ASSIGNMENT-3

Que-1 Explain the components of JDK.

Ans ⇒

(1) Java Compiler —:

The Java Compiler is the tool that converts Java source code into bytecode, which can then be executed by JVM. The bytecode is platform-independent, meaning it can run on any system that has a JVM installed.

Command : javac.

(2) Java Runtime Environment —:

JRE is a part of JDK that provides the necessary libraries, JVM, and other components to run java applications. The JRE does not include development tools such as compiler or debuggers.

It includes a private JRE; which is used for executing the compiled Java code during the development process.

(3) Java Virtual Machine —:

The JVM is an integral part of the JRE and is responsible for running Java bytecode on any platform.

(4.) Java API —:

The Java API is a large collection of ready made software components that provides many useful functionalities. It includes core libraries that are essential for development of Java applications.

(5) Java Debugger —:

It is a tool that allows developers to inspect the execution of their programs. It can be used to set breakpoints, inspect variables, & control the execution flow of the Java Program.

- Command : jdb.

(6) Java Archiver (jar)

(7) Java Document Generator (javadoc)

Que-2 Difference bet" JDK, JVM and JRE.

Ans —

JVM	JDK	JRE
1. Runtime environment that executes Java byte code	2. Software dev. kit used to develop java applications.	1. A subset of JDK provides the environment to run Java applications.

2)	Runs the compiled Java bytecode on any platform.	Previous tools to develop, and debug Java applications.	Provides libraries and the JVM to run Java applications.
3)	Doesn't contains any development tools.	Contains JRE, development tools like compiler & debugger.	Contains JVM and Java class libraries.
4)	Used by anyone running Java Applications.	Used by developers to write, compile and debug Java programs.	Used by end-users who wants to run Java applications.
5)	Typically included with JRE and JDK installations.	Needs to be installed by developer who want to create Java applications.	Installed on systems where Java applications need to run, but not developed.

Que-3 What is the role of JVM in Java? How does JVM execute Java code?

Ans =

1) Platform Independence — Write once run Anywhere
2) Bytecode Execution.
3) Memory Management
4) Security.
5) Exception Handling
6) Thread Management.

How JVM Executes Java Code —

The execution of Java code by the JVM can be broken down into several key steps:

Compilation to Bytecode. → Class Loading → Bytecode Verification → Execution (Interpretation, Just-In-Time Compilation) → Garbage Collection → Execution of Native Methods → Termination.

Que-4 Explain the memory management system of the JVM.

Ans = The memory management system of the Java Virtual Machine (JVM) is designed to optimize the allocation use and release of memory in Java applications. It is a critical component that ensures efficient execution and prevent memory leaks.

(1) Heap Memory —: The heap is where all object in Java are stored. It is the runtime data area from which memory for all class instances and arrays is allocated.

(2) Stack Memory —: Each thread in Java application has its own stack, which is used for storing method calls, local variables and intermediate computation.

(3) Method Area —: The method area stores class-level data such as class structures, method data, runtime constant pool data

(4.) Program Counter (PC): PC registers hold the address of the JVM instruction currently being executed. Each thread has its own PC register.

(5) Native Method stack:- It is used when native methods are invoked. It holds the state of the native method calls.

(6) Garbage Collection:-: Process by which JVM automatically reclaims memory by removing objects that are no longer in use.

(7) Memory leaks in JVM :-

(8) Tuning JVM Memory.

Ques-5 What are the JIT compiler and its role in the JVM? What is the bytecode and why is it important for Java?

Ans :-

JUST-IN-TIME (JustIT) is a crucial component of the Java Virtual Machine (JVM) that enhances the performance of Java applications. It is called "Just in Time" because it compiles Java bytecode into native machine code at the runtime, just before execution. This process occurs dynamically while the application is running.

Role :-

Performance Optimization.

Hotspot Compilation.

3) Adaptive Optimization
4) Inline Methods.
5) Garbage collection Interaction.

Bytecode is an intermediate code generated by the Java compiler after compiling Java source code. It is low-level, platform-independent presentation of the program that the JVM can execute.

IMPORTANCE —

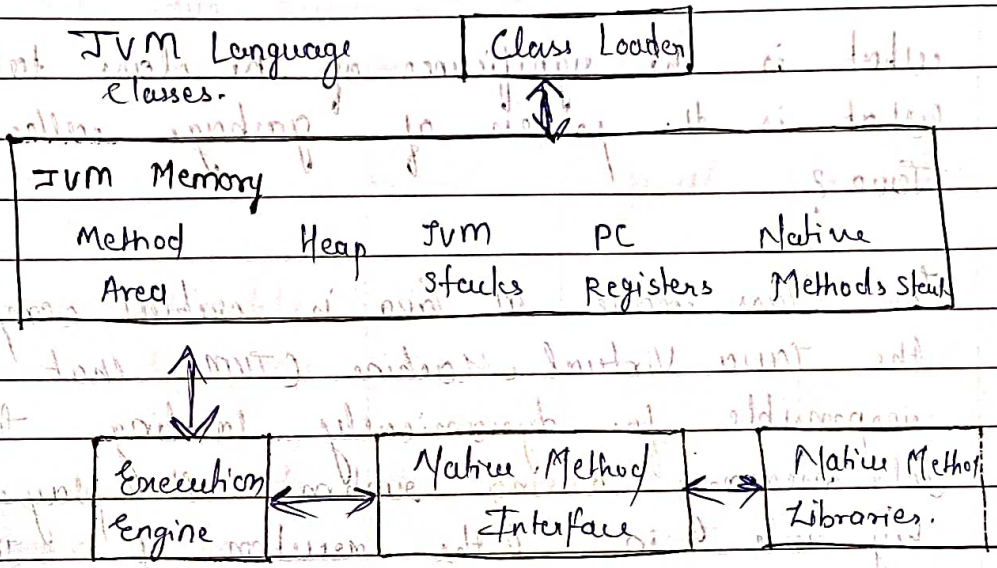
* Platform Independence.
* Portability
* Security
* Efficiency.
* Abstraction.
* Dynamic Linking.

Que-6 Describe the architecture of JVM.

Ans = JVM runs Java applications as a run time engine. JVM is the one that calls the main method present in a Java code. JVM is a part of JRE.

The class loader reads the ".class" file generate the corresponding binary data and save it in the method area. For each ".class" file, JVM stores the following in the method area.

- The fully qualified name of the loaded class and its intermediate parent class.
- Whether the ".class" file is related to Class or interface or Enum.
- Modifier, Variables and Method Information, etc.



Runtime Data Areas
Execution Engine
Native Method Interface
Java Memory Model (JMM)

Que-7 How does Java achieve platform independence through the JVM.?

Ans - Java achieves platform independence through the Java Virtual Machine (JVM), a crucial component of the Java ecosystem that allows Java programs to run on any device or operating system that has a compatible JVM. Here's how Java's platform independence is achieved.

1) Compilation into Bytecode.
2) Java Virtual Machine (JVM).
3) Platform Independence Mechanism.
4) Java API (Application Programming Interface).
5) Classloader Subsystem.
6) Java Runtime Environment (JRE).

Que-8 What is the significance of the class loader in Java? What is the process of garbage collection in Java?

Ans =

The Class loader in Java is critical component of the Java Virtual Machine (JVM) that is responsible for dynamically loading Java classes into memory during runtime. It plays a key role in the execution of Java programs by handling the process of locating loading and linking classes and interfaces.

1. Dynamic Class Loading.
2. Namespacing and Class Isolation.
3. Delegation model.
4. Security.
5. Memory Management & Performance.

Process of garbage collection in Java :-

Garbage collection in Java is the process by which the JVM automatically identifies and reclaims memory that is no longer in use

by the program, freeing it for future allocations. This automatic memory management prevents memory leaks and helps ensure that the applications run efficiently without exhausting available memory.

Que-9 What are four access modifiers in Java, and how do they differ from each other.

Ans —

Usage	Private	Default	Protected	Public
– Members are accessible only within the same class.	Yes	No	No	No
– Members are accessible only within the same package	No	Yes	No	No
– Members are accessible within the same package and in sub classes, even if they are in different packages.	No	No	Yes	No
– Members are accessible from any other class, in any package.	No	No	No	Yes

Que-11 Can you override a method with a different access modifiers in subclass? For example — can a protected method in a superclass be overridden with a private method in a subclass? Explain.

Ans -

When overridding a method, the access modifier in the subclass must be the same or less restrictive than the access modifier of the method in the superclass. This rule ensures that the override method can be accessed at least as broadly as it was in the super class.

```
class SuperClass {
    protected void show() {
        System.out.println("SuperClass show method");
    }
}
class SubClass extends SuperClass {
    public void show() {
        System.out.println("SubClass show method");
    }
}
```

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

Ans =

The primary difference between protected and default (also known as package-private) access in Java lies in their scope of accessibility.
Here's a comparison.

Feature	Protected Access	default (Package-Private) Access
Keyword	protected	No keyword
Within the Same class	Yes	Yes.
With the Same Package	Yes	Yes.
In Subclass (Same Package)	Yes	Yes
In subclass (Different Package)	Yes (through Inheritence)	No.
Outside the Package (Non-Subclass)	No	No

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

Ans - In Java you cannot make a top-level class (a class that is not nested within another class) private. However you can declare a nested or inner class as private.

The private access modifier is restricts access to the class or member within the same class, so a private top-level would be inaccessible to any other class, including those within the

same package, which defeats the purpose of having a top-level class.

Limitations —:

→ Scope Restriction.
→ Encapsulation.

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

Ans = No, a top level class of Java can not be declared as protected or private.
In Java, top level classes can only be declared with the following access modifiers:

* public : The class is accessible from any other class regardless of the package.
* Default : The class is accessible only within the same package.

They can not be protected or private because these access levels are not meaningful for top level classes. A protected or private top-level class would contradict the principle of class visibility and accessibility within Java.

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

Ans = If you declare a variable or method as private in a class, it is accessible only within the class in which it is defined. Attempting to access a private variable or method from another class, even if that class is within the same package, will result in a compile time error.

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

Ans = In Java "package-private" or default access level applied to class member (fields, methods, and inner classes) when no explicit access modifier (public, protected or private) is specified. This access level is also known as default access and it is the most restrictive form of access control than private.

```
class Example {
    int packagePrivateVar;

    void packagePrivateMethod() {
        // Implementation
    }
}
```