1) Explain the components of the JDK.

= All the Java-based Applications and Applets are developed by using JDK. It consist number of software components. One is JRE (Java Runtime Environment) and other tool like Java and Javac. that combinely called as development tool. JDK provides JRE is having JVM and other library files.

- JDK components

① JRE - Java Runtime Environment time.
② Java - It is simply a loader that works for all the java applications.
③ Javac - It is a compiler. It converts source code into Java bytecode.
④ appletviewer - Through this component we can run the Java applets without taking the help of web browser.
⑤ apt - This is used as an annotation processing tool.
⑥ javadoc - It is a documentating generator.
⑦ jar - It is a archiver alongwith related class libraries packages into one Jar file.
⑧ javap - It is a class file disassembler.
⑨ Jconsole - It is a Console Java monitoring and management.
⑩ jstack - A tool prints Java stack traces of java threads.
⑪ VisualVM - It is a visual tool. It is integrated with numerous command line JDK tools.

2) Difference between JDK, JVM and JRE

=

JDK

- JDK (Java development kit) is the core component of Java Environment and provides all the tools, executables and binaries required to compile, debug and execute a java program.
- JDK is a Platform Specific Software and that's why we have separate installers for Windows, Mac and Unix System.
- It contains JRE with Java compiler, debugger and Core Classes.

JVM

- JVM is a heart of Java programming language. When we execute a Java Program, JVM is responsible for converting Byte code to the machine Specific code.
- JVM is also platform dependent and provides Core Java functions such as memory management, garbage collection, security, etc.
- JVM is called Virtual because it provides an interface that does not depend on the underlying Operating System and machine hardware.

Page No.
Date

JRE →

- It provides a platform to execute Java Programs.
- JRE consists of JVM, Java binaries and other classes to execute any program successfully.
- JRE doesn't contain any development tools such as Java compiler, debugger, JShell, etc.
- If you just want to execute a java program you can only install JRE you don't need JDK because there is no development or compilation of java source code is required.

* JDK is for development purpose whereas JRE is for running the java programs.
* JDK and JRE both contains JVM so that we can run our java program.
* JVM is the heart of java programming language and provides platform independence.

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

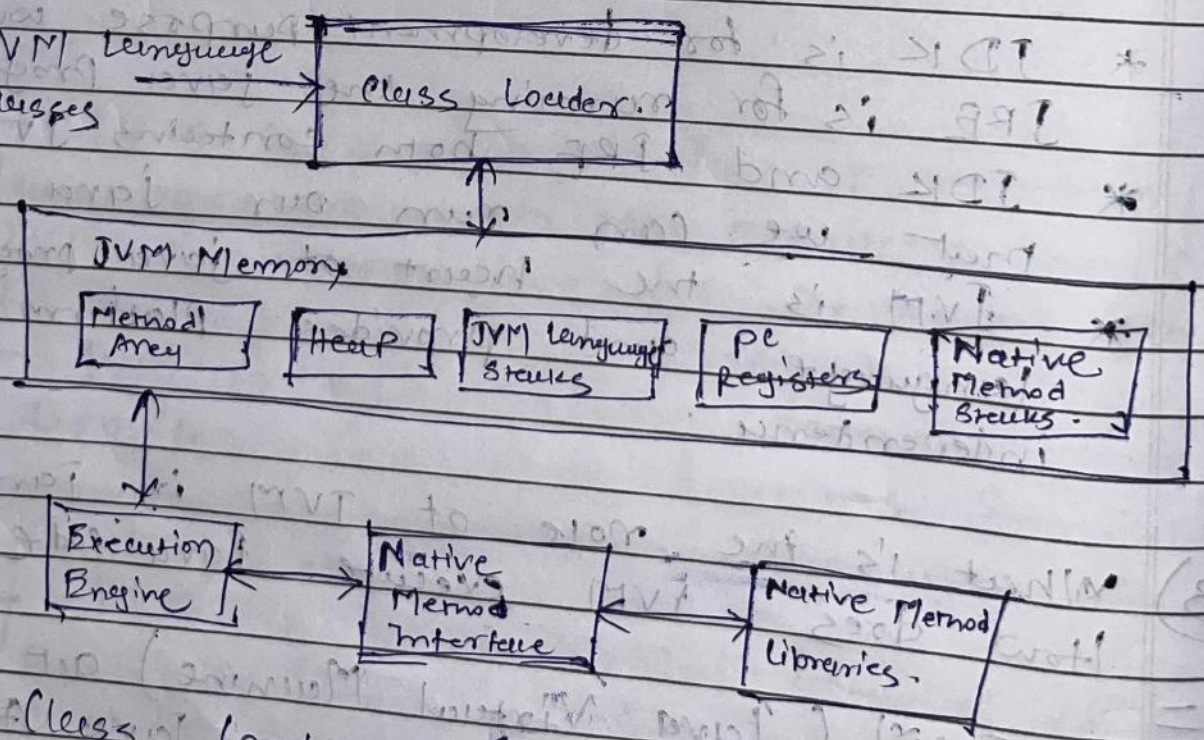
=

JVM (Java Virtual Machine) act as a run time engine to run Java Application. JVM is the one that actually calls the main method present in java code.

JVM is a part of JRE (Java Runtime Environment).

Java applications are called WORA (Write once Run Anywhere). This means a programmer can develop Java Code on one System and can expect it to Run on any other Java-enabled System without any adjustment. This is all possible because of JVM.

When we compile .java files with the same class's names present in .java file are generated by the Java Compiler. This .class file goes into various steps when we run it. These steps together describe the whole JVM.



* Class Loader Subsystem -
It is mainly responsible for three Activities.
① Loading ② Linking ③ Initialization
→ Bringing -

Loading :- The class loader reads the .class file, generate the corresponding binary data and save it in the method area.

Linking :- It performs verification, and resolution.

Initialization :- In this phase all Static Variable are assigned with their values defined in the code and static block. This is executed from top to bottom in a class and from parent to child in the class hierarchy.

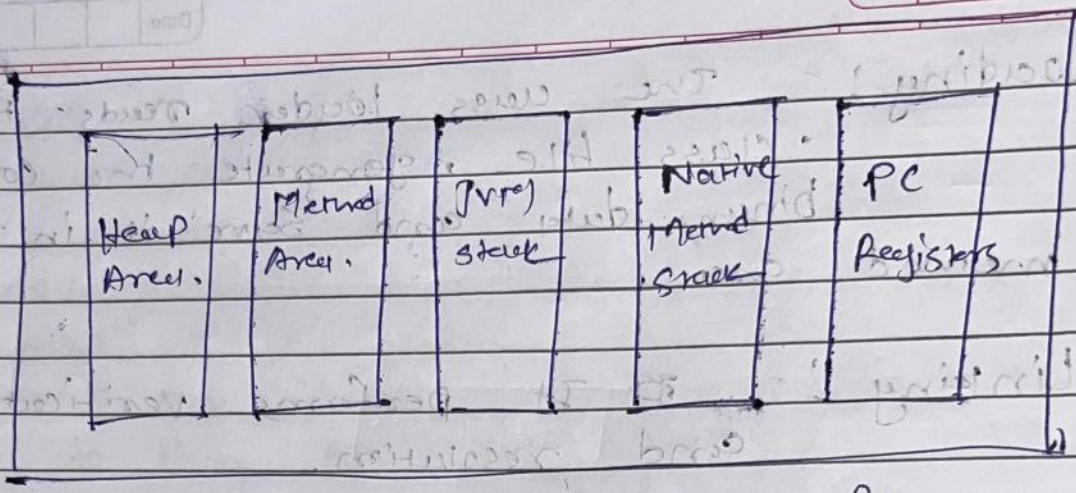
4) Explain the memory management system of the JVM.

=

In Java, JVM and Specific garbage collector has the role of managing memory allocation so that the programme needs not to.

JVM Memory Structure :-

JVM defines various run time data areas which are used during execution of a program. Some of the areas are created by the JVM whereas some are created by the threads that are used in a program.



JVM Memory Area Parts.

Heap :- It is a shared runtime data area and stores the actual object in a memory. This memory is allocated for all classes instances and arrays.

Method Area :- It is a logical part of the heap area and, it's created on virtual machine startup.

JVM stack :- A stack is created at the same time when a thread is created and is used to store data and partial results which will be needed while returning value for method.

Native Method stack :- Also called as C stack, native method stack are not written in Java Language. This memory is allocated for every thread when is created.

Program Counter register :- Every JVM thread contain carries out the task of the specific

method has a Program Counter register associated with it.

5) What are the JIT Compiler and its role in the JVM? What is the bytecode and why it is important for Java?

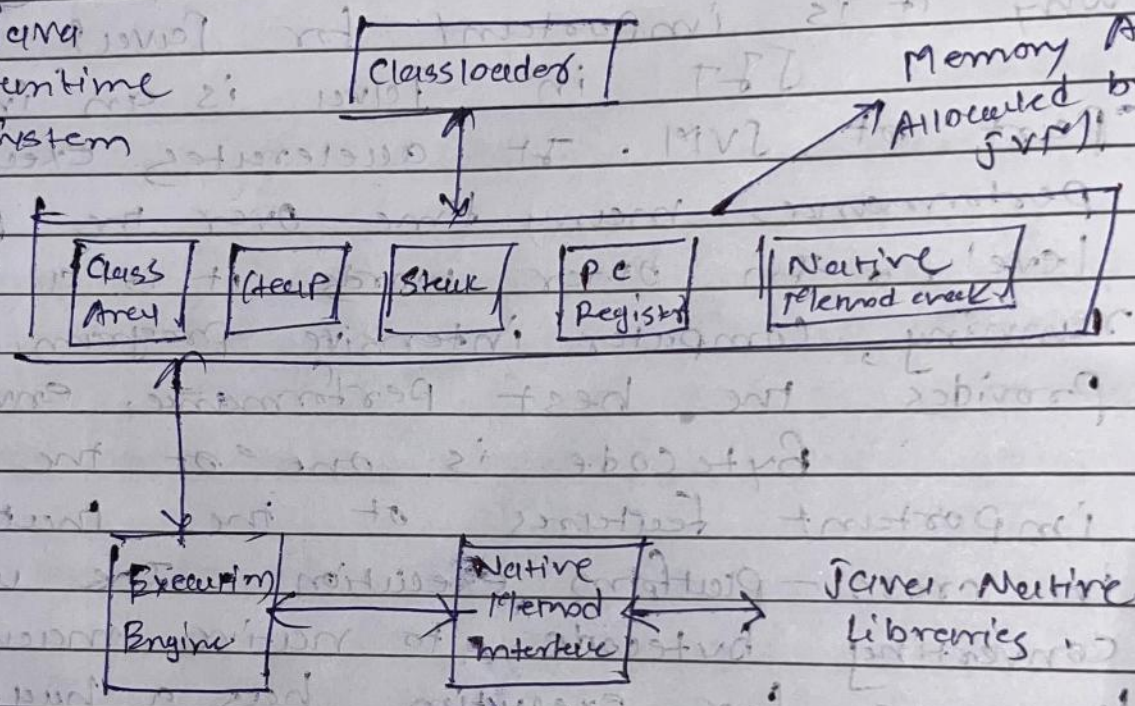
: JIT in Java is an integral part of JVM. It accelerates execution performance many time over the previous level. In other words it is a long running, computer intensive program that provides the best performance environment.

Bytecode is one of the most important features of the that aids in cross-platform execution. The way of converting bytecode to native machine language for execution has a huge impact on its speed of it. These bytecode have to be interpreted or compiled to proper meaning instructions depending on the instructions set architecture.

In order to ~~perform~~ improve performance, JIT compilers interact with Java Virtual Machine (JVM) at run time and compile suitaby bytecode sequences into native machine code.

6) Describe the architecture of JVM.

= JVM architecture contains classloader, memory area, execution engine, etc.



(A) Classloader :- Classloader is a subsystem of JVM which is used to load the class file.

(B) Class Area :- Class area stores per class structures as the runtime constant pool, field and method data, the code for methods.

(C) Heap :- It is the runtime data area in which objects are allocated.

(D) Stack :- Java stack stores frames. It holds local variables and partial results.

(E) Program Counter Register :- Program Counter register containing the address of the Java virtual machine instruction currently being executed.

(F) Native Method Stack :- It contains all the native methods used in applications.

(G) Execution Engine :- It contains A virtual processor, Interpreter, JIT compiler.

(H) Java Native Interface :- Java Native Interface is a framework which provides any interface to communicate with another application written in another language like C, C++, etc.

(7) How does Java achieve platform ~~inde~~ independence through the JVM ?

= (1) Compilation to Bytecode :- When you write java code, it is compiled into an ~~inter~~ intermediate representation called bytecode. This bytecode is not specific to any particular hardware or operating system.

(2) Interpretation or Just in time Compilation :- The bytecode then interpreted or compiled at runtime by the JVM

(3) Abstraction of Hardware and OS :- The JVM abstract away the hardware and Operating System differences.

(4) Java API Java also provides a Standard library that abstract away many Operating System - specific functions.

(8) What is the class loader in Java ?

= The class loader in Java is responsible for loading Java classes into the Java Virtual Machine at Runtime.