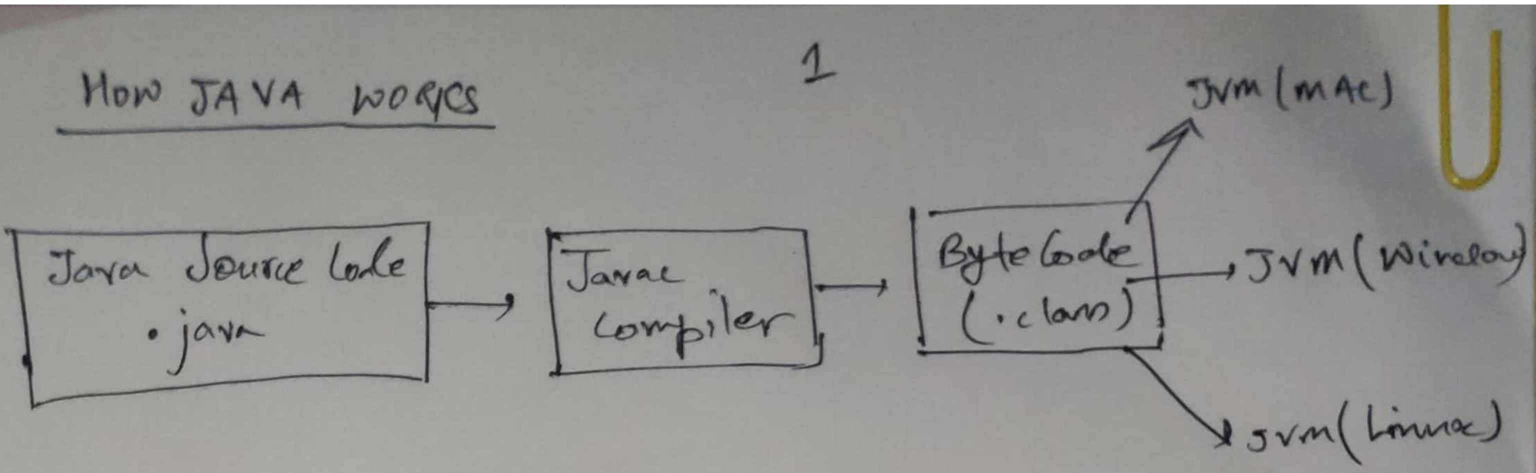
HOW JAVA WORKS



- → Hello world
- → DT
- → Post/Past Increment
- → Type Casting.

JAVA R1

~~OOPS TOPICS~~

- Hello World

```
public class HelloWorld{
        public static void main(strin[] gs)
        { system.out.println ("Hello world");
        }
}
```

- No gap between class name
  ⇓
  CamelCase convention

  Eg. rateOfInterest
  ↑ small (lowercase) · ↓ big.(uppercase).

Also case sensitive

System
String
println ✓

System
string
Println ✗

Data Types.

① byte 8bits
② long 8 bytes
③ int 4 byte
④ short 2 byte.
⑤ double
⑥ boolean
⑦ char
⑧ float

Post | Past Increment

```
b=45
d=b++
System.out.println(b) = 46
                  (d) = 45
```

```
b=45
d=++b
System.out.println(b) = 46
                  (d) = 46
```

TYPE CASTING

: Assign a value of one primitive data type to another.

Automatic Type Casting (ATC)

byte → short → char → int → long → float → double

← going reverse
: Manual Type Casting

Eg.

```
int a = 4
long b = a
print (b)
```

4

→ following ATC

But if

```
double p = 3.14
int q = p          → This line will give error
print (a)
```

→ Reverse of ATC

∴ manual TC is required

Soln:

```
double p = 3.14
int q = (int) p
print(a)
```

3

(3)

Scanner function in Java.

(5)

```
import java.util.Scanner
public class SimpleInterest {
  & public static void main (String [] ga)
   { Scanner sc = new Scanner( System.in)
        int principal = 500        = sc.nextInt();
        float rate = 12.5f         = sc.nextFloat();
        ; int time = 12            = sc.nextInt();

      float SI = principal x rate x float /100;
      System.out.println ("the SI is" + ___);
      }
}
```

(Pb) Take input & a string after a Number.

```
{ int num = sc.nextInt();
  o sc.nextLine();
  String str = sc.nextLine();
 System.out.println(num);
 System.out.println(str);
 }.
```

→ Operators   5 types.

1. Airthematic   +, -, x, /, %, ++, --
2. Bitwise   NOT, AND, OR, XOR   &=, |=
3. Assignment   =, +=, -=, /=, %=, x=, >>=, <<=,
4. Comparison   ==, !=, >, <, >=, <=
5. Logical   &&, ||, !

NOT
opposite

0	1
1	0

AND
Both 1 = 1

OR
atleast 1

XOR
only one 1

		AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Right shift
÷2

1101 >>

a = 13
13 >> 1

1 bit
Right shift

1 1 0 1
→ → → 0 1

1 1 0 1
↘ ↘ ↘ 0
0 1 1 = 6

int a = 13
b = 13 >> 1 6
c = 13 >> 2 $\frac{13}{2^2} = 3$
d = 13 >> 3 $\frac{13}{2^3} = 1$

Left shift
×2

a = 1 1 0 1 = 13
1 1 0 1 0 ⇒ 26

a = 13
b = 13 << 1 13 × 2^1 = 26
c = 13 << 2 13 × 2^2 = 52
d = 13 << 2 13 × 2^3 = 104

$\begin{array}{r} 13 \\ 8 \\ \hline 104 \end{array}$

Trick.

1101. 0000 ... >> 1 ⇒ 110.000
110 ✓ 6

Right shift by x ⇒ Eliminate last x bits from last.
÷ 2^x : Dec form,

1101. 0000 ... << 1 ⇒ 11010.000
= 26

Left shift by x ⇒ Add x zeros in binary form
× 2^x : Dec form

Right shift by x ⇒ /2^x in Decimal.

eliminate last x bits in Binary

Left shift by x ⇒ ×2^x in decimal

append x bits of zeros in binary form

Q4. Shorthand (Ternary Op)

variable (Cond) ? exp1 : exp2.

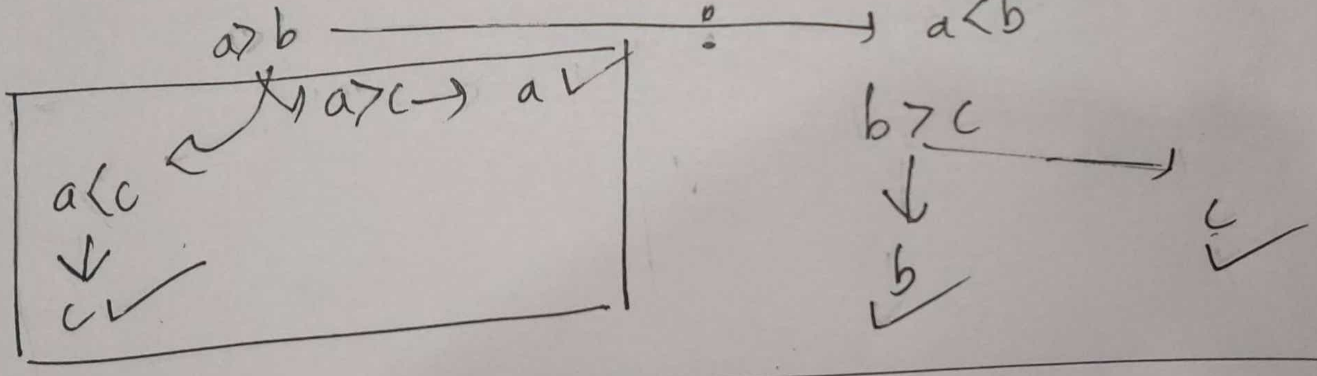
Greatest of three numbers.

→

result = (a>b) ? (a>c ? a:c) : b>c ? b:c;

Explanation.

$\boxed{a>b ?\ a>c ?\ a:c} : \boxed{b>c?b:c}$



Switch Case