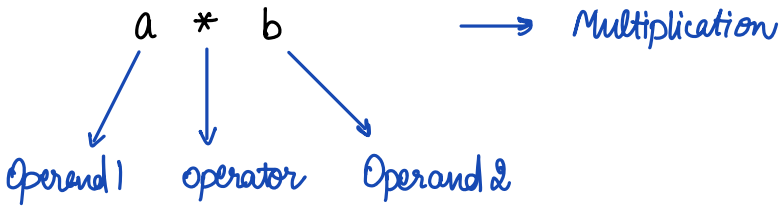
Starting at 7:05 am

Enjoy the song !!

Today's Content

1. Typecasting Revision
2. Rules doing Basic Operations
3. Integer Overflow
4. Operators [Arithmetic, Relational, Logical, Assignment, Unary]



Rule 1: When we do operation between non-decimal [int/long] and decimal [float/double] the output is decimal.

int (op) double ⟶ double
long (op) float ⟶ float

Rule 2: When we do operation between two operands of same category the result is of bigger type

int (op) long ⟶ long
float (op) double ⟶ double
int (op) int ⟶ int

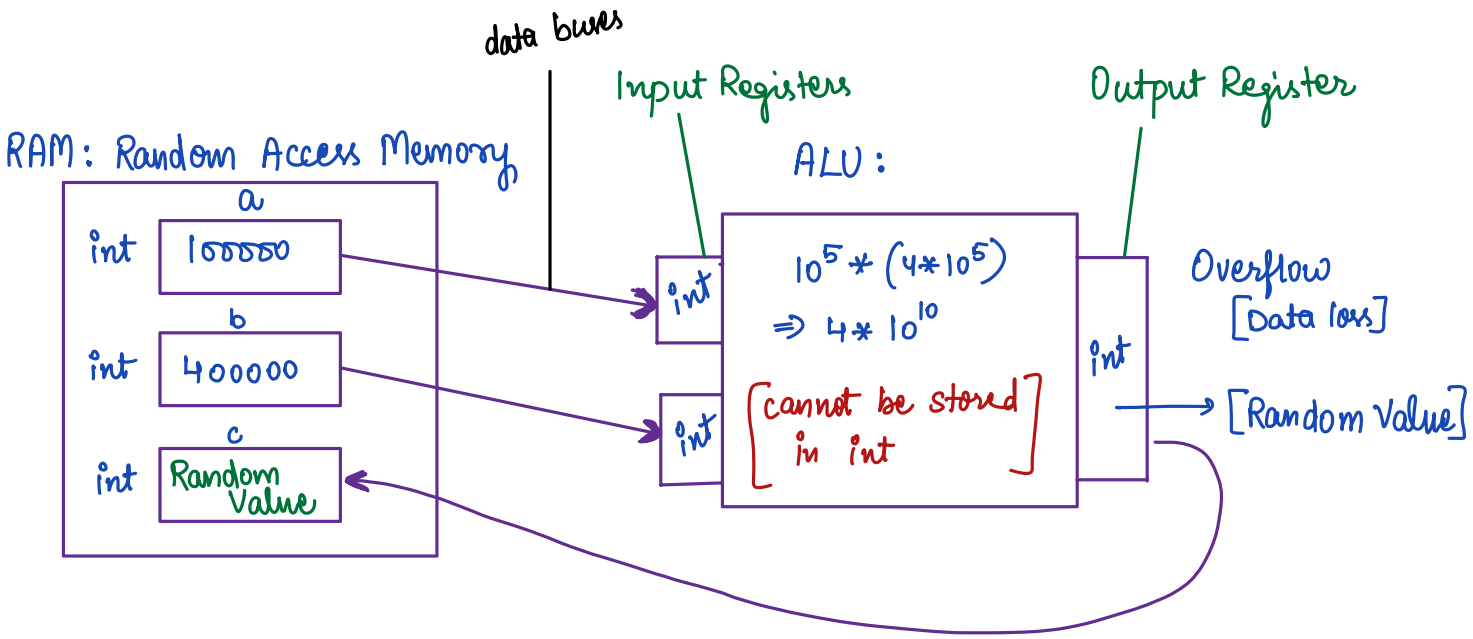
Quiz-10

```
int a = 100000;
int b = 400000;

int c = a*b;        [int * int ⟶ int]
System.out.print(c); ⟶ Random Value
```

Central Processing Unit [C.P.U]

1. ALU ⟶ Arithmetic and Logic Unit
2. Control Unit
3. Registers

Random Access Memory [R.A.M]



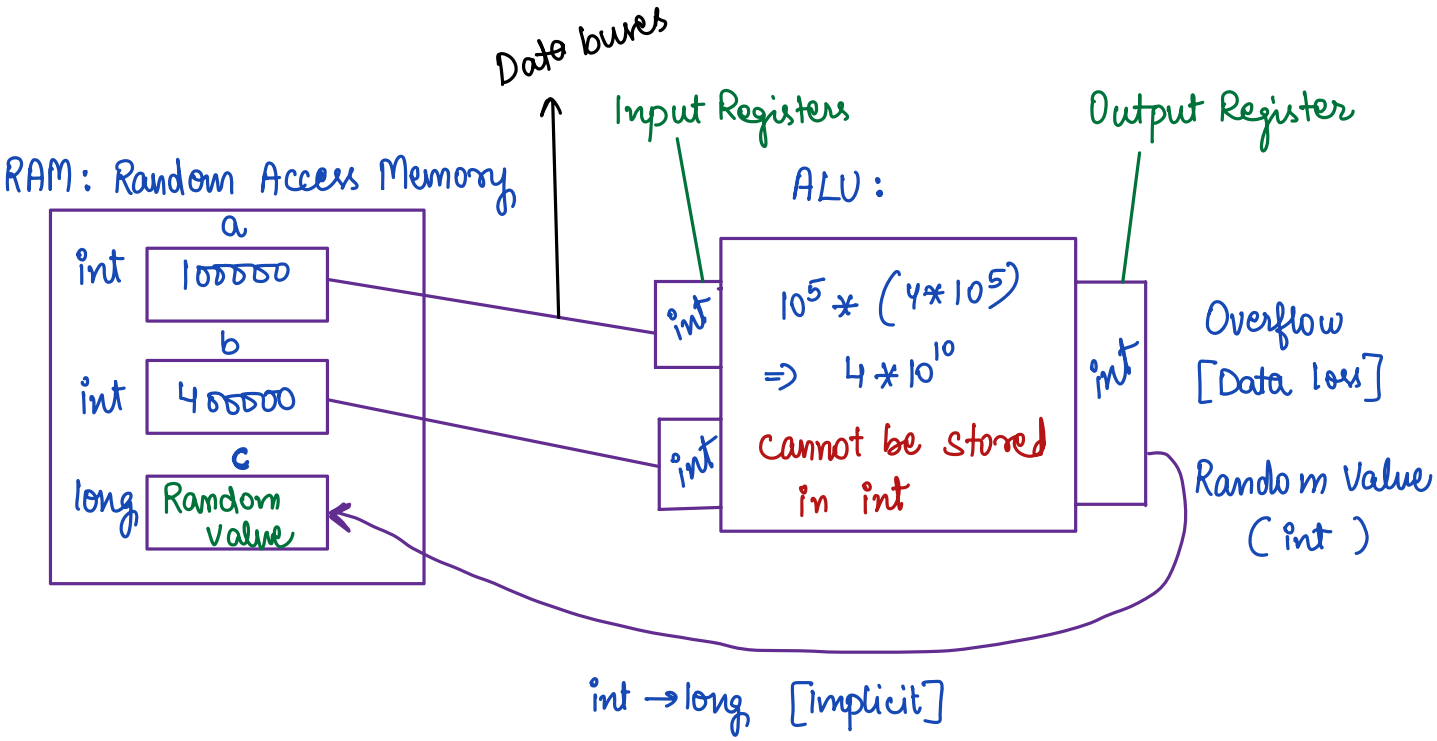
Quiz 11.

```
int a = 100000;
int b = 400000;
long c = a * b;
System.out.print(c);
```

int

long c = a * b; ⟶ [int * int ⟶ int]

System.out.print(c); ⟶ Random Value

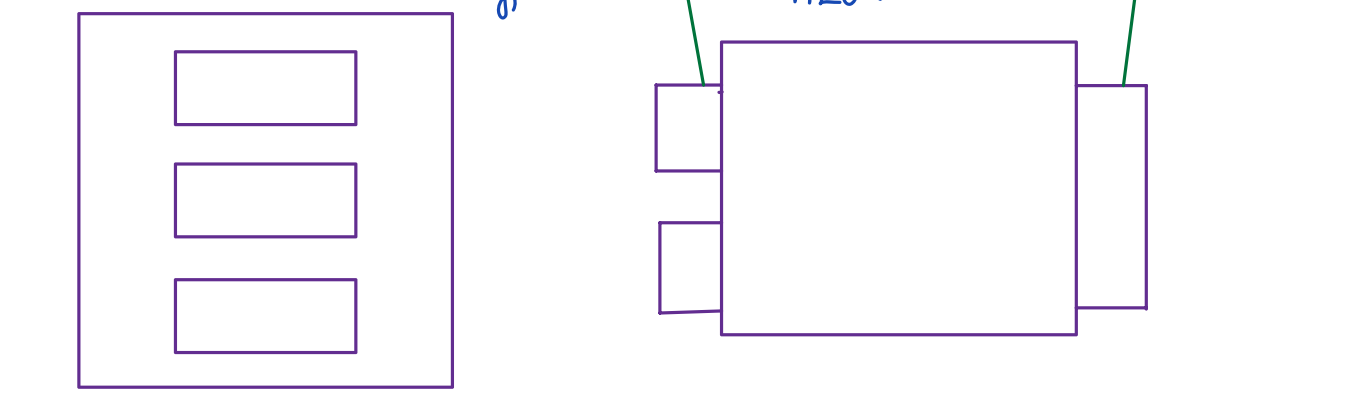


Quiz 12.

```
long a = 100000;
long b = 400000;
int c = a * b;
System.out.print(c);
```

Error: possible lossy conversion from long to int

[long * long → long]

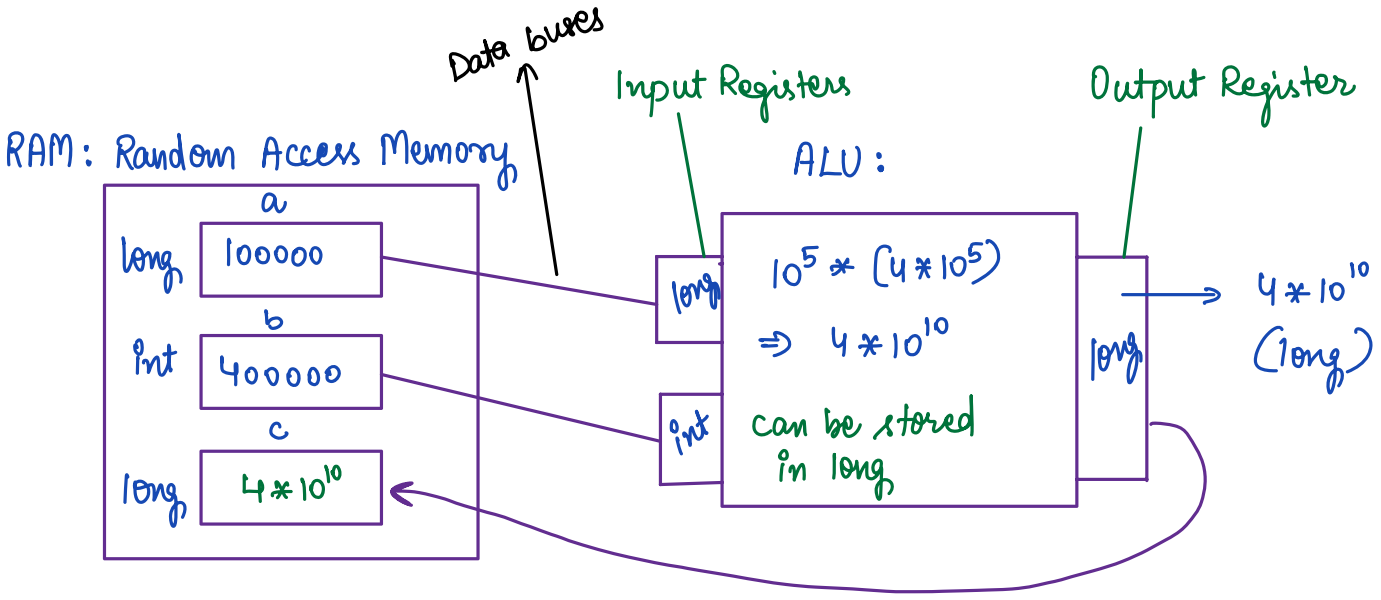


Quiz 13. `long a = 100000;`

`int b = 400000;`

`long c = a * b;` → [long * int → long]

`System.out.print(c);` → 4000000000



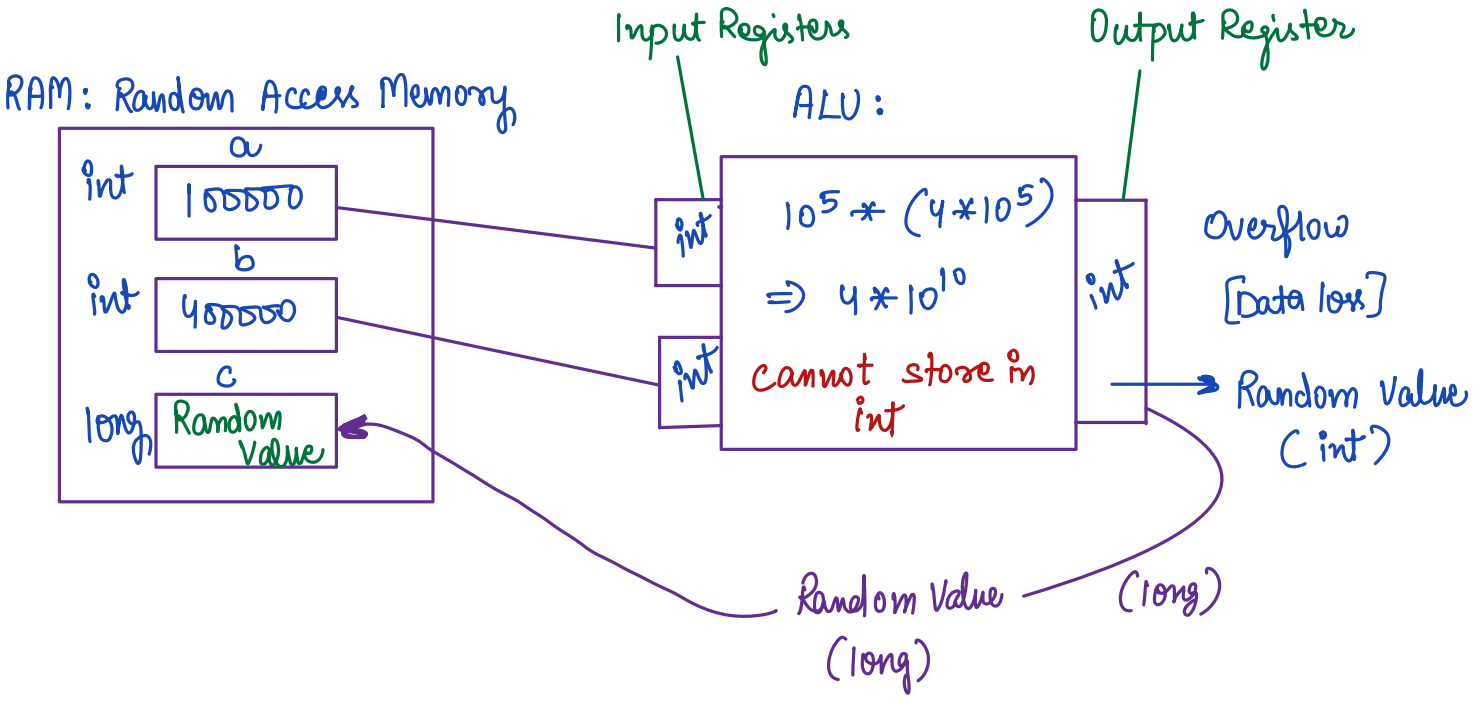
Quiz 14.

```
int a = 100000;
int b = 400000;
long c = (long) (a*b);
System.out.print(c);
```

→ Typecast | product of a & b | to long

→ Random Value

int * int → int



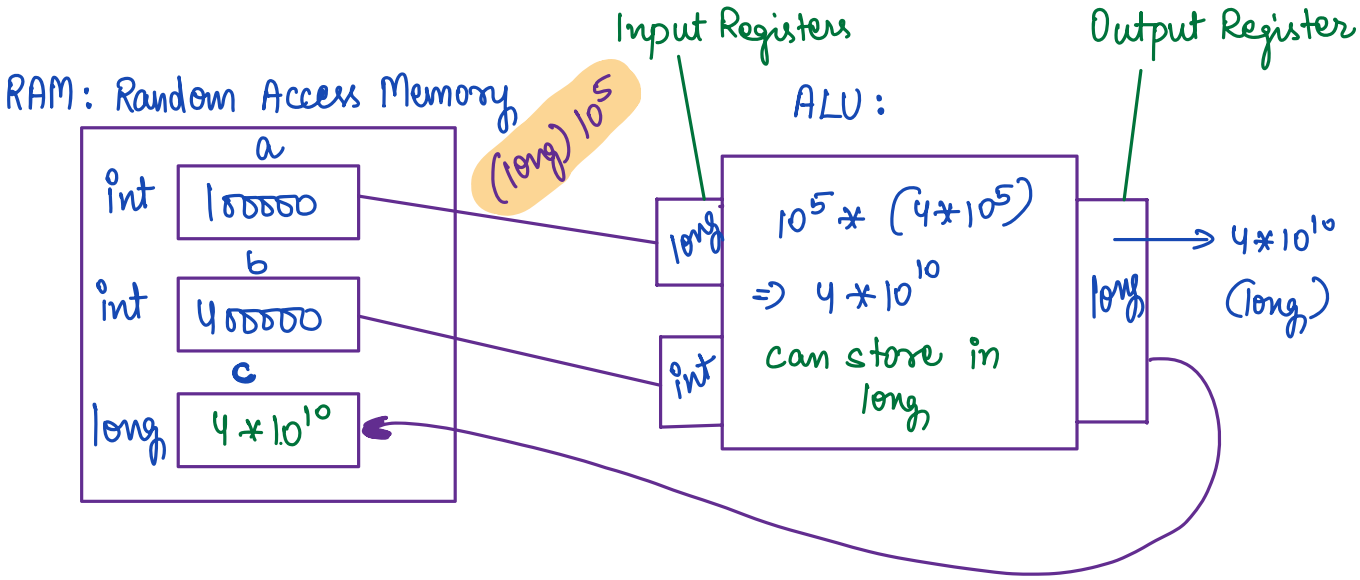
Quiz 15.

```
int a = 100000;
int b = 400000;
long c = (long) a * b;
System.out.print(c);
```

→ Typecast the value of a to long and then multiply with b

→ 4000000000

long * int → long



a * (long) b → int * long → long

(long) a * (long) b → long * long → long

Operators :

- → Arithmetic operators
- → Relational operators
- → Logical operators
- → Assignment operators
- → Unary operators

Arithmetic operators → + , − , / , * , %

```
int a = 10 , b = 24 ;
```

a + b → 10 + 24 = 34

a - b → 10 - 24 = -14

a * b → 10 * 24 = 240

b / a → 24 / 10 = 2

Modulus operator (%) (gives remainder)

12 % 4 = 0

24 % 5 = 4

36 % 6 = 0

5 % 3 = 2

Relational operators → compare two values [true / false]

		a = 45 , b = 14	a = 5 , b = 5
a is greater than b	a > b	true	false
a is less than b	a < b	false	false
a is greater than or equals to b	a >= b	true	true 5 > 5 ✗ 5 == 5 ✓
a is less than or equals to b	a <= b	false	true 5 < 5 ✗ 5 == 5 ✓
a is equals to b	a == b	false	true
a is not equals to b	a != b	true	false

Logical operators → AND OR NOT

1.) Driver's License – a) age should be greater than or equal to 18
b) should know how to drive.

age >= 18 [AND] know how to drive Driver License

true	true	true
true	false	false
false	true	false
false	false	false

AND [&&] → Both conditions should be true to get answer as true.

Note: If the first condition is false with (&&) operators, the evaluation of second is skipped

a.) Eligibility for exam - you need either have diploma or degree

Diploma ?	[OR]	Degree ?	Eligible for exam
true		true	true
true		false	true
false		true	true
false		false	false

OR [||] → Even if one condition is true the answer is true.

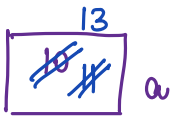
Note : If the first condition is true with (||) operators, the evaluation of second is skipped

NOT operator (!)

A	!A
true	false
false	true

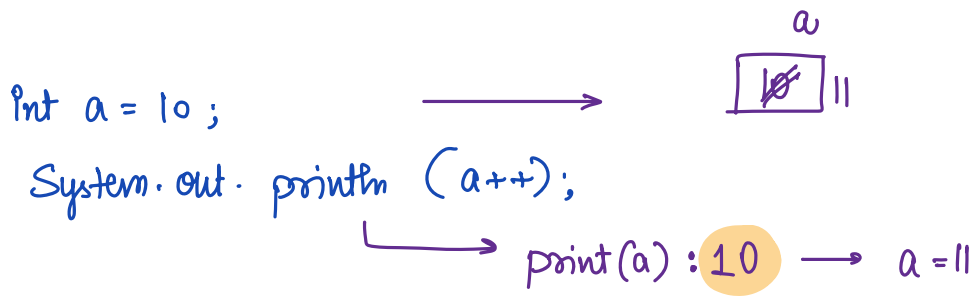
Assignment operators - = , += , -= , *= , /= , %=

```
int a = 10;
```



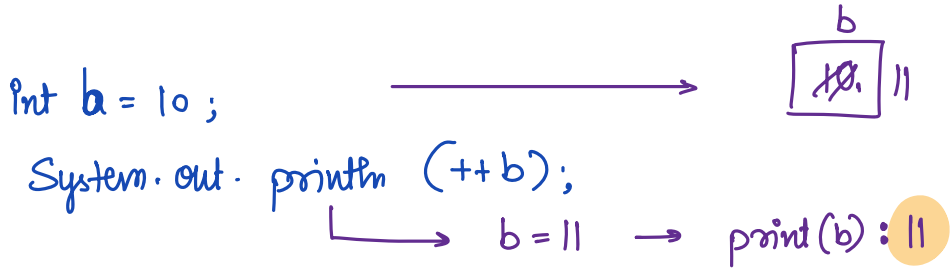

```
a = a + 1 ;      // a = 10+1 => a = 11
System.out.println (a);      ⟶ 11
a = a + 2 ;      // a = 11+2 ⇒ a = 13
System.out.println (a);      ⟶ 13

a += 5 ;    // a = a+5;
   ↳ increment value of a by 5
System.out.println (a);      ⟶ 18

a -= 4 ;    //decrease value of a by 4
System.out.println (a);      ⟶ 14

a *= 2;     //multiply value of a by 2
System.out.println (a);      ⟶ 28

a /= 4 ;    // divide the value of a by 4
System.out.println (a);      ⟶ 7

a %= 5 ;    // a = a %5  ⇒ a = 7 %5  ⇒ a = 2
System.out.println (a);      ⟶ 2
```

Unary operators →



Post increment : a++

↳ use the value first and then increment by 1



Pre increment : ++b

↳ increment by 1 and then use the value

Post decrement : x--

↳ first use the value and decrease by 1

Pre decrement : --x

↳ first decrease by 1 and then use the value.

Q1. `int a = 20;` → a: ~~20~~ 21

`int b = a++;` → use b: 20 and then increment a → 21

`System.out.println(a+b);`

↳ 21 + 20 = 41

Q2. `int a = 10;` → ~~10~~ 11 a

`int b = 20;` → ~~20~~ 21 b

`int c = a++ + b++;`

// c = 10 [a→11] + 20 [b→21] ⇒ c = 30

`System.out.print(a+b+c);`

↳ 11 + 21 + 30 ⇒ 62

Q3. `int a = 10;` → ~~10~~ ~~11~~ ~~12~~ 13 a

`int b = a++ + a++ + a++;`

// b = 10 [a→11] + 11 [a→12] + 12 [a→13] ⇒ b = 33

`System.out.println(b);`

↳ 33

`int c = b++;`

// c = 33 [b→34]

`System.out.print(a+b+c);`

↳ 13 + 34 + 33 ⇒ 80

Priority in operators →

Java Operator Precedence and Associativity

Operators	Precedence	Associativity
postfix increment and decrement	++ --	left to right
prefix increment and decrement, and unary	++ -- + - ~ !	right to left
multiplicative	* / %	left to right
additive	+ -	left to right
shift	<< >> >>>	left to right
relational	< > <= >= ~~instanceof~~	left to right
equality	== !=	left to right
bitwise AND	&	left to right
bitwise exclusive OR	^	left to right
bitwise inclusive OR	|	left to right
logical AND	&&	left to right
logical OR	||	left to right
ternary	? :	right to left
assignment	= += -= *= /= %= ~~&= ^= |= <<= >>= >>>=~~	right to left

```
float x = 5/2;
```

2

x = 2.0

```
double
  ↑
float a = 8.2/2;
          ↓
         int
```

double / int ⟶ double

Error : possible lossy conversion

```
int n = scn.nextInt();
```

[Implicit] `float n = scn.nextInt();` (int)

non-decimal → decimal

[Implicit]
```
double n= scn.nextFloat();
float n = scn.nextDouble();
```

Error : possible lossy conversion