Q1) Explain the components of the JDK

→ It includes :-

① Java Development tools :-
- javac, javap, java, jstack, jdb etc

② Java API Documentation :-
It's a valuable tool for developers to understand & use the Java programming language effectively

③ Java API Libraries.
- rt.jar

④ Execution environment
- Java virtual machine

⑤ Code Samples
- src.zip

Q2) Differentiate between JDK, JVM & JRE

→ ① JDK :- - It is called as Java development Kit
- It is used for developing Java application
- It includes JRE, JVM, Debuggers, Compilers etc

② JVM :- - Java virtual machine
- It is the engine that runs bytecode, enabling Platform independence

It is used by both developers as well as clients.

③ JRE :- Java Runtime Environment

- It provides the necessary components to run Java applications but does not include development tool.

- It includes JVM, Libraries, etc.

Q3) What is the role of the JVM in Java? & How does the JVM execute Java code?

→ Role :- ~~It is a critical~~

JVM is responsible for converting bytecode to machine-specific code & is necessary in both JDK & JRE

Execution :-

JVM enables a computer to run Java program as well as programs written in other languages that are also compiled to Java bytecode

Q4) Explain the memory Management system of the JVM

→ - heap :- It stores objects & class instances.

- stack :- It stores local variables & method call frames.
Stack memory managed & is limited in size

- garbage collection :- The JVM uses garbage collection algorithms to explicitly manage memory

Q5) What is the JIT compiler & its role in the JVM ? What is the bytecode & why is it important for Java.

→ JIT compiler :- Just in time compiler is a component of the JVM that improves the performance of Java applications.
It converts bytecode into native machine code at runtime, allowing the program to run faster

Byte code :- It is the intermediate representation of Java code generated by the java compiler
It is platform independent & is designed to be executed by the JVM

Q6) Describe the architecture of the JVM
→ It has 3 subsystem
(I) Class Loader System :-
It handles dynamic class loading, the classes are loaded at runtime instead of compile time.
It has 3 main parts
(I) Bootstrap, (II) Extension & Application
It involves three step
- loading, Linking & Initialization

(II) Runtime Data area
It contains Method area, Heap, Stack & etc

(III) Execution Engine :-
It containing Interpreter, JIT compiler & Garbage Collector

57) How does Java achieve platform independence through the JVM?

→ Java achieves platform independence through the use of bytecode & JVM. When you write & compile Java code, it is converted into byte code, which is platform - independent. This bytecode can be executed on any platform that has a compatible JVM., regardless of the underlying hardware & operating system.

58) Significance of the class loader in Java & Garbage collection process.

→ Class Loader - It is a part of JVM that loads class files into memory when required. It reads the class bytecode from a .class file & combining classes together & verifying their correctness. It initialize static variables & execute static blocks.

Garbage Collection process :-

- The JVM identifies which objects are still reachable from the root, references, unreachable object are marked for deletion after deletion, memory can be fragmented

Q9) What are the four access modifiers in Java & how do they differ from each other ?

→ ① Public :- It is accessible from any other class

② Protected :- the member is accessible within the same package & by subclasses in other packages

③ Default :- accessible only within the same package

④ Private :- only within the same class

Q10) What is the difference between Public, Protected & default access modifiers ?

→ - Public :- accessible from any other class

- Protected :- accessible within the same package & by subclasses in other packages

- Default :- accessible only within the same package

Q11) Can you override a method with a different access modifier in a subclass? For ex. can a protected method in a super class be overridden with a private method in a sub class? Explain overriding methods with different Access Modifiers

→ you cannot override a method with a more restrictive access modifier ex:- you cannot override a protected method in a super class with a private method in a subclass within because it would break the contract that the superclass provides to its users

Q12) What is the difference between protected & ~~default~~ default (package-private) access?

→

- Protected :- Access within the same package & to subclass in other packages.

- default :- Allows access only within the same package.

VEDHA
DATE : / /201
Page No. :

Q13) Is it possible to make a class private in Java ? If yes, where can it be done & what are the limitations ?

→ A class can be made private only if it is a nested class within another class. A top-level class cannot be declared as private.

Q14) Can a top-level class in Java be declared as protected or private? Why or why not?

→ A top-level class cannot be declared as protected or private It can only be public or have default access because top-level classes need to be accessible by the JVM to run the program.

Q15) What happens if you declare a variable, or method as private in a class & try to access it from another class within the same package?

→ If you declare a variable or method as private, it cannot be accessed from any other classes, even within same package. If you try to do it will an a compilation error

Q16). Explain the concept of "package-private" or "default" access. How does it affect the visibility of class members?

→ When a class member is declared without any access modifier it is said to have default or private access. It means that it is accessible only within the same package & is not accessible from classes in other packages