

ПРИМИТИВНЫЕ ТИПЫ JAVA

ПРИМИТИВНЫЕ ТИПЫ

| | |
|-----------------|-------------------------------|
| void | «пустой» тип |
| boolean (1 бит) | логический тип |
| char (2 байта) | символьный тип |
| byte (1 байт) | целочисленные типы |
| short (2 байта) | |
| int (4 байта) | |
| long (8 байт) | |
| float (4 байта) | типы с плавающей точкой |
| double (8 байт) | |

- 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"
 */
```

ССЫЛОЧНЫЕ ТИПЫ

Все классы наследуются от `java.lang.Object`

Методы в классе `java.lang.Object`:

- `String` `toString()` – все объекты можно приводить к строке (как следствие – печатать на экран)
- `boolean` `equals(Object obj)` – используется для сравнения объектов
- `int` `hashCode()` – хеш-код объекта

...

boolean

ПРИМИТИВНЫЕ ТИПЫ JAVA

ЛОГИЧЕСКИЕ ЗНАЧЕНИЯ

- Литералы: false, true
- Любое сравнение имеет тип boolean:
 - < > ==
 - <= >= !=
- Нет преобразования между boolean и другими примитивными типами. Как следствие нельзя писать

```
int x = 3;  
if (x) {  
    ...  
}
```

НЕЛЬЗЯ

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;
```

```
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)`

- `boolean b = Boolean.parseBoolean("false");`
- `String str = Boolean.toString(b);`

char

ПРИМИТИВНЫЕ ТИПЫ JAVA

СИМВОЛЬНЫЕ ЗНАЧЕНИЯ

- `char` — 16 бит, беззнаковый ($0 \dots 2^{16} - 1$)
- 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 isWhiteSpace(char)
- boolean isLetter(char)
- char toLowerCase(char)
- char toUpperCase(char)
- int getNumericValue(char)

- char c = Character.toLowerCase('a')

Целочисленные типы

ПРИМИТИВНЫЕ ТИПЫ JAVA

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 и одинаков для всех платформ

Все типы знаковые, беззнаковых вариантов нет (не было до появления Java8 😊)

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

ОПЕРАЦИИ

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

Деление целочисленное
(для целых типов)
 $a == (a / b) * b + (a \% b)$

Деление на ноль — исключительная ситуация, бросается `ArithmeticException`

Переполнение не является исключительной ситуацией, лишние старшие биты просто выкидываются

КЛАССЫ-ОБЕРТКИ

Классы:

- `java.lang.Byte`
- `java.lang.Short`
- `java.lang.Integer`
- `java.lang.Long`

Методы в классах:

- `MIN_VALUE`
- `MAX_VALUE`
- `toString(typename)`
- `parsetypename(String)`

Вещественные типы

ПРИМИТИВНЫЕ ТИПЫ JAVA

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

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
- $\text{NaN} \neq \text{NaN}$

Преобразование типов

ПРИМИТИВНЫЕ ТИПЫ JAVA

НЕЯВНОЕ ПРЕОБРАЗОВАНИЕ ТИПОВ

Неявное преобразование возможно в случаях:

- 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

- Операторы приведения типа (typename):
 - `(int)`, `(char)`...
- 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 ТИПА С ПОТЕРЕЙ ДАННЫХ

- Сокращенная запись `var @= expr` раскрывается в `var = (typename) (var @ (expr))`
`@` - здесь любая операция (+, -, *...)
- Неявно срабатывает приведение типа, в том числе с потерей данных