Q1.What is the difference between Compiler and Interpreter?
Ans :

Compiler	Interpreter
The compiler can see code upfront which helps in running the code faster because of performing Optimization.	The Interpreter works by line working of Code,
The compiler saves the Machine Language in form of Machine Code on disks.	The Interpreter does not save the Machine Language.
Compiled codes run faster than Interpreter.	Interpreted codes run slower than Compiler.
Linking-Loading Model is the basic working model of the Compiler.	The Interpretation Model is the basic working model of the Interpreter.

The compiler generates an output in the form of (.exe).	The interpreter does not generate any output.
Any change in the source program after the compilation requires recompiling the entire code.	Any change in the source program during the translation does not require retranslation of the entire code.
Errors are displayed in Compiler after Compiling together at the current time.	Errors are displayed in every single line.

Q2.What is the difference between JDK, JRE, and JVM?
Ans -

Type	JDK	JRE	JVM
Purpose	It is employed in the creation of Java applications.	It is employed to execute Java applications.	Java bytecode is executed using it.

Components	It comes with JRE as well as other Java development tools including a compiler, debugger, and others.	It consists of the JVM, Java class libraries, and other elements needed to run Java programmes.	The virtual machine is responsible for running Java bytecode.
Development	Due to the inclusion of development tools such the Java compiler and Java debugger, it is necessary for the creation of Java applications.	Since it simply contains components for executing Java applications, it is not necessary for the development of Java applications.	Since it is just in charge of running Java bytecode, it is not necessary for the creation of Java applications.
Execution	To execute Java bytecode, it is not necessary.	To execute Java bytecode, it is not necessary.	Java bytecode can be executed by JVM.
Size	JDK has the biggest size.	JRE is smaller than JDK in size.	JVM size is smallest than others.
Memory	More RAM is required by JDK.	JRE uses less RAM than JDK does.	JVM uses less memory than JDK does.

Q3.How many types of memory areas are allocated by JVM?

Heap − Runtime storage allocation for objects (reference types).
Stack − Storage for local variables and partial results. A stack contains frames and allocates one for each thread. Once a thread gets completed, this frame also gets destroyed. It also plays roles in method invocation and returns.
PC Registers − Program Counter Registers contains the address of an instruction that JVM is currently executing.
Execution Engine − It has a virtual processor, interpreter to interpret bytecode instructions one by one and a JIT, just in time compiler.
Native method stacks − It contains all the native methods used by the application.

Q4.What is JIT compiler?

Ans - The JIT compiler helps improve the performance of Java programs by compiling bytecodes into native machine code at run time. The JIT compiler is enabled by default.

Q5.What are the various access specifiers in Java?

Ans-1.private
2. Public
3.default
4.protected)

Q6.What is a compiler in Java?

Ans -

A Java compiler is a program that takes the text file work of a developer and compiles it into a platform-independent Java file. Java compilers include the Java Programming Language Compiler (javac)

Q7.Explain the types of variables in Java?

There are three types of variables in java is local, instance and static.

Q8.What are the Datatypes in Java?

Ans- **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.

Non-primitive data types: The non-primitive data types include Classes, Interfaces, and Arrays.

Q9.What are the identifiers in java?

Ans - Identifiers in Java are symbolic names used for identification. They can be a class name, variable name, method name, package name, constant name, and more. However, In Java, There are some reserved words that can not be used as an identifier.

Q10.Explain the architecture of JVM ?

Ans - 1) Classloader

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

1. **Bootstrap ClassLoader**: This is the first classloader which is the super class of Extension classloader. It loads the *rt.jar* file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes etc.
2. **Extension ClassLoader**: This is the child classloader of Bootstrap and parent classloader of System classloader. It loades the jar files located inside *$JAVA_HOME/jre/lib/ext* directory.
3. **System/Application ClassLoader**: This is the child classloader of Extension classloader. It loads the classfiles from classpath. By default, classpath is set to current directory. You can change the classpath using "-cp" or "-classpath" switch. It is also known as Application classloader.

2) Class(Method) Area

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap

It is the runtime data area in which objects are allocated.

4) Stack

Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register

PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack

It contains all the native methods used in the application.

7) Execution Engine

It contains:

1. **A virtual processor**
2. **Interpreter:** Read bytecode stream then execute the instructions.
3. **Just-In-Time(JIT) compiler:** It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here, the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

8) Java Native Interface

Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc. Java uses JNI framework to send output to the Console or interact with OS libraries.