

Типы данных в Java: примитивы



Филипп
Воронов



Филипп Воронов

Teamlead, VK Group

 [@Филипп Воронов](#)



План занятия

1. [Примитивные и ссылочные типы данных](#)
2. [Целочисленные типы данных](#)
3. [Вещественные типы данных](#)
4. [Тип данных `boolean`](#)
5. [Тип данных `char`](#)
6. [Кодировки символов](#)



Примитивные и ссылочные типы данных



Типы данных в Java


- примитивные (Primitive Data Types);
- ссылочные (Reference Types).



Примитивные типы

Примитивные типы бывают четырех видов:

1. Целочисленный;
2. Вещественный;
3. Символьный;
4. Булевый.




Целочисленные типы данных



Целочисленные типы

1. `byte` (целые числа, 1 байт, -128 до 127)
2. `short` (целые числа, 2 байта, -32768 до 32767)
3. `int` (целые числа, 4 байта, -2147483648 до 2147483647)
4. `long` (целые числа, 8 байтов, -9223372036854775808 до 9223372036854775807)



Представление численных типов в отрицательном виде

Любое число в машинном виде представлено множеством бит, например:

цифра 2 выглядит 00000010 (1 байт)

Вопрос: Как вы думаете, как представлено число -2 в бинарном виде?

Представление численных типов в отрицательном виде

Ответ: Для представления отрицательных чисел в Java (как и во многих других языках программирования), используется понятие дополнительный код.

Чтобы перевести число в дополнительный код нужно:

1. Инвертировать каждый бит;
2. Прибавить к младшему разряду 1.

Пример: Число 2 в бинарном представлении 00000010

1. Инвертируем каждый bit -> 11111101;
2. Прибавим 1 -> 11111110 (дополнительный код).

Дополнительный код используется в вычислительной технике для быстрого вычисления.

Подчеркивание чисел

Визуально разряды чисел можно разделять символом подчеркивания `_`, он не заменяет запятую.

Пример:

```
int number1 = 1_234_000;  
long number2 = 1_000_000L;
```

Операции над целочисленными типами

Над целочисленными типами можно проводить все те же операции как в школьной алгебре:

- сложение;
- вычитание;
- деление;
- умножение.

Нужно запомнить что при сложении двух разных типов например: `byte` и `int` результат вычисления будет `int` (произойдет автоматическое приведение типов к большему).

Пример

Сложение:

```
int a = 10;  
long b = 2021L;  
long result = a + b; // 2031L
```

Вычитание:

```
int a = 500;  
long b = 400L;  
long result = a - b; // 100L
```

Инкремент/декремент:

```
int var1 = 0;  
System.out.println(var1);  
++var1;  
var1--;
```

Сложение byte

Как вы думаете скомпилируется ли следующая программа:

```
byte value1 = 120;  
byte value2 = 3;  
byte value3 = value1 + value2;
```

Сложение byte

Нет программа не скомпилируется, так спецификация языка защищает разработчика от переполнения типов.

Результат вычисления всех примитивных типов меньших `int`, автоматически рассчитываются в типе `int` и результат их вычисления будет тип `int`.

```
byte value1 = 120;  
byte value2 = 3;  
int value3 = value1 + value2; // исправим тип на int
```

Сложение `int`

Как вы думаете запустится ли программа?

```
long value4 = 1_000_000_000;  
long value5 = 3_000_000_000;  
long value6 = value4 + value5;
```


Сложение `int`

Нет, программа не запустится и не скомпилируется потому что:

1. Максимальное число которое, можно положить в тип `int` = 2147483647.
2. По умолчанию все примитивные типы без литерала являются типом `int`. Чтобы исправить проблему нужно добавить литерал типа — `L`.

```
long value4 = 1_000_000_000L;  
long value5 = 3_000_000_000L;  
long value6 = value4 + value5;
```

Деление `int`

Что будет результатом вычисления?

```
int value1 = 123;  
int value2 = 0;  
int value3 = value1 / value2;
```



Деление `int`

Результат вычисления `ArithmeticException` — деление на 0 невозможно.



Вещественные типы данных



Вещественные типы

В Java есть два примитивных типа с плавающей точкой:

1. `float` (вещественные числа, 4 байта, $-3.4E+38$ до $3.4E+38$)
2. `double` (вещественные числа, 8 байтов, $-1.7E+308$ до $1.7E+308$)

Используемый стандарт вещественных чисел IEEE Standard 754, подробнее можно почитать здесь: <http://steve.hollasch.net/cgindex/coding/ieeefloat.html>



Вещественные типы

Float 32 бита (4 байта)

- 23 бита мантисса (около 7 десятичных цифр);
- 8 бит экспонента;
- 1 бит знаковый.

Double 64 бит (8 байт)

- 52 бита используются для мантиссы (около 16 десятичных цифр);
- 11 бит экспонента;
- 1 бит знаковый.

Печать вещественных чисел в консоль

Для вывода вещественных чисел можно использовать метод `System.out.format`, в качестве аргумента нужно указать, сколько символов после запятой, оставить для вывода.

Пример:

```
System.out.format("%.2f", 0.257674);  
System.out.format("%.4f", 0.257674);
```

Операции над вещественными типами

Деление `float`

Какой результат вычисления?

```
float value1 = 123f;  
float value2 = 0f;  
float value3 = value1 / value2;  
  
float value4 = 0f;  
float value5 = 0f;  
float value6 = value4 / value5;
```




Операции над вещественными типами

Результат вычисления Infinity и NaN.

IEEE Standard 754

Представление Double в памяти

бит 0 : знаковый

биты 1 to 11 : экспонента

биты 12 to 63 : мантиса

NaN: 01111111111100

```
-Infinity: 11111111111000000000000000000000000000000000000000000000000000000000
```

```
+Infinity: 01111111111000000000000000000000000000000000000000000000000000000000
```

```
-Minimum: 111111111101111111111111111111111111111111111111111111111111111111
```

```
+Maximum: 0111111111011111111111111111111111111111111111111111111111111111
```



Тип данных `boolean`



Тип данных `boolean`

Может принимать значение `true` или `false`.



Тип данных `char`

Тип данных `char`

Первая таблица ASCII была создана в 1963 году, содержала всего 128 символов и выглядела следующим образом:

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Тип данных `char`

В 1991 году была создана международная универсальная таблица символов Unicode:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01E0	Ā	ā	Ē	ē	Ĝ	g	Ĝ	ĝ	Ķ	ķ	Ų	ų	Ų	ų	Ž	ž
01F0	ǰ	DZ	Dz	dz	Ć	ć	Н	р	Ñ	ñ	Á	á	Æ	æ	Ø	ø
0200	Ă	ă	Â	â	È	è	Ê	ê	Î	î	Î	î	Ï	ï	Ô	ô
0210	Ř	ř	Ŕ	ŕ	Û	û	Û	û	Ş	ş	Ţ	ţ	З	з	Ӧ	ӧ
0220	Ŋ	ŋ	8	8	Z	z	À	à	Е	е	Ö	ö	Ö	ö	Ó	ó
0230	Ō	ō	Ȳ	ȳ	ł	ł	ł	ł	Ѡ	ѡ	А	Б	В	Г	Д	Е
0240	Ƶ	ƶ	Ʒ	ƹ	ƺ	ƻ	Ƽ	ƾ	ƿ	ƽ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ
0250	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ
0260	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ
0270	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ	ƿ



Тип данных `char`

В Java для `char` используется кодировка Unicode, а для хранения одного Unicode-символа используется 2 байта (16 бит).

Диапазон допустимых значений — от 0 до 65536 (отрицательных значений не существует).

Пример объявления переменной `char`

```
// символьный литерал
char letter1 = 'N';

// код символа Unicode в десятичном исчислении
char letter2 = 78;

// '\uxxxx' - символ Unicode,
// где xxxx цифровой код символа Unicode в шестнадцатеричной форме
char letter3 = '\u0053';
```

Вопрос

Как вы думаете каким будет результат вычисления?

```
char value10 = 'A';  
char value11 = 'B';  
char value12 = value10 + value11;
```

Ответ

Программа не скомпилируется. Потому что тип `char` так же является примитивным и результат такого вычисления будет автоматически приведен к типу `int`:

```
char value10 = 'A';  
char value11 = 'B';  
int value12 = value10 + value11; // исправим тип
```

А что если я хочу получить результат символ? Нужно явно привести тип `int` к типу `char`:

```
char value10 = 'A';  
char value11 = 'B';  
char value12 = (char) (value10 + value11);
```



Кодировка символов

Кодировка символов

Кодирование символов это способ представления символов в числовом виде (зачастую в 8-и, 10-и или 16-и ричной системе счисления).

В Java как говорилось раннее применяется кодировка UTF-16, в unix-системах распространена UTF-8, в Windows CP-1251 или CP-1252.

Все ASCII символы в UTF-8 занимают 1 байт остальные от 2-х до 6, чаще 4-х байт. UTF позволяет использовать любые символы (хоть китайские), а CP-1251 только ASCII, кириллицу и еще 62 дополнительных.

Подробнее:

<https://ru.wikipedia.org/wiki/UTF-8> <https://ru.wikipedia.org/wiki/UTF-16>
<https://ru.wikipedia.org/wiki/Windows-1251>

Сравнение

Примитивные типы сравниваются по значению:

```
int a = 3;
int b = 4;
if (a == b) // TODO
if (a >= b) // TODO

char c = 'N';
char d = 'M';
if (d < c) // TODO
```



Как выбрать какой тип использовать в программе

Чтобы выбрать какой тип использовать нужно ответить на два вопроса:

1. Какой диапазон значений необходим для переменной?
2. Переменная будет хранить только целочисленные значения?

Использование заведомо больших типов приводит к избыточному использованию ресурсов процессора и оперативной памяти.

Вопрос

Какой тип использовать для переменной `int` или `long`?



Ответ

Если вы понимаете что диапазон значений переменной будет не больше 2 миллиардов, то смело можно использовать тип `int`.

В большинстве домашних задач достаточно типа `int`

Значения по умолчанию для примитивных типов

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Таблица примитивных типов данных

Целочисленные:

1. `byte` (целые числа, 1 байт, -128 до 127)
2. `short` (целые числа, 2 байта, -32768 до 32767)
3. `int` (целые числа, 4 байта, -2147483648 до 2147483647)
4. `long` (целые числа, 8 байт, -9223372036854775808 до 9223372036854775807)

Вещественные:

5. `float` (вещественные числа, 4 байта, -3.4E+38 до 3,4E+38)
6. `double` (вещественные числа, 8 байт, -1.7E+308 до 1.7E+308)

Другие:

7. `boolean` (значение истина/ложь, 1 байт, `true/false`)
8. `char` (символ Unicode, 2 байта, 1 Unicode символ)



Чему мы научились

- Узнали какие примитивные типы бывают в Java;
- Какие операции можно делать над примитивными типами;
- Выбирать нужный тип под задачу.

Полезные ссылки

- [Unicode:](#)
- [Дополнительный код:](#)
- [Float/Double:](#)
- [NaN.](#)



Пример слайда с домашним заданием

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Филипп Воронов

 [@Филипп Воронов](https://t.me/Филипп_Воронов)