JVM Architecture Interview Questions (Anuja Karse)

Q.1) Explain Components of JDK

→ 1) Java Compiler (javac)

javac is a key component of JDK that transforms .java files to byte code i.e. .class file. This .class file can be executed on any platform using JVM.

2) JVM (Java Virtual Machine)

JVM is a runtime environment that executes bytecode. It enables Java Program to run independently of H/W & OS.

3) JRE (Java Runtime Environment)

It's a subset of JDK, that includes JVM and additional libraries required to run Java appl" on end user system without the need of development tools.

4) API libraries

It is a vast collection of pre built classes & methods. This covers areas such as I/P or O/P, N/W, Database connectivity etc.

5) Java Debugger (JDB)

It is a tool for debugging Java appl". It helps in understanding program flow & logical errors

6) Javadoc

Javadoc is a documentation that auto generates documentation from Java Source code.

Page

(Anusa Korse)

Q 2) Differentiate bet^n JDK, JVM & JRE

Feature	JDK	JVM	JRE
	Java development kit for creating Java appl^n	Java Virtual Machine for executing byte code	Java Runtime environment consists of JVM, class libraries
Components	Compiler, Debugger, Java libraries and other development tools	Provides platform independence by byte code	Contains JVM & libraries but lacks development tools
Usage	Used by developers	Used by developers to run byte code	Used by endusers to run Java appl^n
Dependency	JDK includes JRE along with development tools	JVM is a part of JDK as well as JRE	JRE contains JVM & libraries
Size	Larger in size	Small compared to JDK	Smallest compared to JDK & JVM

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

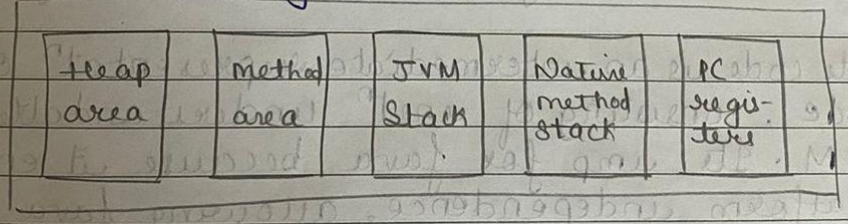
A) Role of JVM

- JVM acts as a bridge bet^n Java code & the H/w or OS.
- It provides runtime environment for Java

appⁿ, ensuring they can run smoothly across different platforms without modification
- JVM also handles memory management, garbage collectⁿ & other runtime task.

B) How JVM executes byte code
- When Java is compiled, it gets translated into platform independent byte code.
- JVM takes this byte code & loads in memory.
- It then interprets this byte code, this is done by Just In Time Compiler (native code specific to H/w/Platform)
- This native code, is then executed by JVM directly on the underlying Platform.

Q.4) Explain memory management Systems of JVM.
Java itself manages the memory and needs no manual intervention of the programmer.
Basic Memory Structure



Heap area: Here Objects are stored during runtime. Its dynamically allocated by JVM. Objects created by Java Programs like variables, arrays etc are allocated memory here.

Method area: Here memory is allocated for class structures, method data & constructor field data. Can be of fixed sized or expanded as required.

(Anusa Korse)

JVM Stacks : Created when thread is created and used to store data & partial results which will be needed while returning value for method.

Native method Stacks (C Stack) : Not written in Java language. Portion of memory reserved for executing native methods, implemented in languages other than Java.

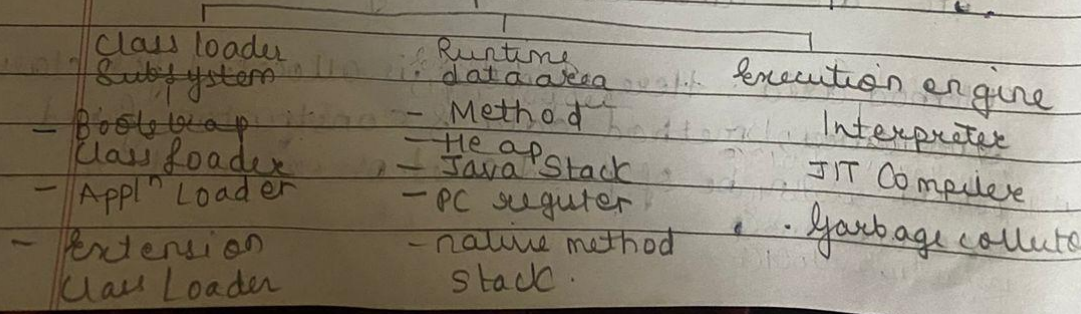
PC (Program Counter) Registers
Stores address of next available JVM inst^n.

Q.5) What are JIT & its role in JVM ? What is byte code & why its imp for Java ?

→ JIT in JVM compiles byte code into m/c code at runtime, optimizing performance by translating byte code into efficient m/c inst^n, increasing the execut^n speed of Java appl^n

Byte code is an intermediate representat^n of Java code, consisting of inst^n understandable by JVM. Its imp for Java because it enables platform independence, allowing Java to run programs on any device with JRE.

Q.6) JVM architecture



(Aruna Korre)

1) Class Loader Subsystem:
Responsible for loading .class file in RAM.
It comprises of Bootstrap, Extension & Appl° Loader

2) Runtime Data Area:
Divided into several memory areas like Method Area, Heap Area, Java Stack, Native Method Stack & PC register. They manage memory allocat°, method execut°.

3) Execution engine: Consists of Interpreter, JIT Compiler & Garbage Collector. Interpreter executes byte code line by line, JIT dynamically compiles frequently executed byte code to m/c specific code. The Garbage collector manages memory by releasing unused objects.

Q.7) How does Java achieve Platform independence through JVM?

→ It achieves Platform independence by compiling Java ~~to~~ source code into byte code. JVM then interprets byte code into m/c code understandable by underlying H/W. ensuring execut° across different platform without modifying actual code.

Q.8) What is the significance of Class Loader in Java? What is the process of Garbage collection in Java?

→ Class loader: dynamically loads Java classes into memory during runtime. It locate these classes from different sources like

(Anusa Kore)

files and makes them ready for use. They ensure that right classes are available when needed.

Garbage Collection is an automated process of clearing memory used. It involves
- Marking : In this, garbage collector identifies all reachable objects & marks them reachable.
- Sweeping : Scans heap to identify unreached objects
- Deletion : Finally it deallocates memory occupied by these unreached objects to efficiently use memory & avoid memory leakage.