Operators and Expression

```
byte   b=10;
short  s=15;
int    i=7;
long   l=50l;
float  f=12.5f;
double d=17.5d;
char   c=65;
```

int

```
byte + byte
short + short
byte + short
int + int
short + int
```

Jab ham ine mai sai kisi bhi type par koi operation karenga tab always hama integer value hi milta hai as a reult.

Coercion

int $X = b + s$ int $X = c + s;$

int $X = s + i$ int $X = c + i;$

float $X = \underset{f}{i} + f$ double $X = f + d;$

float $X = \underset{f}{l} + f$ double $X = \underset{d}{l} + d;$

Increment / Decrement

post ++ , post --

++pre , --pre

Arithametic

$*, /, \%$ $\longrightarrow$ **High precedense**

$+, -$ $\longrightarrow$ **Low precedense**

Bitwise

&, |, ~, ^, <<, >>, >>>

Relational

<, <=, >, >=, ==, !=

Logical

&&, ||, !

⊙ Post increment

Ise mai phela value use hota ha then update hota hai.

⊙ Pre increment

Ise mai phela value update hota ha then use hota hai

Operators and Expression

```
int x=5;

x++; - 6
++x; - 7
```

```
float x=5.3;

x++;
6.3
```

```
byte x=4;

x++;
5
```

Post increment

```
int x=5,y;

y=x++;
y=5 x=6
```

Pre increment

```
int x=5,y;
y=++x;
6   6
y=6 x=6
```

```
char x='A';
          65
x++; 66 'B'
S.o.p(x);
   B
```

```
int a=2,b=5,x=4,c
c=a*++x+b
```

Increment / Decrement

post ++ , post --

++pre , --pre

Arithametic

*, /, %

+, −

Bitwise

&, |, ~, ^, <<, >>, >>>

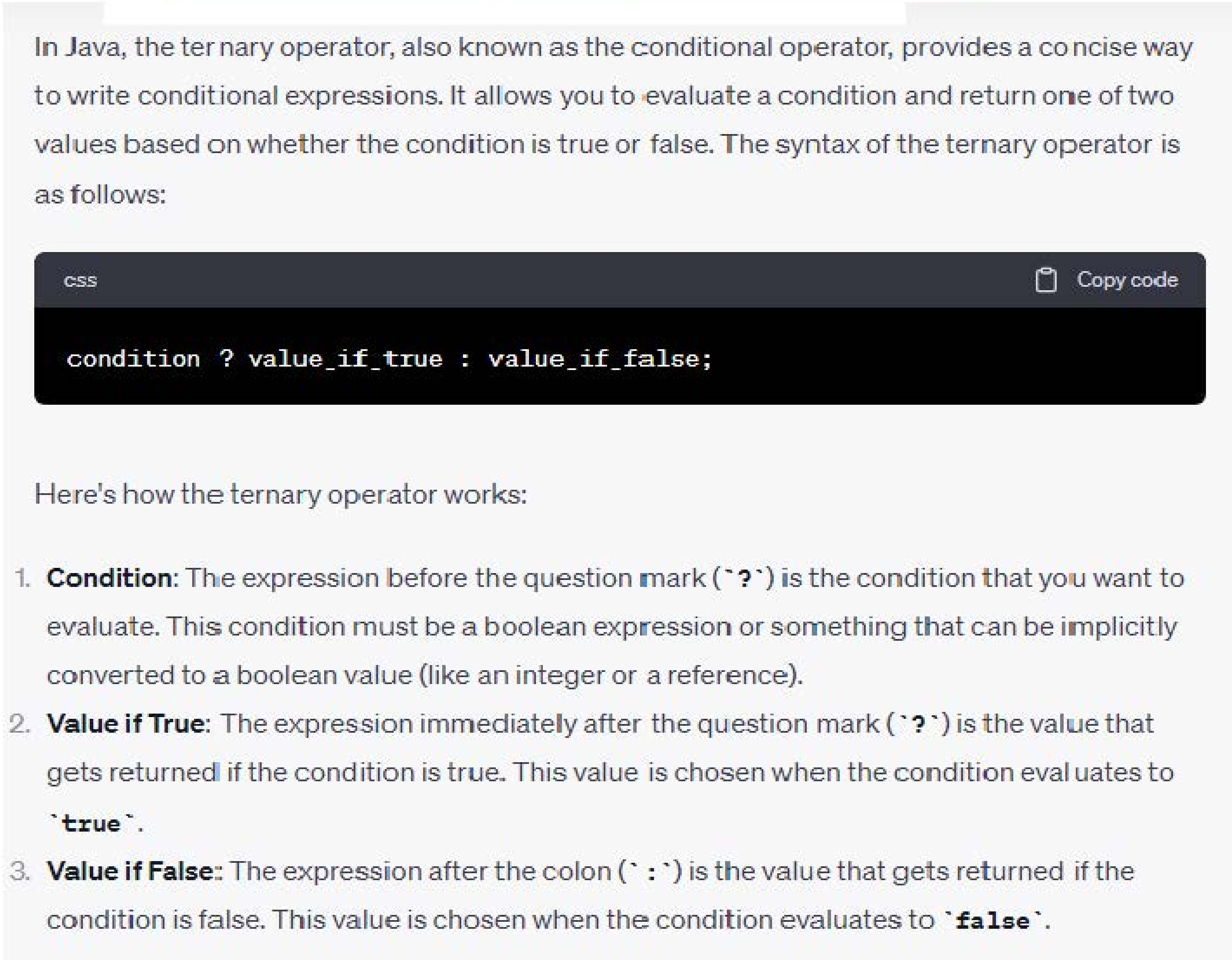
Relational

<, <=, >, >=, ==, !=

Logical

&&, ||, !

Ternary Operator



In Java, the ternary operator, also known as the conditional operator, provides a concise way to write conditional expressions. It allows you to evaluate a condition and return one of two values based on whether the condition is true or false. The syntax of the ternary operator is as follows:

```css
condition ? value_if_true : value_if_false;
```

Here's how the ternary operator works:

1. **Condition**: The expression before the question mark (`?`) is the condition that you want to evaluate. This condition must be a boolean expression or something that can be implicitly converted to a boolean value (like an integer or a reference).
2. **Value if True**: The expression immediately after the question mark (`?`) is the value that gets returned if the condition is true. This value is chosen when the condition evaluates to `true`.
3. **Value if False**: The expression after the colon (`:`) is the value that gets returned if the condition is false. This value is chosen when the condition evaluates to `false`.

Here's an example that uses the ternary operator to determine whether a given number is positive or not:

java Copy code

```java
int number = -5;
String result = (number > 0) ? "Positive" : "Not Positive";
System.out.println(result);
```

In this example:

- The condition is `(number > 0)`, which checks if `number` is greater than 0.
- If the condition is true, `"Positive"` is assigned to `result`.
- If the condition is false, `"Not Positive"` is assigned to `result`.

The ternary operator is often used for short and simple conditional assignments or expressions. However, it's important to use it judiciously; overly complex expressions can lead to code that is difficult to read and understand.

Chapter 2 - operators and Expressions

Operators are used to perform operations on variables and values.

$$\begin{tikzcd}
7 \arrow[d] & + \arrow[d] & 11 \arrow[d] & = & 18 \arrow[d] \\
\text{operand} & \text{operator} & \text{operand} & & \text{Result}
\end{tikzcd}$$

Types of operators

→ Arithmetic Operators → +, -, *, /, %, ++, --
→ Assignment operators → =, +=
→ Comparison operators → ==, >=, <=
→ Logical operators → &&, ||, !
→ Bitwise Operators → &, | (operates bitwise)

Arithmetic operators cannot work with booleans
% operator can work on floats & doubles

Precedence of operators

The operators are applied and evaluated based on precedence. For example (+, -) has less precedence compared to (*, /). Hence * & / are evaluated first.
In case we like to change this order, we use parenthesis

Associativity

Associativity tells the direction of execution of operators
It can either be Left to Right or Right to left

* /	→ L to R
+ -	→ L to R
++, =	→ R to L

Quick Quiz : How will you write the following expressions in Java ?

$\frac{x-y}{2}$, $\frac{b^2-4ac}{2a}$, v^2-u^2 , $a*b-d$

Resulting data type after arithmetic operation

following table summarizes the resulting data types after arithmetic operation on them

R = b + s → int		b → byte f → float
R = s + i → int		s → short d → double
R = l + f → float		i → integer c → character
R = i + f → float		l → long
R = c + i → int		
R = c + s → int		
R = l + d → double		
R = f + d → double		

Increment and Decrement Operators

a++ , ++a → Increment operators → Data type

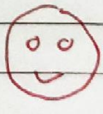
a-- , --a → Decrement operators → remains same

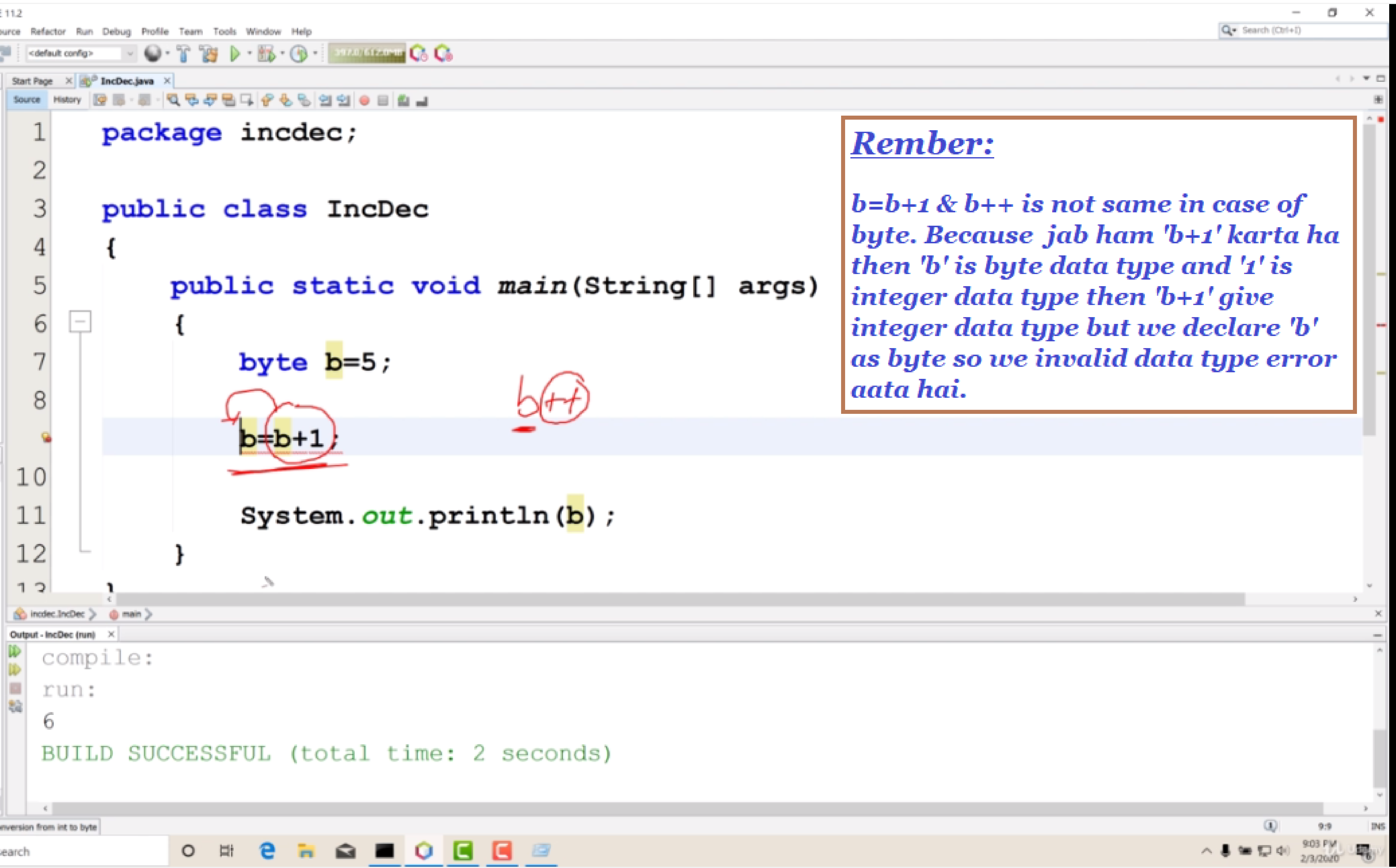
These will operate on all data types except booleans

Quick Quiz : Try increment and decrement operators on a Java Variable

a++ → first use the value and then increment

++a → first increment the value then use it





```java
package incdec;

public class IncDec
{
    public static void main(String[] args)
    {
        byte b=5;

        b=b+1;

        System.out.println(b);
    }
}
```

Rember:

b=b+1 & b++ is not same in case of byte. Because jab ham 'b+1' karta ha then 'b' is byte data type and '1' is integer data type then 'b+1' give integer data type but we declare 'b' as byte so we invalid data type error aata hai.

```
compile:
run:
6
BUILD SUCCESSFUL (total time: 2 seconds)
```

Bitwise Operators

AND	$\wedge$	&
OR	$\vee$	|
NOT	$\neg$	~
XOR	$\oplus$	^
RIGHT SHIFT		>>
UNSIGNED RIGHT SHIFT		>>>
LEFT SHIFT		<<

&

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

\|

A	B	A|B
0	0	0
0	1	1
1	0	1
1	1	1

^

A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

Bitwise Operators

```
int x=10, y=6, z;
```

x → 0 0 0 0 1 0 1 0

y → 0 0 0 0 0 1 1 0

z = x & y   0 0 0 0 0 0 1 0

2

&

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

¦

A	B	A¦B
0	0	0
0	1	1
1	0	1
1	1	1

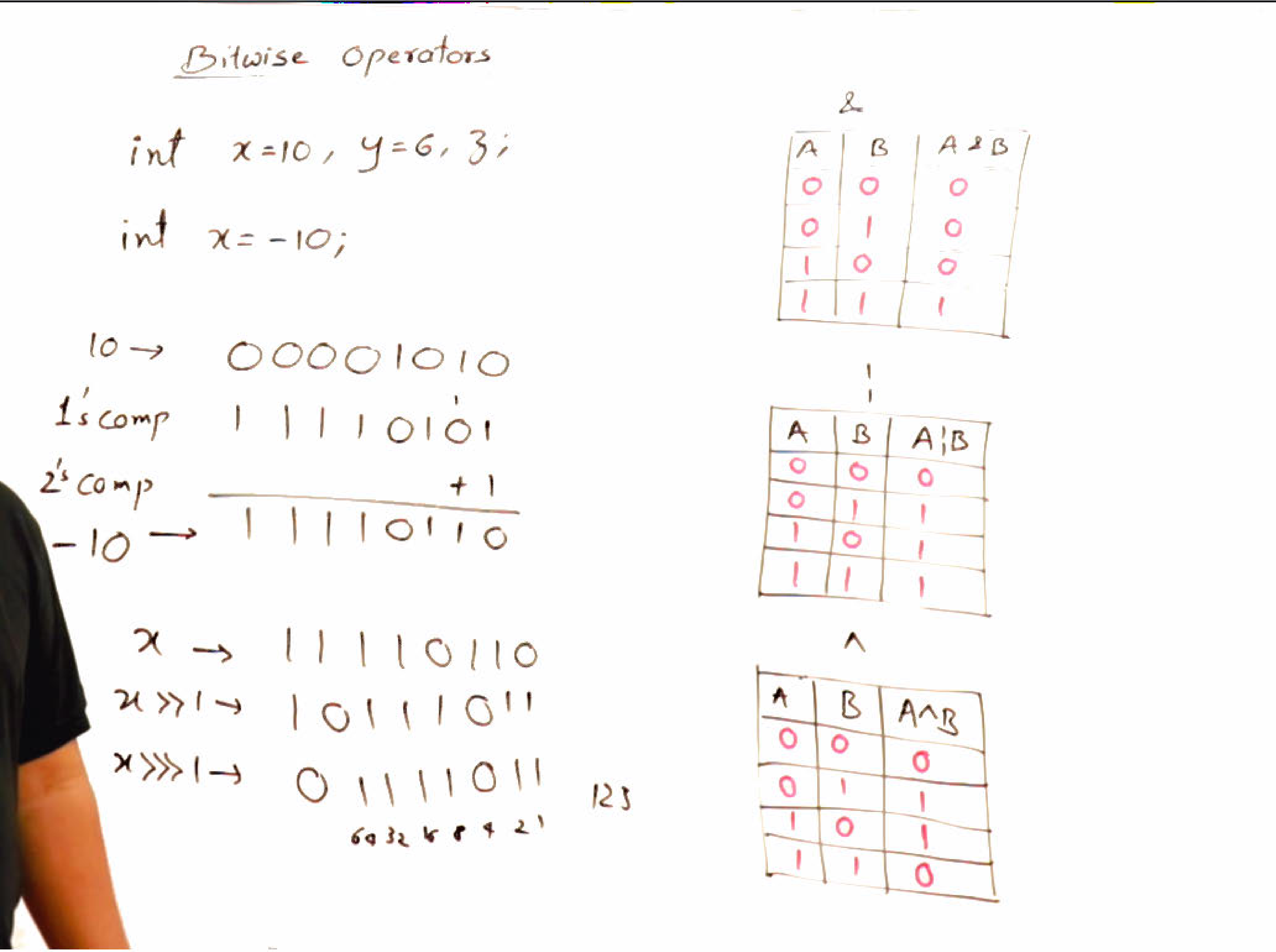
^

A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

$$\begin{array}{rcccccccc}
x \rightarrow & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
y \rightarrow & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
\hline
 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
\cline{6-9}
 & & & & & 8 & 4 & 2 & 1
\end{array}$$

$z = x \mid y$

$z = 14$



Bitwise Operators

int x=10, y=6, 3;

int x=-10;

10 → 00001010

1's comp 11110101

2's comp +1

-10 → 11110110

x → 11110110

x>>1 → 10111011

x>>>1 → 01111011 123

64 32 16 8 4 2 1

&

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

\|

A	B	A|B
0	0	0
0	1	1
1	0	1
1	1	1

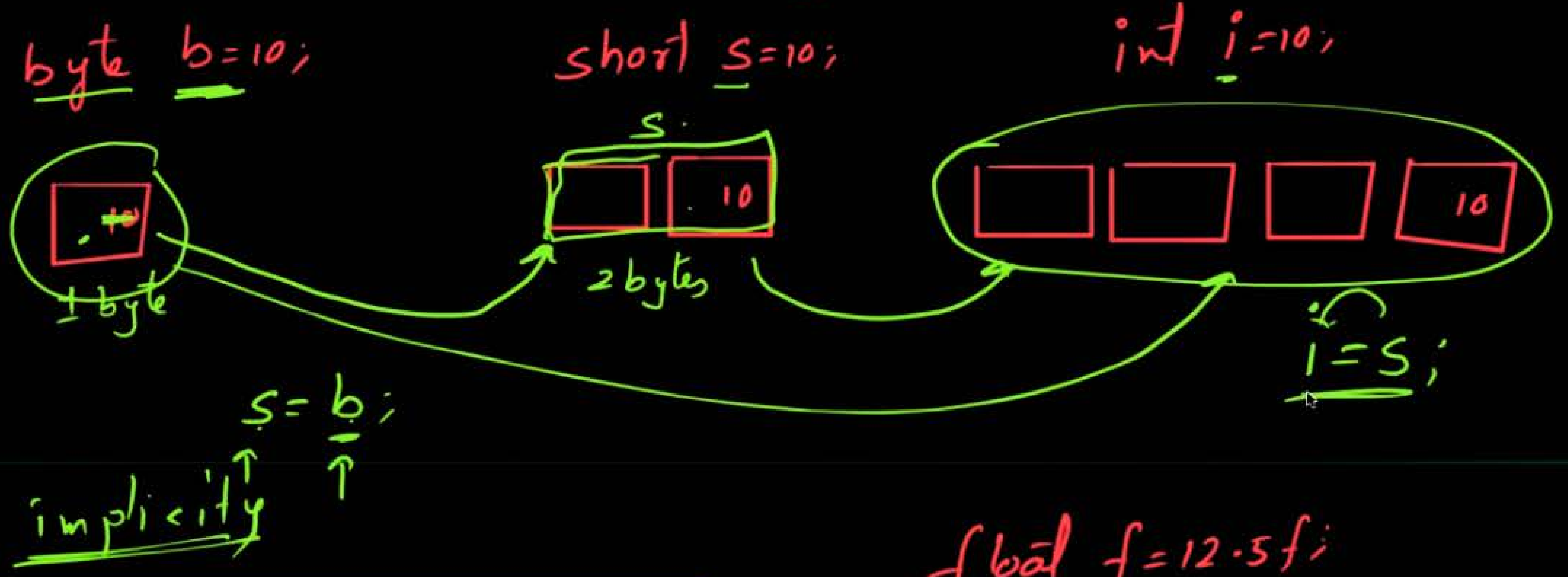
^

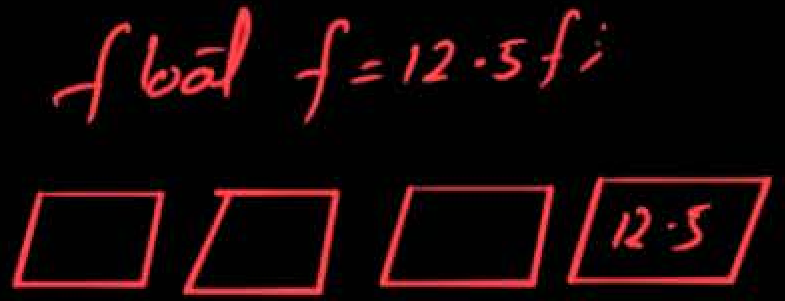
A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

⊗ Java mai koi bhi '-ve' integer hmasa 2's compliment ka form mai save hota hai.

⊗ <u>Note:</u> 1's compliment ka liya ham phela sabhi value koi uska opp. digit sa replace kar deta hai. Then for 2's compliment add '1' in 1's compliment.

Widening and Narrowing





※Widening (Implicit Conversion):
Widening is a process where a value of a smaller data type is automatically converted into a value of a larger data type without any data loss.

※Narrowing (Explicit Conversion or Type Casting):
Narrowing is a process where a value of a larger data type is manually converted into a value of a smaller data type. Since this conversion may lead to data loss or loss of precision.

Deslination left side ↷ =

Souce

right side ↓ =

	byte	short	int	long	float	double	char	boolean
byte	✓	✓	✓	✓	✓	✓	✗	~~✗~~
short	✗	✓	✓	✓	✓	✓	✗	?
int	✗	✗	✓	✓	✓	✓	✓	✗
long	✗	✗	✗	✓	✓	✓	✗	✗
float	✗	✗	✗	✗	✓	✓	✗	✗
double	✗	✗	✗	✗	✗	✓	✗	✗
char	✗	✗	✓	✓	✓	✓	✓	✗
boolean	✗	✗	✗	✗	✗	✗	✗	✓

※. Ise table mai ya show kiya gya hai ki ham kis data type ko kisi mai easily convert kar sakta hai.