DATE

OOPJ- Assignment NO.3 (Interview Questions)

Q1] Explain the ~~concepts~~ components of JDK

ANS: JDK (Java Development kit) has three main component

1. JRE (Java Runtime Environment):
It allows you to run Java Programs, It includes the Java virtual machine (JVM) and core liabraries.

2. JVM (Java virtual Machine):
It is responsible for running the Java bytecode (The compiled version of java code) JVM makes sure the same Java program can run on different computers.

3. Development tools:
These tools are like Java compiler (javac), debugger and other utilities to write, compile and debugg Java programs.

Q2] Differentiate between JDK, JVM, JRE

Difference between JDK, JRE, JVM -

JDK - It is a Software development environment that Provide tools to devlop Java applications. It includes tools like JRE & compiler.

JVM - Its virtual Machine that runs Java bytecode, It allows Java program to be executed on different program.

DATE

JRE - Java Runtime environment - Its a pakage of software that includes the JVM and libraries to run Java applications, but it doesn include development tools like JDK

Q3] What is the role of JVM in Java? How does the JVM execute the Java code.

ANS: The Java Virtual Machine (JVM) is responsible for running a Java program. It provide an environment where Java bytecode (compiled Java code) is executed. The JVM allows Java to be platform independent, meaning the Same Java code can run on any device with a compatible JVM.

• How the JVM executes the Java code -

1. Compilation :- Java source code firsly compiled to bytecode by the Java Compiler

2. class loading :- The JVM's class loader loads the .class files Into memory.

3. Bytecode Verification :- The bytecode is checked for errors ensuring it doesnt violates Javas security rules.

4. Execution : - the JVM's Interpreter or Just in time compiler converts bytecode into machine code and execute on the host machine.

DATE

5. Garbage Collection :- The JVM manages memory by atomatically freeing unused objects during execution.

Q4 Explain the memory management system of JVM

ANS:- The memory management system of JVM is divided into several areas, each responsible for storing different types of data during program execution:

1. Heap : Stores objects and instance variable. This is where the JVM allocates memory dynamically at runtime. Garbage collection happens here to free up unused objects.

2. Stack : Holds local variables and function calls. Each thread has its own stack, and memory is released when method returns.

3. Method Area : Contains class level data like method code strict variable, and runtime constant pool

4. PC Registers : keeps tracks of the current Instruction being executed for each thread.

5. Native Method Stack : Manages calls to native (Non-Java) code

The JVM's Garbage collector is responsible for atomatically managing memory in the heap by removing unused objects.

DATE

Q5] ~~JIT Compl~~ What are the JIT Compiler and its role in the JVM? What is the bytecode and why it is important for Java.?

ANS: JIT compiler and its Role in the JVM:

- JIT (Just In Time) Compiler is a part of the JVM that converts bytecode into machine code at runtime for faster execution.
  The role of JIT compiler is it compiles frequently used code sections, improving performance by avoiding reapeated interpretation.

Byte code :-

It is the intermediate, platform independent code generated by Java compiler
It allows Java to run on any platform with JVM making it portable and enabling the "write once, run anywhere" Feature

Q6] Describe the Architecture of JVM

*ANS: The Architecture of the JVM consist of

1. class loader : loads Java files (complied byte code) Into memory.

2. Byte Code verifier: Ensures the loaded bytecode is valid and doesn't violate any security rules.

3. Interpreter :- Reads and execute the bytecode line byline

DATE

4. Just In Time (JIT) compiler - Converts bytecode into native machine code for foster execution.

5. Runtime Data Areas: Includes:
- Method Area: Stores class structures like methods and variables.
- Heap: Allocates memory for objects.
- Stack: Holds method calls and local variable.
- PC Register: Tracks the next instruction for each thread.
- Native Method Stack: Manages native (non-Java) method calls.

6. Garbage Collector: Automatically reclaim memory by removing unused objects.

Q7] How does Java achieve platform independence through JVM?
Java achieves platform independence through the Java virtual machine (JVM) by using an intermidiate form of code called bytecode. Here's how the process work:

1. compilation to bytecode: When a Java program is compiled, the Java compiled, converts the source code into bytecode, which is platform independent.

2. JVM Execution: The bytecode not specific to any machine or operating system. Instead it is executed by the JVM. which is available for various platform (Windows, Linux, macOS, etc) Each platform has its own implementation of the JVM which translates the bytecode into machine code for that specific system.

Since the JVM abstract away the underlying hardware and operating system differences, a Java program can run any machine that has a compatible JVM installed, achieving the "write once, run anywhere" principle.

Q8] What is the significance of the class loader in Java? What is the process of garbage collection in Java.

Significance of the class loader in Java.

1. Dynamic class loading: classes are loaded when required, saving memory and improving efficiency.

2. Namespace Management: It ensures that each class is loaded within its own namespace, avoiding conflicts.

3. Security: The class loader ensures that classes are loaded from trusted sources and can enforce access restrictions

4. Extensibility: Developers can define custom class loaders for special class-loading mechanisms

• Process of garbage collection.

1 Making: The GC identifies which objects are still reachable in use and marks them.

2. Sweeping: It identifies unreferenced objects that are no longer needed.

3. Compacting:-

DATE

After reclaiming memory from unreferenced objects, the remaining objects are compacted to reduce fragmentation and make memory allocation more efficient.

Java's garbage collection process runs in the background improving memory management without the need for explicit memory deallocation by the developer.

Q9] What are four access modifiers in Java, and how do they differ from each other. ?

Modifier	class	Package	Subclass	World
Public	Yes	Yes	Yes	Yes
Protected	NO	Yes	Yes	NO
Default	NO	Yes	NO	NO
Private	NO	NO	NO	NO

- Public : Maximum visibility ! accessible from anywhere
- Protected : Limited to the same package and subclass.
- default : Limited to the same package.
- Private : Restricted to the deltaring class only.

These access modifiers allow developers to design robust and secure Java application by controlling how different parts of the code interact with each other.

Q 11] Can you override a method with a different access modifiers in a Subclass? for example can a protected method in a superclass be overridden with a private method in a sub class explain.

DATE

No, we can't override a method with more restrictive access modifier. A protected method in a superclass cannot be overriden with a private method in a Subclass. The access level of the orriding method must be the same or more permissive.

Q12] What is the difference between protected and default (Pakage - private) access?

The difference between protected and defalt (Pakage-Private) acess is:

- Protected : The member is accessible within the same pakage and also by subclasses, even if they are in different pakages.

- Default Pakage :- (Pakage Private) The member is only accessible within the Same pakage, not outside it or by Subclasses in other pakages.

Q13] Is it possible to make a class in private in Java? If yes, where can it be done and what are limitatations?

ANS:- Yes It is possible to make a class private in Java but it can be done only for nested (inner) classes. A top level class can not be private.

Limitations:-

- A private nested class can only be accessed within the enclosing class.

• A top level class in Java can only have public or package - private (default) access, not private.

Q14] Can a top level class a Java be declared as protected or private. why or why not?

ANS: No, a top-level class in Java cannot be declared as protected or private.

This is because access modifiers like protected and private are meant to control access within classes and inheritence for top level classes., the only allowed access modifiers are public (accessible everywhere) or pakage private Allowing protected or private would conflict with the visibility rules for top-level classes, which must be accessible to the entire pakage or beyond.

Q15] What happens if you declare a variable or method as private in a class and try to access ~~it~~ it form another class within the same pakage.

ANS: If you declare a variable or method as private in a class and try to access it form another class within the same pakage, you will get compile-time error. Private members are only accessible within the same class and not from the classes, even if they are in the same pakage.

DATE

Q16] Explain the concept of "Package-Private" or "defalt" access
How does it affect the visibility of class members.?

ANS: Package-Private (or "default") access in Java means that a class member (variable, method, or class) is accessible only within the same pakage. It is the default access level when no access modifier (like Public, Protected, Private) is specified.

Visibility:-

- Accessible within the same pakage (including all classes in the pakage).

- Not accessible from classes in other package, even if they are Subclasses

This access level is useful for grouping related classes and restricting access to pakage level functionality.