• Explain the component of the JDK

- JRE : provide libraries, Java virtual machine

- JVM : Execute Java bytecode, providing an environment for running Java application. it's platform independent.

- Development Tools: Javac (Java complier), Java (launcher for Java app.), Javadoc (documentation generator) Jdb (Java debugger) that help in developing, compling and dubbing Java application.

- Java Archive (Jar) Tool: package multiple

④ Explain the memory management system of the JVM.

→
1. Heap: - to store object and arrays.
young Gen: where new object are initally created
Eden space: where most new object are intially created.
- hold obj. that have survied garbage collection cycleing.

2. stack: store method call frame, including local variable method parament, and return ~~from~~ ~~stack~~.address.
- Each thread hay it own steek
- LIFO
- frame are remove when method completed.

3. Method Area: store class level data such as class structure method code, and constants.
- include runtime constant pool, method data, and feild data

. program counter (pc) Register:
- hold address OF currently executing JVM instruction for each thread
- each thread hay its own native method stack.

= Garbage collection process:
· Minor GC: clean yong Gen. by moving surviving object to the survior space or old generation.
· Major GC: clean entire heap, including O.G., more time consum

Refer dig. (back page)

Write more → interview

- Difference betn JDK, JRE and JVM

- Java virtual machine → abstract machine, it ~~also~~ called virtual machine because it does't physically exit

  → provide runtime env.

  Java → bytecode
  - load code
  - verifies code
  - Execute code
  - provide runtime envirnmant.

- JRE → Java Runtime env. → It is also written as Java RTE
  → setof software tool
  → provide runtime env.
  → It is implementation of JVM, It physically exits.
  → contain libraries + other File that JVM uses at runti

- JDK → Java Development Kit → ~~deploy~~ Java application and applets.
  → it physically exits
  → It contain JRE + development tools.

→ Standard Edition Java platform
→ Enterprise Edition Java platform
→ micro Edition Java platform
} JDK implementation, release by oracle copratio

→ JDK contain privat Java virtual machine
→ complier (Javac) & ackchiver (Jar),
→ documentation generoter (Javadoc)

• What is the role of the JVM in Java? How does the JVM execute Java code?

→ The JVM: Is crucial in Java's "write once, run anywhere" It serve layer betn Java code and the underlying hardware, allowing Java application to run on any device with a ~~Javm~~ installed. The JVM perform key task such as memory management, garbage collection and bytecode execution.

How the ~~JVM~~ Execute Java code:

compilation: Java source code (Java file) is compiled (Javac) into bytecode (.class file) which is platform independent.

class loading: The JVM load the compile bytecode into memory using the compiler.

• Bytecode verification: The JVM verifies the bytecode to ensure it is safe and adheres to Java's security constraint.

• Execution: The JVM execute the bytecode via the Just in time (JIT) compiler, which convert bytecode into native machine code at runtime for faster execution.

• ~~Memory~~ management: The JVM manage memory through autom garbage collection, which reclaim memory used by object no longer in use. This process ensure that Java code ~~run~~ consistenly across diff plat

(8) what is the significant of class loader in Java?
what is the process of garbage collection in Java?

→ i) The class loader is a part of Java Runtime Environment (JRE) that dynamically load Java classes into JVM at runtime.

- Dynamic loading: class loader loades classes only when they need
- each class load into seperate
- security
- Developer can crete custom classloader

process:

- JVM automatically identifies and reclaim memory occupied by obj.
- each method where each object has reference count.
- Reachibility analysis - obj still accesible (reachable) from root referen
-

(9) what are access modifier in Java, and how do they differ from each other?

→ - public - protected - Default (package level private) - private

(10) what is difference between public, protected and default access mo

→ public: The member is access from any other class
→ protected: the member is access within the same package or subcla
→ Default: The member is accesible only within same package

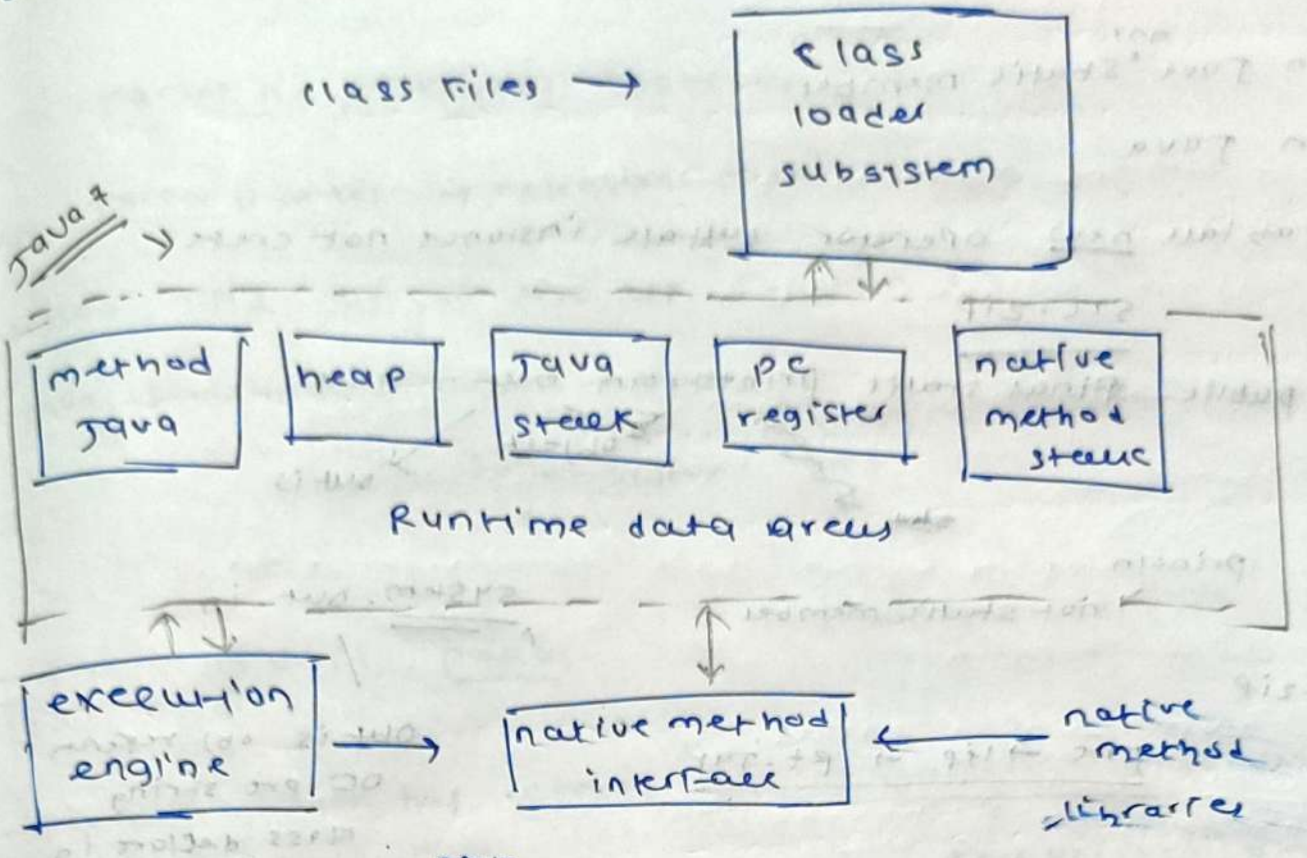
(11) can you override a method with a diff. acess modifier in a subclass? for example, can a protected method in a superclass be overridden with a private method in subclass

→ NO.

For ex → if a method is protected in superclass, it cannot be overridden as private in subclass. Doing so ~~we~~ ~~are~~ would reduce accesibility of method

→ You can override a method with less restritive modifier. For instance, you can override a protected method as public.

• overview of JVM Architecture



use to Java file

• component   Rt. Jar for boot strap

Q. → Describ ~~use to~~ ~~load~~ JVM arch. ~~file~~

→ i) class Loader subsystem: load class file into memory. It is responsible for loading, locating & initializing classes & interfaces
component: Bootstrap classloader → load core Java lib from rt.jar
Extension classloader → load classes from Java Extension (ext) dir.
Application classloader → load classes from the application classpat

ii) Runtime Data area: - stack: each thread has its own stack
- Heap → storage object & arrays created by app.
- program counter (PC): hold the add of the current instru. exe. by thre
- native method stack: manage call to native method (method writt. c or c
iii) Execution Engine: execute bytecode loaded into memory by classlo
component: - Interpreter → exec. code bytecode line by line. it slowe
JIT compiler - Garbage collector
iv) Native methode interface (JNI): allow Java code run into JVM (C,
v) Native libraries: contain lib. required for interfacing with un. OS.
vi) JVM memory management: manage memory allocation & deallocat optimizing the use of system resource throug garbage collecto
stack mangement, heap management.

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

→ Just-In-Time (JIT): The JIT compiler is part of the JVM that optimizes the execution of bytecode by compiling it into machine code at runtime.

- reduce need of repeated interpretation, leading to fast execution
- it optimize code based on program runtime behavior.

• Bytecode: it is an intermediate, platform-independant code generated by the Java compiler from Java source code.

- It store in .class file
- The JVM execute bytecode, either by interpreting it or compiling it into native code using JIT compiler.
- write once, run anywhere.
- The combination of bytecode & JIT compiler make both Java versatile & efficient, ~~[illegible]~~

⑦ How does achieve Platform Independence through JVM?

→ i. compilation to bytecode : ~~in~~ Java → bytecode → .class file.

ii) JVM is an Abstraction Layer: available many platform

- platform independent representation of your code it design to exe.

iii) Developer compile their Java p. into bytecode once

iv) The JVM hide ~~the~~ the detail of the underlying hardware & os from the bytecode

(12) Diff bet? protected and default (package-private) acces

→ Default ~~protected~~ : The mem. is accessible within the same package It cannot be acess. from outside the package, even if they are subclasses. This is more restrictive than protected since it limit ackss to the package level only.

protected: The member is accible within same package and also by subclasses, even if they in diff package. This allows subclassing outside ~~subclasses~~ the package while res. genera clas

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

→ yes, it can only done with nested classes (i.e. inner classes)

→ where it can be done → private class define within another ~~the~~ class (A toplevel class cannot be private)

→ The private class is accesible only within the outer class that contain it.

→ It cannot be accessed directly from outside the outer class

(14) Can a top level class in Java be declared as protected or private? why or why not?

→ NO,

- access control: To control access within the context of enheritance (protected) and encapsulation (private) at the class member level, not for top level classes

- visibility: Top level class are meant to be accessed by other class atleast within the same package. and these thing do to in accesibil

→ public → access any where

- Default → access within same package

(15) what happen if you declare a variable or method as private in a class and try to acess it form another class within same packages

→ it give compilation error.

→ private modifier : it restrict acess to member outside class

(16) ~~Explain the concept~~ private variable or private method from classA, resulting in ~~compilation~~ error.

(18) Explain the concept of "package-private" or default access. How does it affect the visibility of class member

→ ~~when a class member is~~ ~~the~~ mean the member accessible only to other class in the 'same package'. It is not accessible to classes in different package, regardless of whether those classes are subclass.

- when a class member (variable, method or constructor) is declared without an access modifier.