1- Study JVM Architecture.

---

the JVM is divided into three main subsystems:

ClassLoader Subsystem
Runtime Data Area
Execution Engine
1. ClassLoader Subsystem
Java's dynamic class loading functionality is handled by the ClassLoader subsystem. It loads, links. and initializes th e class file when it refers to a class for the first time at runtime, not compile time.

1.1 Loading
Classes will be loaded by this component. BootStrap ClassLoader, Extension ClassLoader, and Application ClassLo ader are the three ClassLoaders that will help in achieving it.

BootStrap ClassLoader – Responsible for loading classes from the bootstrap classpath, nothing but rt.jar. Highest pri ority will be given to this loader.
Extension ClassLoader – Responsible for loading classes which are inside the ext folder (jre\lib).
Application ClassLoader –Responsible for loading Application Level Classpath, path mentioned Environment Varia ble, etc.
The above ClassLoaders will follow Delegation Hierarchy Algorithm while loading the class files.

1.2 Linking
Verify – Bytecode verifier will verify whether the generated bytecode is proper or not if verification fails we will get the verification error.
Prepare – For all static variables memory will be allocated and assigned with default values.
Resolve – All symbolic memory references are replaced with the original references from Method Area.
1.3 Initialization
This is the final phase of ClassLoading; here, all static variables will be assigned with the original values, and the sta tic block will be executed.

2. Runtime Data Area
The Runtime Data Area is divided into five major components:

Method Area – All the class-level data will be stored here, including static variables. There is only one method area per JVM, and it is a shared resource.
Heap Area – All the Objects and their corresponding instance variables and arrays will be stored here. There is also one Heap Area per JVM. Since the Method and Heap areas share memory for multiple threads, the data stored is not thread-safe.
Stack Area – For every thread, a separate runtime stack will be created. For every method call, one entry will be mad e in the stack memory which is called Stack Frame. All local variables will be created in the stack memory. The stac k area is thread-safe since it is not a shared resource. The Stack Frame is divided into three subentities:
Local Variable Array – Related to the method how many local variables are involved and the corresponding values will be stored here.
Operand stack – If any intermediate operation is required to perform, operand stack acts as runtime workspace to per form the operation.
Frame data – All symbols corresponding to the method is stored here. In the case of any exception, the catch block i nformation will be maintained in the frame data.
PC Registers – Each thread will have separate PC Registers, to hold the address of current executing instruction once the instruction is executed the PC register will be updated with the next instruction.
Native Method stacks – Native Method Stack holds native method information. For every thread, a separate native m ethod stack will be created.

3. Execution Engine
The bytecode, which is assigned to the Runtime Data Area, will be executed by the Execution Engine. The Executio n Engine reads the bytecode and executes it piece by piece.

Interpreter – The interpreter interprets the bytecode faster but executes slowly. The disadvantage of the interpreter is that when one method is called multiple times, every time a new interpretation is required.
JIT Compiler – The JIT Compiler neutralizes the disadvantage of the interpreter. The Execution Engine will be using the help of the interpreter in converting byte code, but when it finds repeated code it uses the JIT compiler, which co mpiles the entire bytecode and changes it to native code. This native code will be used directly for repeated method c alls, which improve the performance of the system.
Intermediate Code Generator – Produces intermediate code
Code Optimizer – Responsible for optimizing the intermediate code generated above
Target Code Generator – Responsible for Generating Machine Code or Native Code
Profiler – A special component, responsible for finding hotspots, i.e. whether the method is called multiple times or not.
Garbage Collector: Collects and removes unreferenced objects. Garbage Collection can be triggered by calling Syste m.gc(), but the execution is not guaranteed. Garbage collection of the JVM collects the objects that are created.
Java Native Interface (JNI): JNI will be interacting with the Native Method Libraries and provides the Native Librari es required for the Execution Engine.

Native Method Libraries: This is a collection of the Native Libraries, which is required for the Execution Engine.

Understand the relatonship between JDK, JRE & JVM.

JMV is:

A specification where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Oracle and other companies.
An implementation Its implementation is known as JRE (Java Runtime Environment).
Runtime Instance Whenever you write java command on the command prompt to run the java class, an instance of J VM is created.
What it does
The JVM performs following operation:

Loads code
Verifies code
Executes code
Provides runtime environment
JVM provides definitions for the:

Memory area
Class file format
Register set
Garbage-collected heap
Fatal error reporting etc

JRE
JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment . It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runt ime.

The implementation of JVM is also actively released by other companies besides Sun Micro Systems.

JDK
JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environ ment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tool s.

JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

Standard Edition Java Platform
Enterprise Edition Java Platform
Micro Edition Java Platform
The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (jav a), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.

Study the Features of Java

-A list of the most important features of the Java language is given below.

Java Features
Simple
Object-Oriented
Portable
Platform independent
Secured
Robust
Architecture neutral
Interpreted
High Performance
Multithreaded
Distributed
Dynamic

Q4. Study and prepare real life examples for OOPS pillar.