CDAC Mumbai PG-DAC AUGUST 24

Assignment No- 3

**Note: Write down this Interview questions & answers in your notebook .take a screenshorts ,make word file & upload on Github.**

* Explain the components of the JDK.

Java Virtual Machine (JVM), Debugger, JavaDoc, Jar, Libraries etc.

* Differentiate between JDK, JVM, and JRE.

1. JDK (Java Development Kit) is a Kit that provides the environment to develop and execute(run) the Java program. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), and other tools needed in Java development.

JDK is a kit(or package) that includes two things:

Development Tools(to provide an environment to develop your java programs)

JRE (to execute your java program).

2. JRE (Java Runtime Environment) is an installation package that provides an environment to only run(not develop) the java program(or application)onto your machine. JRE is only used by those who only want to run Java programs that are end-users of your system.

3. JVM (Java Virtual Machine) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as an interpreter.

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

JVM is responsible for converting bytecode to machine-specific code.

The execution process mainly includes:

(1) The Java source code is compiled into bytecode.

(2) Verify the bytecode and load the Java program through the class loader into JVM memory.

(3) Create a class object for each class after the loading and put it into the method area

(4) Initialize bytecode instructions and data into memory

(5) Find the main() method and create a stack frame

(6) Initialize the value inside the program counter as the memory address of the main() method

(7) The program counter increases continuously, executes Java bytecode instructions one by one

* Explain the memory management system of the JVM.

Memory management is the process of allocating new objects and removing unused objects to make space for those new object allocations.

* What are the JIT compiler and its role in the JVM? What is the bytecode and why is it important for Java?
* Describe the architecture of the JVM.

Garbage collector: Automatically frees up memory that is no longer being used by the program.

Interpreter: Interprets bytecode, but is slow and requires a new interpretation each time a method is called.

JIT compiler: Compiles bytecode into native code, which is used for repeated method calls to improve system performance.

Method area: Stores code, constants, and other class data.

Heap: Stores objects and arrays.

Stack: Stores method calls and local variables.

Local variables: Serve the same purpose as registers and are used to pass method arguments.

* How does Java achieve platform independence through the JVM?
* What is the significance of the class loader in Java? What is the process of garbage collection in Java.?

9)What are the four access modifiers in Java, and how do they differ from each other?

Public:

Makes a method, variable, or class accessible to everyone and on every platform.

Private:

Makes a class, method, or constructor only accessible within its own class.

Protected:

Makes a variable or method accessible within its own package and subclasses.

Default:

Makes a class or method only accessible within its own package. This is also known as "package-private" or "friendly" access.

10) What is the difference between public, protected, and default access modifiers?

Public

This access modifier allows any class in a Java project to access the properties or methods it modifies. It's the most permissive access modifier, allowing access from anywhere.

Protected

This access modifier allows access to the class it's applied to, as well as any child classes, even if they're outside the package.

Default

This access modifier, also known as package-private, is the default option in Java. It allows classes within the same package to access the property, but not classes outside of the package.

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.

Yes, the protected method of a superclass can be overridden by a subclass. If the superclass method is protected, the subclass overridden method can have protected or public (but not default or private) which means the subclass overridden method can not have a weaker access specifier.

12) What is the difference between protected and default (package-private) access?

The default access level is simple: The element will be accessible from any code in the same package. The protected access level is more accessible (that is, less protected) than the default access level.

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

Private classes are allowed, but only as inner or nested classes. If you have a private inner or nested class, then access is restricted to the scope of that outer class. If you have a private class on its own as a top-level class, then you can't get access to it from anywhere.

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

No, we cannot declare a top-level class as private or protected. It can be either public or default (no modifier). If it does not have a modifier, it is supposed to have a default access.

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

The methods or data members declared as private are accessible only within the class in which they are declared. Any other class of the same package will not be able to access these members.

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

When no access modifier is specified, Java uses a default access level, often called package-private. This means the member is accessible only within classes in the same package. It is less restrictive than private but more restrictive than protected and public.