

Примитивные типы в Java

Алексей Владыкин

19 сентября 2012

- 1 Прimitives and reference types
- 2 Type `boolean`
- 3 Type `char`
- 4 Integer types
- 5 Floating point types
- 6 Type conversion

- 1 Примитивные и ссылочные типы
- 2 Тип `boolean`
- 3 Тип `char`
- 4 Целочисленные типы
- 5 Вещественные типы
- 6 Преобразование типов

Примитивные типы

- boolean
- char
- byte, short, int, long
- float, double
- void

- Простые типы
- Зарезервированные ключевые слова языка
- Передаются по значению

```
int i = 4;
int j = i;
j++;
// j -> 5, i -> 4
```

Ссылочные типы

- Все остальные, кроме перечисленных на предыдущем слайде
- Являются объектами
- Передаются по ссылке

```
Dog thisDog = new Dog("Sparky");
Dog thatDog = thisDog;
thatDog.setName("Wolfie");
/*
 * thatDog.getName() -> "Wolfie"
 * thisDog.getName() -> "Wolfie"
 */
```

- 1 Примитивные и ссылочные типы
- 2 Тип boolean**
- 3 Тип char
- 4 Целочисленные типы
- 5 Вещественные типы
- 6 Преобразование типов

Логические значения

- Литералы: `false`, `true`
- Любое сравнение имеет тип `boolean`:
 `<` `>` `==`
 `<=` `>=` `!=`
- Нет преобразования между `boolean` и другими примитивными типами

Логические операции

and	&&	&	&=
or			=
xor	^		^=
not	!		

- && и || — вычисление по сокращенной схеме
- & и | — вычисление по полной схеме

Примеры

```
boolean a = true;  
boolean b = false;  
boolean c = (a ^ b) == (a != b);  
boolean d = c &= !b || a;
```

Примеры

```
int m = 0;
int n = 10;
if (m != 0 & n / m >= 1) {
    System.out.println("condition is true");
} else {
    System.out.println("condition is false");
}
```

java.lang.Boolean

- Класс-обертка для `boolean`
- `boolean` `parseBoolean(String)`
- `String` `toString(boolean)`

- 1 Прimitives and reference types
- 2 Тип boolean
- 3 Тип char**
- 4 Целочисленные типы
- 5 Вещественные типы
- 6 Преобразование типов

Символьные значения

- char — 16 бит, беззнаковый ($0..2^{16} - 1$)
- Представляет номер символа в кодировке Unicode
- Литералы:
 - символ в одинарных кавычках: 'a'
 - шестнадцатеричный код символа: '\u78bc'
 - спецпоследовательности: '\t', '\n', '\r', '\"', '\'', '\\'
- Свободно конвертируется в числовые типы и обратно

java.lang.Character

- Класс-обертка для char
- `boolean` `isLowerCase(char)`
- `boolean` `isUpperCase(char)`
- `boolean` `isDigit(char)`
- `boolean` `isLetter(char)`
- `char` `toLowerCase(char)`
- `char` `toUpperCase(char)`
- `int` `getNumericValue(char)`

- 1 Примитивные и ссылочные типы
- 2 Тип `boolean`
- 3 Тип `char`
- 4 Целочисленные типы**
- 5 Вещественные типы
- 6 Преобразование типов

Диапазоны значений

Тип	Бит	Диапазон
byte	8	$-128 .. + 127$
short	16	$-2^{15} .. + 2^{15} - 1$
int	32	$-2^{31} .. + 2^{31} - 1$
long	64	$-2^{63} .. + 2^{63} - 1$

- Размер фиксирован, одинаков для всех платформ
- Все типы знаковые, беззнаковых вариантов нет

Литералы

- Десятичное число: 123
- Восьмеричное число: 0123
- Шестнадцатеричное число: 0x123
- Двоичное число: 0b101 (с Java 7)

- С подчеркиванием: 123_456_789 (с Java 7)
- С суффиксом L для long

Арифметические операции

сложение	+	+=
вычитание	-	-=
умножение	*	*=
деление	/	/=
остаток	%	%=
инкремент	++	
декремент	--	

- Деление целочисленное
- $a == (a / b) * b + (a \% b)$

Особые случаи

- Деление на ноль — исключительная ситуация, бросается `ArithmeticException`
- Переполнение не является исключительной ситуацией, лишние старшие биты просто выкидываются

Побитовые операции

not	~	
and	&	&=
or		=
xor	^	^=
shr	>>	>>=
shr'	>>>	>>>=
shl	<<	<<=

- >> — арифметический сдвиг
- >>> — логический сдвиг

Классы-обертки

- `java.lang.Byte`
- `java.lang.Short`
- `java.lang.Integer`
- `java.lang.Long`
- `MIN_VALUE`
- `MAX_VALUE`
- `toString(typename)`
- `parseTypename(String)`
- `bitCount(typename)`
- `reverse(typename)`
- `reverseBytes(typename)`

Примеры

```
int i = 2_000_000_000;  
long j = 10_000_000_000L;  
long k = j++ + ++i;  
k /= i;
```

Примеры

```
int mask = 0xFF000000;  
int i = mask >>> 16;  
int j = mask >>> 24;  
int k = mask >>> 32;
```

```
int n = ~0;  
n >>>= 1;  
int m = ~n;
```

- 1 Примитивные и ссылочные типы
- 2 Тип `boolean`
- 3 Тип `char`
- 4 Целочисленные типы
- 5 Вещественные типы**
- 6 Преобразование типов

Тип	Бит	Знак	Мантисса	Экспонента
float	32	1	23	8
double	64	1	52	11

- Стандарт IEEE754
- Число в представлено в виде $\pm m \cdot 2^e$

Литералы

- Обычная запись: `-1.234`
- Экспоненциальная запись: `-123.4e-2` ($-123.4 \cdot 10^{-2}$)
- Шестнадцатеричная запись: `0xFFFFpFF` ($FFFF \cdot 2^{FF}$)
- С суффиксом типа:
 - `38f`
 - `3e19d`
 - `123.4e-2f`
 - `444.444d`

Операции

сложение	+	+=
вычитание	-	-=
умножение	*	*=
деление	/	/=
остаток	%	%=
инкремент	++	
декремент	--	

- $a == ((\text{long}) (a / b)) * b + (a \% b)$
- Побитовые операции не поддерживаются

Особые случаи

- Деление положительного числа на 0 дает $+\infty$
- Деление отрицательного числа на 0 дает $-\infty$
- Деление 0 на 0 дает NaN

- Переполнение дает $+\infty$ или $-\infty$,
в зависимости от направления

- Любая арифметическая операция с NaN дает NaN
- `NaN != NaN`

strictfp

- Java использует FPU для вычислений с плавающей точкой
- Регистры FPU могут быть шире 64 бит
- Результаты вычислений могут отличаться

- Модификатор `strictfp` включает режим строгой совместимости, результаты будут идентичны на любом процессоре

Классы-обертки

- `java.lang.Float`
- `java.lang.Double`
- `MIN_VALUE`
- `MAX_VALUE`
- `POSITIVE_INFINITY`
- `NEGATIVE_INFINITY`
- `NaN`
- `boolean isNaN(typename)`
- `toString(typename)`
- `parseTypename(String)`

- 1 Прimitives and reference types
- 2 Тип `boolean`
- 3 Тип `char`
- 4 Целочисленные типы
- 5 Вещественные типы
- 6 Преобразование типов**

Неявное преобразование типов

- Преобразование целочисленных типов в более емкие (byte → short → int → long)
- Преобразование char в int и long
- Преобразование целочисленные типов в типы с плавающей точкой (возможна потеря точности)

Явное преобразование типов

- Оператор приведения типа: (*typename*)
- При приведении более емкого целого типа к менее емкому старшие биты просто отбрасываются
- При приведении типа с плавающей точкой к целому типу дробная часть отбрасывается (никакого округления)
- Слишком большое дробное число при приведении к целому превращается в `MAX_VALUE` или `MIN_VALUE`
- Слишком большой `double` при приведении к `float` превращается в `Float.POSITIVE_INFINITY` или `Float.NEGATIVE_INFINITY`

Автоматическое расширение

- При вычислении выражения $(a @ b)$ аргументы `a` и `b` преобразовываются в числа, имеющие одинаковый тип:
 - если одно из чисел `double`, то в `double`;
 - иначе, если одно из чисел `float`, то в `float`;
 - иначе, если одно из чисел `long`, то в `long`;
 - иначе оба числа преобразуются в `int`.
- **Следствие:** Арифметическое выражение над `byte`, `short` или `char` имеет тип `int`, поэтому для присвоения результата обратно в `byte`, `short` или `char` понадобится явное приведение типа

Неявное приведение с потерей данных

- Сокращенная запись `var @= expr` раскрывается в `var = (typename) (var @ (expr))`
- Неявно срабатывает приведение типа, в том числе с потерей данных

Что сегодня узнали

- Что такое «примитивные типы»
- Какие примитивные типы есть в языке Java
- Как на Java производить логические и арифметические вычисления
- Какие есть подводные камни