Введение в программирование на Java

Лекция 2. Примитивные типы данных. Базовые операции.

Виталий Олегович Афанасьев

18/19 ноября 2025

Примитивные типы данных в Java

- Целочисленные (`byte`, `short`, `int`, `long`)
- Вещественные (`float`, `double`)
- Логический (`boolean`)
- Символьный (`char`)

Целочисленные типы данных

Тип	Размер в байтах	Диапазон значений
byte		
short		
int		
long		

Целочисленные типы данных

Тип	Размер в байтах	Диапазон значений
byte	1	
short		
int		
long		

Целочисленные типы данных

Тип	Размер в байтах	Диапазон значений
byte	1	$[-2^7; 2^7 - 1]$ $[-128; 127]$
short		
int		
long		

Целочисленные типы данных

Тип	Размер в байтах	Диапазон значений
byte	1	$[-2^7; 2^7 - 1]$ $[-128; 127]$
short	2	
int		
long		

Целочисленные типы данных

Тип	Размер в байтах	Диапазон значений
byte	1	$[-2^7; 2^7 - 1]$ $[-128; 127]$
short	2	$[-2^{15}; 2^{15} - 1]$ $[-32\ 768; 32\ 767]$
int		
long		

Целочисленные типы данных

Тип	Размер в байтах	Диапазон значений
byte	1	$[-2^7; 2^7 - 1]$ $[-128; 127]$
short	2	$[-2^{15}; 2^{15} - 1]$ $[-32\ 768; 32\ 767]$
int	4	
long		

Целочисленные типы данных

Тип	Размер в байтах	Диапазон значений
byte	1	$[-2^7; 2^7 - 1]$ $[-128; 127]$
short	2	$[-2^{15}; 2^{15} - 1]$ $[-32\ 768; 32\ 767]$
int	4	$[-2^{31}; 2^{31} - 1]$ $[-2\ 147\ 483\ 648; 2\ 147\ 483\ 647]$
long		

Целочисленные типы данных

Тип	Размер в байтах	Диапазон значений
byte	1	$[-2^7; 2^7 - 1]$ $[-128; 127]$
short	2	$[-2^{15}; 2^{15} - 1]$ $[-32\ 768; 32\ 767]$
int	4	$[-2^{31}; 2^{31} - 1]$ $[-2\ 147\ 483\ 648; 2\ 147\ 483\ 647]$
long	8	

Целочисленные типы данных

Тип	Размер в байтах	Диапазон значений
byte	1	$[-2^7; 2^7 - 1]$ $[-128; 127]$
short	2	$[-2^{15}; 2^{15} - 1]$ $[-32\ 768; 32\ 767]$
int	4	$[-2^{31}; 2^{31} - 1]$ $[-2\ 147\ 483\ 648; 2\ 147\ 483\ 647]$
long	8	$[-2^{63}; 2^{63} - 1]$ $[-9\ 223\ 372\ 036\ 854\ 775\ 808; 9\ 223\ 372\ 036\ 854\ 775\ 807]$

Способы объявления переменных

```
1  int x;
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3  int y = 100500;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3  int y = 100500;
4  int z = x + y;
5
6
7
8
9
10
11
12
13
14
15
16
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3  int y = 100500;
4  int z = x + y;
5
6  int xAndY = x + y, xAndZ = x + z, yAndZ = y + z;
7
8
9
10
11
12
13
14
15
16
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3  int y = 100500;
4  int z = x + y;
5
6  int xAndY = x + y, xAndZ = x + z, yAndZ = y + z;
7
8  int someARBITRARY___name123 = 456;
9
10
11
12
13
14
15
16
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3  int y = 100500;
4  int z = x + y;
5
6  int xAndY = x + y, xAndZ = x + z, yAndZ = y + z;
7
8  int someARBITRARY___name123 = 456;
9  int camelCaseExampleName = 456;
10
11
12
13
14
15
16
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3  int y = 100500;
4  int z = x + y;
5
6  int xAndY = x + y, xAndZ = x + z, yAndZ = y + z;
7
8  int someARBITRARY___name123 = 456;
9  int camelCaseExampleName = 456;
10
11 long oneBillion = 1_000_000_000;
12
13
14
15
16
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3  int y = 100500;
4  int z = x + y;
5
6  int xAndY = x + y, xAndZ = x + z, yAndZ = y + z;
7
8  int someARBITRARY___name123 = 456;
9  int camelCaseExampleName = 456;
10
11 long oneBillion = 1_000_000_000;
12 long oneTrillion = 1_000_000_000_000; // Error
13
14
15
16
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3  int y = 100500;
4  int z = x + y;
5
6  int xAndY = x + y, xAndZ = x + z, yAndZ = y + z;
7
8  int someARBITRARY___name123 = 456;
9  int camelCaseExampleName = 456;
10
11 long oneBillion = 1_000_000_000;
12 long oneTrillion = 1_000_000_000_000; // Error
13 long oneTrillion = 1_000_000_000_000L;
14
15
16
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3  int y = 100500;
4  int z = x + y;
5
6  int xAndY = x + y, xAndZ = x + z, yAndZ = y + z;
7
8  int someARBITRARY___name123 = 456;
9  int camelCaseExampleName = 456;
10
11 long oneBillion = 1_000_000_000;
12 long oneTrillion = 1_000_000_000_000; // Error
13 long oneTrillion = 1_000_000_000_000L;
14
15 int binaryNumber = 0b0110;
16
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3  int y = 100500;
4  int z = x + y;
5
6  int xAndY = x + y, xAndZ = x + z, yAndZ = y + z;
7
8  int someARBITRARY___name123 = 456;
9  int camelCaseExampleName = 456;
10
11 long oneBillion = 1_000_000_000;
12 long oneTrillion = 1_000_000_000_000; // Error
13 long oneTrillion = 1_000_000_000_000L;
14
15 int binaryNumber = 0b0110;
16 int hexNumber = 0x5A;
17
```

Способы объявления переменных

```
1  int x;
2  x = 42;
3  int y = 100500;
4  int z = x + y;
5
6  int xAndY = x + y, xAndZ = x + z, yAndZ = y + z;
7
8  int someARBITRARY___name123 = 456;
9  int camelCaseExampleName = 456;
10
11 long oneBillion = 1_000_000_000;
12 long oneTrillion = 1_000_000_000_000; // Error
13 long oneTrillion = 1_000_000_000_000L;
14
15 int binaryNumber = 0b0110;
16 int hexNumber = 0x5A;
17 int octalNumber = 036;
```

Арифметические операции (1)

```
1 int sum = 10 + 3; // 13
2
3
4
5
```

Арифметические операции (1)

```
1 int sum = 10 + 3; // 13
2 int diff = 10 - 3; // 7
3
4
5
```

Арифметические операции (1)

```
1 int sum = 10 + 3; // 13
2 int diff = 10 - 3; // 7
3 int prod = 10 * 3; // 30
4
5
```

Арифметические операции (1)

```
1 int sum = 10 + 3; // 13
2 int diff = 10 - 3; // 7
3 int prod = 10 * 3; // 30
4 int div = 10 / 3; // 3
5
```

Арифметические операции (1)

```
1 int sum = 10 + 3; // 13
2 int diff = 10 - 3; // 7
3 int prod = 10 * 3; // 30
4 int div = 10 / 3; // 3
5 int mod = 10 % 3; // 1
```

Арифметические операции (2)

```
1  int x = 0;
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6
7
8
9
10
11
12
13
14
15
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6  y = y + 1;
7
8
9
10
11
12
13
14
15
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6  y = y + 1;
7  y += 1;
8
9
10
11
12
13
14
15
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6  y = y + 1;
7  y += 1;
8  y++;
9
10
11
12
13
14
15
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6  y = y + 1;
7  y += 1;
8  y++;
9  ++y;
10
11
12
13
14
15
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6  y = y + 1;
7  y += 1;
8  y++;
9  ++y;
10
11 y--;
12
13
14
15
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6  y = y + 1;
7  y += 1;
8  y++;
9  ++y;
10
11 y--;
12 --y;
13
14
15
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6  y = y + 1;
7  y += 1;
8  y++;
9  ++y;
10
11 y--;
12 --y;
13
14 int a = 1, b = 1;
15
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6  y = y + 1;
7  y += 1;
8  y++;
9  ++y;
10
11 y--;
12 --y;
13
14 int a = 1, b = 1;
15 int c = 10 + a++, d = 10 + ++b;
16
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6  y = y + 1;
7  y += 1;
8  y++;
9  ++y;
10
11 y--;
12 --y;
13
14 int a = 1, b = 1;
15 int c = 10 + a++, d = 10 + ++b;
16 // a = b = 2
17
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6  y = y + 1;
7  y += 1;
8  y++;
9  ++y;
10
11 y--;
12 --y;
13
14 int a = 1, b = 1;
15 int c = 10 + a++, d = 10 + ++b;
16 // a = b = 2
17 // c = 11
18
```

Арифметические операции (2)

```
1  int x = 0;
2  x = x + 100;
3  x += 100;
4
5  int y = 0;
6  y = y + 1;
7  y += 1;
8  y++;
9  ++y;
10
11 y--;
12 --y;
13
14 int a = 1, b = 1;
15 int c = 10 + a++, d = 10 + ++b;
16 // a = b = 2
17 // c = 11
18 // d = 12
```

Об использовании префиксных и постфиксных операций

Крайне не рекомендуется злоупотреблять префиксными и постфиксными операциями.

Существует **очень** ограниченный набор ситуаций, где такие операции будут читаемыми.

```
int x = 0;
x = x++ + ++x;
```

Вопрос: какое значение имеет переменная `x`?

Представление целых чисел в JVM (1)

```
byte x = 37;
```

Так как Java абстрагируется от конкретной аппаратуры, то **один байт** — это всегда **восемь бит**:

0	0	1	0	0	1	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Представление целых чисел в JVM (2)

```
byte x = -37;
```

Для отрицательных чисел используется **дополнительный код**.

Дополнительный код для отрицательного числа можно получить, инвертировав битовое представление модуля числа, а затем прибавив единицу.

37	0	0	1	0	0	1	0	1
~ 37	1	1	0	1	1	0	1	0
$\sim 37 + 1 = -37$	1	1	0	1	1	0	1	1

Таким образом, для знаковых чисел самый старший бит является **битом знака**: для отрицательных чисел он равен 1, а для неотрицательных — 0.

Представление целых чисел в JVM (3)

Вопросы на понимание:

1. Как выглядит **самое большое** число типа `byte` в двоичном представлении?
2. Как выглядит **самое маленькое** число типа `byte` в двоичном представлении?

Представление целых чисел в JVM (3)

Вопросы на понимание:

1. Как выглядит **самое большое** число типа `byte` в двоичном представлении?
2. Как выглядит **самое маленькое** число типа `byte` в двоичном представлении?

127	0	1	1	1	1	1	1	1

Представление целых чисел в JVM (3)

Вопросы на понимание:

1. Как выглядит **самое большое** число типа `byte` в двоичном представлении?
2. Как выглядит **самое маленькое** число типа `byte` в двоичном представлении?

127	0	1	1	1	1	1	1	1
-128	1	0	0	0	0	0	0	0

Битовые операции

```
1  byte x = 37;      // = 0b00100101
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

Битовые операции

```
1  byte x = 37;      // = 0b00100101
2  byte y = 43;      // = 0b00101011
3
4
5
6
7
8
9
10
11
12
13
14
15
```

Битовые операции

```
1  byte x = 37;      // = 0b00100101
2  byte y = 43;      // = 0b00101011
3
4  int and = x & y; // = 0b00100001
5
6
7
8
9
10
11
12
13
14
15
```

Битовые операции

```
1  byte x = 37;      // = 0b00100101
2  byte y = 43;      // = 0b00101011
3
4  int and = x & y; // = 0b00100001
5  int or  = x | y; // = 0b00101111
6
7
8
9
10
11
12
13
14
15
```

Битовые операции

```
1  byte x = 37;      // = 0b00100101
2  byte y = 43;      // = 0b00101011
3
4  int and = x & y; // = 0b00100001
5  int or  = x | y; // = 0b00101111
6  int xor = x ^ y; // = 0b00001110
7
8
9
10
11
12
13
14
15
```

Битовые операции

```
1  byte x = 37;      // = 0b00100101
2  byte y = 43;      // = 0b00101011
3
4  int and = x & y; // = 0b00100001
5  int or  = x | y; // = 0b00101111
6  int xor = x ^ y; // = 0b00001110
7
8  int comp = ~x; // = 0b11011010
9
10
11
12
13
14
15
```

Битовые операции

```
1  byte x = 37;      // = 0b00100101
2  byte y = 43;      // = 0b00101011
3
4  int and = x & y; // = 0b00100001
5  int or  = x | y; // = 0b00101111
6  int xor = x ^ y; // = 0b00001110
7
8  int comp = ~x; // = 0b11011010
9
10 int shl = x << 2; // = 0b10010100
11
12
13
14
15
```

Битовые операции

```
1  byte x = 37;      // = 0b00100101
2  byte y = 43;      // = 0b00101011
3
4  int and = x & y; // = 0b00100001
5  int or  = x | y; // = 0b00101111
6  int xor = x ^ y; // = 0b00001110
7
8  int comp = ~x; // = 0b11011010
9
10 int shl = x << 2; // = 0b10010100
11 int shr = x >> 2; // = 0b00001001
12
13
14
15
```

Битовые операции

```
1  byte x = 37;      // = 0b00100101
2  byte y = 43;      // = 0b00101011
3
4  int and = x & y; // = 0b00100001
5  int or  = x | y; // = 0b00101111
6  int xor = x ^ y; // = 0b00001110
7
8  int comp = ~x; // = 0b11011010
9
10 int shl = x << 2; // = 0b10010100
11 int shr = x >> 2; // = 0b00001001
12
13 int z = -2;           // = 0b11111111111111111111111111111110
14
15
```

Битовые операции

```
1  byte x = 37;      // = 0b00100101
2  byte y = 43;      // = 0b00101011
3
4  int and = x & y; // = 0b00100001
5  int or  = x | y; // = 0b00101111
6  int xor = x ^ y; // = 0b00001110
7
8  int comp = ~x; // = 0b11011010
9
10 int shl = x << 2; // = 0b10010100
11 int shr = x >> 2; // = 0b00001001
12
13 int z = -2;           // = 0b11111111111111111111111111111110
14 int shrZ  = z >>  1; // = 0b11111111111111111111111111111111
15
```

Битовые операции

```
1  byte x = 37;      // = 0b00100101
2  byte y = 43;      // = 0b00101011
3
4  int and = x & y; // = 0b00100001
5  int or  = x | y; // = 0b00101111
6  int xor = x ^ y; // = 0b00001110
7
8  int comp = ~x; // = 0b11011010
9
10 int shl = x << 2; // = 0b10010100
11 int shr = x >> 2; // = 0b00001001
12
13 int z = -2;           // = 0b11111111111111111111111111111110
14 int shrZ  = z >>  1; // = 0b11111111111111111111111111111111
15 int ushrZ = z >>> 1; // = 0b01111111111111111111111111111111
```

Логический тип данных

`boolean` — тип, который имеет только два значения: `true` и `false`

Операции сравнения

```
1  boolean eqTrue  = (2 + 2) == 4; // true
2
3
4
5
6
7
8
9
10
```

Операции сравнения

```
1  boolean eqTrue  = (2 + 2) == 4; // true
2  boolean eqFalse = (2 + 2) == 5; // false
3
4
5
6
7
8
9
10
```

Операции сравнения

```
1  boolean eqTrue  = (2 + 2) == 4; // true
2  boolean eqFalse = (2 + 2) == 5; // false
3
4  boolean neqTrue  = (2 + 2) != 4; // false
5
6
7
8
9
10
```

Операции сравнения

```
1  boolean eqTrue  = (2 + 2) == 4; // true
2  boolean eqFalse = (2 + 2) == 5; // false
3
4  boolean neqTrue  = (2 + 2) != 4; // false
5  boolean neqFalse = (2 + 2) != 5; // true
6
7
8
9
10
```

Операции сравнения

```
1  boolean eqTrue  = (2 + 2) == 4; // true
2  boolean eqFalse = (2 + 2) == 5; // false
3
4  boolean neqTrue  = (2 + 2) != 4; // false
5  boolean neqFalse = (2 + 2) != 5; // true
6
7  boolean less      = 2 < 3; // true
8
9
10
```

Операции сравнения

```
1  boolean eqTrue  = (2 + 2) == 4; // true
2  boolean eqFalse = (2 + 2) == 5; // false
3
4  boolean neqTrue  = (2 + 2) != 4; // false
5  boolean neqFalse = (2 + 2) != 5; // true
6
7  boolean less      = 2 < 3; // true
8  boolean lessEq    = 3 <= 3; // true
9
10
```

Операции сравнения

```
1  boolean eqTrue  = (2 + 2) == 4; // true
2  boolean eqFalse = (2 + 2) == 5; // false
3
4  boolean neqTrue  = (2 + 2) != 4; // false
5  boolean neqFalse = (2 + 2) != 5; // true
6
7  boolean less      = 2 < 3; // true
8  boolean lessEq    = 3 <= 3; // true
9  boolean greater   = 4 > 3; // true
10
```

Операции сравнения

```
1  boolean eqTrue  = (2 + 2) == 4; // true
2  boolean eqFalse = (2 + 2) == 5; // false
3
4  boolean neqTrue  = (2 + 2) != 4; // false
5  boolean neqFalse = (2 + 2) != 5; // true
6
7  boolean less      = 2 < 3; // true
8  boolean lessEq    = 3 <= 3; // true
9  boolean greater   = 4 > 3; // true
10 boolean greaterEq = 3 >= 3; // true
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

Логические операции

```java
boolean tandt = true  && true;  // true
boolean tandf = true  && false; // false
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2  boolean tandf = true  && false; // false
3  boolean fandt = false && true;  // false
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2  boolean tandf = true  && false; // false
3  boolean fandt = false && true;  // false
4  boolean fandf = false && false; // false
5
6
7
8
9
10
11
12
13
14
15
16
17
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2  boolean tandf = true  && false; // false
3  boolean fandt = false && true;  // false
4  boolean fandf = false && false; // false
5
6  boolean tort = true  || true;  // true
7
8
9
10
11
12
13
14
15
16
17
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2  boolean tandf = true  && false; // false
3  boolean fandt = false && true;  // false
4  boolean fandf = false && false; // false
5
6  boolean tort = true  || true;  // true
7  boolean torf = true  || false; // true
8
9
10
11
12
13
14
15
16
17
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2  boolean tandf = true  && false; // false
3  boolean fandt = false && true;  // false
4  boolean fandf = false && false; // false
5
6  boolean tort = true  || true;  // true
7  boolean torf = true  || false; // true
8  boolean fort = false || true;  // true
9
10
11
12
13
14
15
16
17
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2  boolean tandf = true  && false; // false
3  boolean fandt = false && true;  // false
4  boolean fandf = false && false; // false
5
6  boolean tort = true  || true;  // true
7  boolean torf = true  || false; // true
8  boolean fort = false || true;  // true
9  boolean forf = false || false; // false
10
11
12
13
14
15
16
17
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2  boolean tandf = true  && false; // false
3  boolean fandt = false && true;  // false
4  boolean fandf = false && false; // false
5
6  boolean tort = true  || true;  // true
7  boolean torf = true  || false; // true
8  boolean fort = false || true;  // true
9  boolean forf = false || false; // false
10
11 boolean nott = !true;  // false
12
13
14
15
16
17
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2  boolean tandf = true  && false; // false
3  boolean fandt = false && true;  // false
4  boolean fandf = false && false; // false
5
6  boolean tort = true  || true;  // true
7  boolean torf = true  || false; // true
8  boolean fort = false || true;  // true
9  boolean forf = false || false; // false
10
11 boolean nott = !true;  // false
12 boolean notf = !false; // true
13
14
15
16
17
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2  boolean tandf = true  && false; // false
3  boolean fandt = false && true;  // false
4  boolean fandf = false && false; // false
5
6  boolean tort = true  || true;  // true
7  boolean torf = true  || false; // true
8  boolean fort = false || true;  // true
9  boolean forf = false || false; // false
10
11 boolean nott = !true;  // false
12 boolean notf = !false; // true
13
14 boolean oroverand = true || false && false;   // true
15
16
17
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2  boolean tandf = true  && false; // false
3  boolean fandt = false && true;  // false
4  boolean fandf = false && false; // false
5
6  boolean tort = true  || true;  // true
7  boolean torf = true  || false; // true
8  boolean fort = false || true;  // true
9  boolean forf = false || false; // false
10
11 boolean nott = !true;  // false
12 boolean notf = !false; // true
13
14 boolean oroverand = true || false && false;   // true
15 boolean andoveror = (true || false) && false; // false
16
17
```

Логические операции

```
1  boolean tandt = true  && true;  // true
2  boolean tandf = true  && false; // false
3  boolean fandt = false && true;  // false
4  boolean fandf = false && false; // false
5
6  boolean tort = true  || true;  // true
7  boolean torf = true  || false; // true
8  boolean fort = false || true;  // true
9  boolean forf = false || false; // false
10
11 boolean nott = !true;  // false
12 boolean notf = !false; // true
13
14 boolean oroverand = true || false && false;   // true
15 boolean andoveror = (true || false) && false; // false
16
17 boolean example = !(2 > 3) && (1 + 2 == 1 + 1 + 1);
```

Вещественные типы данных

Для вещественных чисел используется формат, описанный в стандарте IEEE754 [1].

Тип	Размер в байтах	Диапазон значений
float	4	$\pm 3.40282347 \cdot 10^{38}$ (6-7 значимых цифр)
double	8	$\pm 1.7976931348623157 \cdot 10^{308}$ (15 значимых цифр)

[1] Для поверхностного ознакомления рекомендую следующий материал: https://neerc.ifmo.ru/wiki/index.php?title=Представление_вещественных_чисел

Операции с вещественными числами

```
1  double x = 0.25;
2
3
4
5
6
7
8
9
10
```

Операции с вещественными числами

```
1  double x = 0.25;
2  x = x + 0.75; // 1.0
3
4
5
6
7
8
9
10
```

Операции с вещественными числами

```
1  double x = 0.25;
2  x = x + 0.75; // 1.0
3
4  float y = 0.2; // Error
5
6
7
8
9
10
```

Операции с вещественными числами

```
1  double x = 0.25;
2  x = x + 0.75; // 1.0
3
4  float y = 0.2; // Error
5  float y = 0.2f;
6
7
8
9
10
```

Операции с вещественными числами

```
1  double x = 0.25;
2  x = x + 0.75; // 1.0
3
4  float y = 0.2; // Error
5  float y = 0.2f;
6
7  double idiv = 10 / 4;   // 2.0
8
9
10
```

Операции с вещественными числами

```
1  double x = 0.25;
2  x = x + 0.75; // 1.0
3
4  float y = 0.2; // Error
5  float y = 0.2f;
6
7  double idiv = 10 / 4;   // 2.0
8  double fdiv = 10 / 4.0; // 2.5
9
10
```

Операции с вещественными числами

```
1  double x = 0.25;
2  x = x + 0.75; // 1.0
3
4  float y = 0.2; // Error
5  float y = 0.2f;
6
7  double idiv = 10 / 4;   // 2.0
8  double fdiv = 10 / 4.0; // 2.5
9
10 double z = 3E-2; // 3 * 10^(-2) = 0.03
```

"Проблемы" с вещественными типами данных (1)

Вещественные числа не такие точные, как хотелось бы:

```
double x = 0.3;
double y = 0.1 + 0.2;
boolean eq = x == y; // false
System.out.println(x); // 0.3
System.out.println(y); // 0.30000000000000004
```

Рекомендую к изучению: 0.30000000000000004.com.

”Проблемы” с вещественными типами данных (2)

Более того, операции с ними не ассоциативны:

```
double x = 0.7 + (0.2 + 0.1);
double y = (0.7 + 0.2) + 0.1;
boolean eq = x == y; // false
System.out.println(x); // 1.0
System.out.println(y); // 0.9999999999999999
```

”Проблемы” с вещественными типами данных (2)

Более того, операции с ними не ассоциативны:

```
double x = 0.7 + (0.2 + 0.1);
double y = (0.7 + 0.2) + 0.1;
boolean eq = x == y; // false
System.out.println(x); // 1.0
System.out.println(y); // 0.9999999999999999
```

Вывод: используйте типы `float` и `double` только если без них **ТОЧНО** невозможно обойтись.

Символьный тип данных

`char` — тип данных для представления символов. Имеет размер в 2 байта и принимает значения от 0 до 65535.

По своей сути, это беззнаковый аналог целочисленного типа `short`, но с небольшими особенностями.

Операции с символами

```
1 char x = 'A';
2
3
4
5
6
7
8
```

Операции с символами

```
1 char x = 'A';
2 System.out.println(x); // A
3
4
5
6
7
8
```

Операции с символами

```
1 char x = 'A';
2 System.out.println(x); // A
3
4 char y = 65;
5
6
7
8
```

Операции с символами

```
1 char x = 'A';
2 System.out.println(x); // A
3
4 char y = 65;
5 System.out.println(y); // A
6
7
8
```

Операции с символами

```
1 char x = 'A';
2 System.out.println(x); // A
3
4 char y = 65;
5 System.out.println(y); // A
6
7 char z = 'A' + 1;
8
```

Операции с символами

```java
char x = 'A';
System.out.println(x); // A

char y = 65;
System.out.println(y); // A

char z = 'A' + 1;
System.out.println(z); // B
```

Приведение типов (1)

Иногда бывает необходимо приводить данные одного типа к другому типу.

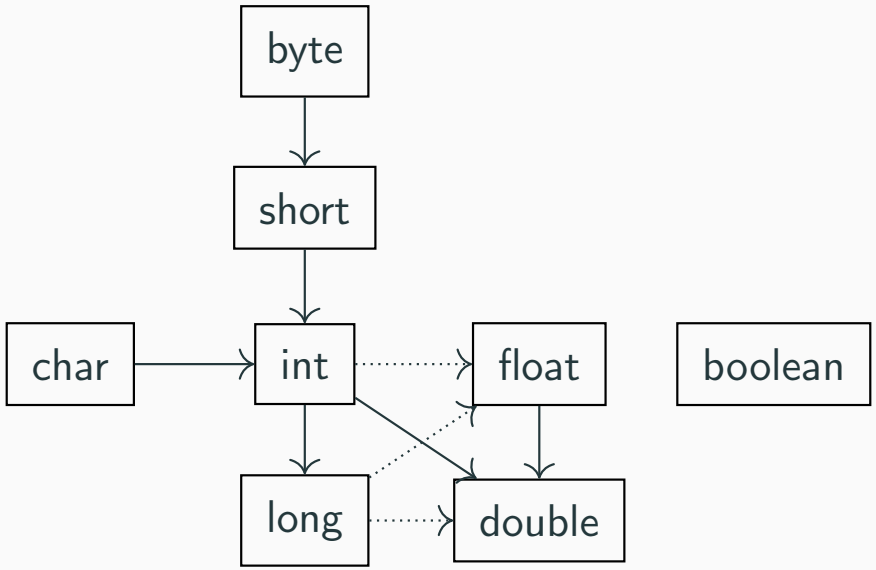
Например, когда значение переменной типа `long` лежит в границах от -128 до 127 и вы хотите сохранить это значение в переменную типа `byte`.

Сделать это автоматически не получится:

```
long x = 10;
byte y = x; // Error: possible lossy conversion from long to byte
```

Приведение типов (2)

Для некоторых ситуаций в Java разрешены неявные приведения типов:



Пример:

```
int x = 123;
long y = x;
```

Приведение типов (3)

Во всех остальных случаях необходимо пользоваться явным приведением:

```
long x = 10;
byte y = (byte) x;
```

Приведение типов (4)

Если значение слишком большое для результирующего типа данных, то старшие биты отбрасываются:

```
short x = 1000;      // = 0b00000011_11101000
byte y = (byte) x; //  =           0b11101000
System.out.println(y); // -24
```

Приведение типов (4)

Если значение слишком большое для результирующего типа данных, то старшие биты отбрасываются:

```
short x = 1000;      // = 0b00000011_11101000
byte y = (byte) x; //  =           0b11101000
System.out.println(y); // -24
```

Ситуация, при которой вычисленное значение выходит за границы, представимые целочисленным типом данных, называют **переполнением**:

```
int x = 2_147_483_647;
x = x + 1;
System.out.println(x); // -2_147_483_648
```

Приведение типов (5)

При приведении из вещественного типа в целый, часть после запятой отбрасывается:

```
double x = 12.34;
int y = (int) x;
System.out.println(y); // 12
```

Константы (1)

Для объявления констант используется ключевое слово `final`.

Переменным с данным модификатором можно присвоить значение лишь единожды:

```
1 public class Main {
2     public static void main(String[] args){
3         final int ultimateAnswer = 42;
4         ultimateAnswer = 12345; // Error
5     }
6 }
```

Константы (2)

Для определения констант на уровне класса, а не метода, пока что будем использовать модификаторы `public static final`:

```
1 public class Main {
2     public static final int ULTIMATE_ANSWER = 42;
3     public static void main(String[] args){
4         ULTIMATE_ANSWER = 12345; // Error
5     }
6 }
```

Вспомогательные операции из класса Math

Иногда основных операций бывает недостаточно, поэтому в стандартной библиотеке определён класс Math со вспомогательными операциями:

```
1  double pow   = Math.pow(2, 4);   // 16.0
2
3
4
5
6
7
8
9
10
11
```

Вспомогательные операции из класса Math

Иногда основных операций бывает недостаточно, поэтому в стандартной библиотеке определён класс Math со вспомогательными операциями:

```
1  double pow   = Math.pow(2, 4);   // 16.0
2  double log   = Math.log(3);      // 1.0986122886681098
3
4
5
6
7
8
9
10
11
```

Вспомогательные операции из класса Math

Иногда основных операций бывает недостаточно, поэтому в стандартной библиотеке определён класс Math со вспомогательными операциями:

```
1  double pow   = Math.pow(2, 4);   // 16.0
2  double log   = Math.log(3);      // 1.0986122886681098
3  double log10 = Math.log10(1000); // 3.0
4
5
6
7
8
9
10
11
```

Вспомогательные операции из класса Math

Иногда основных операций бывает недостаточно, поэтому в стандартной библиотеке определён класс Math со вспомогательными операциями:

```
1  double pow   = Math.pow(2, 4);   // 16.0
2  double log   = Math.log(3);      // 1.0986122886681098
3  double log10 = Math.log10(1000); // 3.0
4  int round    = Math.round(3.6);  // 4
5
6
7
8
9
10
11
```

Вспомогательные операции из класса Math

Иногда основных операций бывает недостаточно, поэтому в стандартной библиотеке определён класс Math со вспомогательными операциями:

```
1  double pow   = Math.pow(2, 4);   // 16.0
2  double log   = Math.log(3);      // 1.0986122886681098
3  double log10 = Math.log10(1000); // 3.0
4  int round    = Math.round(3.6);  // 4
5  int abs      = Math.abs(-1);     // 1
6
7
8
9
10
11
```

Вспомогательные операции из класса Math

Иногда основных операций бывает недостаточно, поэтому в стандартной библиотеке определён класс Math со вспомогательными операциями:

```
1  double pow   = Math.pow(2, 4);   // 16.0
2  double log   = Math.log(3);      // 1.0986122886681098
3  double log10 = Math.log10(1000); // 3.0
4  int round    = Math.round(3.6);  // 4
5  int abs      = Math.abs(-1);     // 1
6  double sin   = Math.sin(0);      // 0.0
7
8
9
10
11
```

Вспомогательные операции из класса Math

Иногда основных операций бывает недостаточно, поэтому в стандартной библиотеке определён класс Math со вспомогательными операциями:

```
1  double pow   = Math.pow(2, 4);   // 16.0
2  double log   = Math.log(3);      // 1.0986122886681098
3  double log10 = Math.log10(1000); // 3.0
4  int round    = Math.round(3.6);  // 4
5  int abs      = Math.abs(-1);     // 1
6  double sin   = Math.sin(0);      // 0.0
7  double cos   = Math.cos(0);      // 1.0
8
9
10
11
```

Вспомогательные операции из класса Math

Иногда основных операций бывает недостаточно, поэтому в стандартной библиотеке определён класс Math со вспомогательными операциями:

```
1  double pow   = Math.pow(2, 4);   // 16.0
2  double log   = Math.log(3);      // 1.0986122886681098
3  double log10 = Math.log10(1000); // 3.0
4  int round    = Math.round(3.6);  // 4
5  int abs      = Math.abs(-1);     // 1
6  double sin   = Math.sin(0);      // 0.0
7  double cos   = Math.cos(0);      // 1.0
8  double tan   = Math.tan(0);      // 0.0
9
10
11
```

Вспомогательные операции из класса Math

Иногда основных операций бывает недостаточно, поэтому в стандартной библиотеке определён класс Math со вспомогательными операциями:

```
1  double pow   = Math.pow(2, 4);   // 16.0
2  double log   = Math.log(3);      // 1.0986122886681098
3  double log10 = Math.log10(1000); // 3.0
4  int round    = Math.round(3.6);  // 4
5  int abs      = Math.abs(-1);     // 1
6  double sin   = Math.sin(0);      // 0.0
7  double cos   = Math.cos(0);      // 1.0
8  double tan   = Math.tan(0);      // 0.0
9  double asin  = Math.asin(1);     // 1.5707963267948966
10
11
```

Вспомогательные операции из класса Math

Иногда основных операций бывает недостаточно, поэтому в стандартной библиотеке определён класс Math со вспомогательными операциями:

```
1  double pow   = Math.pow(2, 4);   // 16.0
2  double log   = Math.log(3);      // 1.0986122886681098
3  double log10 = Math.log10(1000); // 3.0
4  int round    = Math.round(3.6);  // 4
5  int abs      = Math.abs(-1);     // 1
6  double sin   = Math.sin(0);      // 0.0
7  double cos   = Math.cos(0);      // 1.0
8  double tan   = Math.tan(0);      // 0.0
9  double asin  = Math.asin(1);     // 1.5707963267948966
10 double acos  = Math.acos(1);     // 0.0
11
```

Вспомогательные операции из класса Math

Иногда основных операций бывает недостаточно, поэтому в стандартной библиотеке определён класс Math со вспомогательными операциями:

```
1  double pow   = Math.pow(2, 4);   // 16.0
2  double log   = Math.log(3);      // 1.0986122886681098
3  double log10 = Math.log10(1000); // 3.0
4  int round    = Math.round(3.6);  // 4
5  int abs      = Math.abs(-1);     // 1
6  double sin   = Math.sin(0);      // 0.0
7  double cos   = Math.cos(0);      // 1.0
8  double tan   = Math.tan(0);      // 0.0
9  double asin  = Math.asin(1);     // 1.5707963267948966
10 double acos  = Math.acos(1);     // 0.0
11 double atan  = Math.atan(1);     // 0.7853981633974483
```

Вспомогательные константы из класса Math

Также в этом классе есть полезные константы:

```
1 System.out.println(Math.PI);  // 3.141592653589793
2
3
```

Вспомогательные константы из класса Math

Также в этом классе есть полезные константы:

```
1 System.out.println(Math.PI);  // 3.141592653589793
2 System.out.println(Math.TAU); // 6.283185307179586
3
```

Вспомогательные константы из класса Math

Также в этом классе есть полезные константы:

```
1 System.out.println(Math.PI);  // 3.141592653589793
2 System.out.println(Math.TAU); // 6.283185307179586
3 System.out.println(Math.E);   // 2.718281828459045
```