DAY – 1

# JAVA:

## **JDK: Required for coding & compiling.**

* It is a s/w development kit & it is used for to develop Java Applications.
* It includes JRE, Interpreter, Compiler, an archiver (JAR), a documentation generator.
* JDK= JVM+ JRE+ Development tools.
* Contains java libraries.
* Without JDK, you can’t compile Java Code.

**JRE:** Required for running Java Apps.

* It’s a s/w package that provides everything needed to run Java Programs.
* It includes JVM+ Essential libraries (.jar, core Java classes)
* If we want to run only Java apps JRE is enough, no need for JDK.
* It contains JVM.
* Java libraries (. Lang, . util, etc.,)

**JVM:** Brain of Java.

* It is the engine that runs Java Programs.
* It takes the I/P as .class files (byte codes) & executes it on any OS (Windows, Linux, Mac)
* It converts Byte code into Machine code.
* It has responsibilities like
  + Loads Java class files.
  + Verifies Byte code for Security.
  + Allocates memory & manage execution.
  + Performs Garbage collection.
* It executes .class files not .java file

## COMMENTS IN JAVA:

In java Comments are inexecutable and are used to explain or show info. There are 3 types of comments in java, they are:

1. Single line comments.
2. Multi line comments.
3. Java Docs
4. **Single-line Comments**: These comments start with // and extend to the end of the line.

java

// This is a single-line comment.

1. **Multi-line Comments**: These comments start with /\* and end with \*/. They can span multiple lines.

java

/\*

\* This is a multi-line comment

\* It can span multiple lines

\*/

1. **Javadoc Comments**: These are special multi-line comments used to generate documentation. They start with /\*\* and end with \*/.

java

/\*\*

\* @param

\* @return

\*/

Public int add (int a, int b)

{

return a + b;

}

Javadoc comments are used to generate HTML documentation using Javadoc tool.

## LITERAL:

It’s a fixed value written directly in the code, without needing a variable.

Eg: - int num=100; //integer literal

double pi=3.14; //floating-point literal