // дробная часть отбрасывается

Поразрядные операции

Тип операнда - целый

Приоритет	Оператор	Ассоциативность	Название операции
2	~	П	унарное отрицание
7	&	Л	битовое AND
8	^	Л	битовое XOR
9		Л	битовое OR
4	<<, >>, >>>	Л	сдвиг на указанное количество разрядов влево, вправо, вправо с заполнением знак. бита нулем
	&=, ^=, =, <<=, >>=, >>>=		специальное присваивание с соответствующей операцией

Поразрядные операции (продолжение)

Метод кодирования «Дополнение до двух»

Схема кодирования для отриц. чисел

1) **инвертирование всех битов**

2) **+ двоичная 1**

$$-42 = 42(00101010) \rightarrow 11010101 \rightarrow 11010110$$

Схема декодирования для отриц. чисел

1) **инвертирование всех битов**

2) **преобразование в десятичную форму**

3) **+ 1**

4) **изменение знака**

$$11010110 \rightarrow 41 \rightarrow 42 \rightarrow -42$$

Поразрядные операции (продолжение)

Пример операции «*Правый сдвиг*»

□ положительное

```
int a=35; // 00100011
```

```
a >>= 2; // 00001000 – число 8
```

□ отрицательное

```
int b=-8; // 11111000
```

```
b >>= 1; // 11111100 – число -4
```

При *сдвиге вправо* освобождающиеся разряды
заполняются значением знакового бита !

При *сдвиге влево* освобождающиеся разряды
заполняются 0!

Операции отношений

Приоритет	Оператор	Тип операнда	Ассоциативность	Название операции
5	<, <=, >, >=	числовой (целый, веществ., симв.)	Л	меньше, меньше или равно, больше, больше или равно
6	==	простой	Л	равенство
6	!=	простой	Л	неравенство
6	==	объект	Л	равенство ссылок
6	!=	объект	Л	неравенство ссылок

Результат операции – значение типа **boolean**

Операции булевой логики

Приоритет	Оператор	Ассоциативность	Название операции
9	&	Л	булево AND
9		Л	булево OR
8	^	Л	булево исключающее ИЛИ (XOR) (по полной схеме)
10	&&	Л	булево AND (по краткой схеме)
11		Л	булево OR (по краткой схеме)
1	!	П	булево унарное отрицание
	&= , = , ^= , &&= , =		специальное присваивание с соответствующей операцией

Операции булевой логики (продолжение)

Булевы AND и OR по краткой схеме (**&&, ||**) – не выполняется оценка правого операнда перед вычислением выражения

```
if (d !=0 && n/d >5) ... // при d=0 не будет  
исключения
```

```
if (c ==1 & k++ < 70) d=100; // операция  
// инкремента гарантированно выполнится
```

Цепочка присваивания

a=b=c=35;

Условная операция

<имя_поля> = <условие> ? <выраж1> : <выраж2>;

Если условие истинно, то полю присваивается результат выражения 1, в противном случае – выражения 2.

Многострочные в стиле языка C:

/ Любое количество любых строк.
Лишь бы там не было сочетания
звездочки и косой черты */*

Однострочные в стиле языка C++:

*// все написанное до конца строки –
комментарий*

Комментарии документатора :

*/** Многострочный комментарий,
* который войдет в
* программный документ */*