

## • Assignment No. - 3.

### \* Interview Questions.

Q.1] Explain the components of JDK :-

→

- ① Java Compiler (javac) :- Converts Java source code into bytecode.
- ② Java Runtime Environment (JRE) :- Provides the libraries necessary for java development, the JVM & other components needed to run Java applications.
- ③ Java Virtual Machine (JVM) :- Executes Java bytecode. It provides an environment in which java programs can run, regardless of underlying O.S.
- ④ Debugger (JDB) :- A tool to help in debugging java programs.
- ⑤ Documentation Generator (Javadoc) :- Generates API documentation in HTML format from Java source code.
- ⑥ Jar Tool (Jar) :- Packs & unpacks Java archive (JAR) files, which bundle related class files & associated metadata.
- ⑦ Other tools :- This includes tools for generating code, managing documentation & handling security, among other tasks.



Q.2] Differentiate between JDK, JVM & JRE

→

- 1.] JDK (Java development kit) :- It is a full featured software development kit used to develop java app<sup>n</sup>. It includes the JRE, compiler & other tools necessary for development.
- 2.] JRE (Java Runtime Environment) :- It provides the libraries & JVM needed to run java app<sup>n</sup>. It does not include development tools like compiler.
- 3.] JVM (Java Virtual machine) :- The engine that executes java bytecode. It is part of JRE & is responsible for converting bytecode into machine code & executing it.

Q.3] What is role of JVM in java? & How does JVM execute java code?

→

- ① Loading :- The JVM loads the bytecode files (.classfiles) into memory.
- ② Verification :- It verifies the bytecode to ensure it adheres to Java's safety & security constraints.
- ③ Execution :- It converts bytecode into machine code into machine code specific to the host machine & executes it.
- ④ Management :- The JVM manages memory allocation & garbage collection & optimizes performance through Just in Time compilation.



Q.4] Explain the memory management system of the JVM :-

→ The Java virtual machine (JVM's) memory management system includes components like :-

- ① Heap :- The area of memory used for dynamic allocation of objects and arrays. It is divided into the young generation & old generation.
- ② Stack :- Each thread has its own stack, which stores local variables, method calls & partial computations.
- ③ Method area :- Stores class-level data, including runtime constant pool, field & method data & method code.
- ④ PC Register :- Each thread has a program counter (PC) register that keeps track of current instruction being executed.

Q.5] What are JIT compiler & its role in JVM? What is bytecode & why is it important for Java?

→ ① JIT Compiler :-

The Just-In-Time compiler is a part of JVM that improves performance by compiling bytecode into native machine code at the runtime, rather than interpreting it. It helps speed up execution by converting frequently executed bytecode sequences into machine code, which is then cached & reused.



## ② Bytecode :-

This is the intermediate representation of Java code, produced by java compiler. Bytecode is platform independent & can be executed on any JVM. Its importance lies in its ability to provide portability & enable the "write once, run anywhere" capability.

## Q.6) Describe the architecture of JVM.

→ The JVM architecture includes:-

- ① Class Loader Subsystem:- Loads Java classes & interfaces.
- ② Runtime data areas:- Includes, heap, stack, method area & PC register.
- ③ Execution Engine:- Contains the interpreter & JIT compiler. It executes bytecode instructions.
- ④ Native Interface:- Provides access to native libraries & code.
- ⑤ Garbage Collector:- Handles memory deallocation & recycling.

## Q.7) How does achieve platform independence through the JVM?

→ Platform Independence.

Java achieves platform independence through the JVM by compiling Java code into bytecode, which is then interpreted or compiled into native machine code by the JVM on the target platform. This way Java programs can run on any device.



or OS that has compatible JVM.

Q. 8] What is significance of class loader in Java?  
What is process of garbage collection in Java?

→ (A) Class loader :-

The class loader is responsible for loading classes into JVM. It performs three main functions-

Loading :- Finds and loads the class file.

Linking :- Verifies, prepares & resolves symbolic references.

Initialization :- Initializes class variables & executes static blocks.

(B) Garbage Collection :-

It is the process of automatically reclaiming memory by deleting objects that are no longer in use. The JVM uses various GC algorithms (e.g., generational, mark-and-sweep) to identify and collect unused objects, thus preventing memory leaks & optimising memory usage.

Q. 9] What are the four access modifiers in Java, & how do they differ from each other?

→ In Java, the four access modifiers control the visibility of classes, method & variables :-

- Public :- Members are accessible from any other class, regardless of package.

- Protected :- Members are accessible within the same package & by subclass in different packages.

- Package Private (Default) :- Members are accessible only within the same package. No explicit keyword is used; if no access modifier is specified



it defaults to this.

- Private :- Members are accessible only within the class where they are defined.

Q.10] What is the difference between public, protected and default access modifiers?

→

- Public :- Allows access from any other class in any package. It is the least restricted access level.

- Protected :- Allows access within the same package and any subclass, even if they are in different packages. It is less accessible than public.

- Package - Private (Default) :- Allows access only within the same package. This access level is more restrictive than protected & public as it does not allow access from subclass outside the package.

Q.11] Can you override a method with a different access modifier in a subclass? For Ex. Can a protected method in a superclass be overridden with a private method in subclass? Explain.

→

No, we cannot override a method with a more restrictive access modifier. When overriding a method, the access level of overriding method must be same or less restrictive than method in superclass. For ex., we cannot override a protected method with a private method.



in a subclass because private is more restrict-ive than protected. The overriding method must maintain or increase the level of access.

q.12] What is the difference between protected & default (Package-Private) access?

→ Protected :- Members are accessible within the same package & by subclass in other packages. This means that subclass, even if they are in different packages, can access protected members.

default (Package-Private) :- Members are accessible only within the same package. Subclass in other packages cannot access these members. The default access level does not allow access from outside the package, regardless of inheritance.

q.13] Is it possible to make a class private in Java? If yes, where can it be done & what are the limitations?

→ No, we cannot declare a top-level class as private. Top-level classes can only be declared as public or package-private. The reason is that the access modifiers protected & private are meant to control visibility within the scope of a package.



The private modifier is applicable only to nested (inner) class, & it restricts access to within the containing class. Top-level classes need to be accessible to other classes within the package or, if public to any class outside package.

Q. 14] Can a top-level class in Java be declared as protected or private? Why or Why not?

→

No, a top-level class cannot be declared as protected or private. Top-level classes can only be declared as public or package-private. The reason is that the access modifiers protected & private are meant to control visibility within the scope of a package or a specific class, which does not apply to top-level classes. The Java language design requires that top-level classes be either accessible to all classes in package or globally accessible if public.

Q. 15] 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 a variable or method is declared as private, it cannot be accessed from any other class, even if those classes



are in same package. The private modifier restricts access strictly to the class in which it is declared. This is designed to enforce encapsulation & ensure that internal details of a class are not exposed outside of that class.

Q.16] Explain the concept of "package-private" or "default" access. How does it affect visibility of class members?

→ Package-private, also known as default access, occurs when no explicit access modifier is specified. Members with the package-private access are visible only to other classes within the same package. They are not accessible to classes in other packages, even if those classes are subclasses. The access level provides a way to restrict visibility while still allowing related class within the same package to interact with each other.