Q1.What is the difference between Compiler and Interpreter?
→

Compiler	Interpreter
A compiler translates the entire source code in a single run.	An interpreter translates the entire source code line by line.
It consumes less time i.e., it is faster than an interpreter.	It consumes much more time than the compiler i.e., it is slower than the compiler.
It is more efficient.	It is less efficient.
CPU utilization is more.	CPU utilization is less as compared to the compiler.
Both syntactic and semantic errors can be checked simultaneously.	Only syntactic errors are checked.
The compiler is larger.	Interpreters are often smaller than compilers.
It is not flexible.	It is flexible.
The localization of errors is difficult.	The localization of error is easier than the compiler
A presence of an error can cause the whole program to be re-organized.	A presence of an error causes only a part of the program to be re-organized.
The compiler is used by the language such as C, C++.	An interpreter is used by languages such as Java.

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

JDK	JRE	JVM
JDK stands for Java Development Kit.	JRE stands for Java Runtime Environment.	JVM stands for Java Virtual Machine.
The Java Development Kit (JDK) is a software development kit that lets you build Java applications.	The Java Runtime Environment (JRE) is a software package that includes the Java Virtual Machine (JVM), class libraries, and other components required to run Java applications.	The Java Virtual Machine (JVM) is an abstract machine that provides an environment for Java ByteCode execution.
JDK includes tools for creating, monitoring, and debugging Java code.	JRE includes class libraries and other supporting files needed by JVM to run Java code.	JVM has no software development tools.
It is platform-dependent	It is also platform-dependent like JDK	It is platform-independent

The JDK mainly assists in code execution. Its primary purpose is development. JDK = Development Tools + JRE	JRE is primarily tasked with creating an environment for code execution. JRE = Class libraries + JVM	All implementations are specified by JVM. It is in charge of providing all of these implementations to the JRE. JVM = provides a runtime environment.

Q3.How many types of memory areas are allocated by JVM?
→ Types of memory areas allocated by JVM are:
1. Class(Method) Area
2. Heap
3.Stack
4.Program Counter Register
5.Native Method Stack

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

2) Heap : It is the runtime data area in which the memory is allocated to the objects

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

4) Program Counter Register : PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

5) Native Method Stack : It contains all the native methods used in the application.

Q4.What is JIT compiler?
→ JIT stands for Just-In-Time compiler is a component of the runtime environment that improves the performance of Java applications by compiling bytecodes to native machine code at run time.
By default, the JIT compiler is enabled in Java and is activated whenever a Java method is invoked. The JIT compiler then compiles the bytecode of the invoked method into native machine code, compiling it “just in time” to execute.

Q5.What are the various access specifiers in Java?
→ There are 4 types of access variables in Java:
1.Private
2.Public
3.Default
4.Protected

1) Private:

If a variable is declared as private, then it can be accessed within the class. This variable won’t be available outside the class. So, the outside members cannot access the private members.

Classes and interfaces cannot be private.

2) Public:
Methods/variables with public modifiers can be accessed by all the other classes in the project.

3) Protected:
If a variable is declared as protected, then it can be accessed within the same package classes and sub-class of any other packages.

Protected access modifier cannot be used for class and interfaces.

4) Default Access Modifier:
If a variable/method is defined without any access modifier keyword, then that will have a default modifier access.

Q6.What is a compiler in Java?
→ A compiler is a translator that converts the high-level language into the machine language.
Compiler is used to show errors to the programmer.
The main purpose of compiler is to change the code written in one language without changing the meaning of the program.

Q7.Explain the types of variables in Java?
→ There are three types of variables in Java:
1.local variable
2.instance variable
3.static variable

1) Local Variable: A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.A local variable cannot be defined with "static" keyword.

2) Instance Variable: A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static.It is called an instance variable because its value is instance-specific and is not shared among instances.

3) Static variable: A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

Q8.What are the Data types in Java?
→ Data Types in Java are defined as specifiers that allocate different sizes and types of values that can be stored in the variable or an identifier. Java has a rich set of data types.
Data types in Java can be divided into two parts :

Primitive Data Types :- which include integer, character, boolean, and float

Non-primitive Data Types :- which include classes, arrays and interfaces.

Q9.What are the identifiers in java?
→ Names used for classes, variables and methods are called identifiers.
In Java, there are several points to remember about identifiers.

They are as follows -

1.All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).

2.After the first character, identifiers can have any combination of characters.

3.A keyword cannot be used as an identifier.

4.Most importantly, identifiers are case sensitive.

5. Ex:age, $salary, _value, __1_value.// legal identifiers

6.Ex:123abc, -salary.// illegal identifiers

7. A class name should start from a capital case letter and long names should use camel casing.
For example: TaxationDepartment

8. Object name should start from lower case letter and long names should use camel casing.
For example: taxationDepartment

Q10.Explain the architecture of JVM ?
→ The architecture of JVM contains:

*** Class Loader**
*** Memory Area**
*** Execution Engine**

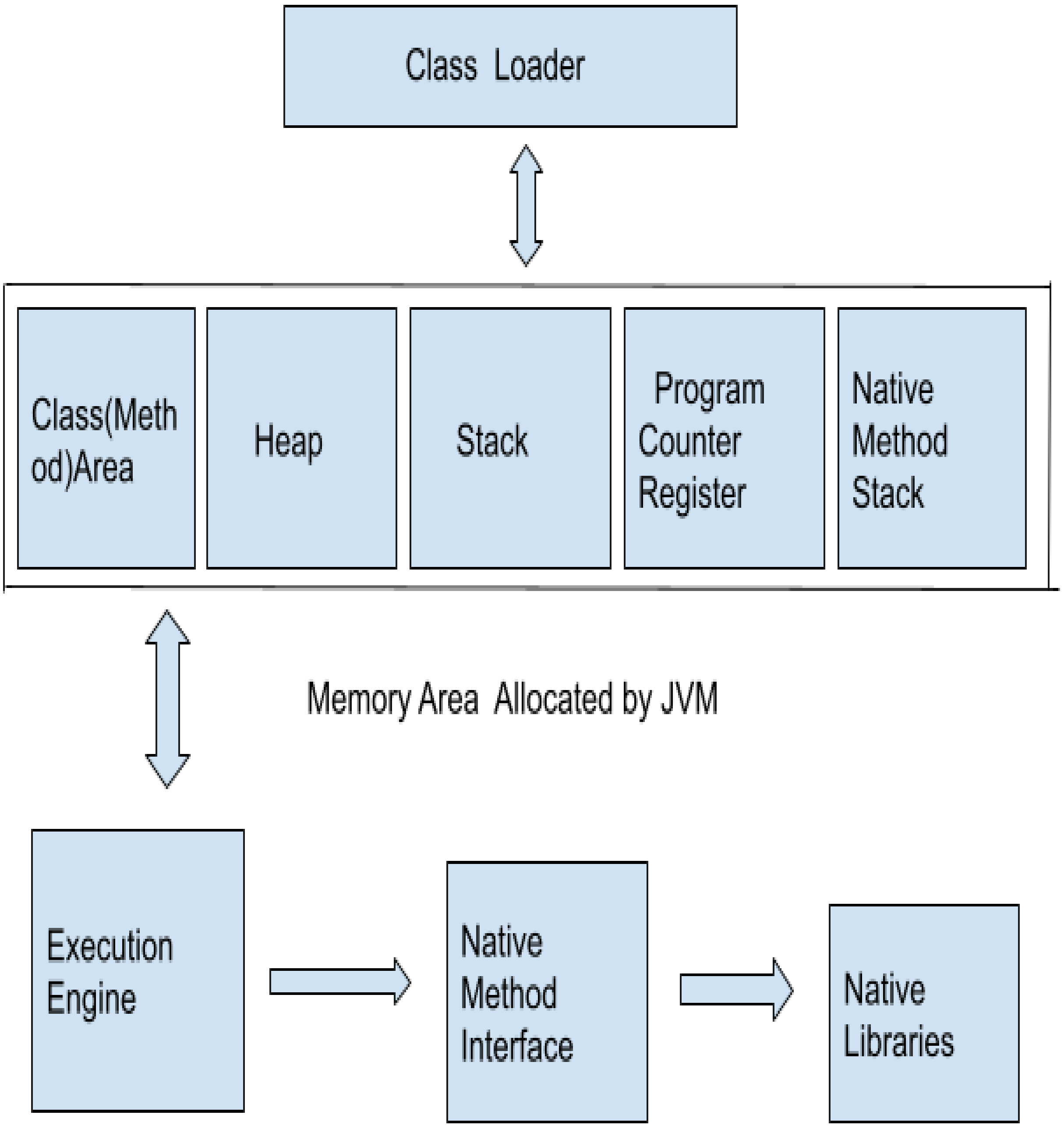
Classloader − Loads the class file into the JVM.

Class Area − Storage areas for a class elements structure like fields, method data, code of method etc.

Heap − Runtime storage allocation for objects.

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.



Native method stack − It contains all the native methods used by the application.

Execution Engine − It has a virtual processor, interpreter to interpret bytecode instructions one by one and a JIT, just in time compiler.

Java Native Interface (JNI) – It 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.