Базовый синтаксис

Алексей Владыкин

Примитивные типы

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

Примитивные типы

```
int a = 100;
```

```
int b = a;
```

Ссылочные типы

```
String s = "Hello world!";
```

```
String t = s;
```

```
String u = null;
```

Тип boolean

```
boolean brushedTeethToday = true;
boolean haveDog = false;

boolean iKnowMath = 1 < 100;


boolean fromInt = 10;
```

Тип boolean

```
boolean haveSpareTime = !isBusy;

boolean canGoToPark =
        haveSpareTime && weatherIsGood;

boolean hadGoodTime =
        learnedJavaOnStepic || wentToPark;

boolean tastesGood =
        addedKetchup ^ addedHoney;
```

Тип boolean

```
value &= expression;
// value = value & expression;

value |= expression;
// value = value | expression;

value ^= expression;
// value = value ^ expression;
```

Целочисленные типы

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

Целочисленные типы

```
int decimal = 99;

int octal = 0755;

int hex = 0xFF;

int binary = 0b101;

int tenMillion = 10_000_000;

long tenBillion = 10_000_000_000L;
```

Арифметические операции

```
int sum = a + b;

int diff = a - b;

int mult = a * b;

int div = a / b;

int rem = a % b;

int inc = a++ + ++b;

int dec = --a - b--;
```

Переполнение

```
byte b = 127;
```

```
b++;
```

Побитовые операции

```
int neg = ~a;

int and = a & b;

int or = a | b;

int xor = a ^ b;

int arithmeticShiftRight = a >> b;

int logicalShiftRight = a >>> b;

int shiftLeft = a << b;
```

Тип char

- 16 бит, беззнаковый, $0 \,..\, 2^{16} - 1$
- Представляет номер символа в кодировке Unicode

Тип char

```
char literal = 'a';

char tab = '\t';

char lineFeed = '\n';

char carriageReturn = '\r';

char singleQuote = '\'';

char backslash = '\\';

char hex = '\u03A9';
```

Вещественные типы

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

Литералы

```
double simple = -1.234;

double exponential = -123.4e-2;

double hex = 0x1.Fp10;

float floatWithSuffix = 36.6f;

double doubleWithSuffix = 4d;
```

Арифметические операции

```
double sum = a + b;

double diff = a - b;

double mult = a * b;

double div = a / b;

double rem = a % b;

double inc = a++ + ++b;

double dec = --a - b--;
```

Особые случаи

```
double positiveInfinify = 1.0 / 0.0;

double negativeInfinify = -1.0 / 0.0;


double nan = 0.0 / 0.0;

boolean notEqualsItself = nan != nan;
```

Особые случаи

```
x + eps == x
```

```
0.1 + 0.1 + ... + 0.1 != 1
```

Класс Math

```
double s = Math.sin(Math.PI);

double q = Math.sqrt(16);

double r = Math.ceil(1.01);

int a = Math.abs(-13);

int m = Math.max(10, 20);
```

Длинная арифметика

```
BigInteger two = BigInteger.valueOf(2);

BigInteger powerOfTwo = two.pow(100);


BigDecimal one = BigDecimal.valueOf(1);

BigDecimal divisionResult =
        one.divide(new BigDecimal(powerOfTwo));
```

Преобразование типов

```
byte byteValue = 123;
short shortValue = byteValue;
int intValue = shortValue;
long longValue = intValue;

char charValue = '@';
int intFromChar = charValue;
long longFromChar = charValue;

double doubleFromFloat = 1f;

float floatFromLong = longValue;
double doubleFromInt = intValue;
```

Преобразование типов

```
int intValue = 1024;
byte byteValue = (byte) intValue;

double pi = 3.14;
int intFromDouble = (int) pi;

float largeFloat = 1e20f;
int intFromLargeFloat = (int) largeFloat;

double largeDouble = 1e100;
float floatFromLargeDouble = (float) largeDouble;
```

Автоматическое расширение

```
double doubleValue = 1d + 1f;

float floatValue = 1f * 1;

long longValue = 1L - '0';

byte a = 1;
byte b = 2;
byte c = (byte) (a + b);
```

Неявное приведение

```
byte a = 1;

a += 3;

// a = (byte) (a + 3);


byte b = -1;

b >>>= 7;

// b = (byte) (b >>> 7);
```

Классы-обертки

`boolean`	`Boolean`
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`char`	`Character`
`float`	`Float`
`double`	`Double`

Классы-обертки

```
int primitive = 0;

Integer reference = Integer.valueOf(primitive);

int backToPrimitive = reference.intValue();
```

Классы-обертки

```
Integer a = 1;

int b = a;


Integer c = 10;

Integer d = 10;

Integer e = c + d;
```

Конвертация в строку и обратно

```
long fromString = Long.parseLong("12345");

String fromLong = Long.toString(12345);

String concatenation = "area" + 51;
```

Полезные методы

```
short maxShortValue = Short.MAX_VALUE;

int bitCount = Integer.bitCount(123);

boolean isLetter = Character.isLetter('a');

float floatInfinity = Float.POSITIVE_INFINITY;

double doubleNaN = Double.NaN;

boolean isNaN = Double.isNaN(doubleNaN);
```

Объявление vs выделение памяти

```
BigInteger number;

number = new BigInteger("12345");
```

Массивы

```
int[] numbers;

String[] args;

boolean bits[];
```

Массивы

```
int[] numbers = new int[100];

String[] args = new String[1];

boolean[] bits = new boolean[0];
```

Массивы

```
int[] numbers = new int[] {1, 2, 3, 4, 5};

boolean[] bits = new boolean[] {true, false};

// this works only in variable declaration
char[] digits = {
        '0', '1', '2', '3', '4',
        '5', '6', '7', '8', '9'};
```

Массивы

```
int[] numbers = {1, 2, 3, 4, 5};

int arrayLength = numbers.length;

int firstNumber = numbers[0];

int lastNumber = numbers[arrayLength - 1];

int indexOutOfBounds = numbers[5];
```

Массивы

```
int[][] matrix1 = new int[2][2];

int[][] matrix2 = {{1, 2}, {3, 4}};
```

```
int[] firstRow = matrix2[0];

int someElement = matrix2[1][1];
```

Массивы

```
int[][] triangle = {
        {1, 2, 3, 4, 5},
        {6, 7, 8, 9},
        {10, 11, 12},
        {13, 14},
        {15}};

int secondRowLength = triangle[1].length;
```

Varargs

```
static int maxArray(int[] numbers) { ... }
```

```
static int maxVarargs(int... numbers) { ... }
```

Сравнение массивов

```
int[] a = {1, 2, 3};
int[] b = {4, 5, 6};

boolean equals1 = a == b;

boolean equals2 = a.equals(b);

boolean equals3 = Arrays.equals(a, b);

boolean equals4 = Arrays.deepEquals(a, b);
```

Как распечатать массив

```
int[] a = {1, 2, 3};

System.out.println(a);

System.out.println(Arrays.toString(a));

System.out.println(Arrays.deepToString(a));
```

Строки

```
String hello = "Hello";

String specialChars = "\r\n\t\"\\";

String empty = "";


char[] charArray = {'a', 'b', 'c'};

String string = new String(charArray);

char[] charsFromString = string.toCharArray();


String zeros = "\u0000\u0000";
```

Строки

```
String s = "stringIsImmutable";

int length = s.length();

char firstChar = s.charAt(0);

boolean endsWithTable = s.endsWith("table");

boolean containsIs = s.contains("Is");
```

Строки

```
String s = "stringIsImmutable";

String substring = s.substring(0, 6);

String afterReplace = s.replace("Imm", "M");

String allCapitals = s.toUpperCase();
```

Строки

```
String hello = "Hello ";
String world = "world!";
String helloWorld = hello + world;
```

```
StringBuilder sb = new StringBuilder();
sb.append(hello);
sb.append(world);
String helloWorld = sb.toString();
```

Строки

```
boolean referenceEquals = s1 == s2;

boolean contentEquals = s1.equals(s2);

boolean contentEqualsIgnoreCase =
        s1.equalsIgnoreCase(s2);
```

Оператор if

```
if (weatherIsGood) {
    walkInThePark();
} else {
    learnJavaOnStepic();
}
```

Оператор ?:

```
if (weatherIsGood) {
    System.out.println("Weather is good");
} else {
    System.out.println("Weather is bad");
}

// same effect, but much shorter
System.out.println("Weather is "
        + (weatherIsGood ? "good" : "bad"));
```

Оператор switch

```
switch (digit) {
    case 0:
        text = "zero";
        break;

    case 1:
        text = "one";
        break;

    // ...

    default:
        text = "???";
}
```

Цикл while

```
while (haveTime() && haveMoney()) {
    goShopping();
}
```

Цикл do while

```
do {
    goShopping();
} while (haveTime() && haveMoney());
```

Цикл for

```
for (int i = 0; i < args.length; i++) {
    System.out.println(args[i]);
}
```

Цикл foreach

```
for (String arg : args) {
    System.out.println(arg);
}
```

Оператор break

```
boolean found = false;
for (String element : haystack) {
    if (needle.equals(element)) {
        found = true;
        break;
    }
}
```

Оператор continue

```
int count = 0;
for (String element : haystack) {
    if (!needle.equals(element)) {
        continue;
    }
    count++;
}
```

Метки

```
boolean found = false;

outer:
for (int[] row : matrix) {
    for (int x : row) {
        if (x > 100) {
            found = true;
            break outer;
        }
    }
}
```

Методы

```
private static String getGreeting(String name) {
    if (name == null) {
        return "Hello anonymous!";
    } else {
        return "Hello " + name + "!";
    }
}
```