LECTURE – 1

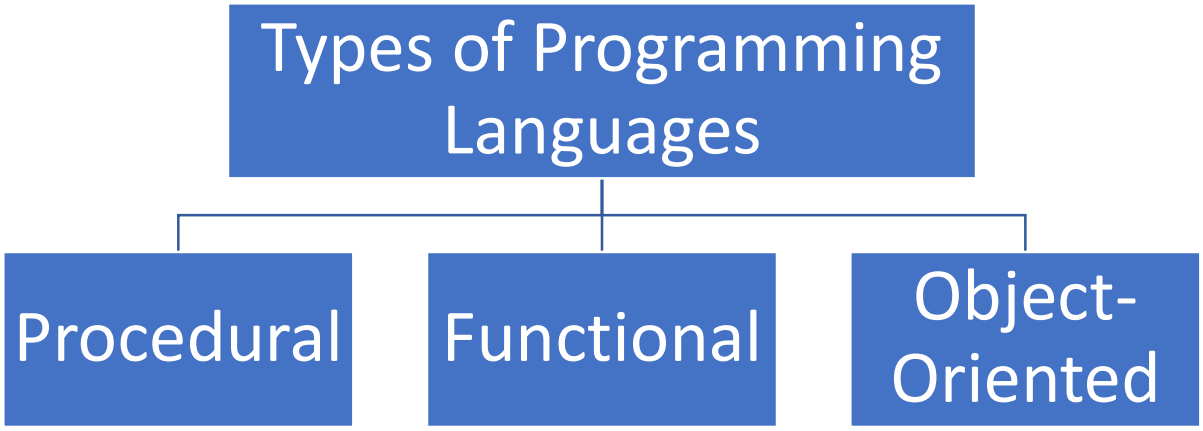
Introduction to programming

Programming is a way to instruct the computer to perform various task.

Computers only understands Binary i.e., 0's and 1's.

Instructing computers in Binary i.e. 0's and 1's are very difficult for humans so, to solve this issue we have programming languages.

Programming language: - It is a computer language used by programmers to communicate with computers.



Procedural

- Specifies a series of well-structured steps and procedures to compose a program.
- Contains a systematic order of statements functions and commands to complete a task.

Functional

- Writing a program only in pure functions i.e., never modify variables but only create new ones as an output.
- Used in a situation where we have to perform lots of different operations on the same set of data like ML.

Object Oriented

- Revolves around objects.
- Code + Data = objects
- Developed to make it easier to develop, debug, reuse and maintain software.

"One programming language can be of all 3 types like- Python"

Java Follows procedural and object oriented both types

Static VS Dynamic Languages

Static	Dynamic
Perform type checking at compile time	Perform type checking at runtime
Errors will show at compile time	Error might not show till programs run
Declare datatypes before use	No need to declare datatype of variables
More control	Saves time in writing code but might give error at runtime.

Memory Management

There are 2 types of memory Stack and Heap

When we declare a variable then the reference variable stored in stack memory points to the object of that variable stored in heap memory.

For ex:- $a = 10$

Here "a" is called reference variable, and "10" is the object of That reference variable

- Reference variable are stored in stack memory.
- Heap memory stores the objects of reference variable.

Points to remember:-

- More than one reference variable can points to the same object.
- If any changes made to the object of any reference variable that will be reflected to all others variable pointing to same object.
- If there is an object without reference variable then object will be destroyed by ***"Garbage Collection"***

FLOW OF THE PROGRAM

Flow Chart :- Visualization of our thought process or Algorithm and represent them diagrammatically is called flow chart.

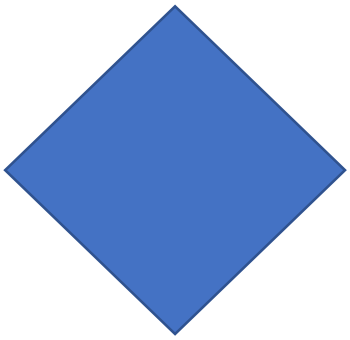
Symbols to Be used in flow chart

1. **Start / Stop** →

2. **Input / Output** →

3. **Processing** →

4. **Condition** →



5. **Flow direction of program** →

Start / Stop: - An ovel shape indicate the starting and ending points of the flow chart.

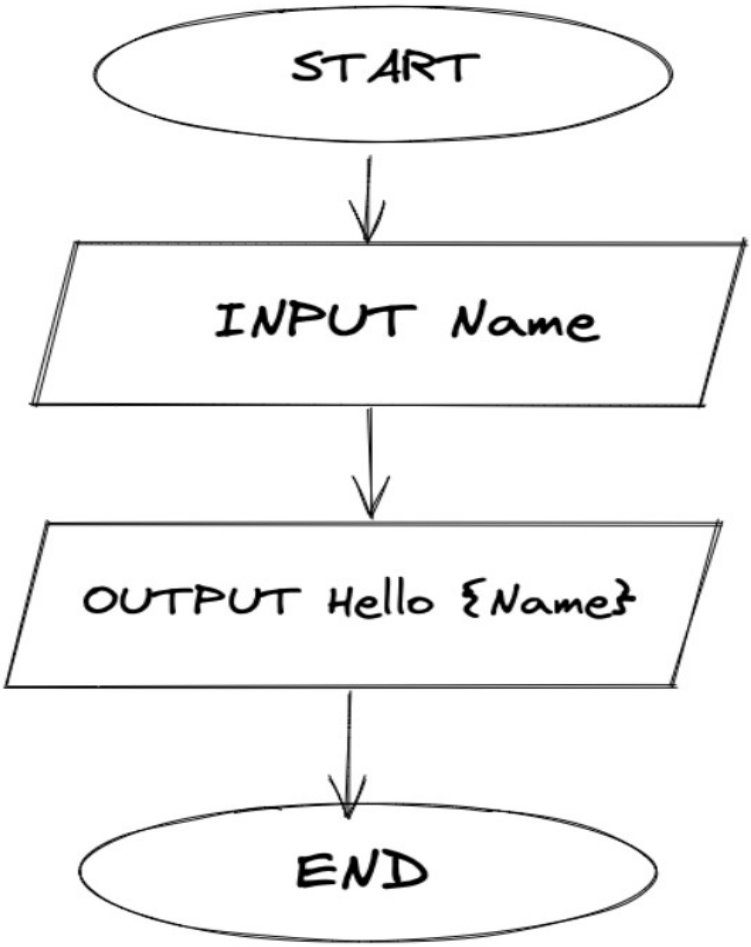
Input / Output: - A parallelogram is used to represent Input and output in flow chart

Processing: - A rectangle is used to represent process such as mathematical computation or variable assignment.

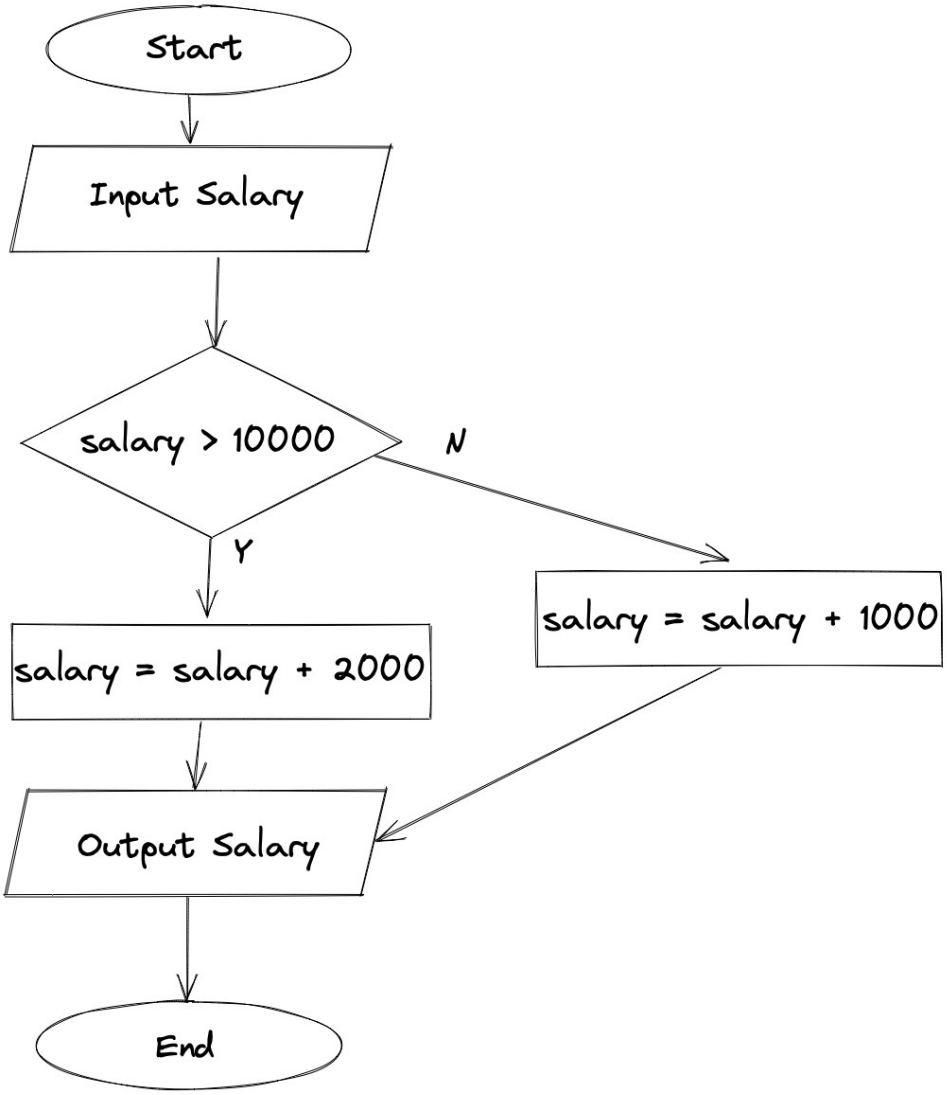
Condition: - A diamond shape is used to represent conditional statement which results in true or false (Yes or No).

Flow direction of program: - An arrow shape is used to represent flow of the program.

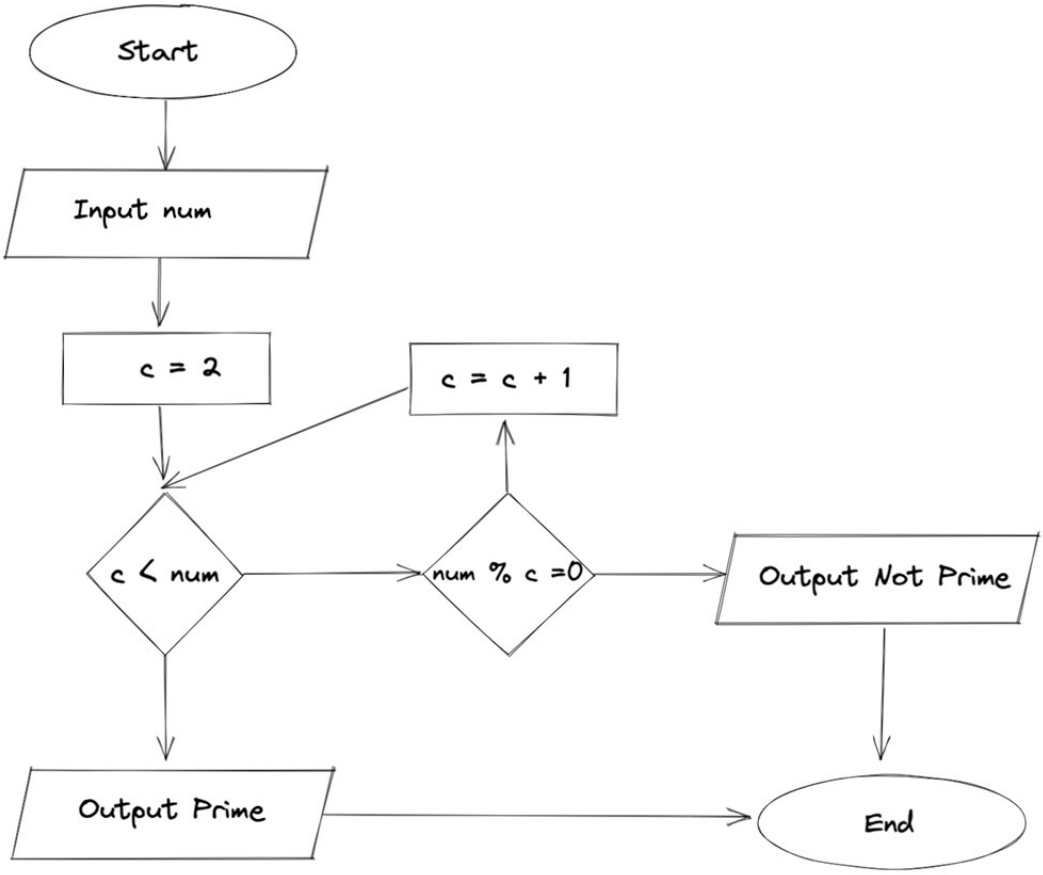
Example 1:- Take a name and output Hello name.



Example 2 :- Take input of a salary. If the salary is greater than 10,000 add bonus 2000, otherwise add bonus as 1000.



Example 3 :- Input a number and print whether it is prime or not.



Pseudocode :- It is like a rough code which represents how the algorithm of a program works.

- Pseudocode does not require syntax.

Pseudocode of Example 2

```
Start
Input Salary
if Salary > 10000 :
    Salary = Salary+2000
else :
    Salary = Salary+1000
Output Salary
exit
```

Pseudocode of Example 3

```
Start
Input num
if num ≤ 1
   print “Nither prime nor composite”
 c = 2
 while c < num
   if num % c = 0
        Output “Not Prime”
        Exit
   c = c+1
 end while
 Output “Prime”
 Exit.
```

Optimization of prime solution

Let's have a number to check it's a prime number of not

→ 36

$$\begin{bmatrix} 1 \times 36 = 36 \\ 2 \times 18 = 36 \\ 3 \times 12 = 36 \\ 4 \times 9 = 36 \end{bmatrix}$$ **…….. (i)**

$$6 \times 6 = 36$$

$$\begin{bmatrix} 9 \times 4 = 36 \\ 12 \times 3 = 36 \\ 18 \times 2 = 36 \\ 36 \times 1 = 36 \end{bmatrix}$$ **……(ii)**

In the above demonstration we have clearly seen that (i) and (ii) are repeated so, to optimize this we can ignore the (ii)

As same as this

We can check the number is prime or not by travelling form
$2\ to\ \sqrt{number}$

For example:-

- **To check 23456786543 is prime or not, we only have to travel from $2\ to\ \sqrt{23456786543}$ (i.e. 153156)**
- **To check 17 is prime or not, we do not have to travel from 2 to 17 we just have to travel from $2\ to\ \sqrt{17}$ (i.e. 4)**

Optimized Pseudocode of Example 3

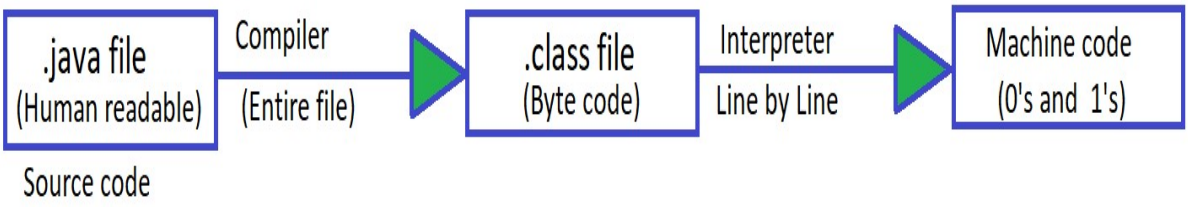
```
start
input n
if n <= 1:
        print("neither prime nor composite")

    c = 2
    while c*c <= n:
        if n % c == 0:
            output "not prime"
            exit
        c += 1
    end while
    output "prime"

exit
```

INTRODUCTION TO JAVA

Ques – Why do we use Programming language?

Ans – Machine only understand 0's and 1's, for humans it is very difficult to instruct computer in 0's and 1's so to avoid this issue we write our code in human readable language (Programming language).

" Java is one of the Programming Language"



- The code written in java is human readable and it is saved using extension **.java**
- This code is known as source code

Java Compiler:-

- Java compiler converts the source code into byte code which have the extension **.class**
- This byte code not directly run on system
- We need JVM to run this
- Reason why java is platform independent

Java Interpreter

- Converts byte code to machine code i.e. 0's and 1's
- It translate the byte code line by line to machine code

More About Platform Independent

→ It means that byte code can run on all operating system

→ We need to convert source code to machine code so computer can understand it.

→ Compiler helps in doing this by turning it into executable code.

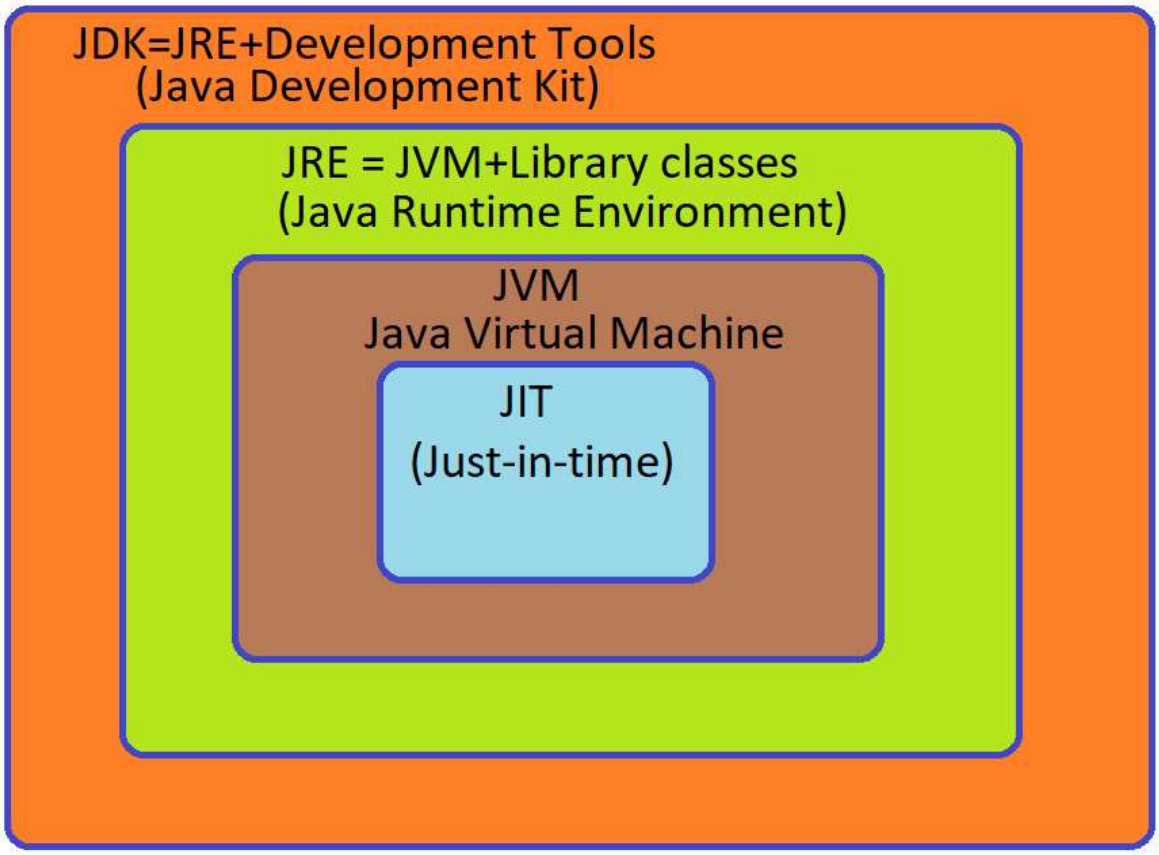
→ This executable code is a set of instruction s for the computer

→ After compiling C/C++ code we get .exe file which is platform dependent.

→ In java we get byte code, JVM converts this to machine code.

→ Java is platform independent but JVM is platform dependent.

Architecture of Java



JDK

- Provide Environment to develop and run the java program.
- It is a package that includes :-
  1. **Development tools :-** To provide an environment to run your program.
  2. **JRE :-** To Execute your program.
  3. **A compiler :-** javac
  4. **Archiver :-** Jar
  5. **Docs generator :-** Javadoc
  6. **Interpreter/loader**

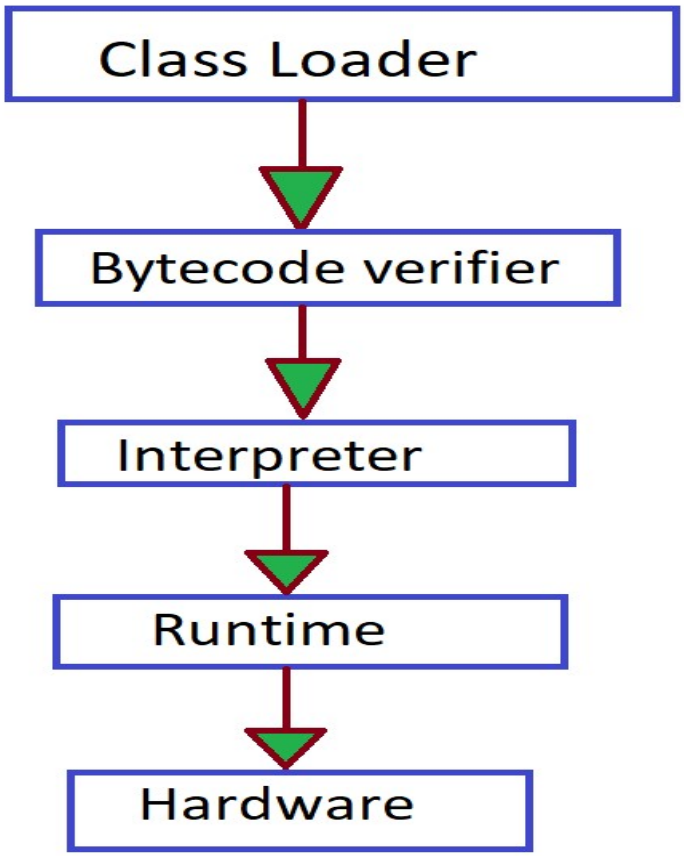
JRE

- It is an installation package that provides environment to only run the program.
- It consist of :-
  1. **Deployment technology**
  2. **User interface toolkit**
  3. **Integration libraries**
  4. **Base libraries**
  5. **JVM** :- Java virtual Machine

Compile Time:-

- After we get the .class file the next thing happen at runtime :
  1. Class loader loads all classes needed to execute the program.
  2. JVM sends code to bytecode verifier to check the format of code.

Runtime :-



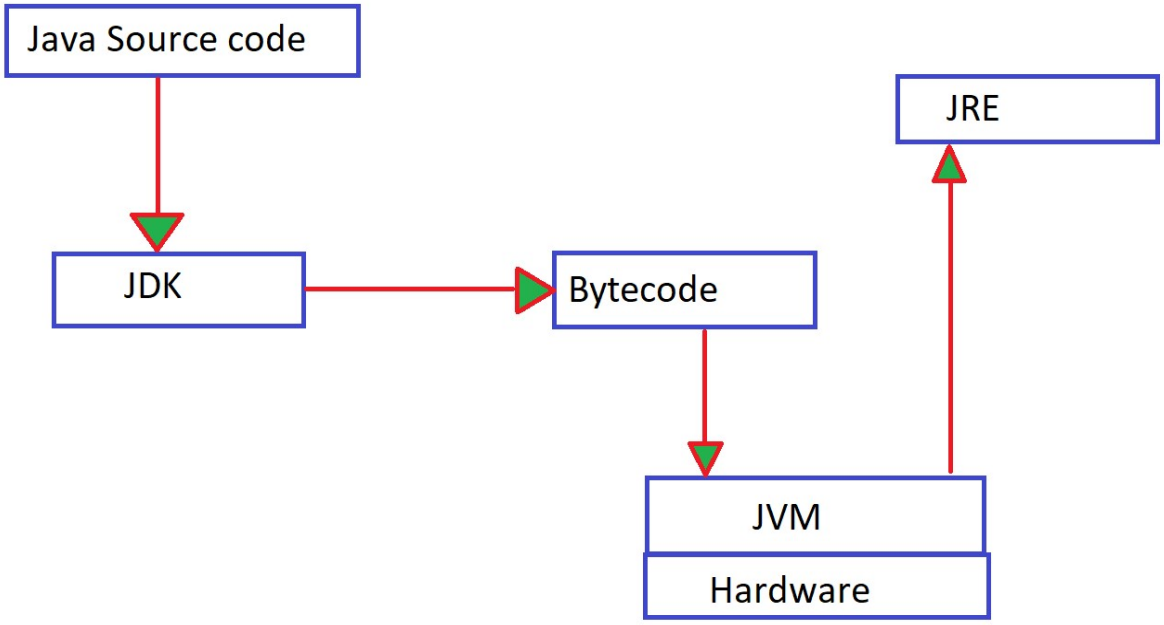
(How JVM Works) class Loader

- ***Loading***
  → Read .class file and generate binary data.
  → an Object of this class is created in heap
- ***Linking***
  → JVM verifies the .class file
  → allocates memory for class variables and default values
  → replace symbolic references from the type with direct reference.
- ***Initialization***
  → All static variables are assigned with their values defined in the code and static bock.
  → JVM contains the stack and heap memory locations.

JVM Execution

- ***Interpreter***
  → Line by line execution
  → When one method is called many times it will interpret again and again
- ***JIT***
  → Those methods that are repeated, JIT provides direct machine code so that interpretation is not required.
  → Makes execution Faster.
  → **Garbage collector**

Working of Java Architecture



Tools required to run java on machine

1. **JDK** https://www.oracle.com/in/java/technologies/javase-downloads.html

2. **InteliJ**
   a) For windows :- https://www.jetbrains.com/idea/download/#section=windows
   b) For macOS :- https://www.jetbrains.com/idea/download/#section=mac
   c) For Linux :- https://www.jetbrains.com/idea/download/#section=linux