8. Explain the components of JDK?

⇒ 1. Java Compiler (Javac) → It translate Java source code into bytecode, which is executable on JVM.

2. Java Virtual Machine (JVM) → It is a runtime engine that executes Java bytecode. It provides platform independence by interpreting bytecode & managing memory, threads, & other resources.

3. Java Runtime Environment (JRE) → It includes JVM & libraries necessary for running Java applications but doesn't include development tools.

4. Java Application Programming interfaces → (JAPI). Collection of libraries & classes that provide pre built functionalities for various tasks.

5. Development Tools → tools used for developing, debugging Java applications. Includes Javac, debugger, Java doc.

6. Additional Tools and Utilities.

Q2

JDK (Java Development Kit)	JVM (Java Virtual Machine)	JRE (Java runtime environment)
* It is software development kit used for develop. Java Applications	* VM that executes Java bytecode.	* Includes JVM and libraries to. Runing Java applications.
* Tools like Javac, debugger and Java doc.	* JVM does not include software development tool.	* It has class libraries & supporting files.
* It is platform dependent. i.e for different platforms different JDK required.	* It is platform independent.	* It is also platform dependent.
* primarily used for code execution and has prime functionality of development.	* It is Majorly Responsible for creating environment for code execution.	* It specifies all the implementation & Responsible to provide these to JRE

Q3 Role of JVM in Java? How JVM executes Java code?

⇒ The Roles of of JVM are →

1. Byte code execution →
2. Memory Management.
3. Security.
4. Optimization.
5. Exception Handling.

→ It serves as an abstract computing machine that enables Java programs to Run on any platform without modification.

→ JVM executes Java code in -:

1. It loads the Bytecode generated by Java compiler into memory.
2. Verification → Then it verifies the loaded bytecode to ensure that it follows the Rules of Java.
3. Execution → JVM interprets the bytecode. During execution, It manages memory allocation, GC, & thread execution.
4. JIT (Just in time) → It involves dynamically translating frequently executed byte code into native machine code.
5. Optimization.
6. Gabbage Collection.

Q4 Memory Management System of JVM?

→

memory management in Java includes →

1. Heap memory → Heap is primary memory area used by JVM to allocate memory for objects & arrays dynamically.

2. Garbage Collection → JVM performs automatic garbage collection to reclaim memory occupied by objects that are no longer reachable.

3. Garbage Collection Algorithms → JVM employs various algorithms to optimize memory such as Serial, Parallel, CMS, G1.

4. Young Generation → It is one kind of Heap memory in which newly created objects are allocated in the eden space. When eden space fills up, a minor garbage collection occurs, during which unreferenced objects are removed and surviving objects moves to survivor space. which are then moved to old generation.

5. Meta Space → It is native memory space that dynamically grows to accomodate class metadata & is Garbage Collected.

Q5. JIT compiler & its Role in JVM? Also bytecode & why it is important?

→ JIT (Just in Time) is responsible for optimizing and translating Java bytecode into native machine code that can be executed by underlying hardware. It does this on-the-fly during Runtime identifying hot spots of code & compiling them for improved performance.

→ Bytecode is intermediate representation of Java code, generated by Java compiler. It is set of instructions understood by JVM. It helps or allows Java programs to be platform independent such it can run on any device or OS. This is key advantage of Java i.e "Write once, Run anywhere".

Q6. Describe the architecture of JVM?

⇒ JVM is crucial component of Java Runtime environment (JRE) Responsible for executing Java bytecode.

1. Class loader → Loads Java class files into the JVM. Responsible for finding & loading classes in bytecode.
2. Bytecode verifier → Verifies the bytecode to ensure it adheres to Java's safety & security.
3. Runtime Data Area.
4. Execution engine → Interpreter, JIT, GC.
5. Native interface → Allow Java code to interact with native code in C or C++.

→ overall, JVM provides platform independent execution.

Q7 How Java achieve platform independence through JVM?

→ When you compile Java code, it's compiled into bytecode, which is a platform independent intermediate representation. The JVM then interprets this bytecode & executes it on the host OS, providing a layer of abstraction from the underlying hardware & OS. This allows Java programs to run on any device or platform that has compatible JVM.

Q. What is significance of class loader in Java? process of GC in Java?

⇒ In Java, class loader is responsible for loading Java classes into JVM dynamically at Runtime. It locates & Reads class files, then creates a class object representing the class.

Garbage collector (GC) is process of automatically reclaiming memory occupied by objects that are no longer referenced by program. It identifies & removes unreferenced objects, freeing up memory for new objects.