07/11/22 ⑧ Operators, Conditionals, Switch Case in Java

① operators in java :-

① Arithmetic operator (+, -, *, /, %)
② Increment & Decrement operator (++, --)
③ logical operator (&&, || or, ! Not)
④ Assignment.
⑤ Conditionals operator ↳ Ternary operator } if-else
⑥ Bitwise & shift operators.
⑦ Relational operator

① **Arithmetic operators** :- Arithmetic operators are used to perform arithmetic operations on variables and data.

Eg :- a+b, a-b, a/b, a*b, a%b.

(i) % → modulo operator
↳ it gives remainder.

a = 30
b = 20
res = 30 % 20
↳ $\boxed{10}$

$20)30(1$
$\;\;\;20$
$\;\;\;\boxed{10}$

(ii) / → division

② logical operator :-

(i) AND operator - &&
(ii) OR operator - ||
(iii) Not operator - !

(i) **AND operator** :- if all True ⇒ T
it Evaluates two statements/ Conditions and returns TRUE only when both statements, are true.

A	B	Y
T	T	T
T	F	F
F	T	F
F	F	F

if all True ⇒ T
if all True & 1 false ⇒ false.

$\boxed{\text{Symbol: \&\&}}$

(ii) **OR operator** :- This operator will only return false when both Conditions are false.

Truth table :-

A	B	Y
T	T	T
T	F	T
F	T	T
F	F	F

(ii) NOT operator: if Condition is true, the operation returns false and vice versa.

① T! ⇒ F

② F! ⇒ T

A	Y
T	F
F	T

9/11/22

③ **Relational operator :-**

→ Comparsion and relating

→ Relational operators are used to check the Relationship between two operands

Java has 6 relational operators :-

① == ⇒ is the Equality operator. this returns true if both the operands are refering to the same object, otherwise false.

② != → is for non- Equality operator.
→ it returns true if both the operands are refferring to the different object, otherwise false

③ < is less than operator.

④ > is greater than operator.

⑤ <= is less than (or Equal to operator

⑥ >= is greater than (or) Equal to operator.

Eg:-

①
```
int a=10;
int b=20;
is  a==b; → false
is (a>b); → false
is (a<b); → True
```

②
```
~~a>=b~~
int a=10;
int b=20;
is (a>=b)
    F F F  → False
is (a<=b)
    T F F  → True
```
→ if one Condn is satisfied then it is true.

③
```
int a=10;
int b=20;
System.out.print(a>b);
output :- false.
```

→ Assignment operator → " = "
→ Equality operator → " == "
(4) Assigment operator :-
(1) Chained assignment :-

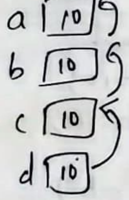
int a;
int b;
int c;
int d;

$\Longleftrightarrow$ int a, b, c, d;

→ all ~~w~~ Variable have Same data type

Eg:- int a, b, c, d;

→ a = b = c = d = 10;
↳ chained assigment



→ a = b = c = d = 10+5;

(2) Compound assignment :-
Compound - assignment operators provide a shorter Syntax for assigning the result of an arithmetic (or bitwise operator.

Eg:- a+ = 20
a - = 20
a * = 20

int a = 10;
a + = 20 → 20 is added to 10.
↳ 20+10 = 30

→ while assigning, operation is happened.

a * = 20 ⇒ a* 20 ⇒ 10×20 ⇒ 200
a - = 20 ⇒ a- 20 ⇒ ~~a~~ 10-20 ⇒ -10
10%0 = 20 ⇒ a%20 ⇒ 10 % 20 ⇒ 10

unary and Binary operator

(i) unary operator :- the unary operator is an operator that can be used only with an operand.

→ unary operators are the types that need only one operand to perform any operation like operation like . increment, decrement, negation etc. (or only one operand is sufficient to perform operation

(ii) Binary operator :- Represents an operation upon two operands of the same type, producing a result of same type as the operands.

→

~~unary operator~~

Example :-

```
int a = 10  }
a + = 20    } one operand is used.
a++ ->      }
```

→ if you are using any operator to perform operation ~~and~~ on Single operand is Called unary operator.

→ one operand is suffient to perform operation.

→ Assignment and increment are unary operators.

→ Relational, logical, Arithmetic are Binary operators.

```
a == b
a <= b
a >= b
```

⑤ Conditional operator.

→ Performing task (or Activity (or operation based on Condition.

(i) if - Else (ii) Else - if (iii) Nested - if else (iv) Ternary operator.

Eg:-

```
int a = 10;
int b = 5;
               -> True
if (a > b) {                    ]
  int res = a-b;                ] -> block ① will get
  S.o.p(res)                    ]    Executed.
}

Else
{                               ]
  int res = a+b;                ] -> block ②
  S.o.p(res)                    ]
}
```

→ Either of them are Executed based on Condition

(ii) Else if( ) :- for checking Multiple statements.

(ii) Else if statement :-

→ for Executing Multiple statements we are Else if.

Eg①
```
int a=10
int b=5;

if (a>b) {
  s.o.p(a-b)
}
Else if(a==b) {
  s.op(a+b)
}
Else if (a<b) {
  s.o.p("a is lesser");
}
```

(iii) Nested if-Else :- The nested if statement represents the if block within another if block.

→ here, inner if ~~the~~ block Condition Executes only when outer if block Condition is true.

Eg:
```
int a=10;
int b=2;

if (a>b) {
  if (a==10) {
  s.o.p(a-b);
  }
  Else
  {
  s.o.p("a is lesser than b");
  }
}
```

→ for checking Multiple Conditions :-

Eg① cheking least number in 3 numbers.
```
int a=10;
int b=20;
int c=30;

if (a<b) {
  if(a<c)
  s.o.p(" A is the least "+a);
}
~~[illegible]~~
  Else {
   s.o.p(" C is the least "+a);
  }
}
Else if (b<c)
   s.op(" B is least ");
Else s.o.p( " c is least ");
```

Program to find least of all 3 numbers

```
int a=10;
int b=20;
int c=5;
if (a<b) {
    if (a<c)
        s.o.p("a is the least"+a);
    Else
        s.o.p("c is the least "+c);
}
Else if (b<c)
{
    s.o.p(" B is least "+b);
}
Else
{
    s.o.p("c is least "+c);
}
```

Alternative:-

```
int a=10;
int b=20;
int c=30;      ✓
if (a<b && a<c)
{        ✓
    s.o.p("A is least"); ✓
}
Else if (b<c)
{
    s.o.p ("B is least");
}
Else
{
    s.o.p(" c is least ");
}
```

(iv) Ternary operator :-

→ the only Conditional operator that takes three operands.

Eg ①

```
int a=10;
int b=20;
if(a>b){
    s.o.p(a)
}
Else {
    s.o.p(a)
}
```

Eg ②

```
int a=10;
int b=20;
int c= (a>b)? a:b;
s.o.p(c);
```

Var3 = (Condition)? Var1 : Var2;

① if Condition is ~~True~~ True Code After question mark(?) and before Colon will be returned.

Eg

```
int c = (a>b)? a : b;
```

② if Condition is false Code after Colon(:) will be returned

```
int c = (a>b)? a: b;
```

→ int Var = (Condition)? T : F;

Eg ③ ~~greatest of~~

→ Least of 3 numbers

```
int a = 10;
int b = 20;
int c = 30;
int res = (a<b)? (a<c? a:c) : (b<c? b:c);
s.o.p(res);   → 10
```

$$res = (a<b) ? \underbrace{\overset{10<20 \to T}{(a<c\ ?\ a:c)}}_{} \overset{10<30 \to T}{} : (b<c\ ?\ b:c);$$

→ Next class

↳ loops → Patterns → loops + Conditionals + operator

④ Switch Case :- Switch statement is a multi-way branch.

→ it executes one statement from multiple Condition

Example ①

```
int number = 100;
Switch(number){
Case 10: System.out.println("1st Case");
Case 20: System.out.println("2nd Case");
Case 100: s.o.p("3rd Case"); ✓
default: s.o.p("no Cases matches)
}
```

① In Switch case if first Case is matching rest of the cases will also Executed.

→ output : 3rd Case

Eg-②
```
Case 105 : s.o.p("1st Case");
Case 100 : s.o.p("2nd Case");
Case 10 ; s.o.p("3rd Case");
```
↳ These two get printed on Executed.

default Case is optional

②
```
Case 10: s.o.p("1st Case")
Case 20: s.o.p("2nd Case")
default: s.o.p("3rd Case")
```

(iii) Break Statement :-

→ Break statement is a loop Control Statement that is used to terminate the loop.

Eg ①
```
int num=10;
Switch (num){
    Case 10: S.o.P("1st Case");
             break;
    Case 20: S.o.P("2nd Case");
             break;
    Case 30: S.o.P("3rd Case");
             break.
}
```

→ Executes and terminated.

~~This~~ This statements will not get printed

Note①:-
→ if no Cases are matching then default will Executed.
→ if default statement is not return then nothing get Executed. if no Cases are matching.