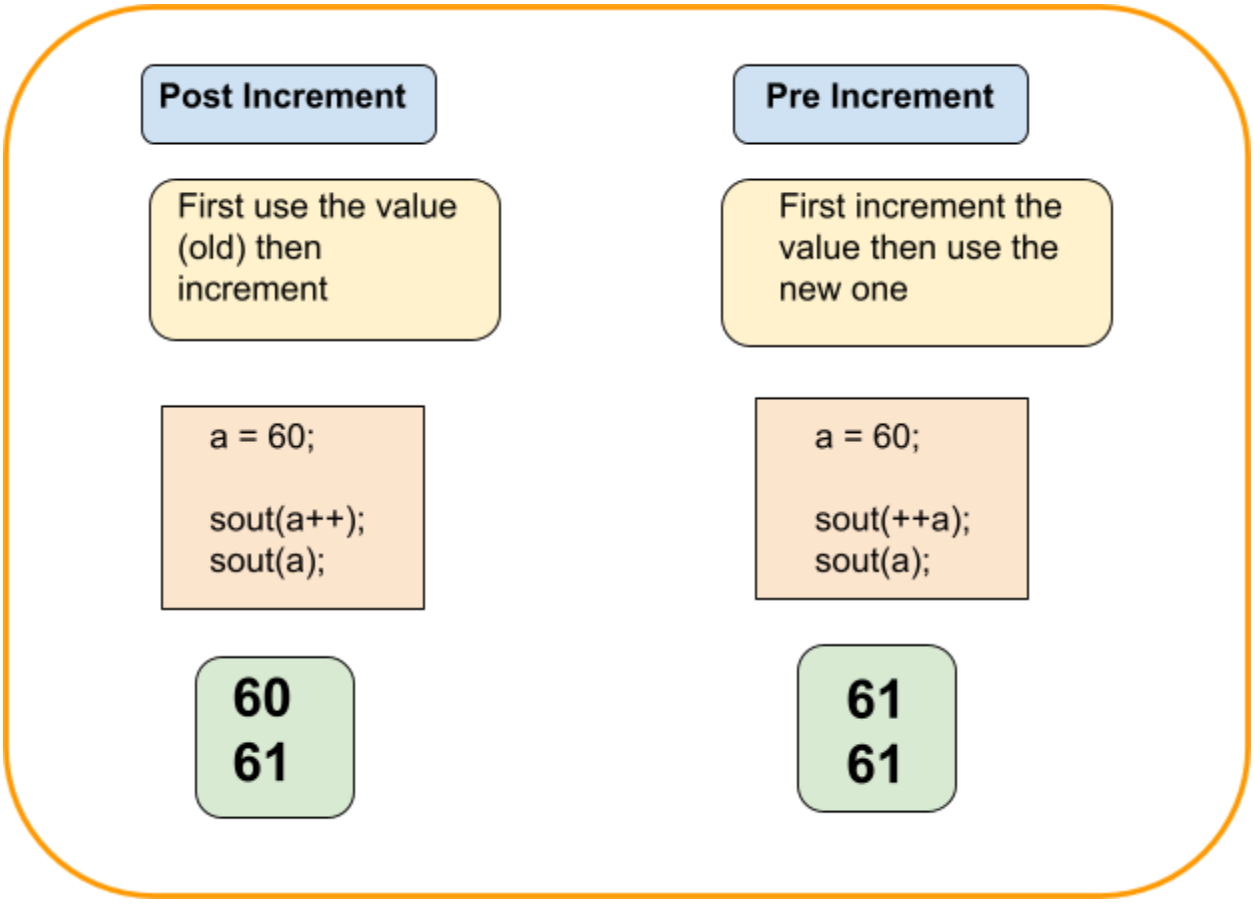
Operators and for loop

Increment and Decrement Operators



Post decrement

First use the value (old) then decrement

```
a = 60;

sout(a- -);
sout(a);
```

60
59

Pre decrement

First decrement the value then use the new one

```
a = 60;

sout(- -a);
sout(a);
```

59
59

Bitwise Operators

Bitwise operators work bit by bit….

int a = 10, b = 2;

AND	a & b	Performs AND operation between bits of a and b	2
OR	a | b	Performs OR operation between bits of a and b	10
XOR	a ^ b	Performs XOR operation between bits of a and b	8
NOT	~a	Toggles every bit of a	-11
Left Shift	a << n	Performs arithmetic left shifts n time on bits of a	$10 * 2^n$
Right Shift	a >> n	Performs arithmetic right shifts n time on bits of a	$10/2^n$
Unsigned right shift	a >>> n	Performs arithmetic right shifts n time on bits of a (instead of sign bit it inserts 0)	$10/2^n$

Assignment Operators

---

int a = 10

a += 10	a = a + 10	20
a -= 2	a = a - 2	8
a *= 3	a = a * 3	30
a /= 5	a = a / 5	2
a ^= 1	a = a ^ 1	11
a <<= 1	a = a << 1	20

Precedence & Associativity

Operator Precedence

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

All binary operators except for the assignment operators are evaluated from left to right; assignment operators are evaluated right to left.

Example:

Which line(s) of the following code would give an error:

```
public static void main (String[] args) {
   int a=10,b=20;
   System.out.println(a+++--b);//line 1
   System.out.println(a--+++b);//line 2
   System.out.println(a++-++b);//line 3
   System.out.println(a+++++b);//line 4
}
```

Solution Description

As + operator and ++ operator have equal precedence.
Similarly - and -- have equal precedence.
So in line 2 +++b and in line 4 +++b gives an error.
To solve this error we can use +(++b).

For Loop

```
for(initialization; test; change){
      statements;
}
```

Note:

- It is an Entry controlled loop, meaning we will only enter the loop if the condition(test) is true.

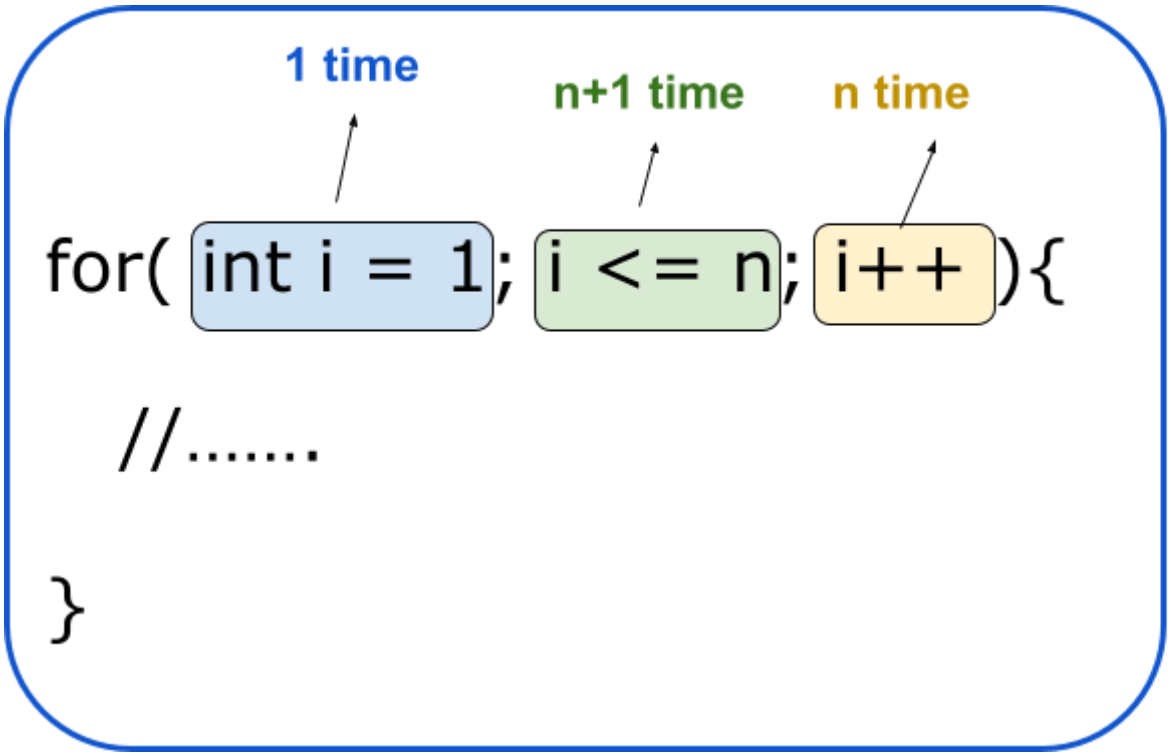
Common Errors:

- For arrays
    - `for(int i=0;i<n;i++)` —----> correct

- `for(int i=0;i<=n;i++)` —----->wrong

- Make sure that your change variable is correct otherwise you will get TIME LIMIT EXCEEDED error

When to use for loop?
-> when we know the exact number times for which we want our loop to run



Enhanced for loop

```
for(data_type element_variable: name_of_collect/array){
      statements;
}
```

```
String[] arr = {“A”, “B”};

for(String curr: arr){
   System.out.println(curr);
}
```

```
 for (int i = 1, j = 100; i <= 5 ; i++, j+= 20) {
            System.out.println(i+" "+j);

       }


    //    1 100
    //    2 120
    //    3 140
    //    4 160
    //    5 180


 for (int i = 1, j = 100; i <= 5 && j <= 200; i++, j+= 40) {
            System.out.println(i+" "+j);

       }
        // 1 100
        // 2 140
        // 3 180
```

Break Keyword

Break statement will exit you out of the loop(immediate loop).

Continue keyword

Continue skips the current iteration of the loop