Unit - I

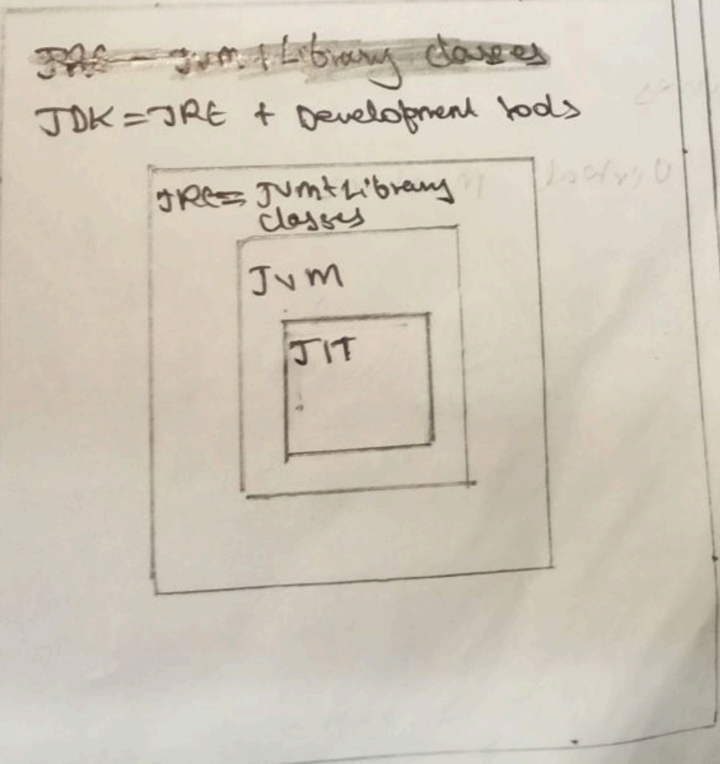
Introduction to Java

⇒ Java is a platform independent general purpose ~~de~~ object oriented programming language. It is platform independent

$$\begin{tikzcd}
\boxed{\text{.java file}} \arrow[r, "\text{compiler}", "\text{(entire file)}"'] & \boxed{\text{.class file (Byte code)}} \arrow[r, "\text{interpreter}", "\text{Line by line}"'] & \boxed{\text{Machine code (0's and 1's)}}
\end{tikzcd}$$

- The code written in .java file human readable thus is known as source code.
- The java compiler converts the code into byte code which has the extension .class. The byte code doesn't directly runs on system. JVM is needed to run this. This is the reason why Java is platform independent.
- Then the java interpretor converts the byte code to machine code. The interpretor converts line by line unlike compiler which complies the whole at once.

ARCHITECTURE OF JAVA


~~JRE = JVM + Library classes~~
JDK = JRE + Development tools
JRE = JVM + Library classes
JVM
JIT

1) JRE ⇒ compiles and generates .class file
2) JVM inside JRE verifies, initializes, and allocates memory
3) If methods are repeated JIT provides the machine code that's being repeated.
4) At last JRE runs the program

JDK

- Provides environment to develop and run the Java program
- It is a package that includes

1) Development tools - To provide an environment to run program

2) JRE - To execute program

3) A compiler : Javac

4) Archiver : Jar

5) Docs generator : Java doc

6) Interpretor/Loader :

JRE

- It is an installation package that provides environment to only run the program.

- It consists of:

1) Deployment technology.

2) User interface toolkit.

3) Integration Libraries.

4) Base libraries

5) JVM : Java Virtual Machine

09/16 23:17

Working of Java or flow of the execution

1) First the .java file is compiled by the javac compiler present in JDK

2) It is then converted to the .class file.

3) After we get the .class file the next thing happens in the run time.

i) The class loader loads the .class file,

ii) Then the bytecode is verified.

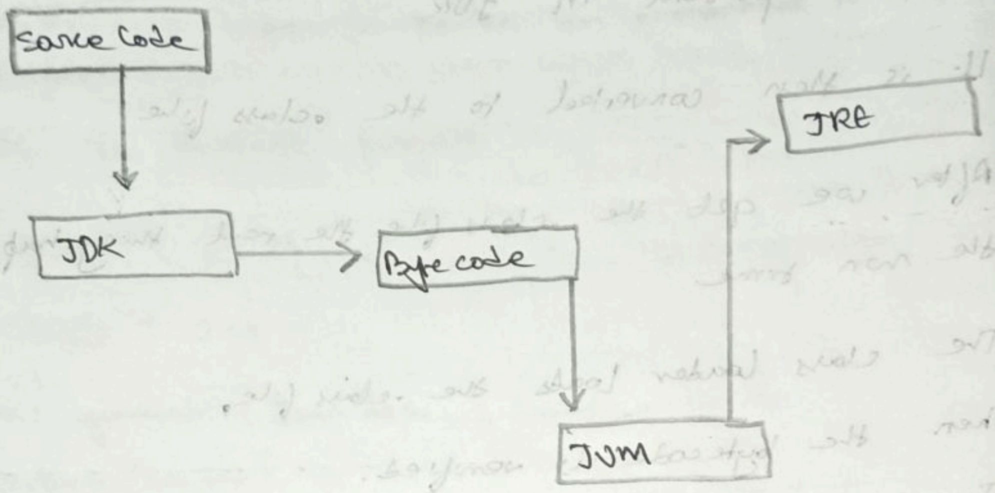
iii) The class is initialiset, allocates memory for all

4) Interpreter interpretes line by line, but when one same method is called many times it will be interpreted again and again.

5) In this case, JIT comes in work. Those method that are repeated, JIT provides direct machine code so that interpretation is not required. This makes execution faster.

6) At last garbage collector collets the garbage values.

Working of Java Architecture



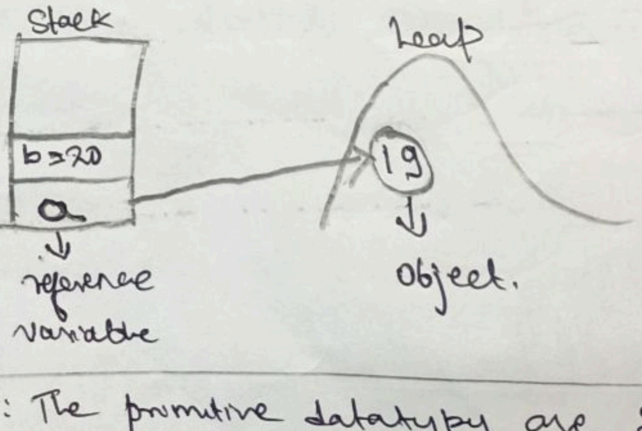
STACK & HEAP MEMORY

~~Heap~~
There are two types of memory, ~~Stack~~ stack & heap.

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

eg.

string a = "19";
int b = 20;



Note *: The primitive datatypes are directly stored in the heap & the non-primitive datatypes have reference variables so, the reference variable is stored in stack and object are stored in heap.

Stack memory: The stack memory is a physical space in RAM allocated to each thread at runtime. It is created when a thread creates. memory management in stack follows LIFO (Last In First Out) order because it is accessible globally. It stores the variable, references to objects and partial results.

Heap memory: It is created when the JVM starts up and used by the application as long as the application runs. It stores objects and JRE classes. It does not follow any order like stack. It dynamically handles the memory blocks.

JAVA TOKENS

A Java program is basically a collection of classes. A class is defined by a set of declarations statements and methods containing executable statements.

Smallest individual units in a program are known as tokens. The compiler recognizes them for building up expressions $ statements. In short, a Java program is a collection of tokens, comments $ white spaces. Java includes five types of tokens.

i) Reserved keywords: eg int, float.

ii) Identifiers: Those things that are used to identify variables, methods etc. eg: void fun(), here fun is a identifier,

iii) Literals: int i= 9; 9 is a literal assigned to i variable.

iv) Operators: +, -, ==, = etc.

v) seperators: ;, { }, ( ), " " etc.

Syntax, executing first Java program.

Structure of a Java file:

- Source code that we write has to be saved using .java extension or in other words the file type should be .java.
- A class with the same name as file name must be present in the file which has main method inside it.
- Class having the main method must be public to be accessed from other files.
- The entry point of the program execution is main function/method.

Converting .java files to .class file

- Using javac compiler we can convert .java file to .class

Syntax & steps:

- Go to the repository where the java files are stored.
- Open terminal: javac filename.java

-> Let the file name be test.java

- So the ~~com~~ command would be

  javac test.java

- The .class file is created. The .class file consists bytecode of the .java file which will be further ~~execut~~ executed by jvm & JRE. Jvm verifies, initializes and allocates memory and JRE provides environment to run the file.

Running the program.

To run the class file following command is used.

```
java <class file name>
```

eg;

```
java test.
```

4) Hello World Program

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

->

1) public (in first line): is an access modifier which allows to access the class from anywhere.

2) class: is a name of group of properties & functions.

3) test: is just the name of the class, as same as name of file.

4) public (in second line): is used to allow the program to use main method from anywhere.

5) static: is a keyword which helps main to run without making object of the class.

6) void: is a keyword when we dont want method to return anything after its execution.

7) main: is the name of the name of the method, also the entry point of the program execution.

8) String []args: It is a command line argument of String type array. The name of the String[] args is not mandatory. The array name can be anything according to the programmer.

9) System: is a final class defined in java.lang package.

10) out: is a variable of printstream type which is public & static member field of the system class.

11) println: is a method of PrintStream class, It prints the arguments passed to it and adds a new line. Insted of 'println' only 'print' can be used too.

Packages in Java

- It is just a folder where java files lies.
- It is used to provide some rules & stuffs to our program.