Interview Questions

1. Explain the components of the JDK.

JDK (Java Development Kit) is a software development kit (sdk) that includes the Java Runtime Environment (JRE), the Java compiler (javac) and other tools.

- Java Runtime Environment (JRE) – It runs Java program. It consist of Java virtual machine (JVM) which helps in executing bytecode and Java class library.
- Java compiler (javac) → converts source code into bytecode.
- Java archive (jar) – It is used to create and manage jar file which are archive of class files & other resource
- JavaDoc – It generates HTML documentation from source code.
- It also includes other tools such as appletviewer, IDL compiler e

2. Differentiate between JDK, JVM and JRE?

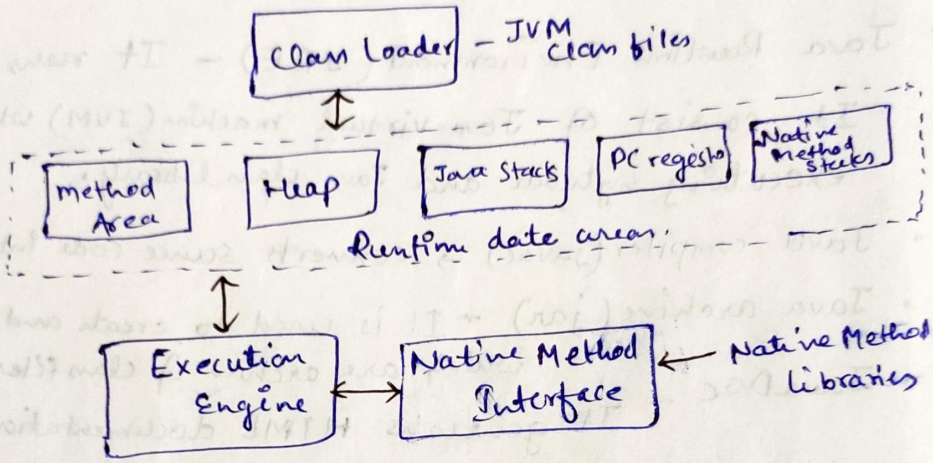
JDK – It is a software development kit. It consist of JVM and JRE. It is used for developing, compiling, testing and debugging.

JVM – It is core component of JRE. It executes the Java bytecode. It interprets and executes bytecode instructions, providing a platform-independent environment

JRE – It is used to run Java applications. Client needs not to install JDK, can install only JRE. It consist of JVM and class libraries.

3. What is the role of the JVM in Java? How does the JVM execute the byte code?

JVM acts as a bridge between Java code and the underlying hardware. It provides a runtime environment for Java applications to run on different platform and OS.



- JVM directly interprets the bytecode instructions one by one, translating them into machine readable instructions.

- JIT compilation — It analyzes frequently executed code sections and compiles them native machine code.

4. Explain the memory management system of the JVM.

JVM uses garbage collector to automatically allocate and deallocate memory. It is used to prevent memory leakage

- Heap Memory - Primary memory area where objects are allocated. It is divided into generation.
  - Young generation — It is for newly created objects.
  - Old generation - For objects that have survived multiple garbage collection cycles.
  - Permanent generation - It was used to store class metadata but removed in Java 8.
- Stack Memory — It is used to store method calls, local variables and return values. for every method invocation there will be a new stack frame.
- Native Method Stack :- It is used to store method which are implemented in C, C++ but not in Java.
- Garbage Collection — It is used to reclaim memory from objects that are no longer in use.

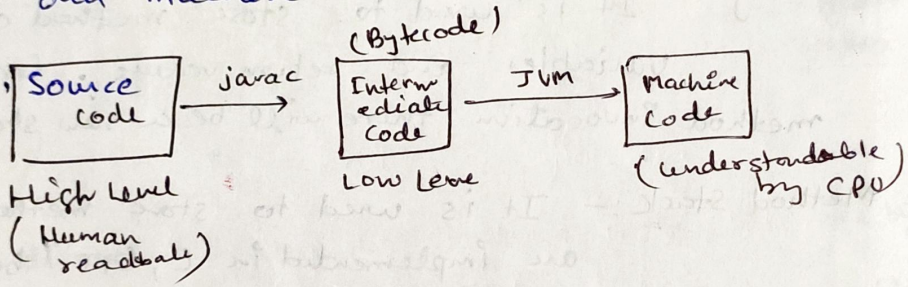
  - Algorithms -
  - Mark and sweep - In this GC marks reachable and sweeps unreachable.
  - Copying — In this old object will be copied into new space.
  - Compacting — Moving live objects ~~to a new space~~ closer to reduce fragmentation.
  - Generational — Optimizes collection based on object age.

5. What are the JIT compiler and its role in the JVM? What is the bytecode and why is it important for Java?

- Just In Time (JIT) compiler is the component of JVM. Its converts bytecode into machine code.

Role- JIT role is to covert bytecode → native m/c code for enhance understanding of CPU. It also optimizes codes based on runtime information, such as frequently called meth

Bytecode- It is basically ".c" file which we get after compiling the source code. It is a intermediat code (low level code) ~~which~~ we between the source code and machine code.



Importance of bytecode -

- Portability — It can run on diffent platfom by platfom specific JVM.

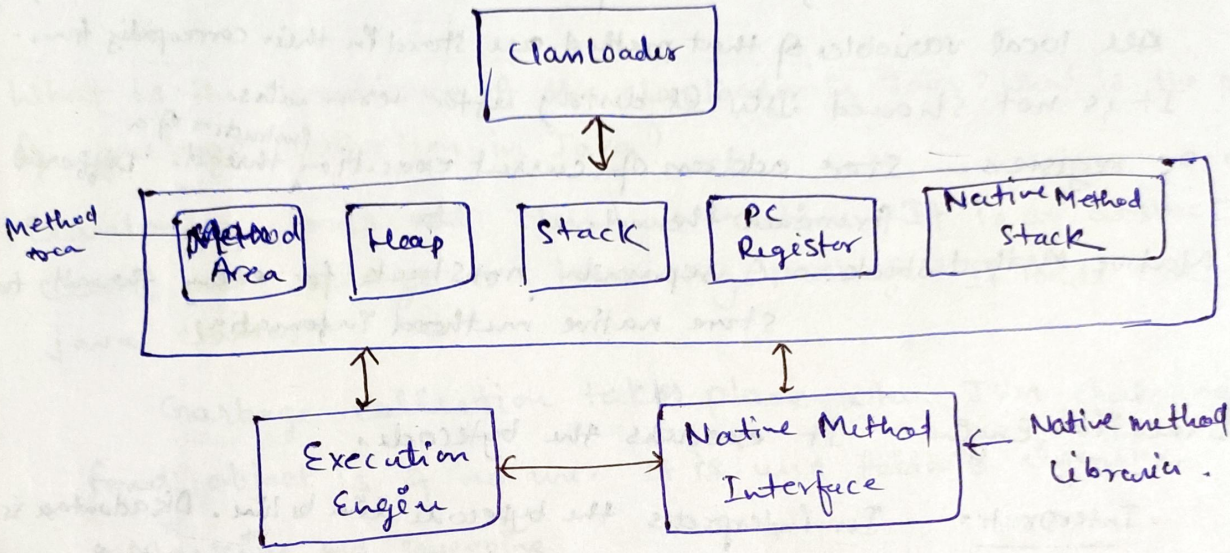
- Security — JVM acts as sandbox, prevents bytecode from directly interacting with the underlying OS & hardwa

- Performance — JIT compilers can optimize bytecode at runtime, improving performance.

6. Describe the architecture of the JVM.

JVM- JVM is a abstract machine. It provides runtime environment where bytecode can be executed. JVM working ~~will be~~ is specified but Implementation depends upon JVM provider.

Architecture of JVM — It contains classloader, memory area, execution engine, Native Method Interface etc



Classloader — It is subsystem of the JVM. It is used to load class files after running the Java program.

There are 3 classloader in Java.

- Bootstrap Classloader — It is first classloader of which is the superclass of Extention classloader.

  It loads rt.jar file which contains all class files of Java SE like java.lang package class, java.util, java.net, java.io, java.sql package class etc.

- Extension Classloader — This is child classloader of Bootstrap and parent classloader of System classloader.

  It loads the jar files located inside - jre/lib/ext.

- System/Application classloader — It is child classloader of extension CL. It loads the classfile from classPATH. By default, it is set to current directory. We can change it using -cp or -classpath.

Memory Area

- Method Area — class (method) area. In this area all class level information like classname, parent class name, methods, variable, static variable information are stored. There is only one Method area per JVM, and it is a shared resource.
- Heap Area — All object information is stored here. It is also only one per JVM.
- Stack Area — for every thread, JVM creates one run-time stack. Every block of stack is called Activation record/Stack frame. All local variables of that method are stored in their corresponding frame. It is not shared. It will destroy after terminates.
- PC registers — Store address of current execution instruction of a thread. Different for each thread.
- Native Method Stack — A separate nm stack for every thread to store native method information.

Execution Engine — It executes the bytecode.

- Interpreter — It interprets the bytecode line by line. Disadvantage is when one method is called multiple times, every time interpretation is required.
- JIT — It increase the efficiency of an interpreter. It compiles the bytecode and changes it to native code so whenever the interpreter sees repeated method calls, it provides direct native code for that.

Garbage Collector — It destroys un-referenced objects.

Java Native Interface — (JNI) — It is an interface that interacts with native method libraries and provides native libraries (C, C++) for the execution.

7. How does Java achieve platform independence through JVM?

In Java source code is compiled to bytecode which is platform independent means It can be compiled using any java compile running on different platform.

JVM takes that code and converts it to machine code. JVM can be of running in different platform. By this way Java achieve platform independe..

8. What is the significance of the classloader in Java? What is the proces of garbage collection in Java?

Classloader loads clas at runtime. It is an abstract clas and belongs to java.lang package. It loade core java clases.

Garbage collection takes place when JVM checks ad found object is of no use. It is used fallwis algorithm.

- Marking and sweeping
- compacting
- Copying
- Generational.

9. What are the four access modifiers in Java, and how do they differ from each other?

There are four access modifier in Java & these are used to control visibility of methods declare inside (Interface/Class/Enum)

- Public
- Private
- Protected
- Default (package-private)

	Same Package			Different Package	
Access Modifier	Same Class	Sub Class	Non Sub Class	Sub Class	Non Sub Class
Private	A	NA	NA	NA	NA
package level private	A	A	A	NA	NA
protected	A	A	A	A	NA
Public	A	A	A	A	A

A - Accesible
NA - Non Accesible

10. What is the difference b/w public, protected and default access modifiers?

public - These are accesible from anywhere from different packages also, # used for classes, methods and fields that need to be widely accesible.

private - These are accesible in same class only.

Protected - Accesible in the same package or in subclass of the class that declare the member. It is used to accesible in subclass of different and in same.

Default - Accesible only in same package. If no access modifier is specified. Default is used.

11. Can you overide a method with a diffent accen modifier in a subclass? for example can a protected method in a superclass be overridden with a privata method in a subclass?

No, we cannot overide a method with diffent accen modifier If a superclass is protected then subclass must be same.

12. What is differnce between protected and defaut(packag-privati) accen?

Protected – can accen in same packcage and subclass in which member deklarn.

default – accensible within package only.

13. Is it possible to make a clan private in Java? If yes, where can it be done, and what are the limitation?

It is not possible becaun by making it privati we can creati an object outside of its enclosing clan.

14. Can a top-level clan in Java be declared as privati? Why or why not?

No, because visibility will be reduced.

15. What happens if you declare a variable or method as privati in a clan and try to accen it from another clan within the same packcage?

Privati member or method defined in clan accensible only in same clan.

16. Explain the concept of "package-private" or default access. How does it affect the visibility of class members?

A method or field declared with a default access modifier, also known as package-private.

It will be visible in same package but also visible in different package subclass of any member function is created

It will be visible only in same package.