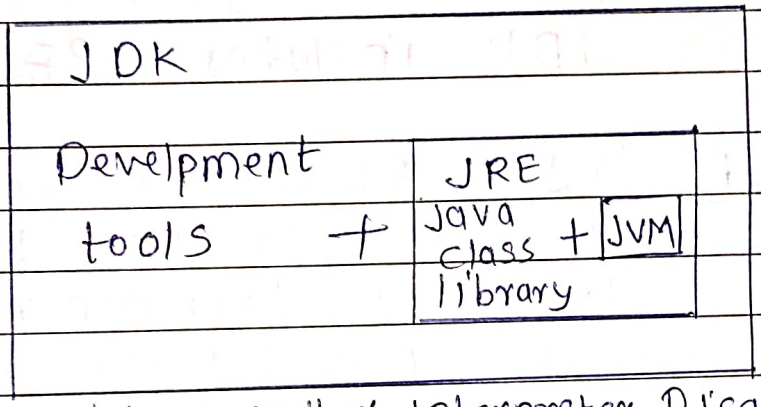
1) Explain the components of the JDK.

→

(i) JDK → Java development kit

(ii) It is a crucial component for Java software development providing essential tools for creating Java-based applications. and applets.

(iii) JDK contains JRE and tools.



(JDK consist compiler, interpreter, Disassembler & more)

(iv) Development tools in JDK:-

a) Javac (Java compiler) For converting source code into bytecode

b) Java (interpreter) for executing Compiled Java classes

c) Javap (Java disassembler) it Disassembles Compiled Java & class files.

and more tools like Javah, Jdb etc.

(v) JDK has Necessary tools for writing Java programs that can be executed by JRE & JVM.

Q) Differentiation Between JDK, JRE & JVM.

→

(i) JDK → Java Development Kit
→ JDK is a software development kit used for developing Java appln
→ JDK contains compiler, debugger, and core classes tools.
→ JDK is used to write, compile and execute Java programs.
→ JDK includes JRE.

(ii) JRE → Java Runtime Environment
→ JRE includes JVM, Java class libraries, Java binaries
→ JRE provides the platform to run Java application without development tools

(iii) JVM → Java virtual Machine
→ JVM executes bytecode by converting it to machine level code.
→ JVM has Memory Management, or garbage collection. etc.
→ JVM includes class loaders, run-time data area like method area & heap area. & an execution engine.

Date:

3) What is the role of the JVM in Java? How does the JVM execute Java code?

→

i) The main and crucial crucial role in executing Java code by converting Java bytecode into machine-specific code.

ii) JVM loads Java bytecode into memory using the class loader subsystem.

iii) JVM interprets and executes Java bytecode, managing memory

iv) JVM is responsible for calling the main method inside the class.

Execution of Java code by JVM:-

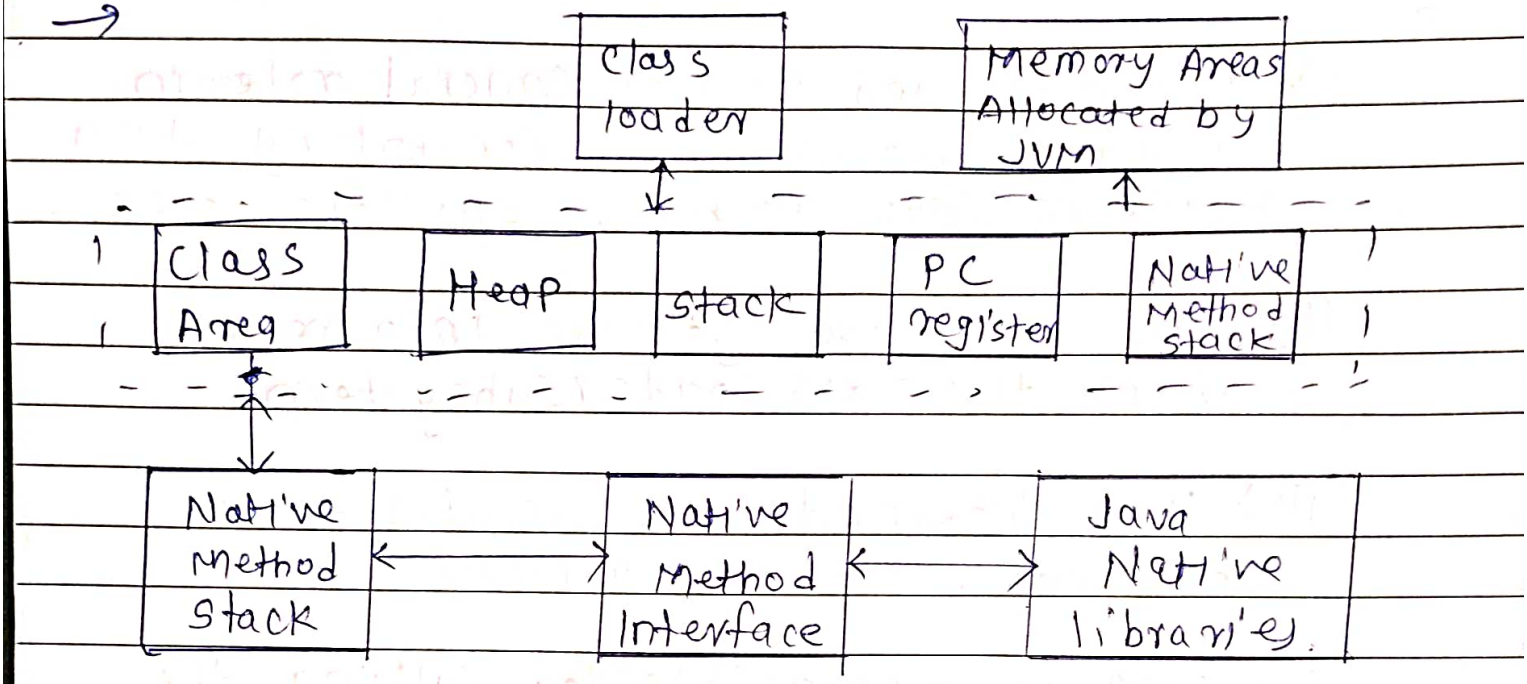
(i) After By code is loaded into memory by class loader JVM checks the wether the byte code is correct or not in accordance to Java language syntax.

(ii) JVM interprets the bytecode and maintains values for instructions, Memory allocation of objects & initialization of static variable. and then goes for execution of code.

(iii) JVM automatically deallocates unsed unused objects and optimizes the memory

4) Explain the memory Management system of the JVM.

→



(1) Java Virtual Machine manages memory through a process called garbage collection. It removes & identifies unused objects from memory to ~~deallocat~~ free that space for new allocation.

~~There are~~ JVM is divided into Parts :-

a) Heap Memory :-
This is a Runtime data section where memory for all Java class Instances & arrays is allocated heap is created when the JVM starts & may increase or decrease in size during execution.

b) Stack :- Stack memory is used for execution of a thread & contains method specific values that are short lived & refrence to other objects in the heap.

c) Method Area :- This area is part of non-heap memory & stores class structure like runtime constants & static variables.

d) Non-Heap Memory :-
Non-Heap memory includes, per class structures such as runtime constant pool, field & method data, constructor & method codes, ~~& interned strings~~.

e) Runtime constant pool :-
This area is a part of method area and contains class runtime constants and static methods.

(ii) JVM uses the mark and sweep garbage collection model for performing garbage collection of whole heap.
This process has two phases :-
(a) The mark phase → where all reachable objects are marked as alive.
(b) The sweep phase → where ~~all~~ the heap is traversed to find gaps betn live objects. which are recorded in a free list and made available for new object allocation.

Edabit to practice

M T W T F S S
Page No.: YOUVA
Date:

5) What are the JIT compiler and it's role it's role in the JVM ? what is the bytecode & why is it important for Java ?

→

the JIT (Just in time) compiler is a crucial component of the Java virtual machine (JVM), and it plays a significant role in improving the performance of Java application.

(i) when the source code is compiled, it's translated into bytecode, which is a platform independent code that can be executed on any device with a JVM.

(ii) JVM interprets bytecode by one by one instruction. then the bytecode gets converted into machine code. This m/c code is then executed directy by the CPU.

(iii) JIT compiler identifies these "hot spots" in the code, where certain section are executed frequently & optimizes them for better performance.

(iv) JVM —bytecode interprets→ checks bytecode frequently (Hotspots)

JIT get invoke when Hotspot identified → translate Hotspot's into m/c code → Performance Improved.

(v) Bytecode :-

a) Bytecode is an code which is genrated by Java compiler.
b) Java source file ".java" gets converted into bytecode ".class" file.
c) bytecode is designed to be executed by JVM, hence we call Java a platform-independent language.
d) bytecode can be run on any device with JVM
e) only JVM controls the execution of bytecode so it can provide security like memory management, access control, runtime checks etc.
f) JIT - compiles byte code into native machine code which allows Java Program to ahieve better performance

6) Describe the architecture of the JVM.

→ In the architecture of JVM it consists :-

i) Class loader :- It loades the java class files into the JVM at runtime. It performs tasks such as locating and reading class files, veryfing their bytecode, & defining the classes within JVM.

ii) Runtime Data Area :-
The Runtime Data Area is the Memory area where the JVM manages data during program execution.
It has
(a) Method Area :- stores class level data, including the bytecode of methods, constant pool, static variables & method data.

(b) Heap :- objects are allocated in heap.

(c) Java Stack :- stores method specific data, including local variables, method arguments, & method invocation records & manages method calls & returns

(d) Native Method Stack :- holds native method-specific data.
It is used for executing native method

(ii) Program Counter :- It keeps track of the currently executing bytecode instruction.

iii) Execution Engine :- It executes Java bytecode using JIT compilation, Interpretation or combination of both.

iv) Native Method stack :-

7) How does Java achieve platform Independence through JVM?

→ i) Java achieves platform Indepedence through the use of JVM & Java bytecode.
ii) Java Compiler (javac) converts the source code into bytecode, which is platform Independent Intermediate representation.
iii) the bytecode can be run on any operating system that has compatible JVM installed.
iv) JVM interprets bytecode and converts it into m/c code that can be executed by specific m/c. Each m/c has it's own JVM, which allows the same bytecode to be executed on different platform without modification.

8) Significance of class loader in Java? what is the process of garbage collection in Java.

→
i) The class loader is crucial component responsible for loading Java classes into memory at runtime.
ii) when Java program is executed one or more class loaders locate and load all classes needed to run the program.

Type :-

A) Bootstrap Class loader :→ It is Responsible for loading core Java API classes that are available in rt.jar file.

b) Extension class loader :-
Child of Bootstrap loader loads all classes from extension directory, which is typically ~~JDK or~~ JDK/JRE/lib/ext. etc.

c) Application class loader :-
It is child ~~loa~~ class of extension class loader & is responsible for loading classes from applications class path.

iii) class loader play a important/crucial role in Java programs to load & execute classes dynamically at runtime. they allow for flexibility & adaptability by enabling appln to load new classes on demand or from remote & locations or networks.