Язык Java

Глава II. Базовый синтаксис Java

Виталий Витальевич Перевощиков

Осенний семестр 2021

Содержание

1. Базовые конструкции языка Java
2. Переменные, литералы, выражения
3. Два типа данных в Java
4. Примитивные типы данных
5. Ссылочные типы данных
   - Массивы
   - Строки
   - Классы-обертки над примитивными типами

Базовый синтаксис: определение и присваивание переменных

```java
// Определение переменной
int numericValue;
String string;
boolean logicalValue;

// Присваивание переменной: <имя переменной> = <выражение>
string = "Name";                // литерал
numericValue = 2 * anotherValue + 1; // арифметическое выражение
logicalValue = true || false;          // логическое выражение
numericValue = string.length();      // метод, возвращающий значение

// Комбинация определения и присваивания:
int result = 100;
```

Базовый синтаксис: условные операторы

```
// Оператор if
if (logicalValue) {
    numericValue = 0;
} else {
    numericValue = 1;
}

// Оператор ?
numericValue = logicalValue ? 0 : 1;

// Оператор switch
switch (numericValue) {
    case 0:
        string = "ноль";
        break;
    case 1:
        string = "один";
        break;
    default:
        string = "несколько";
}
```

Базовые синтаксис: циклы

```
// цикл "for"
for (int i = 0; i < 500; ++i) {
    process(i);
}

// цикл "while"
int i = 0;
while (i < 500) {
    process(i);
    ++i;
}

// цикл "do-while"
i = 0;
do {
    process(i);
    ++i;
} while (i < 500);
```

Базовый синтаксис: методы (функции)

```java
// метод, возвращающий значение
int calculateFactorial(int number) {
    if (number == 0) {
        return 1;
    }
    return number * calculateFactorial(number - 1);
}

// метод, не возвращающий значение
void printNumber(int number) {
    System.out.println(number);
}
```

Переменные

- Определение переменной:
  ```
  <тип переменной> <имя переменной>;
  ```
- Примеры:
  - `int height;`
  - `boolean hasChildren;`
- Имя переменной:
  - Разрешенные символы
    - буквы `A...Z,a...z`
    - цифры `0...9`
    - специальные символы: `_`, `$`
  - Имя переменной зависит от регистра:
    - `height` и `hEIGHT` - разные переменные
  - Имя переменной не может быть ключевым словом Java (`int`, `for`, ...)
- Определение нескольких переменных одного типа:
  - `int height, weight;`

Соглашения об именах переменных

- “Camel case” – запись:
  - `firstName` ✓
  - `FirstName` ✗
  - `first_name` ✗
- Имя должно быть коротким, но содержательным:
  - `firstName` ✓
  - `firstNameOfThePerson` ✗
  - `z` ✗

Литералы (константы)

- Целочисленные
  - десятичные: `0`, `1`, ..., `42`, ...
  - двоичные: `0b110` (число $0 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 = 6$)
  - восьмеричные: `0107` (число $7 \cdot 8^0 + 0 \cdot 8^1 + 1 \cdot 8^2 = 71$)
  - шестандцатеричные: `0x1A`, `0x1a` (число $10 \cdot 16^0 + 1 \cdot 16^1 = 26$)
- Вещественные: `0.012`, `1.2e-2`, `1.2E-2`, `0.012f`
- Логические: `true`, `false`
- Символы: `'a'`, `'1'`, `'ы'`
- Строки: `"Hello"`
- Неопределенное значение: `null`

Выражения

- Выражения строятся из:
  - переменных
  - литералов
  - операторов:
    - арифметических (+, -, *, /, %, ...)
    - сравнения (==, !=, >, >=, ...)
    - логических (&&, ||, !, ...)
    - ...
  - методов (функций)
- Примеры корректных выражений:

```
int value = 2 * (1 + 3); // арифметическое выражение с литералами
int rectangleArea = height * width; // арифметическое выражение с переменными
boolean isNonEmptyRectangle = height > 0 && width > 0; // логическое выражение
int area = calculateRectangleArea(height, width); // функция, возвращающая значение
```

- Примеры некорректных выражений:

```
int height = 0;
height = height || 1; // корректно в JavaScript: 1
int value = true + 1; // корректно в JavaScript: 2
```

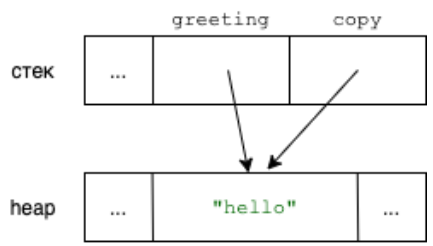
Примитивные типы данных

- Примитивные типы:
  - Целочисленные: `byte`, `short`, `int`, `long`
  - Вещественные: `float`, `double`
  - Символьный: `char`
  - Логический: `boolean`
- Свойства:
  - Базовые элементы для построения более сложных типов (например, классов)
  - Принимают значения из ограниченного диапазона
  - Значение переменной хранится на **стеке** (статическая память):

Ссылочные типы данных

- Ссылочные типы:
  - Классы, например:
    - Класс `String`
    - Классы-обертки над примитивными типами (`Integer`, `Boolean`, ...)
  - Массивы, например:
    - Одномерные массивы (`int[]`, `String[]`, ...)
    - Матрицы (`int[][]`, `String[][]`)
  - Интерфейсы, переменные типов
- Свойства:
  - Данные хранятся в динамической памяти (heap).
  - Значение переменной ссылочного типа - ссылка на участок памяти в heap'е:

```
String greeting = new String("hello");
String copy = greeting;
```



  - Нулевая ссылка: `null` (ссылка на heap отсутствует)

Целочисленные типы

Тип	Размер	Диапазон значений	Default
`byte`	1 байт	-128 ... 127	0
`short`	2 байта	-32768 ... 32767	0
`int`	4 байта	-2147483648 ... 2147483647	0
`long`	8 байтов	-2^{63} ... $2^{63}-1$	0

Операции над целыми числами:

- Арифметические: +, -, *, /, %, ++, --
- Побитовые: ~, &, |, ^, >>, >>>, <<
- Сравнение: ==, !=, <, <=, >, >=

Вещественные типы

Тип	Размер	Точность	Default
`float`	4 байта	до 7 знаков после запятой	`0.0`
`double`	8 байт	до 16 знаков после запятой	`0.0`

- Операции: арифметические, сравнение
- Класс `Math` (пакет `java.lang`) содержит полезные функции для вычислений с вещественными числами:

```
double log = Math.log(Math.E);       // 1.0
double cos = Math.cos(2 * Math.PI); // 1.0
double sqrt = Math.sqrt(16);         // 4.0
double floor = Math.floor(2.1);      // 2.0
double ceil = Math.ceil(2.1);        // 3.0
double max = Math.max(1.1, 2.2);     // 2.2
double abs = Math.abs(-2.1);         // 2.1
```

Логический тип boolean

Тип	Размер	Значения	Default
`boolean`	**1 байт!**	`true`, `false`	`false`

- Операции:
  - Логические: `&&`, `||`, `!`:

```
boolean and = true && false; // false
boolean or  = true || false; // true
boolean not = !true;         // false
```

  - Побитовые
  - Сравнение: `==`, `!=`

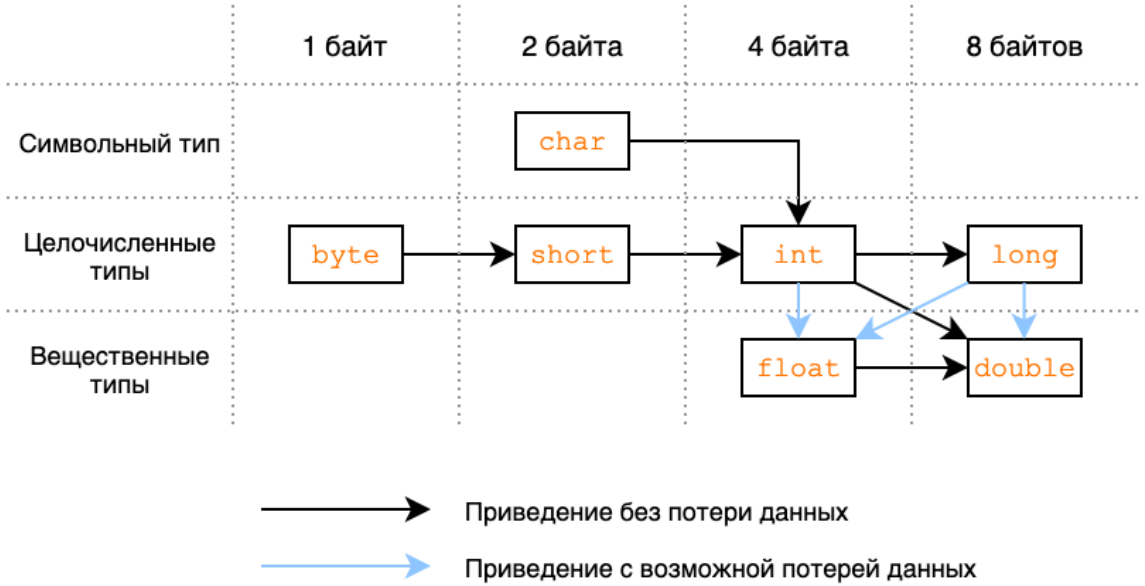
Символьный тип `char`

Тип	Размер	Значения	Default
`char`	2 байта	$0 \ldots 2^{16} - 1$ (номер символа в Unicode)	`'\u0000'`

- Операции: как для целых чисел
- Специальные символы:
  - Символ новой строки: `'\n'`
  - Backslash: `'\\'`
  - Одинарная кавычка: `'\''`
  - Символ табуляции: `'\t'`

Приведение типов: неявное расширение

- Правила расширения типов:



- Примеры:

```
byte byteValue = 1;
short shortValue = byteValue;
int intValue = shortValue;
long longValue = intValue;
```

Приведение типов: явное сужение

- **Проблема**: Возможно ли привести переменную к "меньшему" типу?

```
int intValue = 1;
byte byteValue = intValue;
```

- **Решение**: Явное приведение:

```
int intValue = 1;
byte byteValue = (byte) intValue;
```

- **Внимание**: Возможна потеря данных

```
int intValue = 1000;
byte byteValue = (byte) intValue; // -24
```

Приведение типов в выражениях

- Приведение к общему типу:

```
byte byteValue = 1;
int intValue = 2000;
int sum = byteValue + intValue;
```

- **Внимание**: Арифметические операции не сохраняют тип

```
byte firstByteValue = 1;
byte secondByteValue = 2;
// ошибка компиляции:
byte wrongByteSum = firstByteValue + secondByteValue;
// корректная запись:
int intSum = firstByteValue + secondByteValue;
// явное приведение к byte:
byte correctByteSum = (byte)(firstByteValue + secondByteValue);
```

- Пример необходимости явного приведения:

```
int one = 1;
int two = 2;
int div1 = one / two; // 0
double div2 = one / two; // 0.0
double div3 = (double)one / two; // 0.5
```

Ссылочные типы: массивы

- Объявление:

```
int[] integerArray;
String[] stringArray;
```

- Инициализация:
  - С помощью оператора `new`:

```
// массив длины 200
double[] array = new double[200];
double firstElement = array[0]; // 0.0
double lastElement = array[199]; // 0.0;
```

  - Перечислением элементов массива:

```
int[] array = new int[]{ 1, 2 };
int firstElement = array[0]; // 1
int secondElement = array[1]; // 2;
```

  - Короткая запись:

```
int[] array = { 1, 2 };
```

Массивы

- Длина массива:

```java
int[] array = { 1, 2, 3 };
int length = array.length; // 3
```

- Присваивание значения элементу массива:

```java
array[2] = 42;
```

- Обход массива:

```java
double[] array = { 1.1, 2.2, 3.3 };

// Вариант 1: цикл "for" по индексам элементов массива
for (int index = 0; index < array.length; ++index) {
    System.out.println(array[index]);
}

// Вариант 2: цикл "for-each" по элементам массива
for (double element : array) {
    System.out.println(element);
}
```

Матрицы

- Объявление и инициализация:
  - С помощью оператора `new`:

```
int numRows = 2; // количество строк
int numCols = 3; // количество столбцов
int[][] matrix = new int[numRows][numCols];
```

  - Перечислением элементов матрицы:

```
int[][] matrix = {
        {1, 2, 3}, // 1-я строка
        {4, 5, 6}  // 2-я строка
};
```

- Обращение к элементам матрицы:

```
int[][] matrix = {{1, 2, 3}, {4, 5, 6}};
int firstRowFirstColumn = matrix[0][0]; // 1
int firstRowLastColumn = matrix[0][2]; // 3
int lastRowFirstColumn = matrix[1][0]; // 4
int lastRowLastColumn = matrix[1][2]; // 6
```

Матрицы

- Присваивание значения элементу матрицы:

```
int row = 0;    // 1-я строка
int column = 1; // 2-й столбец
matrix[row][column] = 99;
```

- Обход матрицы (построчно сверху вниз):

```
// Вариант 1: двойной цикл "for" по индексам строк и столбцов матрицы
for (int rowIndex = 0; rowIndex < matrix.length; ++rowIndex) {
    for (int colIndex = 0; colIndex < matrix[rowIndex].length; ++colIndex) {
        System.out.print(matrix[rowIndex][colIndex] + " ");
    }
    System.out.println();
}

// Вариант 2: цикл "for-each" по строкам и вложенный цикл по элементам строк
for (int[] row : matrix) {
    for (int element : row) {
        System.out.print(element + " ");
    }
    System.out.println();
}
```

Строки

- Объявление и инициализация строк:
  - С помощью строкового литерала:

```
String firstName = "Vitaly";
```

  - С помощью оператора "new":

```
// Вариант 1: строковый литерал
String fromLiteral = new String("some string");

// Вариант 2: символьный массив
char[] charArray = { 'V', 'i', 't', 'a', 'l', 'y'};
String fromCharArray = new String(charArray);

// Вариант 3: пустая строка ""
String emptyString = new String();
```

- Длина строки:

```
int length = firstName.length(); // 6
```

- Обращение к символу в строке:

```
char firstLetter = firstName.charAt(0); // 'V'
char lastLetter = firstName.charAt(5); // 'y
```

Строки

- Присваивание значения символу строки:
  - **невозможно**, так как строки неизменяемые (immutable).
  - Для изменения строки необходимо создать новую строку
- Конвертация строк:

```
// Конвертация в массив символов:
char[] charArray = firstName.toCharArray(); // {'V','i','t','a','l','y'}
// Конвертация в целое число:
int number = Integer.parseInt("200"); // 200
int wrongNumber = Integer.parseInt(firstName); // генерация исключения!
// Конвертация в вещественное число:
double realNumber = Double.parseDouble("200.212"); // 200.212
```

- Обход строки:

```
// Вариант 1: цикл "for" по индексам символов строки
for (int index = 0; index < firstName.length(); ++index) {
    System.out.print(firstName.charAt(index));
}

// Вариант 2 (неэффективный): цикл "for-each" по символам строки
// Осторожно! Метод toCharArray создает копию символов строки
for (char symbol : firstName.toCharArray()) {
    System.out.println(symbol);
}
```

Сравнение строк

```
String string = "String";
String stringAlias = string;
String stringCopy = new String(string);
String anotherString = "Another string";
boolean result;

// == проверяет равенство ссылок
result = string == stringAlias;   // true
result = string == stringCopy;    // false
result = string == anotherString; // false

// метод equals проверяет равенство содержимого строк
// осторожно: если string = null, то будет сгенерировано исключение
result = string.equals(stringAlias);   // true
result = string.equals(stringCopy);    // true
result = string.equals(anotherString); // false
```

Конкатенация строк

- Оператор "+":

```
String firstName = "Иван";
String lastName = "Иванов";
String fullName = firstName + " " + lastName; // "Иван Иванов"
```

- **Проблема**: Неэффективность при конкатенации большого количества строк:

```
// strings = { "1", "2", ..., "1000000" }
String[] strings = new String[1000000];
for (int i = 0; i < strings.length; ++i) {
    strings[i] = Integer.toString(i);
}

// join = "1,2,...,1000000"
String join = "";
for (String str : strings) {
    join += str;
}
```

Конкатенация строк

- Метод `String.join`:

```
// join = "1,2,...,1000000"
String join = String.join(",", strings);
```

- Класс `StringBuilder` (пакет `java.lang`):

```
StringBuilder stringBuilder = new StringBuilder();
for (String item : strings) {
    stringBuilder.append(item);
    stringBuilder.append(",");
}
// join = "1,2,...,1000000"
String join = stringBuilder.toString();
```

Полезные функции над строками

```java
String string = "firstName";

// нижний/верхний регистры
String lowerCase = string.toLowerCase(); // "firstname"
String upperCase = string.toUpperCase(); // "FIRSTNAME"
// подстроки
String suffix = string.substring(2);  // "rstName"
String segment = string.substring(2, 5); // "rst"
// поиск подстрок
boolean startsWith = string.startsWith("fi"); // true
boolean contains = string.contains("stN"); // true
boolean endsWidth = string.endsWith("me"); // true
// замена подстрок
String updatedString = string.replace("first", "second"); // "secondName"
```

Классы-обертки над примитивными типами

Примитивный тип	Класс-обертка
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`
`boolean`	`Boolean`
`char`	`Character`

Классы-обертки

- Примеры:

```
// объявление
Integer wrapperInt;
// autoboxing
wrapperInt = 256;
// unboxing
int value = wrapperInt;
// операции
Integer result = wrapperInt + value;
```

- Нулевая ссылка:

```
Integer wrapperValue = null;
int primitiveValue = wrapperValue; // генерация исключения
```

- Полезные методы:

```
byte maxValue = Byte.MAX_VALUE; // 127
int bitCount = Integer.bitCount(4); // 1, т.к. 4 = 0b100
boolean isLetter = Character.isLetter('a'); // true
```