JAVA

DAY - 1 (VARIABLE & DATA STRUCTURE)

→ Class |→ Basic Building block which stores all the code.

```
public class JavaBasics {
    public static void main (String args[]) {
    }
}
```

Name of file & Name of class should always be the same

also called Boiler Plate code

function whose name is main

→ Output in Java

```
System.out.print ("Hello World");
```

; → statement terminator — ends with semicolon

"Hello World" → output.

To run the code. → Compilation of code

```
$ javac Outputinjava.java
$ java Outputinjava.
```

```
System.out.println ("Hello World");
```

ln means next line / space for next line.

also ("Hello World \n"); → also used for next line.

Three types we saw |→

1) print
2) println
3) "\n"

→ Print Pattern |→

```
1) * * * *
   * * *
   * *
   *
```

→ Variables in java

```
int a = 10;
int b = 5;
String name = "Ganesh";
System.out.println (name);
a = b;
```

Value of b will be assigned to a.

a, b, c
JavaBasics
main, Print, Perse } identifiers

⇒ DATA Types in JAVA

Primitive

↳ Already exists in java & we don't have to create them

Size		
1 byte	→ byte	→ 256 numbers can be store [-128 to 127]
2 byte	→ short	→
2 byte	→ char	→ All characters can store
1 byte	→ boolean	→ Stores True or false
4 byte	→ int	→ integer/whole no
8	→ long	→ integer of Big value/size
4	→ float	→ Decimal value
8	→ double	→ Decimal value of Big size

Non-Primit.

↳ Doesn't exists in java & we have to create them

→ String
→ Array
→ Class
→ Object
→ Interface

Doubt |→ Declaring float requires either (float) 1.5; or 1.5f;

⇒ Comments in java

// → Single line comm.

/*
Hi
*/ → Multi-line Comment

⇒ INPUT in JAVA :→

① Add Package :→ import java.util.*;

② Scanner sc = new Scanner (System.in);

③ sc.next(); {To take input)

④ String input = sc.next();

Note: next function only stores before space

Input function :→

next
nextLine
nextInt
nextByte
nextFloat
nextDouble
nextBoolean
nextShort
nextLong

Type conversion ! → widening Conversion/Implicit version

Conversion happen when I
a. type compatible
b. destination type > source type

byte → short → int → float → long → double

→ Typeconversion is done by java automatically.

Eg! float number = sc.nextInt(); → Allow

int b = 10;
float a = b; → Allow
int b = a; → not allowed

Error - Double Float solve

Java Automatically takes decimal value as double I (to solve) either) 1) 10.5f or 2) declare doub

Type Casting :-> Conversion which is not allowed by java, but we make him to do it forcefully.

Eg:-> float a = 25.0;

We are trying, int b = a; -> not allowed by java

int b = (int) a; -> allowed by java.

-> Also called :-> narrowing conversion, Explicit conversion.

char ch = 'a';
int no = ch; -> valid

-> will look later

=> Type Promotion in Expression

$\underset{int}{a} + \underset{float}{b} * \underset{char}{c} / \underset{long}{e}$

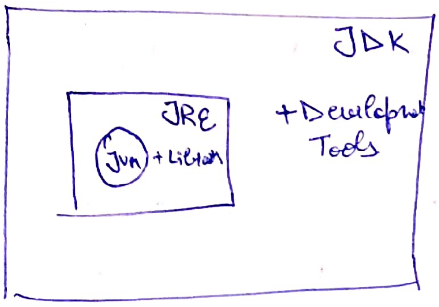
Two Rules :-> 1) Java automatically promotes each byte, short or char operands to int when evaluating expression

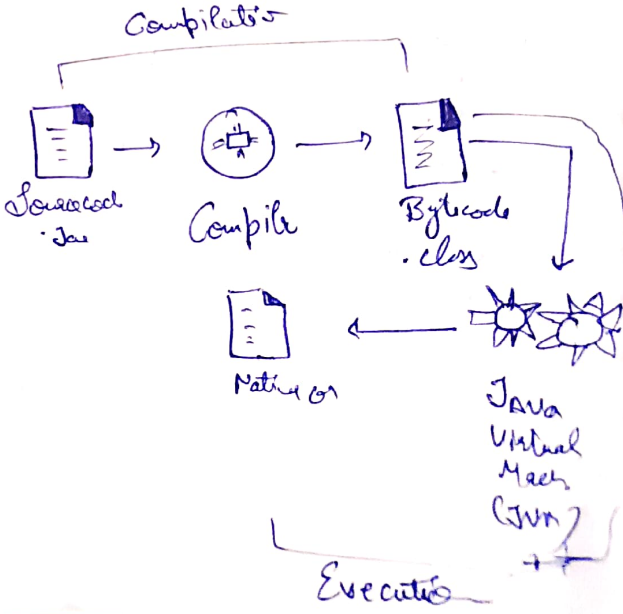
byte, short, char ——> int.

2) If one operand is long, float or double the whole expression is promoted to long, float or double respectively.

mean -> If Largest possible datatype long every other datatype will be converted to ~~doubt~~ long. Same for float & double

How JAVA CODE RUNS :->





Note :-> Names of variables are called identifiers in Java, Identifier rule says, identifiers can start with any alphabet or underscore("_") or dollar ("$")

∴ int $ = 24 is valid etc.

OPERATORS IN JAVA :-

Operators are symbols that tell compiler to perform some operation.

sum = a + b → Operator

Operands

Arithmetic Operators :->

Binary (2 operands)

+

-

*

/

%

Unary (1 operand)

++ → increment operator

-- → decrement operator

		Post	Pre
a = a+1	==	a++	++a
a = a-1	==	a--	--a
		1) Value use 2) Value change	1) Value change 2) Value use

Relational Operators :->

==

!=

>

<

>=

<=

Assignment Operators :->

=

+=

-=

*=

/=

A = A+10 → A += 10

B = B-5 → B += 5

Logical Operator :->

&& (Logical AND)

|| (Logical OR)

! (Logical NOT) → True → False
False → True

! (Stateme)

~~Modular~~ Bitwise operator :->

(Will study further)

Conditional Statements :-

( ) → Para

1) if-else :->

```
if (condition) {

}
else {

}
```

Note!-> we can use multiple if & each satisfying Condition will give desired out.
But in case of else if, if one (if) statement works it will skip other else/else if statements

2) Else if :->

```
if (condition-1) {

}
else if (condition 2) {

}
else {

}
```

3) Ternary Operator
↳ 3 operands

नहीं तो
↑
Variable = condition ? statement 1 : statement 2;

statement 1 ↓ this execute (True)
statement 2 ↓ this exec. (false)

case

Eg :-> int Larger = (5>3) ? 5 : 3;
String s = (5%2 = =0)? "even" : odd;

4) Switch Statement

```
switch (variable) {

Case 1:
break;
case 2:
break;
Case 3:
break;
default:
}
```

Loops :->

1) While Loop :->

```
while (condition) {
        // do something
        i++;        → Remember.
}
```

Infinite loop → when condition is always true.

2) for loop :->

```
for (initialisation ; condition ; updation {
    // do something
}
```

Ques Print Reverse of a number

n = 10899.

num = 10899 /10

2 logical thing → 1) Agar kisi no. ka lastdigit chahiye then %10 → remainder = lastdigit

2) Agar last digit hatana ho /10

To~~day~~ store no. in variable.

rev = (rev* 10) + lastdigit

3) do-while Loop.

```
do {
    // something
} while (condition);
```

→ Break Statement :-> To leave a loop. (In loop use if statement)

→ Continue Statement :-> To skip an iteration (Skip a बारी)