Assignment

Q-1] Explain the componets of JDK?

→ The Java development kit is a comprehsi
Package of tools and libraries neces
For developing java applicatrions.
It consist of several key componets

1. Java compiler (javac) -
This is tthe primary tool used to compile java source code. (.java files) into bytecode (.class files) which can be executed by the JVM.

2. Java Runtime Envirnment (JRE):
The JRE includes the java virtu-al machine (JVM) libraries and other componets necessary for running Java applications. It allows users to execute Java programs without needing to install the full development kit.

3. Java virtual machine :
The JVM is the runtime engine that execute Java bytecode. It provides platform independence by translating bytecode instructions into machine specific instruction into machine specific instructions at runtime.

4. Java Development Tools -

1) javac : Java compiler
2) java : Java interpreter (JVM)
3) jar : Java achrive tool for packag -ing Java applications.
4) ~~java~~ jdb : Java debugger for debug -ging java applications
5) jconsole : Monitoring Tool for Java applications

5. Java API (Application programing Interface)

The JDK includes a vast collection of pre-written classes and interface known as the Java API.
it is a connection between computer and programs.

6. Development Libraries :-

The jdk ~~includes~~ ~~extension~~ contains development libraries which are collections of reusable code modules that developers can use to build java application
2) These are libraries cover a wide range of functionalities including data structure, algorithms networking & graphical User interface

Q-2] Difference between JDK, JVM, JRE

1. JDK (Java development kit)

1) JDK is a development kit used for developing Java applications.

2) It includes tools such as the Java compiler (javac), java interpreter (java) and other development tools like jar, javadoc, and jdb

3) JDK also includes the Java API which is a collection of classes & interfaces for building java applic-ations.

4) Developers use the JDK to write, compile, debug, and run java program

5) It is required for developing java application.

2) JVM (Java virtual machine)

1) JVM is a virtual machine that executes java bytecode.

2) It provides platform independence by translating bytecode instructions

into machine specific instructins at runtime.

3] JVM responsible for memory managem -ent, garbage collection and excepti -on handling.

4] It runs Java applications on vario -us platforms without needing to recompile code.

5] JVM is part of both JDK & JRE

3] JRE (Java Runtime Envirnment)

1] JRE is a runtime envirnment used for executing Java applications

2] It include the JVM, libraries and other componets necessary for runni- Java applications.

3] JRE doesnot contain development tools like the Java compiler or debug -er.

4] User need JRE to run java applicat on their machines.

5] It is smaller in size compared to JDK because it doen't include development tool.

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

→ includes:

1] Platform Independence:-
The JVM provides platform independ-ce by executing Java bytecode, whic-is generated by compiling Java source code.

2] Memory Management:-
The JVM manages memory allocation and deallocations including the allocation of objects on the heap and cleanup of unused object through garbage collection.

3] Exception Handling:-
The JVM handles exceptions that occur during program execution ensuring graceful error handling and preventing abnormal termi-nation of programs.

4] Just-In-Time (JIT) compilation-
The JVM includes a Just-In-Time compiler that dynamically translate bytecode into native machine code at runtime.

5] Security

6] Thread management-

The JVM supports multithreding allowing java applications to execute multiple threads concurrently.

• The JVM execute Java code through the following steps:

1] Loading :-
The classloader loads compiled Java bytecode (.class file) into memory from the file system or network.

2] verification -
The bytecode verifier checks the loaded bytecode to ensure it adhers to the rules of Java langage and not violate security constrains

3) Execution :- The interpreter or Just-In-Time (JIT) compiler execute the bytecode instructions.
Initially the bytecode is interprete-d but frequently executed code paths are compile into native machine code by the JIT compile for better performance.

4] Memory management -
The JVM manages memory allocation and deallocation including the garbage collection to ensure efficien memory usage & prevent memory leaks.

5] Runtime Envirment-

It provides a runtime environment for java programs., including libraries and other runtime support necessary for program execution.

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

→ The memory management system of the JVM is responsible for managing memory allocation and de-allocation to ensure efficient memory usage and prevent memory leaks.

1. Heap memory

The JVM divides memory into several regions with the most significant portion for the heap is where created by your Java application are stored.

2. Garbage collection-

The JVM garbage collector periodically scans the heap to identify objects that are no longer in use or reachable by the application.

3] Generational Garbage collection-

Most modern JVM implementations use generational garbage collection dividing the heap into multiple generations.

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

→

The JIT (Just-In-Time) compiler is a critical componets of the java virtual machine (JVM) responsible for improving the performance of Java application

1) JIT compiler-

1) The JIT compiler is a part of the JVM that optimizes Java bytecode into native machine code at runtime

2] It analyzes the bytecode of frequently executed methods and translate them into equivalent native machine code specific to the underlying hardware architecture

3] By compiling bytecode into native code The ~~JV~~ JIT compiler can significantly improve the performance of the Java applications, as native code can execute faster than interpreted bytecode

4] The JIT compiler employes various optimization techniques such as method inlining, loop unrolling and dead code elimination, to generate efficient native code

2] Bytecode-

1) Bytecode is the intermediate representation of Java source code compiled by the Java compiler

2) Java source code is compiled into bytecode which is a platform independent format that can be executed by any JVM.

3] The use of bytecode allow Java programs to be "write onces run anywhere" meaning that compiled bytecode can execute any platform that supports the Java platform without the need for recompilation

4] Bytecode serves as a crucial abstraction layer between Java source code and the underlying hardware providing portability and security features inherent to the Java platform.

Q-6] Describe the architecture of the JVM.

→ • components of Java virtual Machine

- Class loader subsystem-
  - Bootstrap class loader
  - Extension class loader
  - System class loader
  - custom class Loader

- Runtime data areas
  - Method Area
  - Heap
  - Java stack
  - PC Register
  - Native Method stacks

- Execute engine
  - Interpreter
  - Just-In-Time (JIT) compiler
  - Garbage collector

1) class Loader- The class Loader sub-system loads Java classes into the JVM from various sources, such as file network streams or dynamically generated code. Its responsible for loading and loading class file into memory.

2° Runtime Data Areas :-

Method area -

The method Area stores class m
-data, including method bytecode,
field information, and static variab
each loaded class has its own
entry in the method Area.

Heap :- The Heap is the runtime d
area where objects are allocated.
It s shared among all threads an
i s divided into generations, suc
as the young generations and th
old Generation.

Native method stack -

similar to the java stack the
native method stack i s used for
native method invocation it store
native method invocation & local
variable

* Execution Engine -

Interpreter -

The Interpreter reads and execu
bytecode instructions one by one.
its simple and Portable way to
execute java code but can be
slower

* Just-In-Time (JIT) compiler-
The JIT compiler dynamically compil-es dynamically compiles frequenty executed bytecode into native machine code at runtime, optimizing performance by genera-ting efficient code tailored to the underlying hardware

Q-7] How does Java achieve platform indep-endente through the JVM?

→

1) Java achieves platform independence through the Java virtual machine (JVM) The JVM is an abstract computing machine that provides a runtime envirn-ment for Java bytecode to be excuted.

2) Java code is compiled into byte-code, which can run on any device with a JVM installed, regardless of the underlying hardware or operating system.

This allows Java program to be developed and deployed across differ-nt platforms without modificatio

Q-8] What is the significant of the class loader in Java? what is the process of garbage collection in Java.

1) The class loader in Java plays a crucial role in the dynamic loading of class into the Java Virtual machine, at runtime.
2) It is responsible for locating Java class files from various sources, such as the file system, network, or other repositories into memory as needed by the JVM.
3) Class loaders follows a hierarchical delegation model, where each class loader delegates the request to load a class to its parent before attempting to load the class itself.

Garbage collection -

Garbage collection in Java is the automatic process of reclaiming memory occupied by object that are no longer in use by the program.
Java's garbage collector periodically identifiers and frees memory occupied by unreachable objects, thereby preventing memory leak and ensuring efficient memory usage.