Interview Questions.

PAGE No.
DATE / /

- Explain the components of the JDK.

→ Java compiler : Converts java source code into bytecode for JVM to execute.

JVM: Runs the bytecode, enabling platform indepedent execution of Java applications.

JRE : Provides JVM and core libraries required to run Java app^n.

Java API : A collection of standard libraries and classes that offer reusable functionalities for development.

Java Debugger: A tool for debugging Java programs by setting breakpoints and inspecting variables.

Java Documentation: Generates HTML doc from source code comments for easier understanding of API's.

Java Archieve tool: Bundles java class files and resources into a single JAR file for distribution.

DATE / /

- Difference between JDK, JRE, JVM.

→ JDK is for development purpose whereas JRE is for running java programs. JRE & JDK both contains JVM so that we can run our java program. JVM is foundation of java programming lang and provides platform independence.

- Role of JVM in Java ? How does JVM execute Java code.

→ JVM is virtual machine that enables execution of Java bytecode. JVM acts as an interpreter between Java lang and underlying hardware. It provides a runtime environment for app's to run on different platforms & OS.

- Memory management system of JVM.

→ In Java, it is an automatic process that is managed by JVM and one that does not need explicit intervention.
Java being a block structured lang, uses a model where its memory is divided into 2 types : stack & heap.

PAGE No.
DATE / /

- JIT compiler and its role in JVM.

→ It is a component of JVM.
Improves performance of Java app^n
Translates bytecode into native machine code at runtime.
Instead of interpreting each bytecode instruction repeatedly.

- What is bytecode and why it is imp in Java.

→ It is an intermediate code that Java compiler generates from Java source code.
It is a set of instructions that the JVM interprets or compiles (via JIT) into machine specific code.
makes java portable and secure & platform independent via JIT.

- JVM architecture:

① Class loader - used to load class files.
② ↳ Bootstrap - rt.jar file is loaded. constains all files of JSE.
↳ Extension - $JAVA_HOME/jre/lib/ext directory.
Child classloader of bootstrap and loads files located inside this directory.

↳ System/ App^n classloader : Child classloader of to extension classloader

It loads files from classpath, by default current directory. (-cp)

② class (Method) area - Includes heap (for object storage)
stacks (for method calls & local variables)
Method area (for storing class structures & metadata)
PC registers (for storing address of currently executing JVM instructions.

③ Execution Engine - Consists of:
Interpreter (interprets bytecode) and JIT (just in time) compiler - compiles bytecode to native machine code for performance optimization.

④ Garbage Collector: manages automatic memory management by reclaiming memory used by objects no longer in use.

- How does Java achieve platform independence through JVM.

→ Java compiles code into → bytecode which is platform independent. JVM interprets bytecode and runs it on any machine with a JVM, making Java platform independent. "Write once, run anywhere"

- Class loader : loads Java classes into memory dynamically at runtime.

Garbage collector : automatically frees memory by removing unused objects to prevent 'memory leakage'.

- 4 Modifiers & its diff:

⇒ Private: Accessible only within same class.

Default: Accessible within same pacakage.

Protected: Same pacakage & subclasses

Public: Accessible anywhere.

- Overriding with different access modifiers:

→ No, you cannot override a method with a more restrictive access modifier.

For eg: a protected method in a superclass cannot be overriden as private in a subclass.
You can only increase the visibility (eg. protected to → public).

- Difference → Protected & default.

→ Protected: Accessible within same package & by subclasses.

Default: Accessible only within the same package, not by subclasses outside the package.

- Is it possible to make a class private in Java?

→ A nested class can be declared as private within another class. but a top-level class cannot be private! The private class is only accessible within outer class.

- Can a top-level class in Java be declared as protected or private?

→ No, a top-level class cannot be protected or private. Only public or default access is allowed for top-level classes because top-level classes need to be accessible by the JVM & also the other classes.

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

→ If a method / variable is private it cannot be accessed from another class, even if it's within same package. They are only accessible within a class in which they are declared.

- Explain concepts of package private or default access. How does it affect visibility of class members?

→ Default - (package - private) access:

If a class member has default access, it is accessible only to classes within the same package. But not from classes in other packages. This restricts access more than public or protected.