Assignment 2

1) Explain componant of the JDK?

Ans Follwing componant of JDK :-

① Java compiler (Javac) :

The compiler (Javac) is a key componant of JDK that transforms Java source code (.Java file) into bytecode (.class file). the genrated byte code can be executed on any platform with a Java virtual machine installed, ensuring the "write once, run anywhere" philosophy of Java.

② Java virtual machine (JVM):-

The Java virtual Machine is runtime engine that execute java bytecode. it provide an abstraction layer between the Java application & the underlying operating system.

③ Java Runtime Environment (JRE) :- The Java Runtime Environment (JRE) is a subset of JDK that include the JVM & essenssial class liborraries. it required to run Java application on ~~ede~~ end-user system without the need for development tool.

④ Java API :- An Application programing ~~language~~ interface (API) is connection between computer or between computer program. In a more simply way - API is set of ways & ~~us~~ rules for interaction & data Exchange between differant program & computer.

② Differant between JVM JDK JRE

JDK	JVM	JRE
Defination:- software development kit for Java, including tools & librories for develo-ping java application	virtual machines that executes Java bytecode & provide a runtime environment for Java application	subset of the JDK that includes the JVM & essential librarie required for execu Java application.
components :- Java compiler (Javac) deve-lopment tools (debugger, archive tool, etc) librories & API's for development	-Iterpreter for Java bytecode - Just-In-time (JIT) complier (in some) JVM implemetation - garbage collector - Runtime librories	- Java virtual mac (JVM) - Java runtime libra-ries, additional required for run Java applications
3) **Purpose & used** & API for develop-ment, including writing, compiling & debugging code	Execute Java byte code & provides a platform. independant runtime enviornment for Java application	provides the run environment necessa for executing Jav application. but does no include development tool like compiler & debugger.
4) **Example usage** :- Developing, compiling & debugging & deploy Java application	Running Java applica-tions on various platforms	Running standalone Java applications or Java applets within web browsers.

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

1* Role of JVM:

- it serves as an interpreter for Java bytecode, executing Java program on any platform.
- it provide a runtime environment that abstract away the underlying hardware & operating system details
- it manages memory allocation & garbage collection, ensuring efficient use of resources.

2) Execution of Java code:

- The Java compilee translate source code into platform-dependant bytecode.
- The JVM then interpretee or optionally compiles this bytecode into machine-specific instructions.

(3) - Just-in-time (JIT) compilees in modeen JVMs further optimize performance by translating frequently executed bytecode into native

- overall, the JVM provides a runtime environment where Java code can be executed efficiently & reliably across diverse platforms.

(4) Explain the memory management system of the JVM

Ans:- The JVM manages memory through automatic garbage collection, where unreachable objects are identified & removed to free up memory for new allocations. This process ensures efficient memory utilization & prevents memory leaks. Aditionally, the JVM can optimize memory uses through techniques like generational garbage collection & adaptive sizing of memory areas.

The memory management system of the JVM involves three main areas:

* Heap memory:

- This is where object & their instance variable are stored.
- The heap is divided into two main section:

@ The young generation & old generation.

- New object ~~is~~ are allocated in the young genration & when become full, a garbage collection process called minor GC is triggered to reclaim memory from unreachable object.

* Method Area:

- This area stores class metadata, static variables, & constant pool information.
- In older JVM implementation, this was known as the permant genration in newer version. it is called metaspace

* Stack Memory :-

- Each thread in a java application has its own stack memory.
- Stack memory is used to storing method invocation, local variable & partial result
- it operates in last-in, first-out (LIFO) manner
- Stack memory is typically smaller than heap memory & is relesed when the method completes execution.

5) What are the JIT compiler & its role in JVM? What is the bytecode & why is it important for java?

* JIT compiler :- (just-in-time compiler)

- The JIT compiler is a componant of JVM that improves the performance of Java application by dynamically translating java bytecode into native machine code during runtime.
- it defines frequently executed section of bytecodes & compile them. into highly optimized native code, which can execute more efficiently on underlaying hardware.
- The JIT compiler help reduce the interpretation overload of bytecode, resulting in faster execution of java program

* Bytecode:

- bytecode is the intermediate representation of Java source code after compilation by the Java compiler. (Javac)
- It is platform independant format that can execute on any system with a compictable JVM, making Java program ~~into~~ inhearatly portable.

(6) Describe the Architecture of JVM?

Ans- The architecture of the java virtual machine (JVM) can be bracen down into sever key componanets, each playing a crucial role in executing java bytecode efficiently. they are:

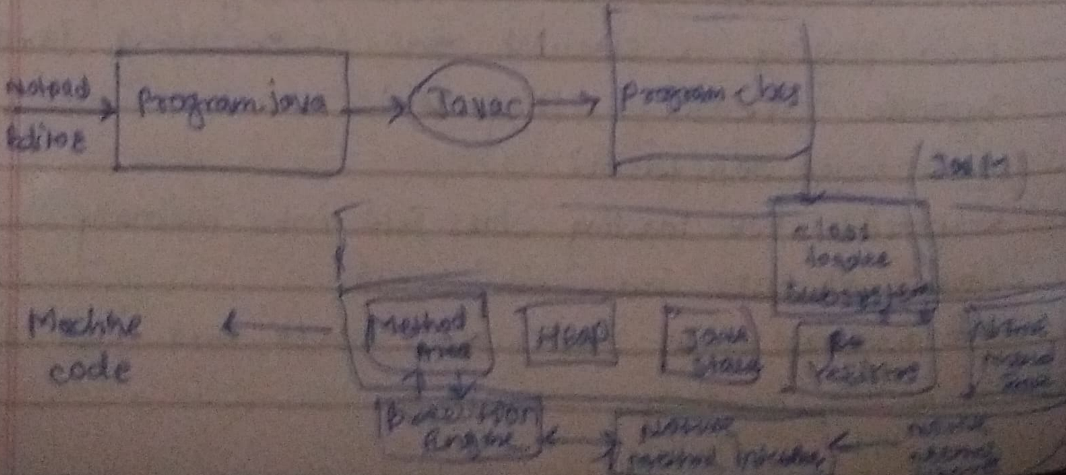
Class Loader Subsystem:

- Responsible for loading class file into memory
- consist of three main components
- ** bootstrap class loader **: Loads core java classes from the bootstrap classpath

- **Extension class Loader** :- Loads class from the extension directories
- ** Application class Loader ** :- Loads classes from the application classpath

* Runtime Data Area:
- Divided into seeval memory areas:
- Method Area (Metaspace): stores class metadata Static veriable, & costant pool data.
- Heap: store the object & their instance veriable, divi into young & old generations.
- Stack: Each thread has its ~~town~~ own stack me invocation & local veriable.
- PC resistoes: Hold the address of the current instruct-ion being executed
- Native method Stack: store ~~method~~ Native method information.

* Execution Engine:
- Responsible for executing java bytecode
- consist of !-
- intrepreter: Interpreter bytecode instructions & execute them sequentially.



(7) How does Java achive platform independence though the JVM?

→ ① Bytecode :- When you compile a Java source file, it's translated into platform-idependent bytecode. This bytecode is set of instruction meant to be executed by the JVM. its specific to hardwary Nor any operating system

② JVM :-

The JVM is an abstract computing machine that provides a runtime environment for executing Java bytecode. Each operating system has its own implemetation of the JVM, tailored to that specific system. When you run a Java program, you don't execute the bytecode directly; instead, it's interpreted or compiled by the JVM into machine code that's specific to the underlying hardware & operating system.

(8) What is the significance of the class loader in java? What is process of garbage collection in java.

- Class loader in Java:-

The class loader in java is responsible for loading classes into the Java virtual Machine (JVM) dynamicaly at runtime.

- It locate & reads the binary data for a class file, which typically resides in the file system.

① Bootstrap class Loader
② Extension class Loader
~~③ Java Home (lib / ext)~~
③ Application class Loder.

Garbage collection in Java:-

garbage collection in Java is the process of automaticaly reclaiming memory accuipied by objects that are no longer in use program.

① Marking: The garbage collector identifies which object in the heap are reachable & which one no

② Sweeping: once the reachable objects are identified, the garbage collector sweeps through the heap & dellocates memory for objects that are not marked as reachable.

③ Compacting: Some garbage collector perform heap compaction, where live objects are moved to contigous memory location to reduce fragmentation & improve memory locality.