Name:- Ashish Jadhav - JH

classmate
Date 5-09-24
Page 1

Assignment No-3

1) Components of JVM

a) Class Loader
b) Memory Area
c) Execution Engine
d) Native Method Interface
d) Native Method Library.

a) Class loader:- Class loader has three major responsibility
- Class Loader:-
- Main Responsibility is taking the class and loading into the memory. We can find two loaders in Java. One is JVM and second one is custom-defined class loader.

- Linking: Divided into 3 parts
  > Verification :- It has byte code verifier in the JVM. From there, it will check whether the particular class is safe to execute or Not.
  > Preperation:- If you use an instance level variable or static variable in your class, the prepeparation part will assign a default value for that.
  > Resolution:- JVM replaces students with a specific memory location with a newly created student object.

- Initialization:- It will assign actual values. It will also execute static blocks. This execution occurs from top to bottom in class & from parent to child hierarchy.

b) Memory Area:

- Method area
- Heap area
- Stacks
- PC Registers
- Native Method area

- Method area / Heap area :- loads all the class information used to keep type info about the class. For every JVM, you can have only one method area and Heap area. This is a thread of memory area

- Stacks :- will keep the method info. otherwise, local variables those type of thing A seperate runtime stack is created for every new thread. AKA runtime stack.

- PC Register :- If it is not native method pc registers will hold the info. about the next execution. This memory area is relatively small & upto fixed size.

- Native Method area :- This is a stack that can support native methods that are written in diff. language

c) Execution Engine :- The execution engine is responsible in order to run the particular program. The execution engine is devided into three components as :

- Interpreter
- JIT compiler
- Garbage Collector.

> Interpreter :- It's primary responsibility is to read, interpret, and execute the java program line by line.

> JIT compiler :- Main job of JIT compiler is to overcome for the interpreter's disadvantage of slowness during execution. The JIT compiler only works for repeated methods.

> Garbage Collection :- It is used to free up space memory by removing & collecting all the objects from heap area when there is no particular reference.

d) Native Method Libraries :- This contains the libraries which are required by the execution engine while executing byte code.

e) Native method Interface :- It acts as a bridge bt^n execution Engine & Native method libraries while executing.

Q2 Differentiate between JDK, JVM & JRE

JDK:- The Java development kit is a develop ment platform that includes tool of coding, debugging and compilation. It's a superset of JRE like it contains all the tools of JRE, plus more.

JVM:- The Java virtual machine is the foundation of java programing and it's responsible for executing Java Programs. It contains both JDK & JRE & it's what ensures that JAVA programs can run on diff. platforms

JRE:- The Java Runtime Environment providrs the minimum enviroment needed to run Java programs. It includes the JVM & other components to execut to run JAVA programs.

Q3) What is the role of JVM in Java & how does the JVM executes Java code?

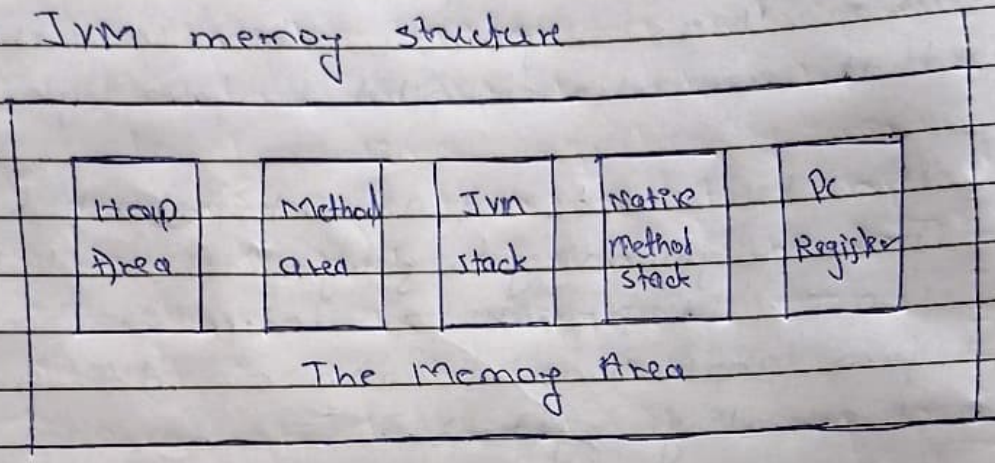
→ i) Java Virtual Machine (JVM) is a virtual m/c that enables a computer to run Java programs as well as programs written in other lang. that are also compiled to java bytecode.

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 diff. platforms & operating systems.

Q 4 Java Memory Management:- The major concepts of in Java Memory mangement

1) JVM memory structure
2) Working of Garbage collector

1) JVM memory structure



- Heap area:- It is runtime data area and stores the actual object in a memory. It is instantiated during the virtual m/c startup. There exists only one heap for a running JVM process

- Method area:- It is a logical part of the heap area & it is created on virtual m/c startup this memory is allocated for class structur method data & constructor field data & also for interfaces or special method used inclus

- Stacks:- Will keep the method info or local variable those type. A seperate runtime stack is created for every new thread

- Native method area :- It also called as C stacks. native methods are supported that are written in diff lang. This memory is allocated for each thread when it's created.

- PC Register :- If it is not native method, PC registers will hold that info. about the next execution. PC register is capable of storing the return adress or a native pointer on some specific platform

2) Working of a Garbage Collector :-

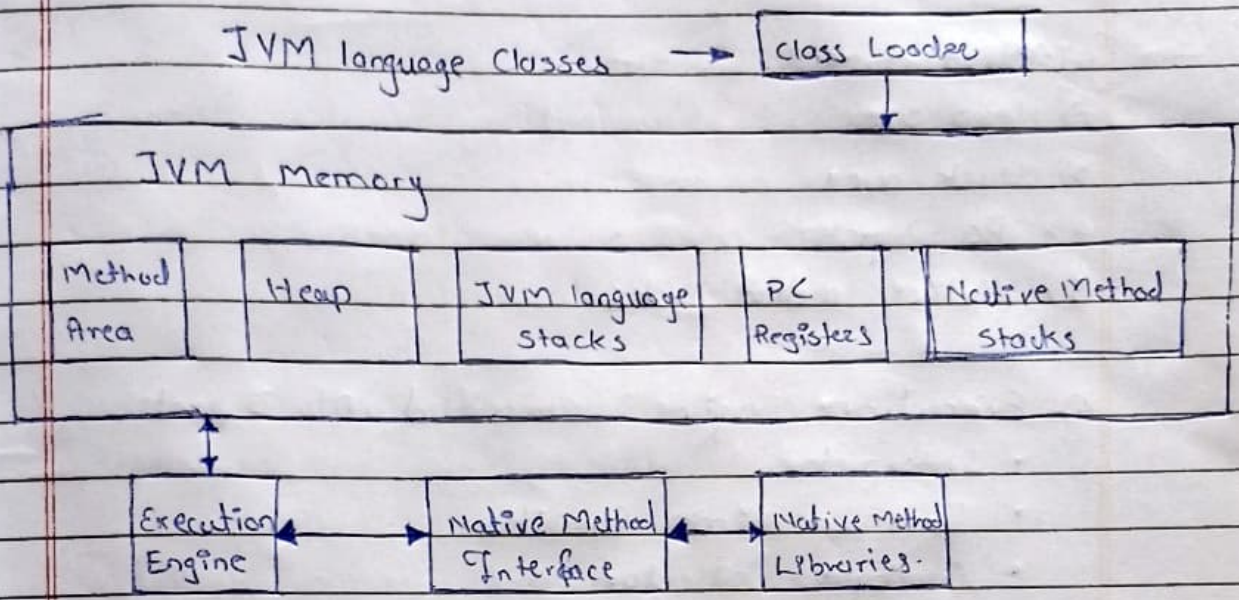
Garbage collection process causes the rest of the processes or threads to be paused & thus it's costly in nature. This problem is unacceptable for the client but can be eliminated by applying algorythm based several garbage collector.

Q5 What are the JIT compilers and it's role in the JVM? What is the bytecode & why is it important for Java?

A JIT compiler, is a componet within the Java virtual Machine (JVM) that dynamically translates Java bytcode into native machine code at runtime, significantly improving the performance of JAva applications by allowing for optimised execution on the specific hardware it's running on.

Since bytecode is not tied to any specific m/c architecture, the same java program can run on diffrent OS with a compatible with a compatible JVM. The JVM interprets the bytcode & exicuts it on current hardware Java sourcecode is first compiled into bytecode using a java compiler & which is then stored in class files.

06 Describe the architecture of the JVM.



1) Class Loader subsystem: It is mainly responsible for 3 activities
   1) Loading
   2) Linking
   3) Initialization

2) Class Loaders: Three primary types of class loader.

- Bootstrap class loader:- loads core java API classes from the JAVA_HOME/lib directory
- Extension class loader:- Loads class from JAVA_HOME /jre/lib/ext directory
- Sys/App class loader: loads classes from the application class path which is specified by the Java.class.path environment variable

3) JVM areas (Memory Area)

1) Method area & ~~for~~ the
2) Heap area
3) Stack area
4) PC Registers
5) Native method stacks

4) Execution Engine: Classified into 3 parts.
- Interpreter
- Just-In-Time Compiler
- Garbage Collector

5) Java Native Interface: It is an Interface that interacts with the Native method libraries & provides the native libraries (c, c++) required for the execution.

6) Native Method Libraries:- These are collection of native libraries required for executing native methods.

Q7 How does Java achieve platform indipendereethrough IVM?

→ Java is platform Indipendant because it uses a "Write once, Run Anywhere" approach. Java source code is compiled into bytecode, which is platform-neutral This bytecode can be executed on any platform that has a java machine (JVM) compatible with that bytecode.

Q8 What is the significance of the class loader in Java? What is the process of garbage collection in Java?

→ The class loader is responsible for locating libraries, reading their contents, and loading the classes contained in the libraries.
Garbage collection in java is the automated process of deleting code that's no longer needed or used. This automatically frees up memory space and ideally makes coding java apps easier for developers.

Q9 What are 4 access modifiers in Java? How do they differ from each other?

→ The four access modifiers in Java are; public, private, protected and default.

- Public :- can be accessed from any class regardless of package.
- Private :- They are only accessible within the class where they are declared
- Protected :- Can be accessed within same package and by subclasses in other packages
- Default : If no access modifier is specified, The member is considered as 'package-private' - access within same packages.

Q 10) What is the difference between public, protected and default access?

→ 1) Public :-
- Can be accessed by any class regardless of package
- Considered as most open access level.

2) Protected :-
- Can be accessed within the class itself and it's subclasses, even if they are in a different package.
- Used to control access when inheritance is solved

3) Default : (Package-private)
- Can be accessed by classes within the same package
- Considered the default access level if no explicit access modifier is specified.

Q 11) Can you override a method with a diff access modifier in a subclass? Can a protected method in superclass be overridden with put method subclass? Explain

→ Yes, The protected method of a superclass can be overridden by a subclass. If the superclass method is protected, the subclass overriden method can have protected or public which means subclass over-ridden method cannot have a weaker access specifier.

Q12) What is the difference between Protected & default access?

→ • Default :-

The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify the any access level, it will be the default.

• Protected :

The access level of a protected modifier is within the package and outside the package through child class.

Q13) Is it possible to make a class private in Java? If yes where it can be done and what are the limitations?

→ Yes, private classes are allowed, but only as inner or nested classes. If you have a private/nested class, then the access is restricted to the scope of that outer class. If you have a private class on own as a top level class, then you can't get access to it from anywhere.

Q14) 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. It can be either public or default (no modifier). If it does not have a modifier it is supposed to have a default class.

• Because the whole point of private is to disable the usage of the item outside the class they have been defined in.

Q15) What happens if you declare a variable or method as private in class and try to access it from another class within the same package

→ The methods or data members declared as private are accessible only within the class in which they are declared. Any other class from the same package will not be able to access these members. private means "Only visible within the enclosing class".

Q16) Explain the concept of "Package Private" or "default" access. How does it affect the visibility of class members?

→
- 'Private' restricts access to the elements only within the class they are declared.
- 'Protected' allows access within the same package or in subclasses, which might be in diffrent packages.
- Lastly, the 'default' access (No modifier) limits the visibility to classes within the same package.