

Objektovo orientované programovanie

Učiteľ:
Ing. Jozef Wagner PhD.

Učebnica:
<https://oop.wagjo.com/>

OPG

Cvičenie 5

1. Literály
2. null
3. Typová konverzia
4. Obalené typy

Literál

Literál je zápis konkrétneho údaja priamo v kóde

```
123          // int literál
123L         // long literál
0xFF         // int literál v hexadecimálnej sústave
0b1010       // int literál v binárnej sústave
07           // int literál v osmičkovej sústave
3.14         // double literál
1_000_000    // int literál s podtržníkom
'a'          // char literál
"Line1\nLine2" // String literál s escape sekvenciou
"Unicode:\u263A" // String literál s Unicode znakom ☺
{1, 2, 3}     // pole (zložený typ, prvky sú literály)
```

Null

`null` vyjadruje neprítomnosť hodnoty

Používa sa pri nasledovných situáciách:

- Inicializácia premenných, keď ešte nemáme vypočítanú hodnotu
- Ak metóda, ktorá má vyhľadať objekt ho nenašla, alebo nedokáže vypočítať výsledok
- Keď už nechceme, aby premenná viac ukazovala na konkrétny objekt

Výnimka `NullPointerException` sa vyhodí, ak program očakával objekt a nie `null`

Typová konverzia

Typová konverzia umožňuje konvertovať medzi dátovými typmi

Implicitná konverzia - widening, automaticky konvertuje menšie typy na väčšie

byte → short → int → long → float → double

char → int

Explicitná konverzia - narrowing alebo casting, manuálna konverzia. Môžu nastať nasledovné problémy:

- odrezanie desatinných miest
- pretečenie (overflow)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Obalené typy - wrapper classes

Pre každý primitívny dátový typ existuje príslušný obalený typ

Byte, Short, Integer, Long, Float, Double, Character, Boolean

Autoboxing a **auto-unboxing** je automatická konverzia z a do balených typov, keď je to potrebné

Uprednostňujeme primitívne typy pred obalenými, Java robí konverzie za nás. 2 obalené objekty porovnávam cez `equals()`

Obalené triedy obsahujú užitočné metódy a konštanty