1) Components of JDK:

-JRE: Java Runtime Environment
-java: It’s simply a loader that works for all the java applications. Also, this particular tool interprets the class file generated by javac. The single java launcher is used for the purpose of development and deployment. Gone are the days when JRE was used as the -deployment launcher. Nowadays, JRE doesn’t exist with the Sun JDK and totally replaced by this new java loader.
-javac: It’s a compiler. And, basically, it converts source code into Java bytecode
-appletviewer: Through this component, we can run the Java applets without taking the help of a web browser and even debug them.
-apt: This is used as an annotation-processing tool
-extcheck: For identifying the conflicts, this next check is used
-idlj: An IDL-to-Java compiler used to generate Java bindings from the provided Java IDL file
-jabswitch: It is a Java Access Bridge
-javadoc: It is a documentation generator, which produces documentation from source code comments, automatically
-jar: It is an archiver. Along with related class libraries packages into one JAR file. Also, it manages those files
-javafxpackager: It is a tool to attain packages and sign JavaFX applications
-jarsigner: It’s a verification and jar signing tool
-javah: It’s a stub generator and ‘C’ header that is used for writing built-in methods
-javap: It’s a class file disassembler
-javaws: It’s a Java Web Start launcher for JNLP applications
-JConsole: It’s a Console used for Java Monitoring and Management
-jdb: It’s a debugger
-jhat: It’s a Java Heap Analysis Tool (experimental)
-jinfo: It particularly used to get the configuration information from a running Java process
-jmap: It’s an Oracle jmap which is also a Memory Map. This gives the result of the memory map for Java. On the other hand, it is   useful in printing heap memory, or shared object memory maps details of a particular core dump or process.
-jmc: It’s known as a Java Mission Control
-jps: it’s a JVM Process, Status Tool. It is capable of listing the instrumented HotSpot JVMs on the target system
-jrunscript: It’s a Java command-line script shell
-jstack: A tool prints Java stack traces of Java threads
-jstat: Java Virtual Machine statistic monitoring tool
-keytool: A tool for manipulating the Keystore
-pack200: JAR compression tool
-Policytool: Utility that determines the Java runtime. That means it is a policy creation and management tool.
-VisualVM: It’s a visual tool. It is integrated with numerous command-line JDK tools.
-wsimport: It produces portable JAX-WS artifacts with an aim to invoke a web service
-xjc: It’s the part of the Java API boosting the XML Binding (JAXB) API. After accepting the XML schema, it generates Java classes

2) JDK (Java Development Kit):

The JDK is a full-featured software development kit used for developing Java applications.
It includes tools and utilities necessary for writing, compiling, debugging, and running Java programs.
Key components of the JDK include the Java compiler (javac), Java Virtual Machine (JVM), Java Runtime Environment (JRE), development tools (such as javadoc, jar, and jdb), and Java API libraries.
Developers typically use the JDK to create new Java applications from scratch.

JVM (Java Virtual Machine):

The JVM is an abstract computing machine that provides an execution environment for Java bytecode.
It translates compiled Java bytecode into machine-specific instructions that can be executed by the underlying hardware.
JVM is responsible for various tasks, including memory management, garbage collection, and bytecode interpretation or just-in-time (JIT) compilation.
JVM implementations are available for different platforms, allowing Java programs to be executed consistently across diverse environments.
JVM is a crucial part of both the JDK and JRE.

JRE (Java Runtime Environment):

The JRE is a subset of the JDK that is required for running Java applications.
It includes the JVM, along with necessary libraries and files needed to execute Java bytecode.
Unlike the JDK, the JRE does not contain development tools like compilers and debuggers.
End-users who want to run Java applications on their systems typically only need to install the JRE.
The JRE provides a runtime environment for executing Java applications but lacks the development capabilities of the JDK.

3) Role of Java:

-A Java virtual machine (JVM) is a virtual machine that enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java bytecode.

Execution of Java code:
----
(1) The execution process mainly includes.
(2) The Java source code is compiled into bytecode.
(3) Verify the bytecode and load the Java program through the class loader into JVM memory.

4) Memory management system in java:
   ----
a)Automatic Memory Allocation: JVM automatically allocates memory for objects when they are created.

b)Garbage Collection: JVM’s garbage collector reclaims memory from objects that are no longer in use, preventing memory leaks.

c)Memory Optimization Techniques: Techniques like object pooling, escape analysis, string interning, and class data sharing optimize memory usage and improve performance.

5) JIT Compiler:

   The JIT (Just-In-Time) compiler is a component of the JVM that dynamically compiles Java bytecode into native machine code at runtime. Its role is to improve the performance of Java applications by optimizing frequently executed code, adapting to runtime conditions, and caching compiled code for reuse.

Bytecode and its importance in java:

-Bytecode is a object oriented assembly language code which is designed for JVM.
-Bytecode is a highly optimized set of instructions that is executed by the Java Virtual Machine. Bytecode helps Java achieve both portability and security.
-It is platform independent, security, performance, Dynamic loading.

6) JVM Architecture:

Classloader: it’s a subsystem of JVM used to load class files. When we execute the code, it’s loaded by the classloader. There are built-in classloaders in Java.
Bootstrap ClassLoader: This is the first classloader, the superclass of the Extension classloader. It loads the rt.jar file, which contains all files of Java Standard Edition like Java.lang package classes, java.net package classes.
2. Extension ClassLoader: This is the child classloader of Bootstrap and parent classloader of System classloader.
3. System/Application Classloader: It loads the class files from classpath. It is also known as the Application classloader.

7) How does Java achieve platform independence through the JVM:

Java achieves platform independence through the Java Virtual Machine (JVM), which executes platform-independent bytecode. Java source code is compiled into bytecode, which can run on any system with a compatible JVM installed, abstracting away hardware and operating system differences. The JVM ensures consistent execution across diverse platforms, enabling Java programs to be truly portable.

8) significance of the class loader in Java:

The Class Loader in Java is essential for dynamically loading classes into the JVM during runtime.
It allows Java programs to load classes from various sources like the file system, network, or even memory.
Class loaders follow a hierarchical structure and enable features like dynamic class loading, facilitating Java’s flexibility and extensibility.
Process of Garbage Collection in Java:

Garbage collection in Java is the automatic process of reclaiming memory occupied by objects no longer in use.
The JVM’s garbage collector identifies and marks reachable objects, then sweeps through the heap, reclaiming memory from unreached objects.
Some implementations may involve compacting memory to optimize space.
Garbage collection is transparent to the programmer, managed by the JVM to automatically handle memory allocation and deallocation.