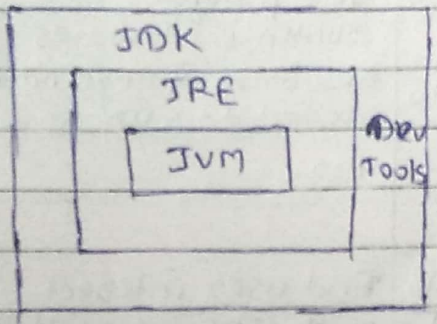
Interview Questions.

Q.1 Explain the components of JDK?

- JDK is cross platform software development environment that offers collection of tools and libraries necessary for developin java based app^{n} & applet.
- It is core package along with JVM & JRE

JDK = JRE + Development Tools.



Components:

1) JRE: A component of JDK that includes JVM, core classes & supporting files. It allow user launch Java app^{n}
2) Compiler:
   A tool included in JDK that is used for compiling Java program.
3) Archive (jar): A tool that is used to create JAR file JAR files can be used to package code & shared with other developers.
4) Javadoc - It is documentation generator.
5) Jconsole: A tool can be used to monitor & manages Java app^{n}. It can connect local or remote JVM.
6) JavaFx:
   This component of JDK used for building & deploying Android apps.

Q.2 Differentiate betⁿ JDK, JVM & JRE

JDK	JRE	JVM
1) JDK provides tools & resources to develop java appⁿ	JRE allows the execution of Java appⁿ on end-user	JVM acts virtual computer with physical computer to interpret & execute Java bytecode
2) JDK include compiler to convert human-readable code into machine code	JRE provides the runtime libraries resources & environment needed to run Java Prg.	JVM manages memory handles exception & ensures smooth running of Java prg.
3) Developer interact with JDK during the development phase to write, compile & debug code	End user interact with JRE when they run Java appⁿ on their system.	JVM interact with java bytecode interpreting & executing it to produce desired o/p

Q.3 What is the role of the JVM in java & How does the JVM execute Java code?

→ JVM is virtual machine that enables the execution of java bytecode. T

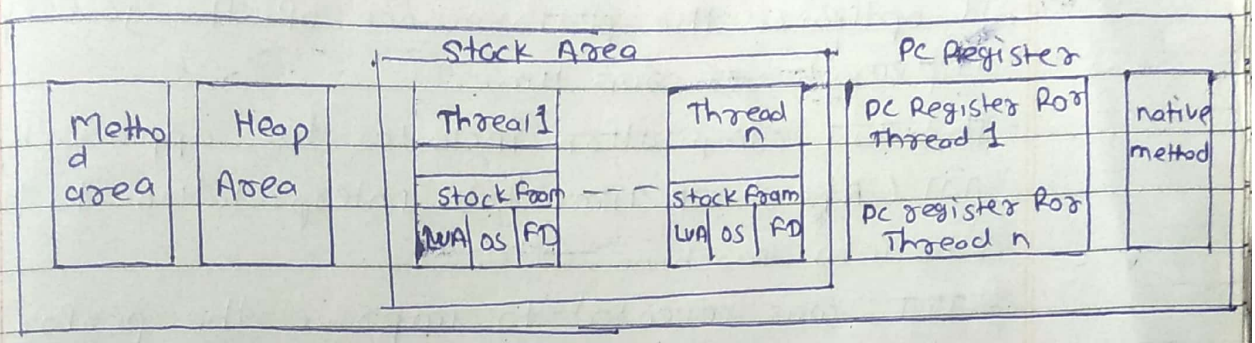
- The JVM acts as an interpreter between the java programming language & the underlying hardware.
- It provides a runtime environment for java applications to run on different platform and operating systems.

- Execution:

Java code firstly compiled by java compiler into bytecode which stored in .class files.

- JVM class loader load .class in memory, link them
- bytecode verifier checks valid or not.
- then JVM executes bytecode using interpreter or Just-in-Time (JIT) compiler.

Q.4 Explain memory management system of the JVM.



① Method Area:
- It is part heap memory shared among all the threads. It creates when JVM starts up.
- It is used to store class structure, name, interface name.

② Heap Area:
- Heap stores the actual object. user can control heap if needed. It can be fixed or dynamic size.
- when you use new keyword, the JVM creates instance for the object in heap

③ Stack Area:
- It can be of either fixed or dynamic size
- The stack memory is allocated per thread.
- It store data & partial results

④ PC Register:
PC Register stores the return address or native pointer.

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

→ JIT :

- JIT in java is integral part of JVM. It provide best performance environment.
- It optimize the performance of the java app^n at compile or run time.
- The JIT compilation includes two approaches AOT (Ahead-of Time) & Interpretation. to translate code into machine code.
- JIT was invented to improve the performance of JVM. ~~of te~~
- JIT compiles only the reasoble byte code to machine code (no ~~extra cod~~ complete code)

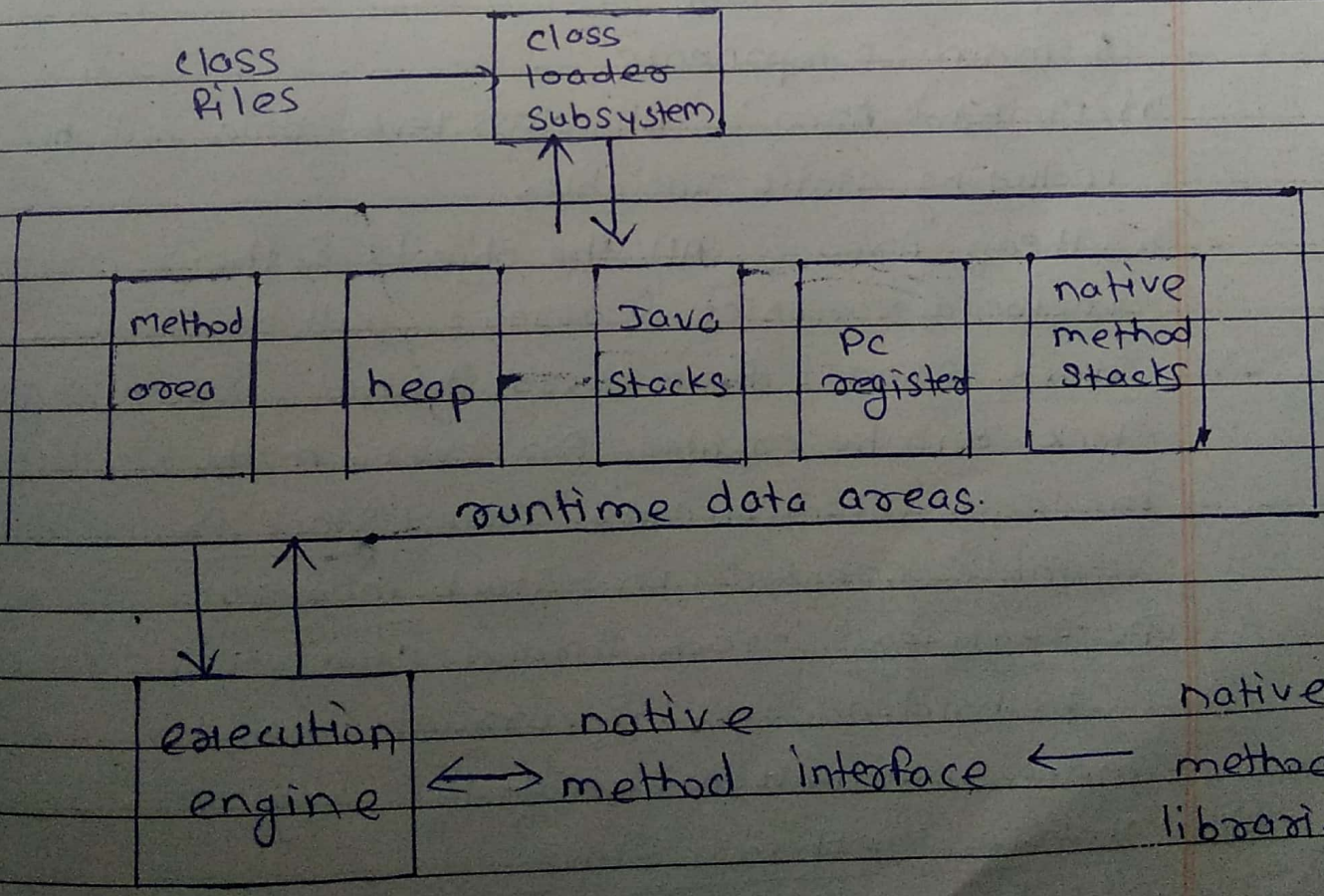
- Bytecode :

- Java bytecode is instruction set of JVM. It is similar to assembler.
- If we compiled java program. java bytecode is generated.
- bytecode is machine code in form of .class file

- Byte code in important for java. It helps Java to achieve both portability & security

JVM Architecture.

A virtual Machine is software implementation of a physical machine. Java was developed with concept of write once Run Anywhere. The compiler compiles the java file into .class file then that .class file is input into JVM which load & execute class file



1) classLoader Subsystem.
-Java's dynamic class loading Functionality is handled by the classloader subsystem. It loads links & initializes the class File
1) Loading.
classes will be loaded by this component. Bootstrap classloader, Extension classloader, Application classloader are three will help achieve it.
2) Linking.
• veriRy : it veriRy bytecode is proper or not.
• perpare : all static variable assign default value
• Resolve : Symbolic memory ref. replaced with original ref.
3) Initialization :
-Static variable assign original values.

2) Runtime Data area :
5 main components.
1) Method Area : All class level data will be Stored including static variables.
2) Heap Area : All the objects & their corresponding instance variables & arrays will be Stored.
3) Stack Area : For every thread, Separate runtime Stack will be created. For every method call, one entry made in Stack memory called Stack Frame. Local variable created in stack memory.
4) PC Registers : each thread separate PC Register to hold address of current executing instruction once instruction executed PC register updated.
5) Native Methods Stacks
-It holds native method information. like C, C++ method info

Q.7 How does Java achieve platform independence through the JVM?

→ - Java is platform independent means - Java compiled code can run on all operating system.
- If we write code that is in human readable it not understand machine. It need to language understood by machines.
- Here role of compiler it convert high-level language to machine understood language.
- It executed by CPU directly or it may intermediate representation, interpreted by virtual machine. that is Java byte code.
- So that byte code require the JVM so if we create java prg. & compiled it the byte code generate that byte code we can run any platform with the help of JVM.

Q.8 What is the significance of the class loader in Java. what is the process of garbage collection in Java?

→ class loader is abstract class in a java.lang. Package It loads the classes at run time, It helps to manage security by ensuring classes come from trusted locations.
- class loader load classes in different sources, like files or network.

Garbage collection :
- It is automatically manage memory it remove objects / variables in a program that are no longer used.
- Free up the memory space, so that prevent memory leaks.

We can overload main method in java

Q.9 What are the four access modifier in Java & how

* Java Modifiers :

There are two types of

① Access Modifiers ② non-access modifier.

① Access Modifier.

1) Private :

- In this methods or data members declared as private are accessible only within the same class/ class in which are declared. A
- Any other class or same package will not able to access these
- If we declared 2 two classes & in one class declared method as private and in 2nd class try to access this method in it give error.

Error : The method is not visible

2) Public :

- The public access modifier has widest scope among all other access modifier
- classes, methods or data members that are declared as public are accessible from everywhere in the program.

3) Protected :

The method or data member declared as protected accessible within same package or subclasses in different packages.

10. What is different between public, protected, & default, access modifier.

⇒ - public : The member of public class or public variable is access everywhere.

- protected :
  protected ac modifier accessible within same class, same package & subclass (different package.

- Default : Accessible only within the same class and same package

Access modifier	Same class	Same Package	Subclass diff. package	outside Package
public	Yes	Yes	Yes	Yes
protected	Yes	Yes	Yes	No
Default	Yes	Yes	No	No

Q.11 can you override a method with a different access modifiers in a subclass? for example can a protected method in a superclass be overridden with a private method in subclass?

→ - Yes, we can override by using different access modifiers.

- No, a protected method in a superclass cannot be overridden with a private method in a subclass, the access level of overriden method in the subclass must be same or less restrictive than the method in the superclass.

Q.12 What is the difference between protected and default (package level private) class.

→ • Default (package level private) :
- Default modifier accessible within same class and same package
- This is used when you don't specify a modifier

• Protected :
- It is only accessible within same class, same package & subclass (different package).

13. It is possible to make a class private in Java IF yes, where can it be done & what are the limitations ?

→ Yes, It is possible to declare class with private access specifier. But not in main class.
You can declare private in inner classes.
- IF we declare top-level class as private then it would be completely useless because nothing would have access it.

Q.14 can a top-level class in Java be declared as protected or private ? why or why not !

→ - No, we cannot declare a top-level class as private or protected.
- private classes only accessible through owner class & protected owner class & subclass.
- But in nested classes we can use this access modifier.

Q.15 What happens if you declare a variable or method as private in a class & try to access it from another class within same package.

- If you declare a variable or method private & access it from another class Then compilation error occur
- because private variable or method only accessible within same class

Q.16 Explain the concept of package-private or default access, how does it affect the visibility of class members?

- Package-level-private (default) means the member is accessible only within its own package
- If we do not specify any access modifier java automatically treats, class field or method as package private.
- visibility of class is accessible within the same class or within some package.