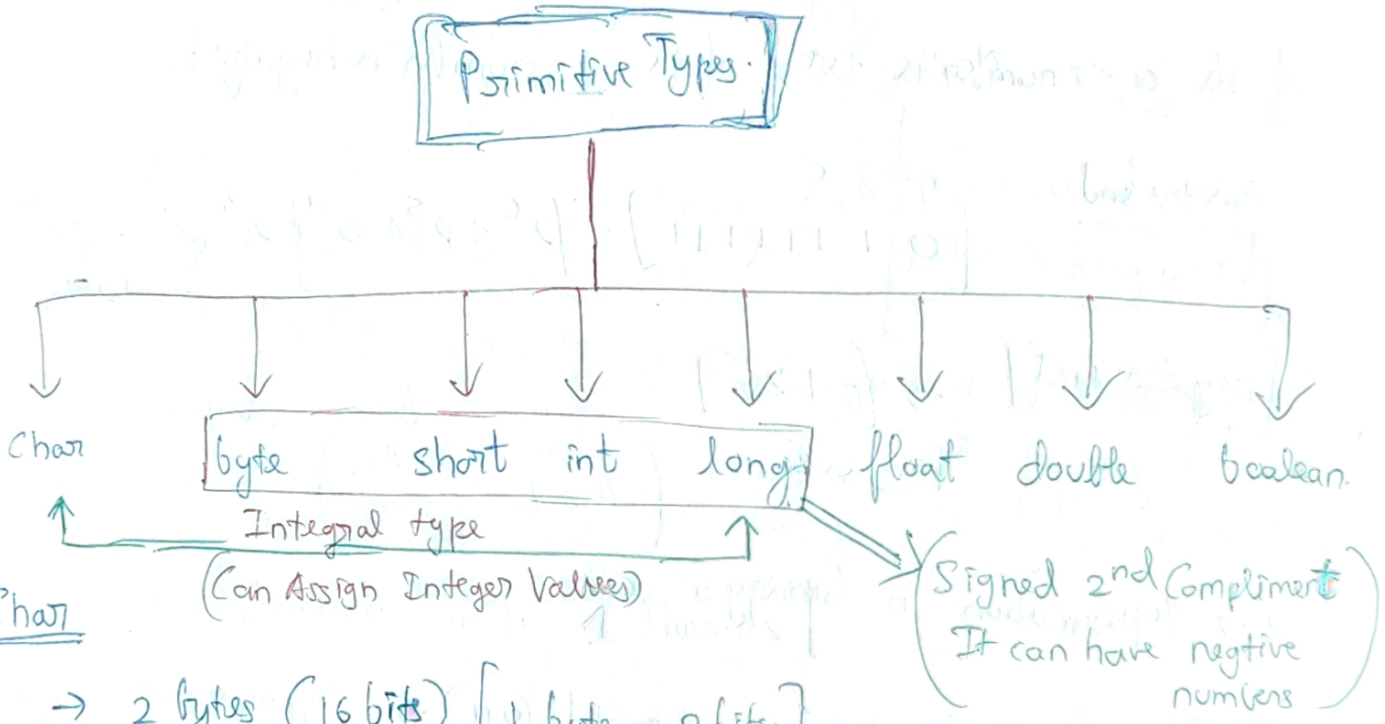
4. Java Variables | Primitive Data Types In Depth.

Variable Naming Convention.

* Variables can start with $\boxed{\$, \_, \text{letters}}$ only
* For constant, Variable name should be define in CAPITAL Letters,

```
static final int JAIPUR = 2.
```



Char

→ 2 bytes (16 bits) [1 byte = 8 bits.]

→ Range 0 to 2^{15}

i.e. 0 to 65535 $\quad 2^{15}$ - - - - - - - - | - - - - - - - - 2^{0}

i.e.

"\0000" to "\ffff"
(NUL)

byte

* 1 byte (8 bits)
* default value 0
* (Signed 2's compliment). $\underset{2^7}{\_}$ _ _ _ _ _ _ $\underset{2^0}{\_}$

(Explanation)

Imagine all the bits are 1, val = $2^7 + 2^6 + \dots + 2^0 = 255$

But instead we use 2's complitment. The 2^{7th} bit.

if its 0 → number is +ve, if its 1 number is negative,

positive end [0 1 1 1 1 1 1 1] = $2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$

$= 127$

[negative end] → [−128]

~~10000000~~

+3 → Representation in binary → 011

−3 → In binary is 2's compliment of (011) → 1st compliment 100 $+1$ $\boxed{101}$

Negative of a no is

2's compliment = (1's Compliment + 1)

Short

* 2 byte (16 bits)
* Signed 2's complement [Can store negative numbers]

int
→ 4 bytes (32 bits)
→ Signed 2's Compliment

long
→ 8 bytes (64 bits)
→ Signed 2's compliment.

Fractional Data Types

float → 32 bit IEE 754 value

double → 64 bit IEE 754 value

↕ ⟷ Explained in next video/pdf.

⇘ float/double {Not reliable for precision}

boolean → true/false.
* default value is false.

Types of Conversions

* Widening/Automatic Conversion.
* Narrowing/Downcasting/Explicit Conversion
* Promotion during Expression
* Explicit casting during Expression

1. Widening / Automatic Conversion [Lower to higher]

$$\begin{tikzcd}
\text{byte (1byte)} \arrow[d, bend left] \\
\text{short (2byte)} \arrow[d, bend left] \\
\text{int (4byte)} \arrow[d, bend left] \\
\text{long (8byte)}
\end{tikzcd}$$

Through lower data type to higher data type Conversion is automatic

```
int x = 10;
long x1 = x;
```
(automatic type casting happened)

2. Down Casting / Explicit Casting [Higher to lower]

$\boxed{\begin{array}{l}\text{int } x = 10; \\ \text{byte } x1 = x;\end{array}} \Rightarrow$ ERROR

Explicitly we have to downcast it

```
int x = 10;
byte x1 = (byte)(x);
```

{Issue with this}
will lead to errors in the case, where number is outside

```
int x = 128
byte b = x;
```

byte range → -128 --- 127

(127+1) will bring you here, Basically roams in a circle

output ~~b~~ = -128

3) Promotion during Expression. (Interesting)

(Case I)

```
byte a = 127
byte b = 1
```

$\boxed{\text{byte sum = a+b;}}$ → Compile ERROR

Ideally this should return (-128) because byte [-128 to 127]

because compiler internally promotes it to int since it exceeds byte Range

Solution 1 → $\boxed{\text{byte sum = (byte) (a+b)}}$ → (Explicit Casting)

Solution 2 → int sum = a+b.

(Case II) → When two variables are getting evaluated together and both are of different datatype. Then all the variables are promoted to the highest data type

```
int x = 10;
double y = 20D;
```

$\boxed{\text{int sum = x+y;}}$ → (Compile ERROR)

[Because ~~x~~ has ~~x~~ ~~been~~ been promoted to double]

(Solution 1) double sum = x+y

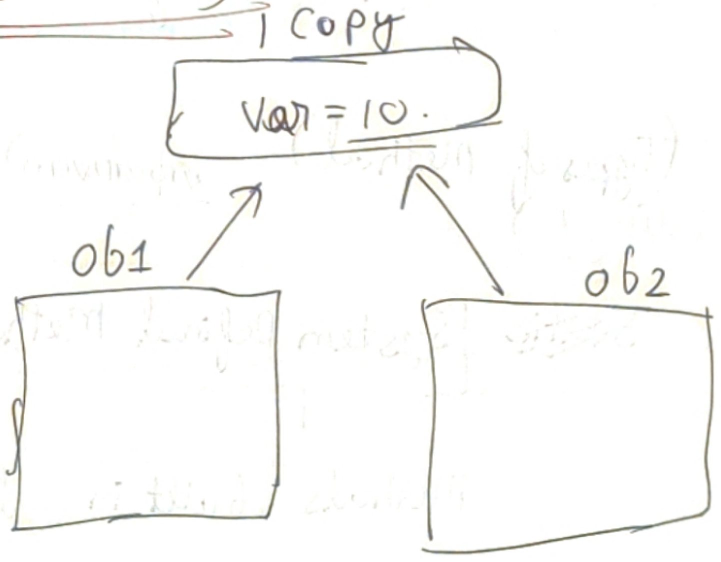
(Solution 2) int sum = int (x+y); → Explicit Casting.

$\boxed{\text{Static Variable}}$ (Class Variable)

```
class Employee {

    static int var=10;

}.
```



~~[illegible]~~ Static Variables can only be accessed by ClassName.

Employee. var