

Специализация:
данные и
функции





Специализация: данные и функции



На прошлом уроке

- 📌 Краткая история
- 📌 Базовый инструментарий
- 📌 Что нужно скачать
- 📌 JDK, JRE, JVM
- 📌 Структура проекта
- 📌 CLI: сборка, пакеты, запуск
- 📌 Javadoc
- 📌 Makefile, Docker





Что будет на уроке сегодня

- 📌 Данные
- 📌 Примитивные типы данных
- 📌 Ссылочные типы данных, массивы
- 📌 Базовый функционал языка
- 📌 Функции





Данные

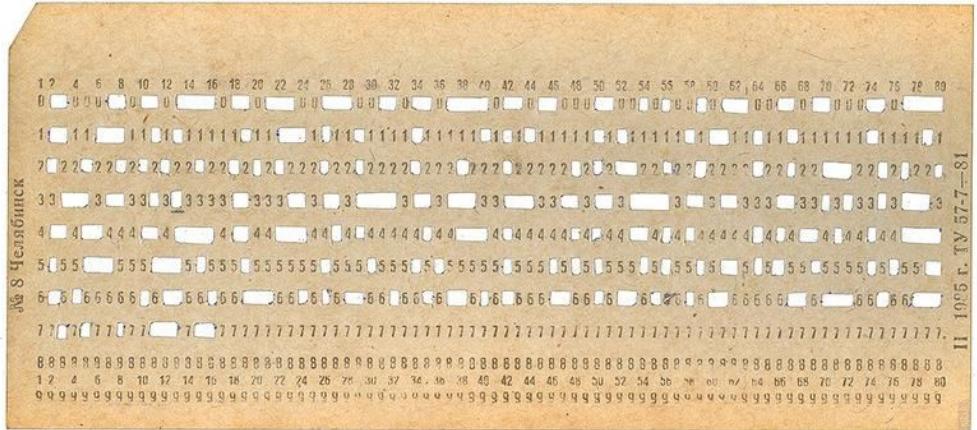


Типизация языков программирования





Безтиповые языки





Типизация Java

Java является языком со **строгой** (также можно встретить термин «**сильной**»)
явной статической типизацией



Типизация языков

Язык	Типизация		
Java	Статическая	Сильная	Явная
C#4.0	Статическая	Сильная	Неявная
C	Статическая	Слабая	Явная
C++	Статическая	Слабая	Явная
Python	Динамическая	Сильная	Неявная
JavaScript	Динамическая	Слабая	Неявная
PHP	Динамическая	Слабая	Неявная





Основные типы данных

Тип	Пояснение	Диапазон
byte	Самый маленький из адресуемых типов, 8 бит, знаковый	[- 128, +127]
short	Тип короткого целого числа, 16 бит, знаковый	[-32 768, +32 767]
char	Целочисленный тип для хранения символов в кодировке UTF-8, 16 бит, беззнаковый	[0, +65 535]
int	Основной тип целого числа, 32 бита, знаковый	[- 2 147 483 648, +2 147 483 647]
long	Тип длинного целого числа, 64 бита, знаковый	[- 9 223 372 036 854 775 808, +9 223 372 036 854 775 807]
float	Тип вещественного числа с плавающей запятой (одинарной точности, 32 бита)	
double	Тип вещественного числа с плавающей запятой (двойной точности, 64 бита)	
boolean	Логический тип данных	true, false



Антипаттерн “магические числа”



3

6

14

18

72

45

21



Начальное значение

Начальное значение - значение по-умолчанию, т.е. значение, которое можно прочитать из примитива, если ничего ему не присвоить.





Примитивные типы данных



Основные типы данных

Тип	Пояснение	Диапазон
byte	Самый маленький из адресуемых типов, 8 бит, знаковый	[- 128, +127]
short	Тип короткого целого числа, 16 бит, знаковый	[-32 768, +32 767]
char	Целочисленный тип для хранения символов в кодировке UTF-8, 16 бит, беззнаковый	[0, +65 535]
int	Основной тип целого числа, 32 бита, знаковый	[- 2 147 483 648, +2 147 483 647]
long	Тип длинного целого числа, 64 бита, знаковый	[- 9 223 372 036 854 775 808, +9 223 372 036 854 775 807]
float	Тип вещественного числа с плавающей запятой (одинарной точности, 32 бита)	
double	Тип вещественного числа с плавающей запятой (двойной точности, 64 бита)	
boolean	Логический тип данных	true, false



Переполнение переменной

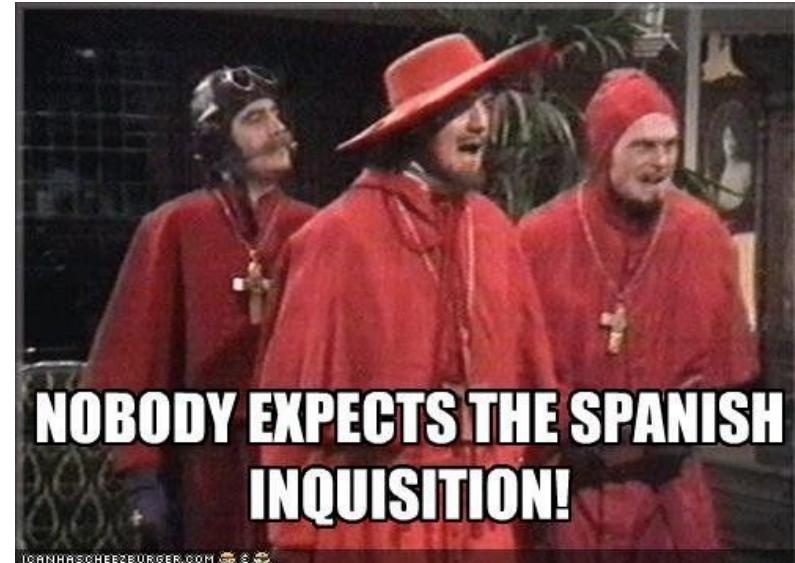




Переполнение переменной

```
9     byte b0 = 100;  
10    byte b1 = 200;
```

Required type: byte
Provided: int
Cast to 'byte' More actions...



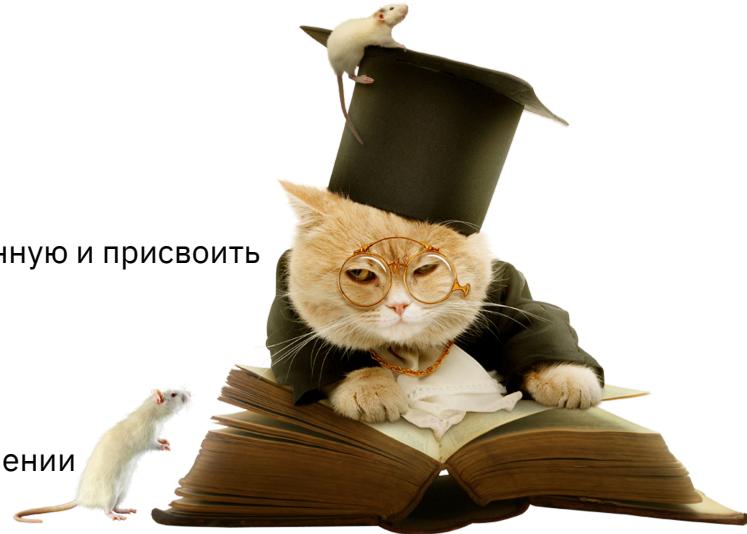
Теперь ваша очередь!



Ответьте на вопросы сообщением в чат

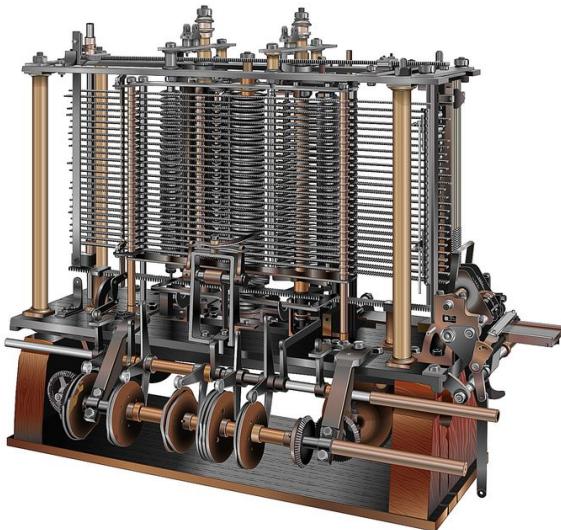
Вопросы:

1. Возможно ли объявить в Java целочисленную переменную и присвоить ей дробное значение?
2. Магическое число - это:
 - a. числовая константа без пояснений;
 - b. число, помогающее в вычислениях;
 - c. числовая константа, присваиваемая при объявлении переменной.
3. Переполнение переменной - это:
 - a. слишком длинное название переменной;
 - b. слишком большое значение переменной;
 - c. расширение переменной вследствие записи большого значения.





Бинарное (битовое) представление





Двоичная система счисления



Бинарное представление

Десятичная	Двоичная
00	00000
01	00001
02	00010
03	00011
04	00100
05	00101
06	00110
07	00111
08	01000

Десятичная	Двоичная
09	01001
10	01010
11	01011
12	01100
13	01101
14	01110
15	01111
16	10000



Бинарное представление

Десятичная	Двоичная	Шестнадцатиричная
00	00000	0x00
01	00001	0x01
02	00010	0x02
03	00011	0x03
04	00100	0x04
05	00101	0x05
06	00110	0x06
07	00111	0x07
08	01000	0x08

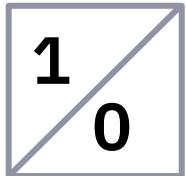


Десятичная	Двоичная	Шестнадцатиричная
09	01001	0x09
10	01010	0x0a
11	01011	0x0b
12	01100	0x0c
13	01101	0x0d
14	01110	0x0e
15	01111	0x0f
16	10000	0x10

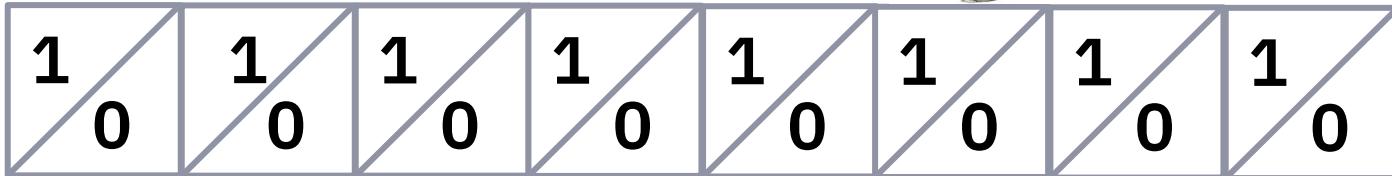


Хранение информации

бит



байт





Один байт данных

байт = 8 бит, [-128, 127]

S

0	1	1	1	1	1	1	1	= 127
---	---	---	---	---	---	---	---	-------





Отрицательные числа

00000000 = 0

~ 11111110 = -1

00000001 = 1

~ 11111110 = -2

00000010 = 2

~ 11111101 = -3





Основные типы данных

Тип	Пояснение	Диапазон
byte	Самый маленький из адресуемых типов, 8 бит, знаковый	[- 128, +127]
short	Тип короткого целого числа, 16 бит, знаковый	[-32 768, +32 767]
char	Целочисленный тип для хранения символов в кодировке UTF-8, 16 бит, беззнаковый	[0, +65 535]
int	Основной тип целого числа, 32 бита, знаковый	[- 2 147 483 648, +2 147 483 647]
long	Тип длинного целого числа, 64 бита, знаковый	[- 9 223 372 036 854 775 808, +9 223 372 036 854 775 807]
float	Тип вещественного числа с плавающей запятой (одинарной точности, 32 бита)	
double	Тип вещественного числа с плавающей запятой (двойной точности, 64 бита)	
boolean	Логический тип данных	true, false



Переполнение переменной

	?	?	?	?	?	?	?	?
1	1	1	1	1	1	1	1	1

Переполнение



Теперь ваша очередь!



Ответьте на вопросы сообщением в чат

Вопросы:

1. Возможно ли число 3000000000 (3 миллиарда) записать в двоичном представлении?
2. Как вы думаете, почему шестнадцатеричная система счисления вытеснила восьмеричную?





Основные типы данных

Тип	Пояснение	Диапазон
byte	Самый маленький из адресуемых типов, 8 бит, знаковый	[- 128, +127]
short	Тип короткого целого числа, 16 бит, знаковый	[-32 768, +32 767]
char	Целочисленный тип для хранения символов в кодировке UTF-8, 16 бит, беззнаковый	[0, +65 535]
int	Основной тип целого числа, 32 бита, знаковый	[- 2 147 483 648, +2 147 483 647]
long	Тип длинного целого числа, 64 бита, знаковый	[- 9 223 372 036 854 775 808, +9 223 372 036 854 775 807]
float	Тип вещественного числа с плавающей запятой (одинарной точности, 32 бита)	
double	Тип вещественного числа с плавающей запятой (двойной точности, 64 бита)	
boolean	Логический тип данных	true, false



Значения по умолчанию

```
9      byte b0 = 100;  
10     byte b1 = 200;  
11  
12     Required type: byte :  
13     Provided: int  
14  
15     Cast to 'byte' ↗↔ More actions... ↗↔
```



Значения по умолчанию

```
9      byte b0 = 100;  
10     byte b1 = 200;  
11     long l0 = 5_000_000_000;  
12  
13
```

Integer number too large



Решение проблемы строгой типизации

```
9      byte b0 = 100;
10     byte b1 = 200;
11     long l0 = 5_000_000_000;
12     long l1 = 5_000_000_000L;
13     float f0 = 0.123;
14     float f1 = 0.123f;
```

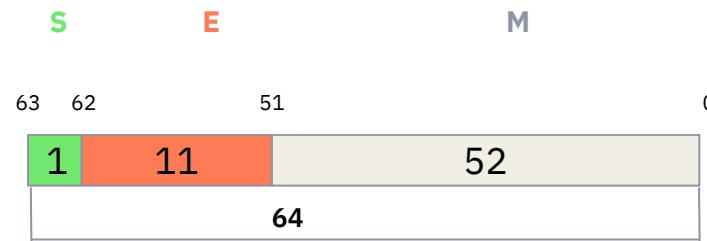
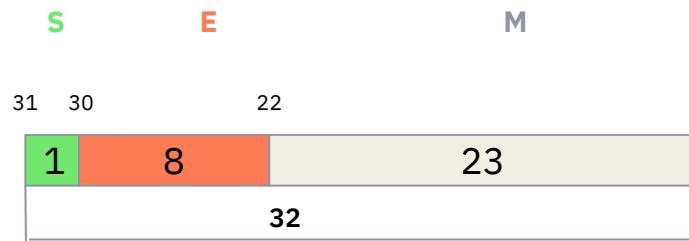


Основные типы данных

Тип	Пояснение	Диапазон
byte	Самый маленький из адресуемых типов, 8 бит, знаковый	[- 128, +127]
short	Тип короткого целого числа, 16 бит, знаковый	[-32 768, +32 767]
char	Целочисленный тип для хранения символов в кодировке UTF-8, 16 бит, беззнаковый	[0, +65 535]
int	Основной тип целого числа, 32 бита, знаковый	[- 2 147 483 648, +2 147 483 647]
long	Тип длинного целого числа, 64 бита, знаковый	[- 9 223 372 036 854 775 808, +9 223 372 036 854 775 807]
float	Тип вещественного числа с плавающей запятой (одинарной точности, 32 бита)	
double	Тип вещественного числа с плавающей запятой (двойной точности, 64 бита)	
boolean	Логический тип данных	true, false



Форматы с плавающей запятой



S- Sign (знак числа)

E- Exponent (8 или 11 бит смещённого на 127 или 1023 порядков числа)

M-Mantissa (23 или 52 бита мантиссы, дробная часть числа)



Пример вычисления

$$+0,5 = 2^{-1}$$

0_01111110_00000000000000000000000000000000

знак = 0

порядок = 126

мантиssa = 0

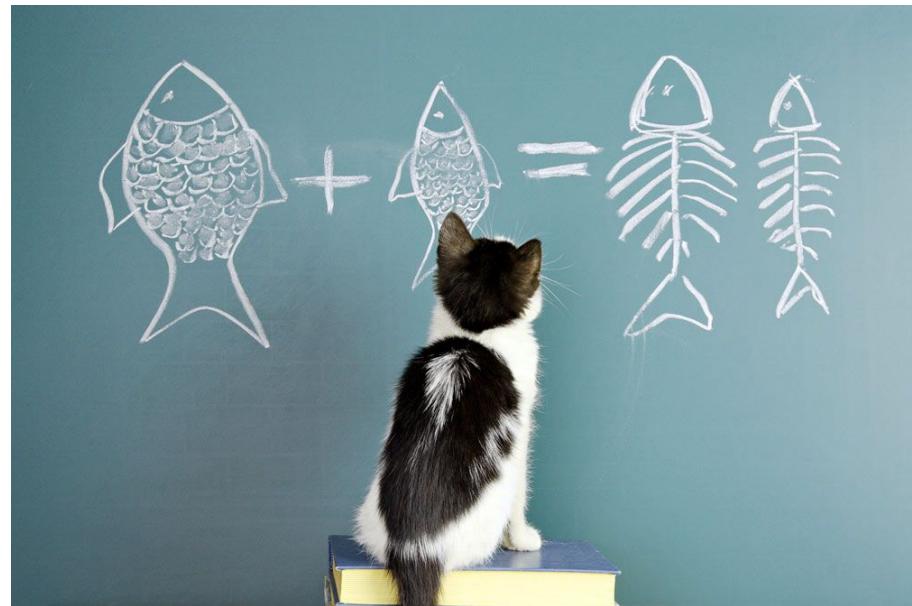
$$-1^0 \times 2^{-1} \times (1 + 0) = 0,5$$





Пример вычисления

-0,15625





Отрицательные степени числа 2

$$2^1 = 2$$

$$2^0 = 1.0$$

$$2^{-1} = 0.5$$

$$2^{-2} = 0.25$$

$$2^{-3} = 0.125$$

$$2^{-4} = 0.0625$$

$$2^{-5} = 0.03125$$

$$2^{-6} = 0.015625$$

$$2^{-7} = 0.0078125$$

$$2^{-8} = 0.00390625$$





Пример вычисления и результат

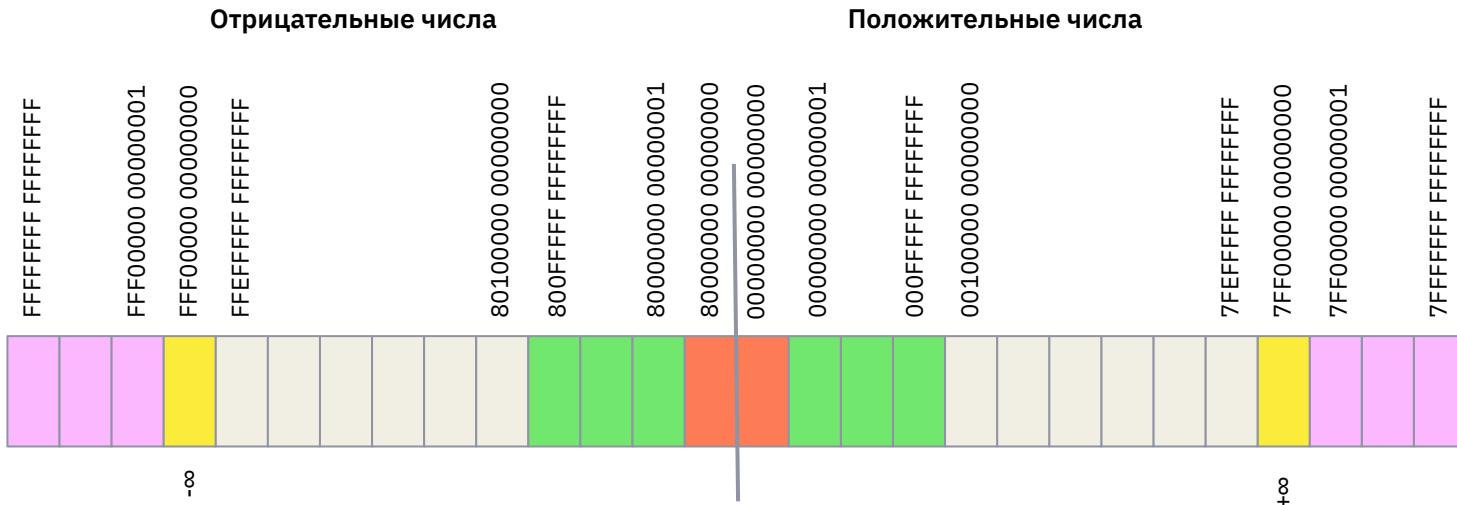
$$(-1)^1 \times 2^{(124-127)} \times (1 + 2097152/2^{23}) = -0,15652$$

$$\begin{aligned} (-1)^1 \times 1,01e-3 &= 1 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5} = \\ &= 1 \times 0,125 + 0 \times 0,0625 + 1 \times 0,03125 = \\ &= 0,125 + 0,03125 = -0,15625 \end{aligned}$$





Особенности чисел с плавающей запятой

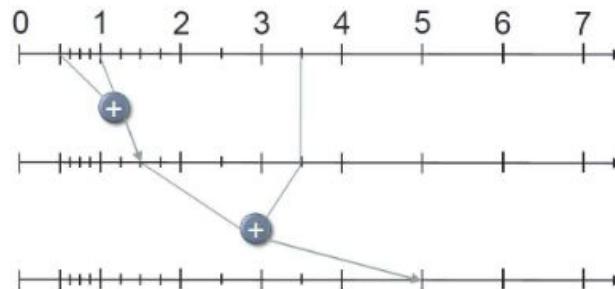


- нулевые числа
 - денормализованные числа
 - нормализованные числа
 - бесконечность
 - не числа (NANs)

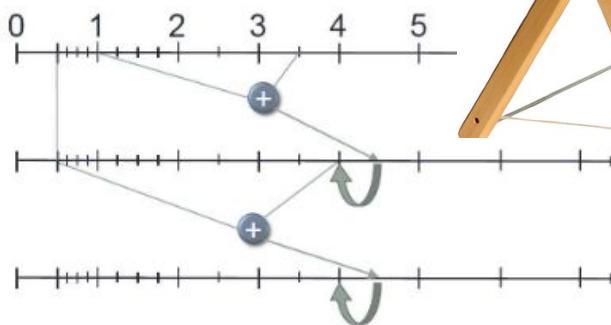




Пример вычислений



$$(0,5 + 1,0) + 3,5 = 5$$



$$0,5 + (1,0 + 3,5) = 4$$



Теперь ваша очередь!



Ответьте на вопросы сообщением в чат

Вопросы:

1. Сколько байт данных занимает самый большой целочисленный тип?
2. Почему нельзя напрямую сравнивать целочисленные данные и числа с плавающей запятой, даже если там точно лежит число без дробной части?
3. Внутри переполненной переменной остаётся значение:
 - a. переданное - максимальное для типа;
 - b. максимальное для типа;
 - c. не определено





Основные типы данных

Тип	Пояснение	Диапазон
byte	Самый маленький из адресуемых типов, 8 бит, знаковый	[- 128, +127]
short	Тип короткого целого числа, 16 бит, знаковый	[-32 768, +32 767]
char	Целочисленный тип для хранения символов в кодировке UTF-8, 16 бит, беззнаковый	[0, +65 535]
int	Основной тип целого числа, 32 бита, знаковый	[- 2 147 483 648, +2 147 483 647]
long	Тип длинного целого числа, 64 бита, знаковый	[- 9 223 372 036 854 775 808, +9 223 372 036 854 775 807]
float	Тип вещественного числа с плавающей запятой (одинарной точности, 32 бита)	
double	Тип вещественного числа с плавающей запятой (двойной точности, 64 бита)	
boolean	Логический тип данных	true, false



UTF-8

dec	hex	val	dec	hex	val	dec	hex	val	dec	hex	val
000	0x00	(nul)	032	0x20	□	064	0x40	@	096	0x60	'
001	0x01	(soh)	033	0x21	!	065	0x41	A	097	0x61	a
002	0x02	(stx)	034	0x22	"	066	0x42	B	098	0x62	b
003	0x03	(etx)	035	0x23	#	067	0x43	C	099	0x63	c
004	0x04	(eot)	036	0x24	\$	068	0x44	D	100	0x64	d
005	0x05	(enq)	037	0x25	%	069	0x45	E	101	0x65	e
006	0x06	(ack)	038	0x26	&	070	0x46	F	102	0x66	f
007	0x07	(bel)	039	0x27	'	071	0x47	G	103	0x67	g
008	0x08	(bs)	040	0x28	(072	0x48	H	104	0x68	h
009	0x09	(tab)	041	0x29)	073	0x49	I	105	0x69	i
010	0x0A	(lf)	042	0x2A	*	074	0x4A	J	106	0x6A	j
011	0x0B	(vt)	043	0x2B	+	075	0x4B	K	107	0x6B	k
012	0x0C	(np)	044	0x2C	,	076	0x4C	L	108	0x6C	l
013	0x0D	(cr)	045	0x2D	-	077	0x4D	M	109	0x6D	m
014	0x0E	(so)	046	0x2E	.	078	0x4E	N	110	0x6E	n
015	0x0F	(si)	047	0x2F	/	079	0x4F	O	111	0x6F	o
016	0x10	(dle)	048	0x30	0	080	0x50	P	112	0x70	p
017	0x11	(dc1)	049	0x31	1	081	0x51	Q	113	0x71	q
018	0x12	(dc2)	050	0x32	2	082	0x52	R	114	0x72	r
019	0x13	(dc3)	051	0x33	3	083	0x53	S	115	0x73	s
020	0x14	(dc4)	052	0x34	4	084	0x54	T	116	0x74	t
021	0x15	(nak)	053	0x35	5	085	0x55	U	117	0x75	u
022	0x16	(syn)	054	0x36	6	086	0x56	V	118	0x76	v
023	0x17	(etb)	055	0x37	7	087	0x57	W	119	0x77	w
024	0x18	(can)	056	0x38	8	088	0x58	X	120	0x78	x
025	0x19	(em)	057	0x39	9	089	0x59	Y	121	0x79	y
026	0x1A	(eof)	058	0x3A	:	090	0x5A	Z	122	0x7A	z
027	0x1B	(esc)	059	0x3B	;	091	0x5B	[123	0x7B	{
028	0x1C	(fs)	060	0x3C	<	092	0x5C	\	124	0x7C	
029	0x1D	(gs)	061	0x3D	=	093	0x5D]	125	0x7D	}
030	0x1E	(rs)	062	0x3E	>	094	0x5E	^	126	0x7E	~
031	0x1F	(us)	063	0x3F	?	095	0x5F	_	127	0x7F	\DEL





Строгая типизация и кавычки

```
16     char c0 = 72;
17     char c1 = 'H';
18     char c2 = "H";
19     String s0 = "H";
20     String s1 = 'H';
```

Required type: String

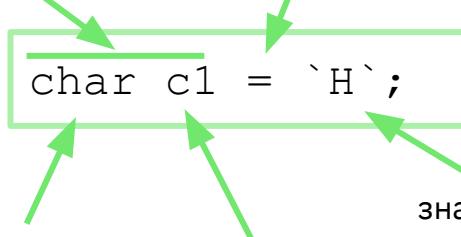
Provided: char

Wrap using 'String.valueOf()' Alt+Shift+Enter



Инициализация переменной

```
16      char c0;  
17      c0 = 'H';  
18      char c1 = 'H';
```

определение присваивание

тип идентификатор значение

инициализация

КАЖЕТСЯ Я
ВСЁ ЗАПИСАЛ





Преобразование типов

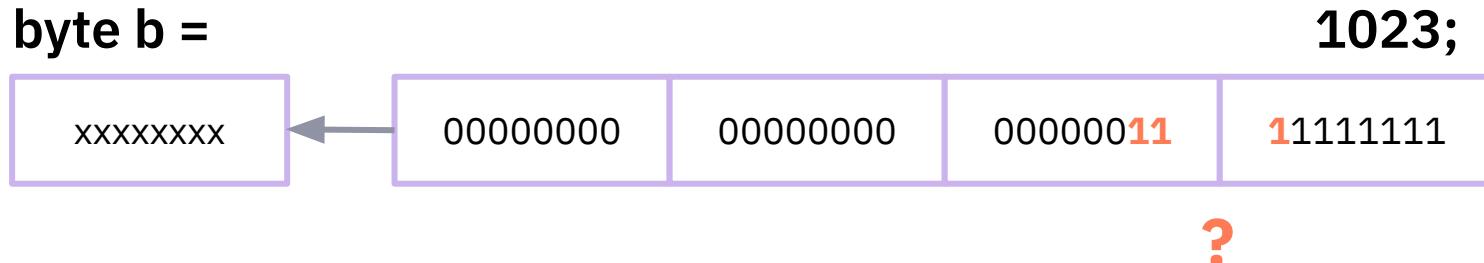
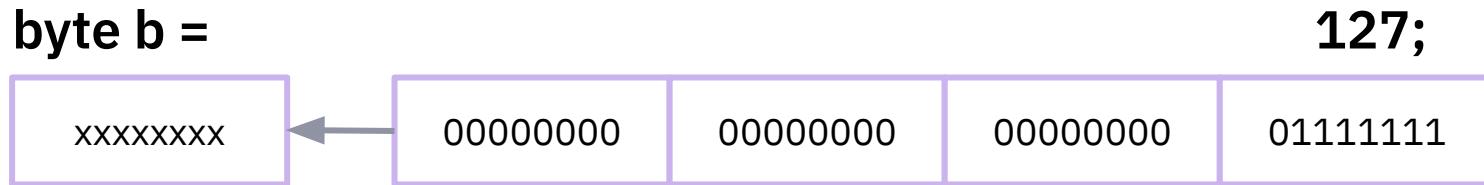


Преобразование типов (typecasting)





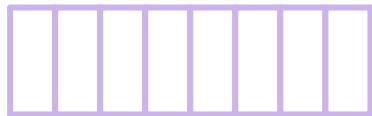
Неявное преобразование типов





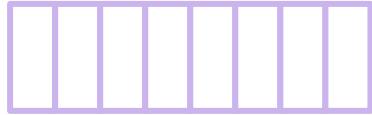
Явное преобразование типов

`long l`



0000
0111

> 2 млрд (int)



0000 0000 0000 0000
0000 0000 0000 0111

L



Ошибка присваивания

```
9      int i0 = 100;
10     byte b0 = i0;           Required type: byte
11                                         Provided: int
12                                         Cast to 'byte' More actions...   :::
13
14     int i0 = 100
15
16 sources-draft
17
```



Явное приведение к меньшему типу

```
9      int i0 = 100;
10     byte b0 = (byte) i0;
11
12
13
14
15
```



Константность

Constare - (лат. стоять твёрдо)





Константность

final- переменная с конечным значением



Теперь ваша очередь!



Ответьте на вопросы сообщением в чат

Вопросы:

1. Какая таблица перекодировки используется для представления символов?
2. Каких действий требует от программиста явное преобразование типов?
3. Какое значение будет содержаться в переменной a после выполнения строки
`int a = 10.0f/3.0f;`





Ссылочные типы данных



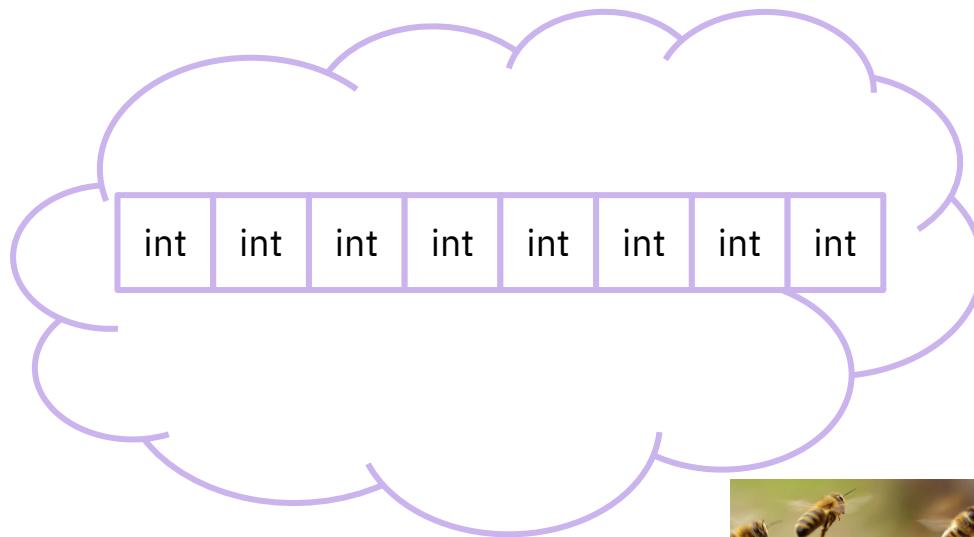
Сылочные типы данных

Бесконечное
множество, которое
держится на восьми
примитивах





Массив





Отличие хранения

Примитивный

```
int i = 10 ;
```

00001010

Ссылочный

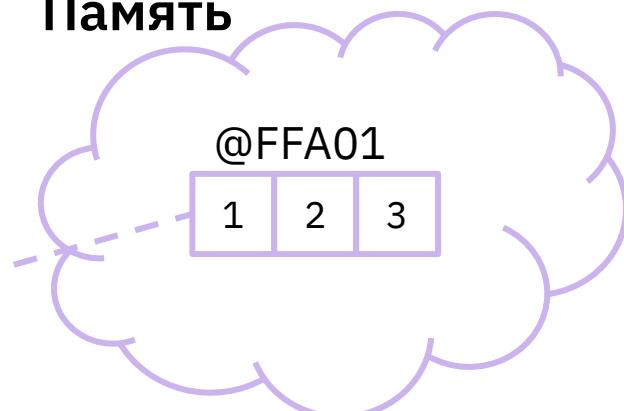
```
int[] a = {1, 2, 3};
```

000@FFA01

Память

@FFA01

1	2	3
---	---	---





null

Non-zero value



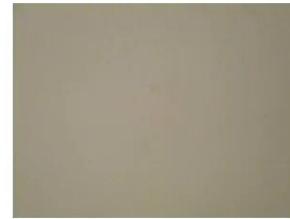
null



0

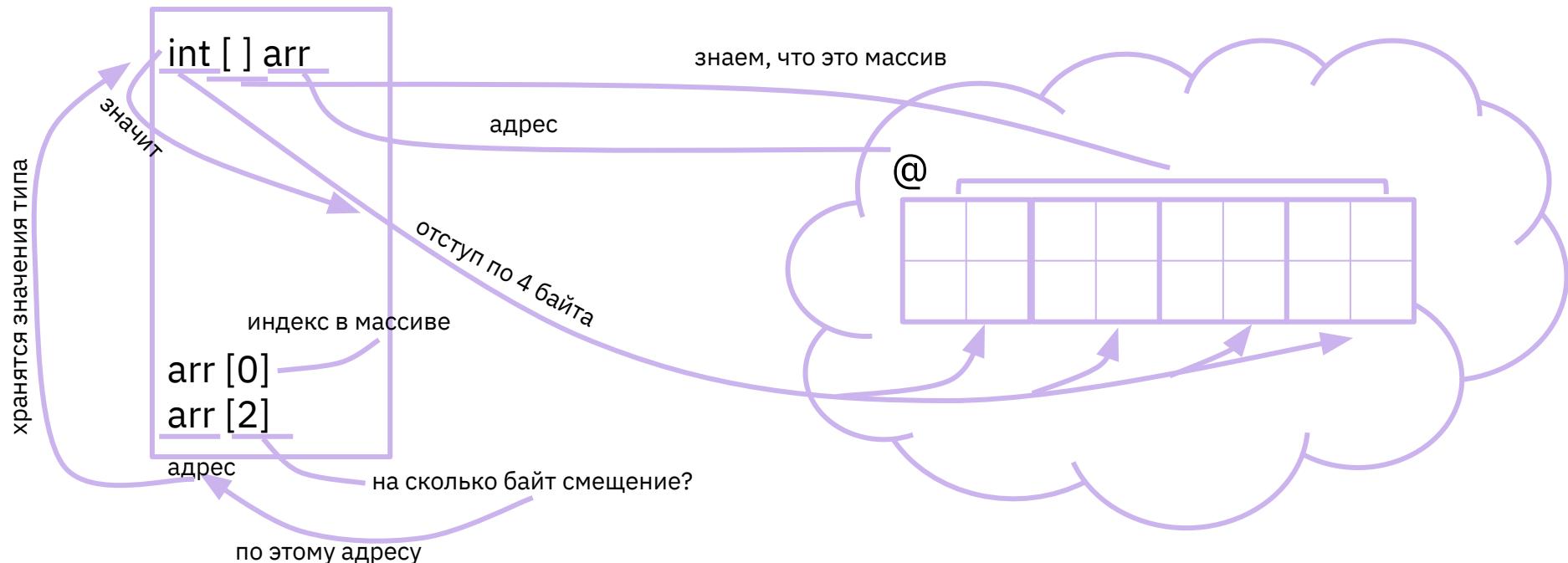


undefined





Индексация массива





Ошибка присваивания

```
9      int[] array3 = {5, 4, 3, 2, 1};  
10     int[] array2 = new int[5];  
11     int[] array1 = array2;  
12     int[] array0;  
13  
14     array2 = {1, 2, 3, 4, 5};  
15  
16
```

Array initializer is not allowed here

Add 'new int[]'

More actions...

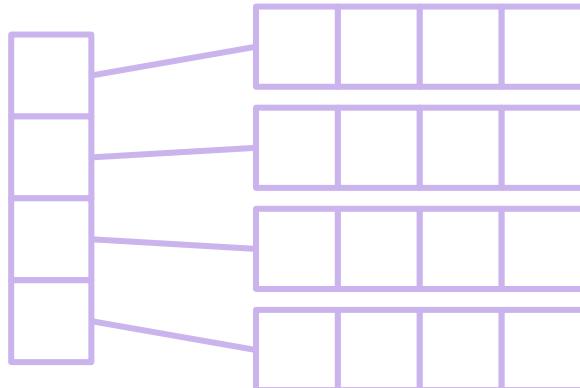


Многомерные массивы

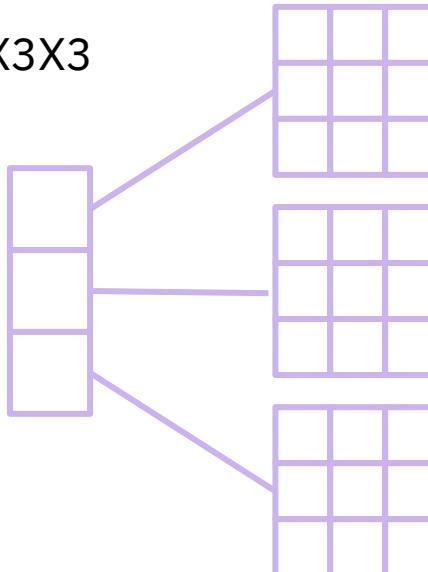
1×6



4×5

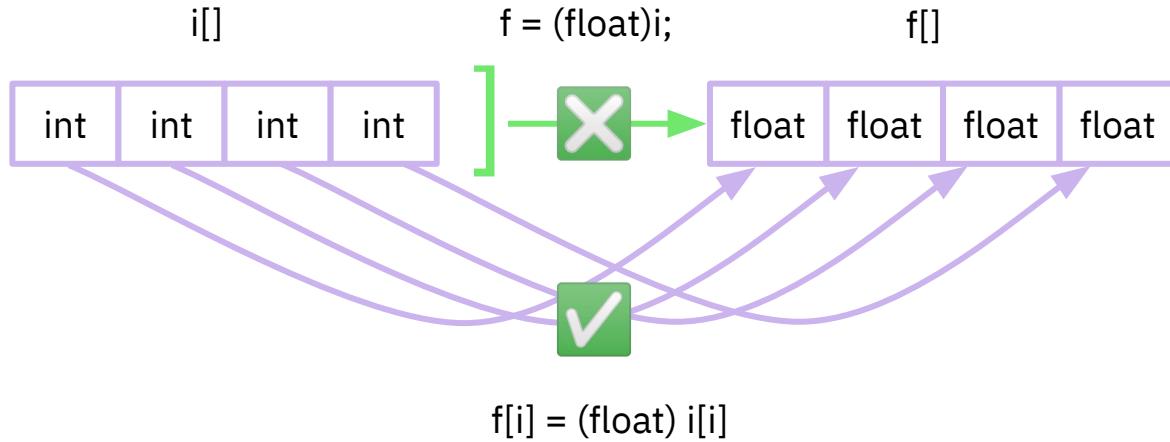


$3 \times 3 \times 3$





Преобразование типов массива





final массив

```
final a[] = {1, 2, 3}
```

a = b ✗

a[1] = 5 ✓





Особенности инициализации массивов

```
12          // memory leak
13          int[][] arr0 = new int[3][3];
14          arr0[0] = new int[3];
15          arr0[1] = new int[5];
16          arr0[2] = new int[7];
17
18          int[][] arr = new int[3][];
19          arr[0] = new int[3];
20          arr[1] = new int[5];
21          arr[2] = new int[7];
```



Индексный доступ и длина массива

```
int[] arr = {1, 2, 3, 4, 5};  
System.out.println(arr[0]);  
arr[0] = 10;  
System.out.println(arr[0]);  
System.out.println(arr.length);
```



Теперь ваша очередь!



Ответьте на вопросы сообщением в чат

Вопросы:

1. Почему индексация массива начинается с нуля?
2. Какой индекс будет последним в массиве из 100 элементов?
3. Сколько будет создано одномерных массивов при инициализации массива 3x3x3?





Базовая функциональность языка



Арифметические операции

$$2 + 2 = 4$$

$$3 * 3 = 9$$

$$10 / 3 = 3$$

$$10 \% 3 = 1$$

a = 
присваивание





Условный оператор

```
12          int adult = 18;  
13          int age = 10;  
14          if (age < adult) {  
15              // deny something  
16          } else if (age == adult) {  
17              //congratulations  
18          } else {  
19              //allow something  
20      }
```



Оператор множественного выбора

```
11      int address = 5;
12      switch (address) {
13          case 1:
14              // director's desk
15          case 2:
16              // manager's desk
17          case 3:
18              // developer's desk
19          default:
20              // unknown recipient
21 }
```



Оператор безусловного перехода

```
11      int address = 5;
12      switch (address) {
13          case 1:
14              // director's desk
15              break;
16          case 2:
17              // manager's desk
18              break;
19          case 3:
20              // developer's desk
21              break;
22          default:
23              // unknown recipient
24      }
```



Циклы

```
while () {}  
do {} while () ;  
for ( ; ; ) {}
```





Цикл **foreach**

```
for ( : ) { }
```





Бинарные операторы





Бинарные операторы

Битовые

& битовое и;
| битовое или;
~ битовое не;
^ исключающее или;
<< сдвиг влево;
>> сдвиг вправо.

Литеральные

&& литеральное и;
|| литеральное или;
! литеральное не;





Таблицы истинности

A	Б	И
0	0	0
0	1	0
1	0	0
1	1	1

A	Б	ИЛИ
0	0	0
0	1	1
1	0	1
1	1	1

A	Б	ИСК ИЛИ
0	0	0
0	1	1
1	0	1
1	1	0

A	НЕ
0	1
1	0





Бинарная арифметика

00000100 = 4

& 00000111 = 7

= 00000100 = 4

00000100 = 4

| 00000111 = 7

= 00000111 = 7

00000100 = 4

^ 00000111 = 7

= 00000011 = 3

~ 00000100 = 4

= 11111011 = -5





Битовые сдвиги

```
| 2 | 000000010 | 2 << 0 |
| 4 | 000000100 | 2 << 1 |
| 8 | 000001000 | 2 << 2 |
| 16 | 000010000 | 2 << 3 |
| 32 | 000100000 | 2 << 4 |
| 64 | 001000000 | 2 << 5 |
| 128 | 010000000 | 2 << 6 |
```





Битовые сдвиги

$$x * 2^N = x \ll N$$





Битовые сдвиги

	128		010000000		128 >> 0	
	64		001000000		128 >> 1	
	32		000100000		128 >> 2	
	16		000010000		128 >> 3	
	8		000001000		128 >> 4	
	4		000000100		128 >> 5	
	2		000000010		128 >> 6	





Бинарные операторы



- $X \&& Y$ - литеральная;
- $X || Y$ - литеральная;
- $!X$ - литеральная;
- $N << K$ - $N * 2^K$;
- $N >> K$ - $N / 2^K$;
- $x \& y$ - битовая. 1 если оба $x = 1$ и $y = 1$;
- $x | y$ - битовая. 1 если хотя бы один из $x = 1$ или $y = 1$;
- $\sim x$ - битовая. 1 если $x = 0$;
- $x ^ y$ - битовая. 1 если x отличается от y .

Теперь ваша очередь!



Ответьте на вопросы сообщением в чат

Вопросы:

1. Почему нежелательно использовать оператор `switch` если нужно проверить диапазон значений?
2. Возможно ли записать бесконечный цикл с помощью оператора `for`?
3. $2 + 2 * 2 == 2 << 2 >> 1$?





Методы

Функция - это исполняемый блок
кода на языке Java

Функция в классе это **метод**.



Параметры и аргументы функций

```
функция (параметр 1, параметр 2) {  
    тело  
}
```

```
-----  
функция (аргумент 1, аргумент 2);
```

lowerCamelCase

Функция - это глагол



Ограничение на создание функций

```
функция (параметр 1, параметр 2) {  
    функция (параметр 1, параметр 2) {  
        тело  
    }  
}
```



Нельзя создавать функцию внутри функции



Передача аргументов в функцию

```
функция (параметр 1, параметр 2, параметр 3) {
```

тело

```
}
```



```
-----  
функция (аргумент 1, аргумент 2, аргумент 3);
```



Возвращаемое значение

```
целое функция () {  
    вернуть 10;  
}
```

```
int i = функция (); тоже самое, что и int i = 10 ;
```





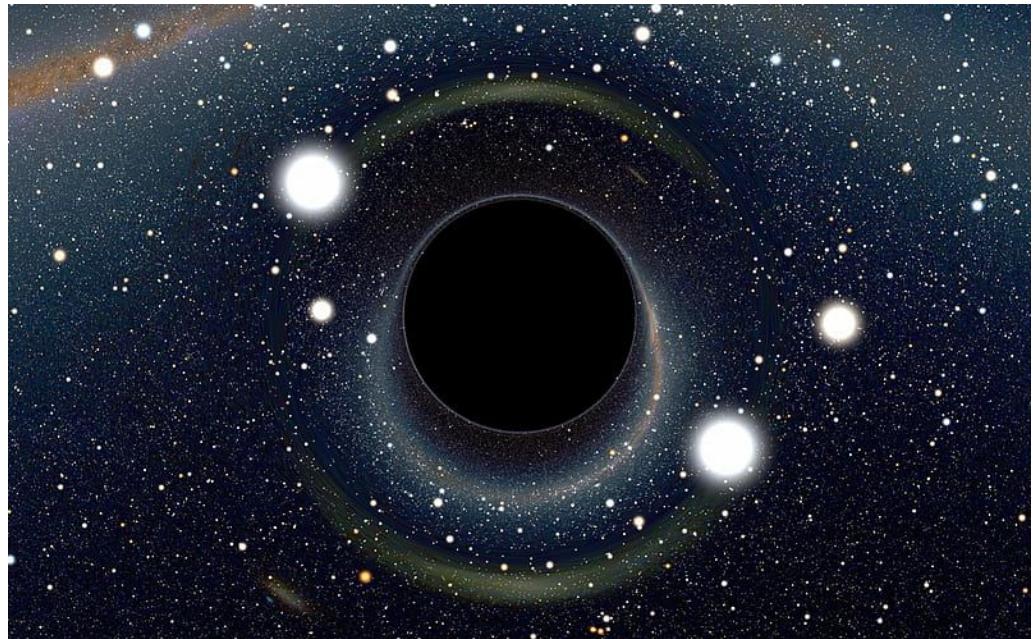
Возвращаемое значение void

void *noun* [no plural]

a large hole or empty space

пустое место, дыра

(Cambridge dictionary)





Возвращаемое значение

```
целое функция () {  
    вернуть 10;  
}
```

```
-----  
int i = функция (); тоже самое, что и int i = 10 ;
```

```
public static void main(String[] args)
```



Сигнатура метода

Сигнатура метода — это имя метода и его параметры. В сигнатуру метода не входит возвращаемое значение.

Нельзя написать два метода с одинаковой сигнатурой.





Перегрузка методов

Перегрузка методов - это механизм языка, позволяющий описать методы с одинаковыми названиями и разными оставшимися частями сигнатурой, чтобы получить единообразие при вызове семантически схожих методов с разнотипными данными.





Пример перегрузки методов

The screenshot shows a Java code editor with a dark theme. The code defines two methods named `sum` and one `main` method.

```
Project: Main.java ×
1 usage
5     private static float sum (int a, int b) {
6         return a + b;
7     }
8
9     1 usage
10    private static float sum (float a, float b) {
11        return 0;
12    }
13 ▶     public static void main(String[] args) {
14         System.out.println(sum( a: 2, b: 2));
15         System.out.println(sum( a: 2.0f, b: 2.0f));
16 }
```

The `main` method calls the `sum` method twice with different argument types: `int` and `float`. The output window at the bottom shows the results:

Run: Main ×

Output
4.0
0.0



На этом уроке

- 📌 Рассмотрели базовый функционал языка
- 📌 Примитивные типы данных
- 📌 Ссылочные типы данных, массивы
- 📌 Разобрали способы хранения и представления данных
- 📌 Поговорили о способах манипуляции данными





Практическое задание

- 📌 Написать метод «Шифр Цезаря», с булевым параметром зашифрования и расшифрования и числовым ключом;
- 📌 Написать метод, принимающий на вход массив чисел и параметр n. Метод должен осуществить циклический (последний элемент при сдвиге становится первым) сдвиг всех элементов массива на n позиций;
- 📌 Написать метод, которому можно передать в качестве аргумента массив, состоящий строго из единиц и нулей (целые числа типа int). Метод должен заменить единицы массиве на нули, а нули на единицы. Написать как можно больше вариантов





Цель учения — достичь наибольшего удовлетворения в получении знаний.

Сюнь-цзы