



# Типове данни. Изрази. Операции. Приоритет.

Програмиране на Java

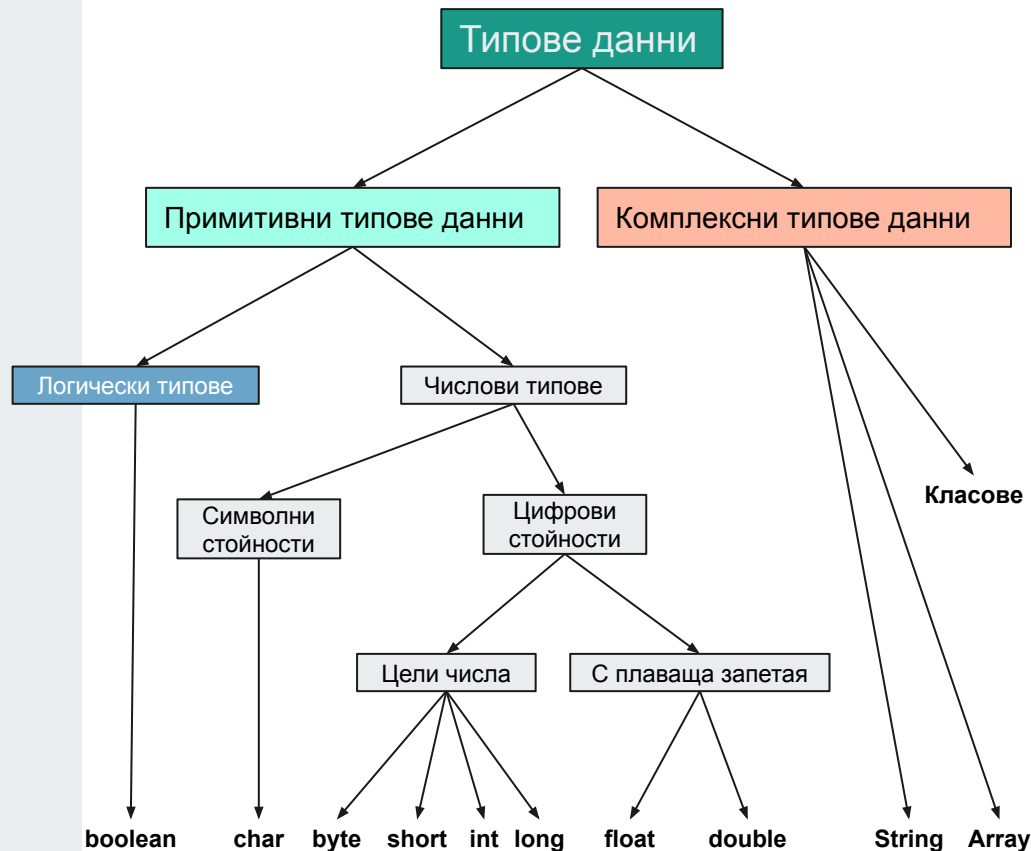


# Съдържание

1. Примитивни типове данни
2. Деклариране и присвояване на стойности
3. Област на видимост
4. Константи
5. Типа var
6. Изрази и операции
7. Приоритет
8. Коментари

# Типове данни в Java

Java е стриктно типизиран език - не се позволява промяна на типа на променлива след като веднъж е декларирана.





# Примитивни типове данни в Java

Тип данни	Размер (битове)	Диапазон	
Ключова дума		Минимум	Максимум
boolean	1	false	true
byte	8	-128	127
short	16	- 32 768	32 767
char	16	'\u0000' (0)	'\uFFFF' (65535)
int	32	-2 147 483 648	2 147 483 647
long	64	$-2^{63}$	$2^{63}-1$
float	32	$\sim 1.4e^{-045}$	$\sim 3.4e^{+038}$
double	64	$\sim 4.9e^{-324}$	$\sim 1.8e^{+308}$



# Примитивни стойности и променливи

- Деклариране на променлива
- Инициализиране на променлива
- Едновременно деклариране и инициализиране

# Деклариране на променлива

Задължително променливите трябва да бъдат декларирани преди да могат да бъдат използвани!

**тип** име\_на\_променлива;

`int x;`

`double y;`

`boolean b;`

`String s;`

Декларацията на променлива “казва” на компилатора какво е името на променливата и какъв е типът на данните, които ще съдържа.

При декларацията променливите **не се** инициализират със стойности по подразбиране.



# Инициализиране на променлива

име\_на\_променлива = стойност;

x = 10;

y = 2.5;

b = true;

s = "Пловдив";

Инициализацията се извършва след декларацията.

Стойностите трябва да съответстват на типа на променливата при декларацията ѝ.

# Едновременно деклариране и инициализиране на променлива

**тип** име\_на\_променлива = стойност;

```
int x = 2;
```

```
double y = 3.5;
```

```
boolean b = true;
```

```
String s = "Пловдив";
```

В общия случай аналогично на декларация и последваща инициализация:

```
int x;  
x = 2;
```





# Именуване на променлива

[Code Conventions for the Java Programming Language: 9. Naming Conventions](#)

1. Името на променливата започва винаги с малка буква.
2. В общия случай името на променливата е съществително.
3. Ако в името на променливата има повече от една дума, всяка следваща след първата започва с главна буква:

```
String specialityName = "Информатика";  
int yearOfGraduation = 2025;
```

4. Името на променливата не бива да започва с \_ или \$, въпреки че и двата символа са допустими
5. Еднобуквените имена на променливи трябва да бъдат избягвани, освен за "временни" променливи и такива, които са с класическо предназначение (напр. броячи). Най-често използваните временни променливи са:

- o i, j, k, m, n - за целочислени типове



# Област на видимост (variable scope)

В Java (както и в C/C++) всички променливи имат статична област на видимост, която е известна по време на компилиране и не зависи от логиката на изпълнение на програмата.

Java позволява деклариране на променливи във всеки блок. Блокът дефинира област, която започва с **{** и завършва с **}**.

Областта на видимост на променлива започва от мястото на деклариране на променливата и се простира до края на блока, в който е декларирана.

Променливите са видими и във всички нива на вложеност на блокове.

# Област на видимост (variable scope)

```
public static void main(String[] args) {  
    System.out.println("b = " + b);  
    boolean b = true;  
}
```

Начало на блок

Край на блок

Грешка при компилиране: b не е декларирана.



## Област на видимост (variable scope)

```
public static void main(String[] args) {  
  
    boolean b = true;  
  
    System.out.println("b = " + b);  
  
}
```

Правилна употреба.



## Област на видимост (variable scope)

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

```
{  
    boolean b = true;  
}
```

```
System.out.println("b = " + b);
```

```
}
```

Грешка при компилиране: b не е декларирана.



## Област на видимост (variable scope)

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

```
    boolean b = false;
```

```
    {
```

```
        b = true;
```

```
    }
```

```
    System.out.println("b = " + b);
```

```
}
```

Правилна употреба.



# Литерали

- Стойност зададена в сорс кода
- Целочислени литерали
  - В десетична, шестнайсетична (0x) и двоична (0b) бройни системи;
  - long с L/l
  - int по подразбиране
  - Символа '\_'
- Литерали на числа с плаваща запетая
  - double по подразбиране (D/d)
  - 'e' за научна нотация (степен на 10)
  - float с (F/f)
- Литерали от тип символ или низ
  - Може да съдържат Unicode (UTF-16) символи: '\u0108' - голямо С
  - Специални символи: \b (backspace), \t (tab), \n (line feed), \f (form feed), \r (carriage return), \" (double quote), \' (single quote), and \\ (backslash)
  - Стойност null



# Заделяне на памет

Памет = стойност;





# Числа с плаваща запетая

- Типове: `float` и `double`
  - Пример: `47281.97`
- Могат да бъдат представяни в научна нотация
  - Пример: `47281.97` се представя като  $4.728197 \times 10^4$
- Java използва “E” нотацията за представяне на стойности в научна нотация
  - Пример: `47281.97` се представя като `4.728197e4`



# Операции с числа с плаваща запетая

- Джеймс Гослинг: "95% от хората не разбират какво е плаващата запетая."
- Java използва подмножество на [IEEE 754 binary floating point representation](#) за представяне на числа с плаваща запетая.
- Типа **float** използва 32 бита и всяка комбинация от битове представя едно реално число. Т.е. float може да представи най-много  $2^{32}$  реални числа.
  - Покрива обхват от  $\pm 1.40129846432481707 \times 10^{-45}$  до  $\pm 3.40282346638528860 \times 10^{38}$  със 6 или 7 значещи цифри, включително + безкрайност, -безкрайност и NaN (not a number).
- IEEE стандарта използва вътрешно представяне подобно на научната нотация, но с двоична вместо десетична основа.



# Операции с числа с плаваща запетая

- Десетичната стойност се представя като:  $(-1)^s \times m \times 2^{(e-127)}$
- $s$  - бит за знак (31): 1 - отрицателно, 0 - положително число
- $e$  - експонента (30-23): по конвенция експонентата се намалява със 127.
  - Двоична експонента 5 се представя като  $127+5 = 132$  (двоично 10000100)
  - Двоична експонента -5 се представя като  $127-5 = 122$  (двоично 01111010)
- $m$  - мантиа (22-0): нормализирана между 0 и 1.
  - 0.1101 представя  $1/2 + 1/4 + 1/16 = 13/16 = 0.8125$
  - Не всяка десетична дроб може да бъде представена като двоична:  
 $1/10 = 1/16 + 1/32 + 1/256 + 1/512 + 1/4096 + 1/8192 + \dots$ 
    - В този случай 0.1 се представя с най-близката 23 битова двоична дроб:  
0.000110011001100110011...



# Операции с числа с плаваща запетая

Пример: десетичното число 0.085 се представя двоично така: 00111101101011100001010001111011

битове:	31	30-23	22-0
двоично:	0	01111011	01011100001010001111011
десетично:	0	123	3019899

Което е:

$$2^{-127} (1 + m / 2^{23}) = 2^{-4} (1 + 3019899 / 8388608) = 11408507 / 134217728 = 0.085000000894069671630859375$$



## Операции с числа с плаваща запетая

- `double` е подобно на `float`, но представя реалните числа с 64 бита, от които 11 за експонента с отклонение от 1023 и 52 за мантия.
- Това покрива диапазона  $\pm 4.94065645841246544 \times 10^{-324}$  до  $\pm 1.79769313486231570 \times 10^{+308}$  с 15 или 16 значими цифри.



## Операции с числа с плаваща запетая

```
double x1 = 0.3;  
double x2 = 0.1 + 0.1 + 0.1;  
System.out.println(x1 == x2);  
  
double z1 = 0.5;  
double z2 = 0.1 + 0.1 + 0.1 + 0.1 + 0.1;  
System.out.println(z1 == z2);
```

**НИКОГА** не използвайте типове данни с плаваща запетая (**float**, **double**) за работа с критични откъм точност реални числа - напр. финансова информация.

В Java съществува специален тип - **BigDecimal**

# Символен тип (char)

## ASCII Таблица

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	Space	64	40	100	64	8	96	60	140	96	`
1	1	001	SOH	(start of heading)	33	21	041	!	65	41	101	65	A	97	61	141	97	a
2	2	002	STX	(start of text)	34	22	042	"	66	42	102	66	B	98	62	142	98	b
3	3	003	ETX	(end of text)	35	23	043	#	67	43	103	67	C	99	63	143	99	c
4	4	004	EOT	(end of transmission)	36	24	044	\$	68	44	104	68	D	100	64	144	100	d
5	5	005	ENQ	(enquiry)	37	25	045	%	69	45	105	69	E	101	65	145	101	e
6	6	006	ACK	(acknowledge)	38	26	046	&	70	46	106	70	F	102	66	146	102	f
7	7	007	BEL	(bell)	39	27	047	'	71	47	107	71	G	103	67	147	103	g
8	8	010	BS	(backspace)	40	28	050	(	72	48	110	72	H	104	68	150	104	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	73	49	111	73	I	105	69	151	105	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	74	4A	112	74	J	106	6A	152	106	j
11	B	013	VT	(vertical tab)	43	2B	053	+	75	4B	113	75	K	107	6B	153	107	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	76	4C	114	76	L	108	6C	154	108	l
13	D	015	CR	(carriage return)	45	2D	055	-	77	4D	115	77	M	109	6D	155	109	m
14	E	016	SO	(shift out)	46	2E	056	.	78	4E	116	78	N	110	6E	156	110	n
15	F	017	SI	(shift in)	47	2F	057	/	79	4F	117	79	O	111	6F	157	111	o
16	10	020	DLE	(data link escape)	48	30	060	0	80	50	120	80	P	112	70	160	112	p
17	11	021	DC1	(device control 1)	49	31	061	1	81	51	121	81	Q	113	71	161	113	q
18	12	022	DC2	(device control 2)	50	32	062	2	82	52	122	82	R	114	72	162	114	r
19	13	023	DC3	(device control 3)	51	33	063	3	83	53	123	83	S	115	73	163	115	s
20	14	024	DC4	(device control 4)	52	34	064	4	84	54	124	84	T	116	74	164	116	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	85	55	125	85	U	117	75	165	117	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	86	56	126	86	V	118	76	166	118	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	87	57	127	87	W	119	77	167	119	w
24	18	030	CAN	(cancel)	56	38	070	8	88	58	130	88	X	120	78	170	120	x
25	19	031	EH	(end of medium)	57	39	071	9	89	59	131	89	Y	121	79	171	121	y
26	1A	032	SUB	(substitute)	58	3A	072	;	90	5A	132	90	Z	122	7A	172	122	z
27	1B	033	ESC	(escape)	59	3B	073	:	91	5B	133	91	[	123	7B	173	123	{
28	1C	034	FS	(file separator)	60	3C	074	<	92	5C	134	92	\	124	7C	174	124	
29	1D	035	GS	(group separator)	61	3D	075	=	93	5D	135	93	]	125	7D	175	125	}
30	1E	036	RS	(record separator)	62	3E	076	>	94	5E	136	94	^	126	7E	176	126	~
31	1F	037	US	(unit separator)	63	3F	077	?	95	5F	137	95	_	127	7F	177	127	DEL

Source: [www.asciitable.com](http://www.asciitable.com)

## Разширена ASCII Таблица

128	Ѓ	144	É	160	á	176	ð	193	±	209	ƒ	225	ß	241	±
129	ü	145	æ	161	í	177	ñ	194	ˆ	210	π	226	Γ	242	≥
130	é	146	Æ	162	ó	178	ï	195	˜	211	ℓ	227	π	243	≤
131	â	147	ô	163	ú	179		196	—	212	ℓ	228	Σ	244	∫
132	ä	148	ö	164	û	180	ı	197	†	213	ƒ	229	σ	245	∫
133	à	149	ò	165	ñ	181	ı	198	†	214	ƒ	230	μ	246	+
134	â	150	û	166	ˆ	182	ı	199	ı	215	ı	231	τ	247	≈
135	ç	151	ù	167	°	183	ı	200	ı	216	ı	232	Φ	248	°
136	ê	152	—	168	ı	184	ı	201	ı	217	ı	233	⊙	249	.
137	ë	153	Ö	169	—	185	ı	202	ı	218	ı	234	Ω	250	.
138	è	154	Û	170	ı	186	ı	203	ı	219	ı	235	δ	251	√
139	ı	156	£	171	½	187	ı	204	ı	220	ı	236	∞	252	—
140	ı	157	¥	172	¾	188	ı	205	ı	221	ı	237	φ	253	²
141	ı	158	—	173	ı	189	ı	206	ı	222	ı	238	ε	254	■
142	Ä	159	ƒ	174	»	190	ı	207	ı	223	ı	239	∩	255	
143	Å	192	Ł	175	»	191	ı	208	ı	224	α	240	≡		

Source: [www.asciitable.com](http://www.asciitable.com)

# Символен тип (char) - Unicode

Cyrillic 04FF															
	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	040A	040B	040C	040D	040E
0	Ё	А	Р	а	р	ѐ	Ѡ	Ѣ	Ѥ	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ
1	Ѣ	Б	С	б	с	ѣ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
2	Ѥ	В	Т	в	т	ѥ	Ѣ	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ
3	Ѧ	Г	У	г	у	ѧ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ	ѵ
4	Ѩ	Д	Ф	д	ф	ѩ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
5	Ѭ	Е	Х	е	х	ѭ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
6	Ѯ	І	Ж	і	ж	ѯ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
7	Ѱ	Ѕ	Ч	ѕ	ч	ѱ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
8	Ѳ	Ј	И	ј	и	ѳ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
9	Ѵ	Љ	Ш	љ	ш	ѵ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
A	Ѷ	Њ	Ѣ	њ	ѣ	ѷ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
B	Ѹ	Ѣ	Ѥ	ѥ	Ѧ	ѹ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
C	Ѻ	Ѣ	Ѥ	ѥ	Ѧ	ѻ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
D	Ѽ	Ѣ	Ѥ	ѥ	Ѧ	ѽ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
E	ѿ	Ѣ	Ѥ	ѥ	Ѧ	Ѿ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ
F	ѿ	Ѣ	Ѥ	ѥ	Ѧ	ѿ	ѡ	ѣ	ѥ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ

The Unicode Standard, Version 15.0, Copyright © 1991-2022 Unicode, Inc. All rights reserved.

Arabic 06FF															
	0600	0601	0602	0603	0604	0605	0606	0607	0608	0609	060A	060B	060C	060D	060E
0	٠	١	٢	٣	٤	٥	٦	٧	٨	٩	١٠	١١	١٢	١٣	١٤
1	١٥	١٦	١٧	١٨	١٩	٢٠	٢١	٢٢	٢٣	٢٤	٢٥	٢٦	٢٧	٢٨	٢٩
2	٣٠	٣١	٣٢	٣٣	٣٤	٣٥	٣٦	٣٧	٣٨	٣٩	٤٠	٤١	٤٢	٤٣	٤٤
3	٤٥	٤٦	٤٧	٤٨	٤٩	٥٠	٥١	٥٢	٥٣	٥٤	٥٥	٥٦	٥٧	٥٨	٥٩
4	٦٠	٦١	٦٢	٦٣	٦٤	٦٥	٦٦	٦٧	٦٨	٦٩	٧٠	٧١	٧٢	٧٣	٧٤
5	٧٥	٧٦	٧٧	٧٨	٧٩	٨٠	٨١	٨٢	٨٣	٨٤	٨٥	٨٦	٨٧	٨٨	٨٩
6	٩٠	٩١	٩٢	٩٣	٩٤	٩٥	٩٦	٩٧	٩٨	٩٩	١٠٠	١٠١	١٠٢	١٠٣	١٠٤
7	١٠٥	١٠٦	١٠٧	١٠٨	١٠٩	١١٠	١١١	١١٢	١١٣	١١٤	١١٥	١١٦	١١٧	١١٨	١١٩
8	١٢٠	١٢١	١٢٢	١٢٣	١٢٤	١٢٥	١٢٦	١٢٧	١٢٨	١٢٩	١٣٠	١٣١	١٣٢	١٣٣	١٣٤
9	١٣٥	١٣٦	١٣٧	١٣٨	١٣٩	١٤٠	١٤١	١٤٢	١٤٣	١٤٤	١٤٥	١٤٦	١٤٧	١٤٨	١٤٩
A	١٥٠	١٥١	١٥٢	١٥٣	١٥٤	١٥٥	١٥٦	١٥٧	١٥٨	١٥٩	١٦٠	١٦١	١٦٢	١٦٣	١٦٤
B	١٦٥	١٦٦	١٦٧	١٦٨	١٦٩	١٧٠	١٧١	١٧٢	١٧٣	١٧٤	١٧٥	١٧٦	١٧٧	١٧٨	١٧٩
C	١٨٠	١٨١	١٨٢	١٨٣	١٨٤	١٨٥	١٨٦	١٨٧	١٨٨	١٨٩	١٩٠	١٩١	١٩٢	١٩٣	١٩٤
D	١٩٥	١٩٦	١٩٧	١٩٨	١٩٩	٢٠٠	٢٠١	٢٠٢	٢٠٣	٢٠٤	٢٠٥	٢٠٦	٢٠٧	٢٠٨	٢٠٩
E	٢١٠	٢١١	٢١٢	٢١٣	٢١٤	٢١٥	٢١٦	٢١٧	٢١٨	٢١٩	٢٢٠	٢٢١	٢٢٢	٢٢٣	٢٢٤
F	٢٢٥	٢٢٦	٢٢٧	٢٢٨	٢٢٩	٢٣٠	٢٣١	٢٣٢	٢٣٣	٢٣٤	٢٣٥	٢٣٦	٢٣٧	٢٣٨	٢٣٩

The Unicode Standard, Version 15.0, Copyright © 1991-2022 Unicode, Inc. All rights reserved.

Devanagari 097F															
	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	090A	090B	090C	090D	090E
0	०	१	२	३	४	५	६	७	८	९	१०	११	१२	१३	१४
1	१५	१६	१७	१८	१९	२०	२१	२२	२३	२४	२५	२६	२७	२८	२९
2	३०	३१	३२	३३	३४	३५	३६	३७	३८	३९	४०	४१	४२	४३	४४
3	४५	४६	४७	४८	४९	५०	५१	५२	५३	५४	५५	५६	५७	५८	५९
4	६०	६१	६२	६३	६४	६५	६६	६७	६८	६९	७०	७१	७२	७३	७४
5	७५	७६	७७	७८	७९	८०	८१	८२	८३	८४	८५	८६	८७	८८	८९
6	९०	९१	९२	९३	९४	९५	९६	९७	९८	९९	१००	१०१	१०२	१०३	१०४
7	१०५	१०६	१०७	१०८	१०९	११०	१११	११२	११३	११४	११५	११६	११७	११८	११९
8	१२०	१२१	१२२	१२३	१२४	१२५	१२६	१२७	१२८	१२९	१३०	१३१	१३२	१३३	१३४
9	१३५	१३६	१३७	१३८	१३९	१४०	१४१	१४२	१४३	१४४	१४५	१४६	१४७	१४८	१४९
A	१५०	१५१	१५२	१५३	१५४	१५५	१५६	१५७	१५८	१५९	१६०	१६१	१६२	१६३	१६४
B	१६५	१६६	१६७	१६८	१६९	१७०	१७१	१७२	१७३	१७४	१७५	१७६	१७७	१७८	१७९
C	१८०	१८१	१८२	१८३	१८४	१८५	१८६	१८७	१८८	१८९	१९०	१९१	१९२	१९३	१९४
D	१९५	१९६	१९७	१९८	१९९	२००	२०१	२०२	२०३	२०४	२०५	२०६	२०७	२०८	२०९
E	२१०	२११	२१२	२१३	२१४	२१५	२१६	२१७	२१८	२१९	२२०	२२१	२२२	२२३	२२४
F	२२५	२२६	२२७	२२८	२२९	२३०	२३१	२३२	२३३	२३४	२३५	२३६	२३७	२३८	२३९

The Unicode Standard, Version 15.0, Copyright © 1991-2022 Unicode, Inc. All rights reserved.





## Аритметични операции

+	Положителен	$n$
-	Отрицателен	$-n$
+	Сума	$a + b$
-	Разлика	$a - b$
*	Умножение	$a * b$
/	Деление	$a / b$
%	Остатък от деление	$a \bmod b$
++	Предварително увеличение	++a става $a+1$ , увеличава $a$ с 1
++	Последващо увеличение	a++ става $a$ , увеличава $a$ с 1
--	Предварително намаление	--a става $a-1$ , намалява $a$ с 1
--	Последващо намаление	a-- става $a$ , намалява $a$ с 1



# Видове операции

- Infix:  $a + b$ ,  $a < b$
- Postfix:  $a++$
- Prefix:  $++a$
- Трипозиционни:  $a > b ? a : b$



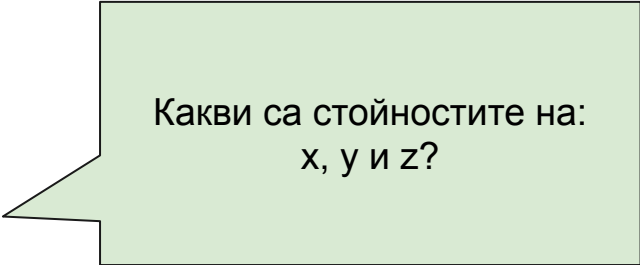
# Странични ефекти

```
int x = 2, y = 5, z = 1;
```

```
x = (z++ - (y = y + x));
```

```
x = (++z - (y = y + x));
```

До тук на 08.03.2023



Какви са стойностите на:  
x, y и z?



## Релационни операции

==	равно	$a == b$
!=	неравно	$a != b$
<	по-малко	$a < b$
<=	по-малко равно	$a \leq b$
>	по-голямо	$a > b$
>=	по-голямо равно	$a \geq b$



# Логически операции

!	отрицание	$\sim a$
&&	AND с частично оценяване	$a \wedge b$
	OR с частично оценяване	$a \vee b$
&	AND с пълно оценяване	$a \wedge b$
	OR с пълно оценяване	$a \vee b$
^	EXCLUSIVE-OR (или ... или)	$a \otimes b$



# Операции с boolean

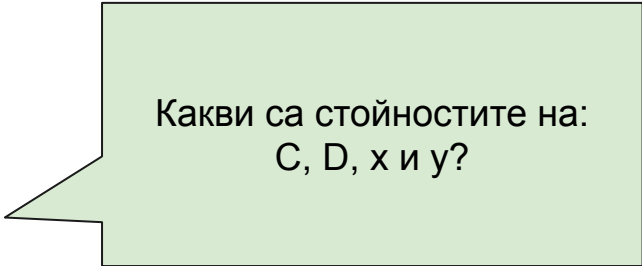
- Пълно оценяване |, &
- Кратко оценяване ||, &&



# Логически операции

```
boolean A = true;  
boolean B = false;  
int x = 0;  
int y = 0;
```

```
boolean C = A || (++x < 0);  
boolean D = B & (++y < 0);
```



Какви са стойностите на:  
C, D, x и y?



# Побитови операции

До тук на 15.03.2023

~	Единичен комплимент	$\sim a$ : инвертиране битовете на $a$
	Побитов OR	$a   b$ : побитов $a_i \vee b_i$
&	Побитов AND	$a \& b$ : побитов $a_i \wedge b_i$
^	Побитов XOR	$a \wedge b$ : побитов $a_i \otimes b_i$
>>	Отместване на дясно със знак	$a >> b$ : битовете на $a$ се отместват с $b$ позиции надясно, знак като при $a$
>>>	Отместване на дясно без знак	$a >>> b$ : битовете на $a$ се отместват с $b$ позиции надясно, попълване с 0, припокриване на знак (0)
<<	Отместване на ляво	$a << b$ : битовете на $a$ се отместват с $b$ позиции наляво, попълване с 0, знак като при $a$





## Оператор за присвояване

EBNF:

Оператор за присвояване ::= = | += | \*= | -= | /=  
| %= | &= | ^=  
| <<= | >>= | >>>=.



# Преобразуване на типове

- Автоматично преобразуване в по-голям тип - няма загуба на информация

```
byte a = 120;  
int z = a;
```

Няма загуба на информация - 8 битови стойности се побират в 32 битов тип.



# Преобразуване на типове

- Автоматично преобразуване в по-малък тип - не е възможно

```
int a = 130;  
byte z = a;
```

Грешка при компилиране:  
компиляторът не може да побере  
32 битови стойности в 8 битов тип.



# Преобразуване на типове

- Явно преобразуване - с евентуална загуба на информация

```
int a = 130;  
byte z = (byte)a;
```

Тук явно принуждаваме компилаторът да извърши преобразуване на типовете дори със загуба на информация.

Резултатът е:  $z = -126$



# Приоритети

Правила: По-ниска група - по-висок приоритет  
Вътре в една група: приоритет по асоциативност

Операция	Типизиране	Асоциативност	Означение
Група 1			
++	N	R	Увеличаване
--	N	R	Намаляване
+	N	R	<u>Унарен</u> плюс
-	N	R	<u>Унарен</u> минус
~	I	R	<u>Onecomplement</u>
!	L	R	Отрицание
(type)	A	R	Преобразуване
Група 2			
*	N,N	L	Умножение
/	N,N	L	Деление
%	N,N	L	Остатък
Група 3			
+	N,N	L	Събиране
-	N,N	L	Изваждане
+	S,S	L	Конкатенация на низове
Група 4			
<<	I,I	L	Изместване на ляво
>>	I,I	L	Изместване на дясно
>>>	I,I	L	Изместване на дясно с допълване с 0




# Приоритети

Операция	Типизиране	Асоциативност	Означение
Група 5			
<	N,N	L	По-малко
<=	N,N	L	По-малко или равно
>	N,N	L	По-голямо
>=	N,N	L	По-голямо или равно
instanceof	R,R	L	Инстанция на клас
Група 6			
==	P,P	L	Равенство
!=	P,P	L	Неравенство
==	R,R	L	Равенство по реф.
!=	R,R	L	Неравенство по реф.
Група 7			
&	I,I	L	Побитов И
&	L,L	L	Логически И
Група 8			
^	I,I	L	Побитов XOR
^	L,L	L	Логически XOR
Група 9			
	I,I	L	Побитов ИЛИ
	L,L	L	Логически ИЛИ
Група 10			
&&	L,L	L	Логически И, частично

# Приоритети

Операция	Типизиране	Асоциативност	Означение
Група 11			
	L,L	L	Логическо ИЛИ, част.
Група 12			
? :	L,A,A	R	Условно изчисление
Група 13			
=	V,A	R	Присвояване
+=	V,N	R	Addition <u>assignment</u>
-=	V,N	R	Extraction assignment
*=	V,N	R	Multiplication ssignment
/=	V,N	R	Division assignment
%=	V,N	R	Rest value assignment
&=	V,N	R	Bitw.-AND-Assignment
=	V,L	R	Log.-AND-Assignment
	V,N	R	Bitw.-OR-Assignment
^=	V,L	R	Log.-OR-Assignment
	V,N	R	Bitw.-XOR-Assignment
>>=	V,L	R	Log.-XOR-Assignment
	V,I	R	Left-Write-Assignment
>>>=	V,I	R	Right-Write-Assignment
>>>>=	V,I	R	Right-Write-Assignment with Nullexpansion



## Още за декларирането на локални променливи

Java 10 Local Variable Type Inference

```
var a = 5;  
var name = "Иван";  
var c = 5/2;
```

```
a = 10; // ok  
a = "Петър"; // compile error
```





## var

- Само за локални променливи - т.е. променливи декларирани в тялото на методи.
- Не може да се присвоява null
- Задължително се инициализира с декларацията
- Типа се определя по време на компилиране и не може да се променя - т.е. все така е статичен тип



# Константи

Keyword: **final**

```
final int HOURS = 24;
```

```
final double PI = 22/7d;
```

```
final String JANUARY = "Януари";
```

```
HOURS = 12; // compile error
```



# Именуване на константи

1. Цялото име се изписва само с главни букви
2. Ако името се състои от повече от една дума, то те се разделят с \_

```
final double PI = 3.14;
```

```
final int MAX_BACHELOR_EDUCATION_YEARS = 4;
```



# Коментари

Видове коментари:

- inline - от **//** до края на реда

```
int a = 18; // my very secret magic number
```



# Коментари

Видове коментари:

- блок коментар - затворен между **/\*** и **\*/**

```
/* Годишите, при които се навършва  
пълнолетие в повечето държави от ЕС */  
int a = 18;
```



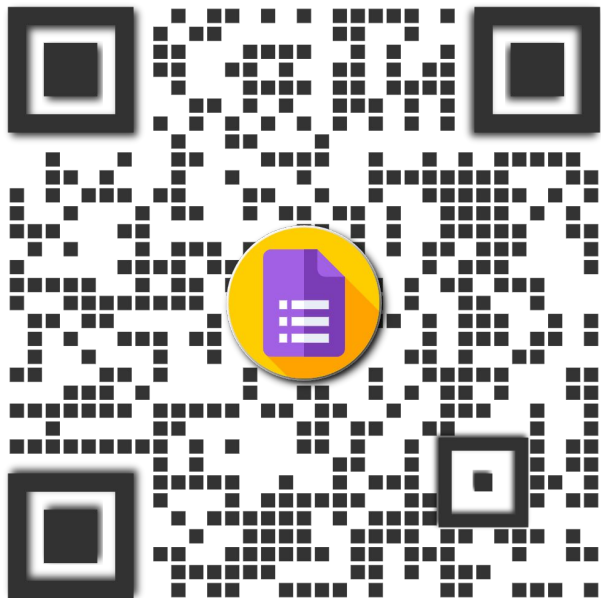
# Коментари

Кога се пишат коментари?

```
/* Годините, при които се навършва  
пълнолетие в повечето държави от ЕС */  
int a = 18;
```

```
int ageOfMajority = 18;
```

## Регистриране на присъствие

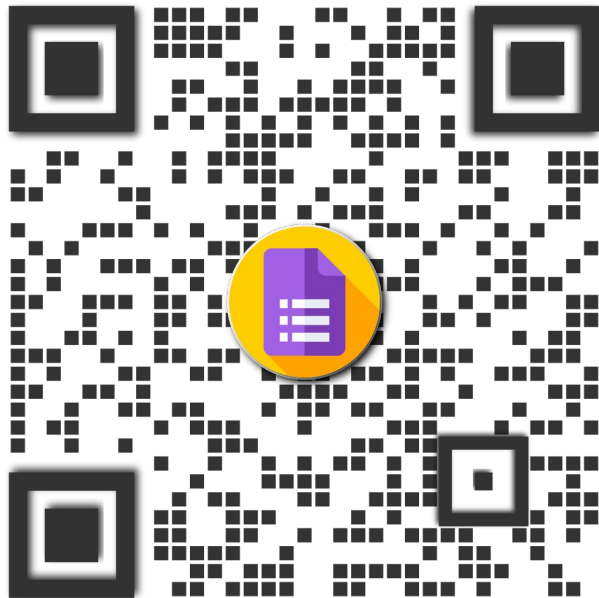


<https://t.ly/oEKHU>

Отговор: var



## Регистриране на присъствие



**<https://t.ly/md-h>**

Отговор: comment