PROGRAMMING

- **Introduction to Programming**
  - Simple ways for us to give instructions to the computer to get tasks/work done.
  - Way of writing efficient instructions using particular LANG syntax+grammar+rules , is called PROGRAMMING LANG.

Types :

- ML or BL or LowLevel(1920-1945)
  - Hard to write & understand , programming written using 0's 1's
  - -No need of translation tools , we are writing programs directly in Binary bits.
- Assembly Level (30-40)(upto 1970)
  - Slightly easier to understand compared to BinaryLang.
  - Programs written using Mnemonics.
  - Need assembler to convert back to 0's 1's
- High Level
  - Easy to understand and write for humans
  - Programming written using an English set of instructions.
  - Need a compiler/Interpreter to convert back to machine code.
  - Ex: Java , Python , Scala , JavaScript , C , C++

Compiler

- It is a translator which takes input i.e., High-Level Language, and produces an output of low-level language i.e. machine or assembly language.
- A compiler is more intelligent than an assembler; it checks all kinds of limits, ranges, errors, etc.
- The compiler scans the whole program in one go.The errors (if any) are shown at the end together.
- Execution of the program takes place only after the whole program is compiled.
- It is more efficient.
- C, C++, C#, etc are programming languages that are compiler-based.

Interpreter

- An interpreter is a program that translates a programming language into a comprehensible language.
- It translates only one statement of the program at a time.
- Considering it scans code one line at a time, errors are shown line by line.
- Execution of the program happens after every line is checked or evaluated.
- CPU utilization is less.
- Python, Ruby, Perl, SNOBOL, MATLAB

Java Introduction.

- Java Introduction
- Java History
- Java Terminologies
- Features
- Sample Java Program Structure
- Installation of Java

OBSERVATIONS

==========

- **Javac** (Java Compiler) will take care all java syntax checking and generates BYTECODE

.

- **JVM** will load byte code into the memory and starting execution from main() method

- JDK(JVM+JRE) is platform dependent because each OS will have different instruction sets.

- ByteCode is PLATFORM INDEPENDENT , which java as a platform independent language.

  *Compile : **javac** fileName.java*
  *Running : **java** clasName/ByteCodeName*

- We can compile a class without a main method() , but we can't run , it will raise a Runtime Exception.

- When a **class is public , fileName must be the same as className**.
- When Class is not public.
  - compile with fileName
  - run with className
  - by default it will generate a byte with className.

- order of public static can be of different ways
  - public static
  - static public
  -
  - String[] args
  - String []args
  - String args[]
  - String… args
- final synchronized strictfp keywords for main() method

- ➔ You can compile a java program , without the main method , but we can't run it.
- ➔ It is Possible to have multiple classes in one file , it will create multiple .class files.
- ➔ whenever we have multiple .class files while running , execute with individual names.

```
java A
java B
```