Java Interview Questions

Assignment No:- 03

Date
Page

1. Explain the Components of the JDK

→ - The JDK Consists of several key Components:

① JVM (Java Virtual Machine): Executes java bytecode

② JRE (Java Runtime Environment):
Bundles libraries, java virtual machine (JVM), and other components to run application written in java.

③ Compiler (Javac):
Translate java source code into bytecode.

④ Java API Libraries:
A set of standard libraries and API's that provide useful functionalities.

⑤ Development Tools:
Includes utilities such as javap (disassembler), javadoc (documentation generater), and jdb (debugger).

2. Differentiate between JDK, JVM and JRE

→ ① JDK (Java Development kit):
A full featured software development kit used for developing java application. It include the JRE, Compiler and other development tools.

② JVM (Java Virtual Machine):
- The engine that provides the runtime environment to execute java bytecode. It is part of the JRE

③ JRE (Java Runtime Environment):
Provides libraries and the JVM necessary to run java application. It does not include development tools like Compiler.

3 What is the role of the JVM in java?

→ The JVM is responsible for executing Java bytecode. It performs several tasks:

① Loading: Loads Java bytecodes from the class files.

② Linking: Verifies and prepare code, resolving symbolic reference.

③ Initialization: Initialize classes and their static variables.

④ Execution: Executes the bytecode instruction using an interpreter or JIT Compiler. JVM uses a combination of interpretation (executing bytecode line by line) and Just-In-Time (JIT) compilation (compiling bytecode into native machine code at runtime) to execute Java code efficiently.

4. Explain the memory management system of Java.

→ ① Heap:
Stores objects and their associated data. Its managed by the garbage collector.

② Stack:
Holds method call and local variables. Each thread has its own stack.

③ Method Area:
Stored ~~data~~ class definition and static variables.

④ Program Counter (PC) Register:
Keep track of the current instruction being executed.

⑤ Native Method Stack:
Manages native method calls (methods written in language other than Java)

- Memory management in the JVM includes automatic garbage collection, which reclaims memory used by objects that are no longer referenced.

5. What are the JIT compiler and its role in the JVM? What is the bytecode and why is it important for java?

→ ① JIT Compiler (Just-In-Time Compiler):
A component of the JVM that improve performance by compiling bytecode into native machine code at runtime. This machine code is then executed directly by the CPU, which speeds up the execution of Java application

② Bytecode:
The intermediate representation of java code that is platform-independent. Bytecode is important because it allows Java programs to be written once and run everywhere, as the JVM on each platform can interpret or complier bytecode into native machine code

6. Describe the architecture of the JVM.

→ ① Class Loader Subsystem:
Loads Java classes into the JVM

② Runtime Data Areas:
Includes heap, stack, method area, and PC register.

③ Execution Engine:
Execute bytecode using an interpreter or JIT Compiler

④ Native Interface:
provides access to native code (eg. C/C++) through the Java Native Interface (JNI)
⑤ Garbage Collector:
Manages memory by reclaiming unused objects

7. How does Java achieve platform independence through the JVM?
→ Java code is compiled into bytecode, which can run on any device with a JVM. This makes Java programs portable across different system
- JVM implementation:
Each platform has its own JVM implementation that translates bytecode into native machine code for that platform
- Abstraction:
The JVM provides a uniform execution environment, allowing Java program to run consistently across different systems.

8. What is the significance of the class loader in Java? What is the process of garbage collection in java?
→ - Role:
Loads classes into the JVM at runtime
It handles the process of finding and loading class files from the filesystem or network
- Process:
Involves loading, linking (verifying and preparing classes), and initializing classes.

Garbage Collector:

- Role:
  Automatically reclaimed Mover memory by identifying and discarding objects that are no longer in use
- Process:
  The garbage collector periodically check the heap for unused objects and clean them up, freeing memory for new objects

9. What are the four access modifiers in java, and how do they differ from each ones.

→ (1) Public:
~~As~~ Accessible from any other class or package
(2) Protected:
Accessible within the same package and by subclass, even if they are in diffent packages.
(3) default (package-private):
Accessible only within the same package
(4) Private:
Accessible only within the defining class

10. What is diffence between public, protected and default access modifier?

→ (1) Public:
Members are accessible from any class, regardless of package or class.
(2) Protected:
members are accessible within the same package and subclass, even if they are in diffent packages.
(3) default (package-private)
Members are accessible only within the same package and not from outside the package or from

Garbage Collector:

- Role:
  Automatically reclaimed Maven memory by identifying and discarding objects that are no longer in use.
- Process:
  The garbage collector periodically check the heap for unused objects and clean them up, freeing memory for new objects.

9. What are the four access modifiers in java and how do they differ from each other.

→ (1) Public:
As Accessible from any other class or packa

(2) Protected:
Accessible within the same package and by subclass, even if they are in diffent package

(3) default (package-private):
Accessible only within the same package

(4) Private:
Accessible only within the defining class

10. What is diffence between public, protected and default access modifier?

→ (1) Public:
members are accessible from any class, regardl of package or class.

(2) Protected:
members are accessible within the same package and subclass, even if they are in diffent packages.

(3) default (package-private)

Subclass in different packages

11. Can you override a method with a different access modifier in a subclass? For example, can a protected method in a superclass be overridden with a private method in a subclass? Explain

→ You cannot override a method with a more restrictive access modifier. For example, a protected method in a superclass cannot be overridden by a private method in a subclass. The access level of the overriding method must be the same or more permissive than the method in the superclass.

12. What is difference between protected and default access modifier?

→ Protected:
Accessible within the same package and by subclass outside the package
Default (package-private):
Accessible only within the same package and ~~^~~ not by subclass outside the package

13. Is it possible to make a class private in java? If yes, where can it be done, and what are the limitations?

→ NO. You cannot declare a top-~~class~~ level class as private in java. Top-level classes can be public or default (package-private). Nested classes (inner class) can be declare as private.

14. 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 package-private (default). This is because top-level classes need to be accessible to the java runtime system and other classes, and restricting access with protected or private would limit their visibility in ways that are inconsistent with the intended use of top-level classes.

15. What happen if you declare a variable or method as private in a class and try to access it from another ~~from~~ class within the same packages?

→ If you declare a variable or method as private in a class, it cannot be accessed from another class, even if they are in the same package. The private modifier restricts access to the defining class only.

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

→ Default (Package-private) access:

Definition:

When no access modifier is specified, the member is package-private, also known as default access.

Visibility:

Members with package-private access are visible to all classes with the same package but are not accessible from classes in other packages.