Q1. What is the difference between compiler and interpreter?

Compiler	Interpreter
* It is a Software which takes Source code (HLL) as the input and generates MLL (Machine Level Language) code as the output. To convert the High Level Language (HLL) code to MLL code compiler will scan the HLL code only once.	It is a Software which takes Source code (HLL) as the input and generates MLL code as the output. To convert the HLL code to MLL code interpreter will scan the HLL code multiple times (depends on the instructions).
* Compiler will speed up the process	Interpreter will slow down the process.
* Compiler in one scan will identify all the problems in the code (if found).	Interpreter will do scanning line by line so it takes more time for identifying the problem.

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
* JDK is a Software development kit that develops applications in java. Along with JRE, the JDK also consists of various development tools (Java Debugger,	JRE is an implementation of JVM. It is a type of Software packages that provides class libraries of java, JVM & various other components for running the java application.	JVM Specifies all of the implementations. It is responsible for providing all of these implementations to the JRE.

JDK	JRE	JVM
JavaDoc, compilers etc)		
* JDK = Development tools + JRE	JRE = Libraries for running the application + JVM	Jvm = only runtime environment that helps in executing the java byte code.

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

The memory in the JVM is devided into 5 different parts :

1. Class (Method) Area
2. Heap.
3. Stack
4. Program Counter Register
5. Native Method Stack

1. Class (Method) Area

The Class (method) area is the memory block that stores the class code, variable code (static variable, runtime constant), method code and the constructor of a java program. (Here method means the function which is – written inside the class). It stores class-level data of every class such as runtime constant pool, field and method data & the code for the methods.

2. Heap

The heap area is the memory block where objects are created or objects are stored. Heap memory allocates memory for classes, interfaces and arrays (an array is an object). It is used to allocate memory to objects at runtime.

3. Stack

Each thread has a private JVM stack, created at the same time as the thread. It is used to store data and partial results which will be needed while returning value for method and performing dynamic linking.

Java stack stores frames and a new frame is created each time at every invocation of the method. A frame is destroyed when its method invocation completes.

4. program Counter Register

Each JVM thread that carries out the task of a specific method has a program counter register associated with it.

The non-native method has a PC that stores the address of the available JVM instructions, whereas In native method, the value of the program counter is undefined.

5. Native Method Stacks

Native method stacks are not written in java Language.

This memory is allocated for each thread when it is created and it can be fixed or dynamic in nature.

Q4. What is JIT Compiler?

The Just Intime Compiler is an essential part of the JRE, It is responsible for performance optimization of java based applications during run time.

- At run time, the JVM loads the class files, the semantics of each are determined, and appropriate computations are performed. The additional processor and memory usage during interpretation make a java application perform slowly.
- The JIT Compiler aids in improving the performance of java programs by compiling byte code into native machine code at runtime.
- The JIT Compiler is enabled throughout, while it gets activated when a method is invoked. For a compiled method, the JVM directly calls the compiled code. instead of interpreting it.

Q 5. What are the various access specifiers in java?

Access modifiers defines the scope or accessibility of the classes, interfaces, variables etc. It defines how they can be accessed from the other part of the program. Various access modifiers used in java are;

1. public
2. private
3. protected
4. static
5. strictfp
6. synchronized
7. final
8. abstract
9. native
10. transient
11. volatile

Q6. What is a compiler in java ?

The java Compiler (Javac) is a command line tool that reads java Source code files and compiles them into executable java bytecode classes. The java source code must be contained in files whose file names end with .java extension.

The java compiler takes files with this extension and generates executable class file with .class extension. It is possible to define more than one class in a single source file and if it happens the java compiler will generate exactly one class file for each class defined in the Source file.

Q7. Explain the types of variables in java ?

Java variable is a name given to a memory location. It is the basic unit of Storage in a program.

Types of variables used in java are:

1. Local variables
2. Instance Variables
3. Static variables

1. Local Variables :

- These variables are created when the block is entered, or the function is Called and destroyed after exiting from the block or when the call returns from the function

- The scope of these variables exists only within the block in which the variables are declared. We can access these variables only within that block.

- Initialization of the local variable is mandatory before using it in the defined scope.

2. Instance Variables :

Instance variables are non-static variables and are declared in a class & outside of any method, constructor or block.

- As instance variables are declared in a class, these variables are created when an object of the class is created & destroyed when the object is destroyed.
- Unlike local variables, we may use access specifier for instance variable. If we do not specify any access specifier, then the default access specifier will be used.

- Initialization of an instance variable is not mandatory. Its default value dependent on the data type of variable.

- Instance variables can be accessed only by creating objects.

- We initialize instance variables using constructors while creating an object. We can also use instance block to initialize the instance variables.

3. Static variables

Static variables are also known as class variables.

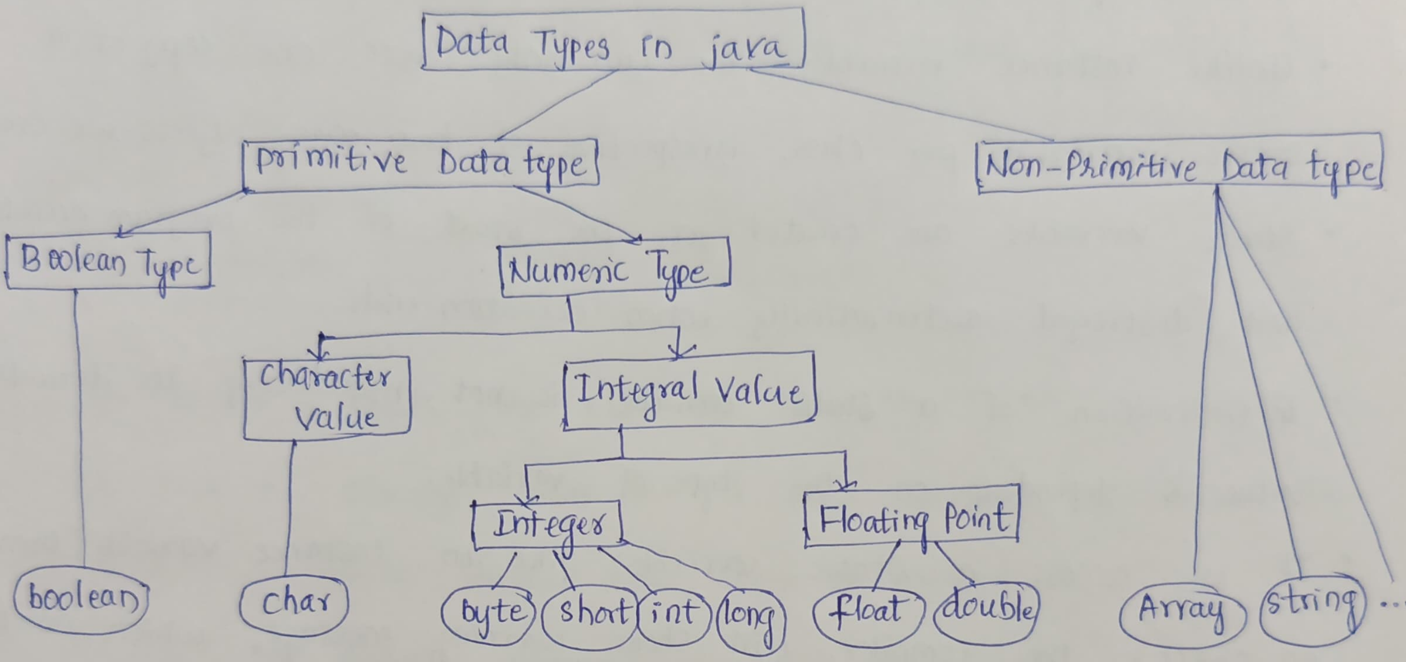
- These variables are declared similarly to instance variables. The difference is that static variables are declared using the static keyword.
- Unlike instance variables, we can only have one copy of a static variable per class, irrespective of how many objects we create.
- Static variables are created at the start of the program execution and destroyed automatically when execution ends.
- Initialization of a static variable is not mandatory. Its default value is dependent on the type of variable.
- If we access a static variable like an instance variable (through an object), the compiler will show warning message, which won't halt the program. The compiler will replace the object name with the class name automatically.
- If we access a static variable without the class name, the compiler will automatically append the class name. But for accessing the static variable of a different class, we must mention the class name, as 2 different classes might have a static variable with the same name.
- Static variables cannot be declared locally inside an instance method.
- Static blocks can be used to initialize static variables.

Q8. What are the data types in java?

Java has two categories in which data types are segregated

1. Primitive Data type : byte, short, char, int, $\overset{long,}{}$ float, double & boolean

2. Non-primitive Data type: String, Array etc.

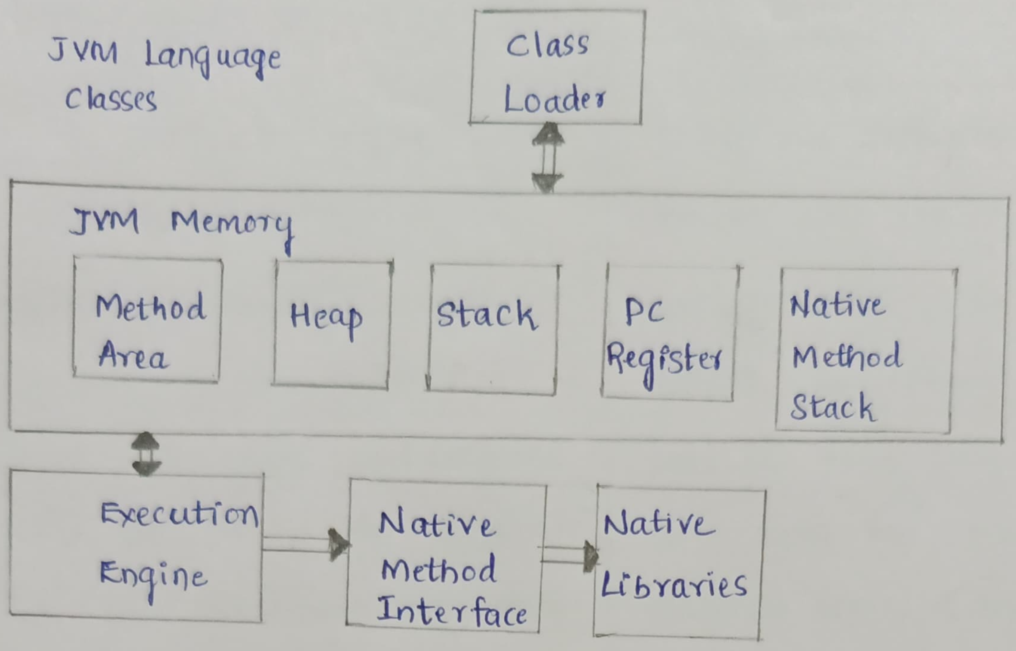


Q9. What are the identifiers in java?

All java variables must be identified with unique names. These unique names are called identifiers.

- Identifier can be class name, method name, variable name, label name.
- The only characters allowed for java identifiers are: a to z, A to Z, 0 to 9, $, _
- Identifiers should not start with digit.
- We can't use reserve words as an identifier.
- We can use inbuilt class names and variable names as identifiers, but it is not a good programming practice to use.

10. Explain the architecture of JVM.



- Class Loader - 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 - PC Registers contains the address of an instruction that JVM is currently executing.
- Execution Engine - It has a virtual Processor, interpreter, JIT Compiler
- Native method Stack - It contains all the native methods used by the application.