```
int[] arr = new int[10];
```

index →

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0

```
string[] arr = new string[3];
```

null	null	null.

// Default value of char.

0 → 0
1 → 1
2 → 10
3 → 11
4 → 100
5 → 101
6 → 110
7 → 111
8 → 1000
9 → 1001
10 → 1010

2^1 2^2 2^3 2^2

-2^{n-1} to $2^{n-1}-1$

-2^7 to 2^7-1

-128 to 127

byte (8 bits) short (16 bits) int (32 bits) long (64 bits)

sign bit (MSB)

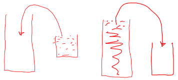
1	1	0	1	0	0	1	0

$2^8 = 256$

byte → 2^7

-127 → 1010010

short → 2 bytes
int → 4 bytes
long → 8 bytes
char → 2 bytes
double → 8 bytes
float → 4 bytes.



○ Typecasting: Typecasting is the process of assigning one primitive value to another primitive type. Compiler does this conversion.

int → -2^{31} to $2^{31}-1$

Int.MIN-VALUE

Int.MAX-VALUE

-128
-127
-126
-124
0
1
2
...
126
127
128
129
130
131

-48 + 1 = -47

-128 + 1 = -127

Loss of precision / loss of information.

○ Array:

Syntax:

data-type[] name-of-array; → Declaration.

name-of-array = new data-type[size]; → initialization

○ Typecast which compiler allow (no loss of precision) is called widening / upcasting / Implicit typecasting.

Ex:
```
int x = 10;
long y = x;
```

○ Vice-versa is called narrowing / down-casting or Explicit typecasting.

Ex:
```
long x = 10;
int y = (int) x;
```

○ Widening:

byte → short → int → long → float → double
char → int

loss of magnitude

```
int z = 130
byte x = z;   // error.
sout(x);
```
-127

○ Narrowing

byte ← short ← int ← long ← float ← double
int → char

loss of magnitude