JAVA

→ High level language

→ language & platform (Java has runtime environment & API)

→ supports UNICODE values (all characters & languages (vocal lang.))

→ Robust

→ platform/os independent : WORA (write once run anywhere)

→ Secured : No explicit pointer
 : classloader (load Java classes into JVM dynamically)
 : Bytecode verifier (check code if they violate access rights to objects)

→ Architectural-Neutral

*No pass by reference is there

→ lack of pointers avoid security problems (pointers may sometimes lead to unauthorized access)

→ Case Sensitive : Name of java file must match class name

→ Multi-threaded

→ Applications : Desktop app i.e. media player, antivirus etc
 : web app i.e. irctc.co.in etc
 : enterprise app i.e. Banking app
 : mobile
 : embedded system
 : smart card
 : Robotics
 : games

→ Run Java program

compiler

```
javac filename.java
java class/fileName
```

→ Syntax :

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello");
    }
}
```

class name

can be access from anywhere

entry pt. of javacode

to make main fn run without any object.

return type

fn

• cmd line arguments
• array of strings

in-built class

variable in System class of type Printstream

method is in printstream

to change location of bytecode file (.class)
javac -d . Demo.java / javac -d .. Demo.java
previous directory

→ **to change location of bytecode file (.class) :**

```
javac -d . Demo.java
or
javac -d .. Demo.java
           previous directory
```

→ **package** : in simple words → files lies in a folder, that folder is said to be package

: used to provide rules & systematic way to our programs & classes.

→ **Scanner** : java.util.Scanner (package)

Syntax.

```
import java.util.Scanner;

public class Demo
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
   inbuilt class  variable  creating  scanner   corresponds
                            object    object    to keyboard
                                                input
      allow us to                               (standard i/p)
      take input
        System.out.println(input.next());
                                 fns of Scanner class
    }
}
```

→ **Primitive Datatypes** : not further breakable

* **Data type** → type/category of data belongs to.

* int - numeric (4 bits)

* char - character/alphabets

* float - floating pnt. numbers (float a = 9.2f;)

* double - larger decimal numbers

*(8) long - long integer numbers (long l = ...l;)

defaut value → false ← * boolean - true or false (no fix size)

: In float and long, f & l use to denote the number is of which data type.

→ **Comments** : cntrl + /
Single line → //
Multi-line → /* . . . */

→ **Variables** : names to store data in
eg. int a = 5;
float b = 5.2f;
: **rules** → case sensitive
→ Start with alphabets, _ or $
→ not be keyword
→ CamelCase
→ not be class name
→ no length limit

→ int $\boxed{a}$ = $\boxed{10}$;
reference. literals
variable of 10 → identifier

→ **Type casting** : one type of data assigned to another type of variable i.e. changing data type of any data.
: Typecasting can occur if the two datatypes are compatible
&
Destination data type greater than source type.
Eg. int num = (int)(67.6f);
O/P. $\boxed{67}$

*automatic type promotion in expressions.
→ Java automatically promotes each byte, short or char operand to int when evaluating expression.
→ If one operand is a long, float or double the whole expression is promoted to long, float or double respectively.
Eg. byte a = 40, b = 50, c = 100;
int d = (a* b)/(c);