Date ___/___/___
Page ___

* Type Casting

(Done by compiler) Implicit — Explicit (Done by programmer)

Implicit :- When we come Top to Bottom (Bottom to Top) means lower data type to higher.

Ex → ~~int~~ ~~num~~ ~~float~~ int num1 = 45;
~~int~~ ~~num~~ float n;
n = num1; (Implicitly done by compiler).
sout(n);

output
45.0

int → float.
lower → higher

Explicit : While going lower to higher, to lower.
Compiler ~~System~~ can't do that so we explicitly cast into lower data type.

Ex → float n1 = 46.7
int n;
n = (int) n1;

(Correct syntax)

float n1 = 46.7
int n;
n = n1

(wrong syntax)

* Operators

- Arthemetic Operators :

1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/) gives Quotient
5. Modulo (%) gives reminder

Date / /
Page

Increment and Decrement Operators:

- ++ (increase value by 1)
- -- (decrease value by 1)

Ex→
```
a = 5;
a++;    // a = 6
a--;    // a = 5
```

post Increment (a++): It use the value first than update value.

Ex→
```
int a = 5;
int b = a++;
```
Output:
a = 6
b = 5 // Now if b = 6 after output.

preincrement (++a): It update value first than use it.

Ex→
```
int a = 5;
int b = ++a;
```
Output
a = 6
b = 6

Ex→
```
int a = 5;
int b = (a++) + (++a) + (--a);
```
(5) + (7) + (6)

Output
a = 6 // $a\overset{\rightarrow}{++}^{6}$, $++\overrightarrow{a}^{7}$, $--\overrightarrow{a}^{6} = 6$
b = 18 // 5 + 7 + 6

Ex → int a = 5;
int b = (a++) + (++a) + (++a) + (a++) +
(--a) + (a--);
5 7 8 8 8 8

Output;
a : 7;
b : ~~422~~ 44;

* Ternary Operator

Syntax = (condition) ? a : b ;

If True than a execute.

If false than b execute.

Example:- int a = 10;
int b = 20;
int res = (a > b) ? a : b; // b goes in res.
print(res);

Output
20

Example 2:-
int num = 1;
int num2 = 4;
System.out.println((num < num) ? num : num2);

Output:
1

Example 3

```
int a = 5;
int b = 10;
String result = (a > b) ? "a is greater" :
                          "b is greater";

Sout(result);
```

Output

b is greater.

Example 4: Greatest in a, b, c.

```
int a = 10;
int b = 20;
int c = 15;
int result = (a > b) ? (a > c) ? a : c : (b > c) ? b :
```

Output

20