Name - Purva Salvi
PRN - 240340520065

14-3-24

Moodle Assignment - 1

Q1. Explain the components of JDK.

→ The main components of JDK are:-

① Java compiler (javac) -
This is a tool used to compile Java source code files (.java) into Byte code files (.class files) that can be executed on Java Virtual machine (JVM).

② Java Virtual Machine (JVM) -
The JVM is an essential component of JDK. It is responsible for executing java bytecode. The JVM provides a runtime environment in which java applications can run independently of the underlying hardware and OS.

③ Java Runtime Environment (JRE) -
The JRE is the subset of JDK. It includes the JVM & other libraries necessary for running Java applications but does not include development tools like the compiler. The JRE is needed to run & compiled Java programs on a users machine.

④ Java API's -
The JDK includes a vast collection of libraries, packages, and API's that provides various functionalities to Java developers. These APIs cover a wide range of areas including networking, database connectivity, user interface development (Swing, JavaFX), file I/O, security & more.

⑤ Java Development Tools -
The JDK comes with several development tools that aid in the creating, debugging & profiling of Java applications. Some of the prominent tools include:

(i) Java Debugger (jdb)
(ii) Java Archieve Tool (jar)
(iii) JavaDoc
(iv) JavaFX Packager

⑥ JavaFX –

JavaFX is a platform for creating rich internet applications (RIA) and modern, cross platform graphical user interface (GUI's) for Java Applications. It includes set of libraries and tools for building interactive multimedia applications.

Q2. Difference between JDK, JVM, JRE.

→ ① JDK –

Purpose: The JDK is a comphrehensive software development kit for building Java Applications.

Components: It includes tools & libraries necessary for Java development, such as javac, jre, java API's, development tools and javafx.

Usage: Developer used the JDK to write, compile, debug and deploy Java applications. It is essential in ~~new~~ creating new Java software.

② JVM –

Purpose: The JDK is an abstract computing machine that provides runtime environment for executing java bytecode.

Components: The JVM includes various subsystem, such as class loader, bytecode verifier, interpreter, JIT compiler, garbage collector & runtime libraries.

Usage: JVM is responsible for running compiled Java bytecode on different hardware & OS.

③ JRE –

Purpose: The JRE is a subset of JDK & provides the runtime environment for executing Java applications.

Components: It includes the JVM, core Java class libraries, and supporting files required for running Java applications.

Usage: End users install the JRE on their system to run Java Applications. It doesn't include development tools like the compiler but it is sufficient for executing Java programs.

Summary: (i) JDK is used by developers for Java Applications development and includes the development tools, runtime environment and libraries.

(ii) JVM is responsible for executing Java bytecode and providing platform independence to Java applications.

(iii) JRE is a subset of the JDK and includes the JVM and necessary runtime libraries for running Java applications on end-user machine.

Q3. What is the role of JVM in Java? How does the JVM execute Java code?

→ The JVM plays a very crucial role in the execution of Java code. It's primary responsibilities include:

(i) Platform Independence
(ii) Execution of bytecode
(iii) Memory management
(iv) Security
(v) Exception Handling
(vi) Dynamic class loading

Regarding how the JVM executes Java code:

① Class Loading –
The JVM loads compiled Java classes into memory as they are referenced by the program. This includes loading classes from classpath, JAR files & other locations specified by the applications.

② Bytecode verification –
Before executing bytecode, the JVM performs bytecode verification to ensure that it confirms to the rules of the Java language & does not violate security constraints. This step helps prevent potential security vulnerabilities and ensure the integrity of program.

③ Interpretation & Compilation –
Interpretation involves executing bytecode instructions one at a time while JIT compilation converts entire sections of bytecode into optimized native code that can be executed directly by the CPU.

④ Execution & Runtime Management –
Once bytecode is loaded and verified, the JVM executes it according to the rules of Java language. During execution, the JVM manages memory, handles exception, enforces security policies and performs other runtime task to ensure the correct and secure operation of the program.

Q4. Explain the memory management system of JVM.

→ The memory management system of the JVM is responsible for allocating & deallocating memory resources used by Java applications, it includes several key components and techniques to efficiently manage memory and ensure optimal performance

(i) Heap memory
(ii) Method Area
(iii) JVM stack
(iv) Native method stack
(v) PC registers

Q5. What are the JIT compiler & its role in JVM? What is bytecode and why it is important for Java?

→ JIT Compiler –

The JIT compiler is a part of the JVM that dynamically compiles Java bytecode into native machine code at runtime. The primary role of the JIT compiler is to improve the performance of Java applications by generating optimized native code that executes more efficiently than interpreted bytecode.

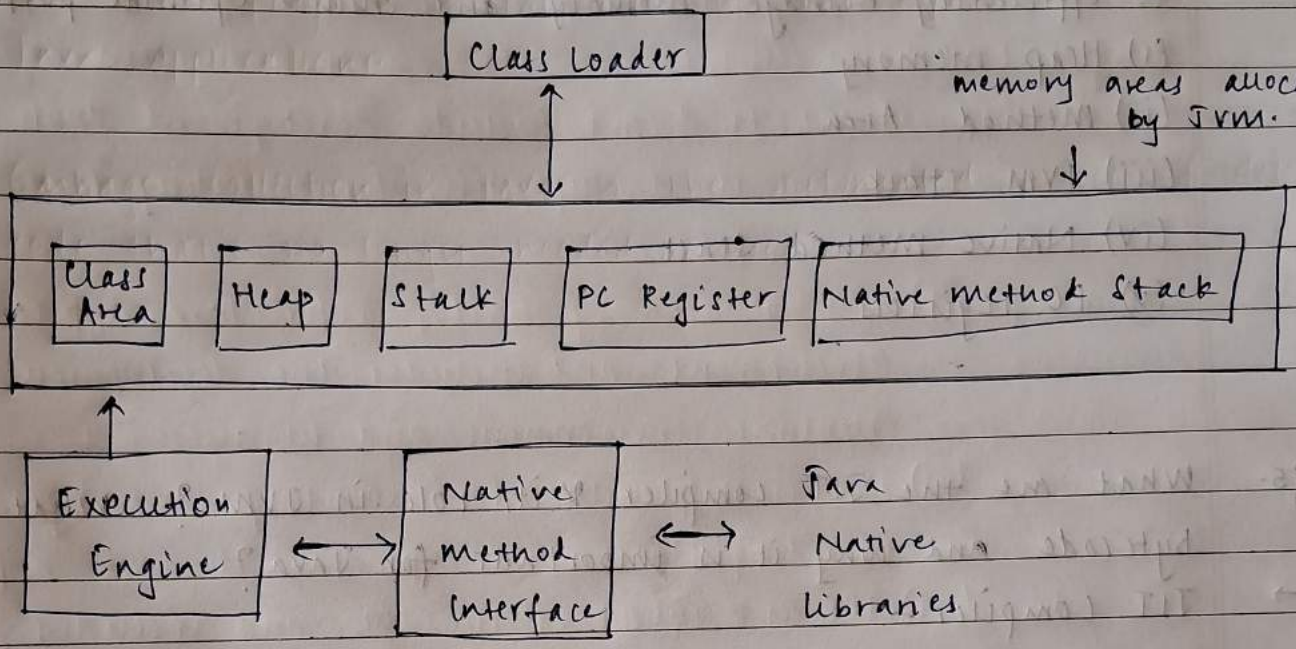
Bytecode is an intermediate representation of Java source code that is compiled by the Java compiler. It consists of machine independent instructions that are designed to be executed by the JVM.

In Summary, bytecode is the key component of the Java platform that enables platform independence, security & flexibility in executing Java applications, while the JIT compiler enhances performance by dynamically translating bytecode into optimized native code at runtime.

Q6. Describe the architecture of JVM?

→ Architecture of JVM-

It contains class loader, memory area, execution engine, etc.



Class Loader-

It is a sub system of JVM which is used to load class file.

Class Area-

Class (method) area stores per-class structure such as the runtime constant pool, field & method data, the code for methods.

Heap-

It is a runtime data area in which objects are allocated.

Stack-

Java Stack stores frames. It holds local variable & partial results, and plays a part in method invocation and return.

Program Counter Register (PC Register) –
PC Register contains the address of the JVM instruction currently being executed.

Native Method Stack –
It contains all the native methods used in the application.

Execution Engine –
It contains : (i) A virtual processor
(ii) Interpreter
(iii) JIT compiler

Java Native Interface –
JNI is a framework which provides an interface to communicate with another application written in another language. like C, C++, Assembly, etc. Java uses JNI framework to send o/p to the console or interact with OS libraries.

Q7. How does Java achieve platform independence through the JVM?

→ Java is platform independent because it is compiled to a bytecode that can be run on any device that has a JVM. This means that you can write a Java program on one platform and then run it on a different platform without making any changes to the code.

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

→ In Java, a class loader is responsible for loading Java classes into the JVM at runtime. It is an integral part of JRE and plays a crucial role in the dynamic nature of Java applications.

Garbage collection in Java is the automated process of deleting code that's no longer needed or used.

—x— x——