Q. (1) Explain the components of the JDK.

→ Java Development Kit (JDK):

- JDK toolkit provides a comprehensive set of tools to develop, compile & run Java applications.
- JDK is a software development environment provided
- It includes various tools, libraries & executables required for Java Development.
- The JDK forms the foundation for Java Platform, Standard Edition (Java SE), which is the platform used to develop & deploy Java applications for desktops, servers etc.

Components of JDK:

1) Java Compiler (javac)

- It is a key component of JDK that transforms Java source code (.java) into bytecode (.class).
- The generated bytecode can be executed on any platform with a Java Virtual Machine (JVM), ensuring the run anywhere.

2) Java Virtual Machine (JVM)

- It is the runtime engine that executes Java bytecode.
- It provides an abstraction layer between the Java application & the underlying operating system.
- The JVM enables Java programs to run independently of the hardware & operating system, enhancing portability & security.

3) Java Runtime Environment (JRE)

- It is a subset of JDK that includes the JVM & essential class libraries.
- It is required to run Java applications on end-user

systems without the need for development tools.
- Users can execute Java applications using the JRE, ensuring a seamless experience.

4) Java API Libraries
- It provides a vast collection of pre-built classes & methods that simplify common programming tasks.
- These libraries cover areas such as input/output, networking, database connectivity, GUI development.
- Developers can leverage these APIs to accelerate application development & improve code quality.

5) Java Debugger (jdb)
- The Java Debugger allows developers to set breakpoints, inspect variables & step through the code to identify & fix issues during development.
- The debugger significantly aids in understanding the program flow & identifying logical errors.

6) Java Documentation Generator (Javadoc)
- It automatically generates documentation from Java source code.
- It helps in creating comprehensive API documentation, making it easier for developers to understand & use the classes & methods provided by the application's codebase.

7) appletviewer
- This tool is used to run & debug Java applets without a web browser.

8) javap
- It is a class file disassembler.

d) jar
- This file format is used to aggregate many Java class files & associated metadata & resources into one file.

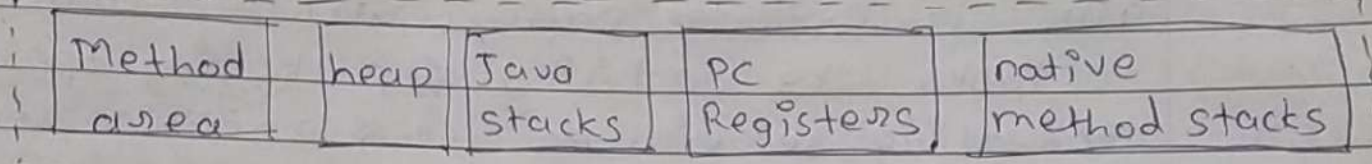
- Apart from the major components mentioned above, JDK also includes various utilities, which cater to tasks like archiving & packaging applications, generating security policies & managing key stores.

Q.(2) Differentiate between JDK, JVM & JRE.
→- JVM converts bytecode to machine specific code. JRE Executes java programs & includes JVM JDK It includes JRE, a Java compiler, a debugger & more components.
- So main difference is:
- JVM is the foundation & ensures the program's Java source code will be performed on any platform.
Whereas JDK is the development platform while JRE is for execution.

Q.(3) What is the role of the JVM in Java & how does it executes Java code?
→- A Java Virtual machine is a virtual machine that enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java bytecode.
- Execution of code through JVM:
1) The Java source code is compiled into bytecode.
2) verify the bytecode & load the Java program through the †class loader into JVM memory.

Q. (4) Explain the memory management system of JVM.

→ - JVM creates various run time data areas in a heap which are used during execution.
- The memory areas are destroyed when JVM exits, whereas the data areas are destroyed when the thread exits



JVM memory structure.

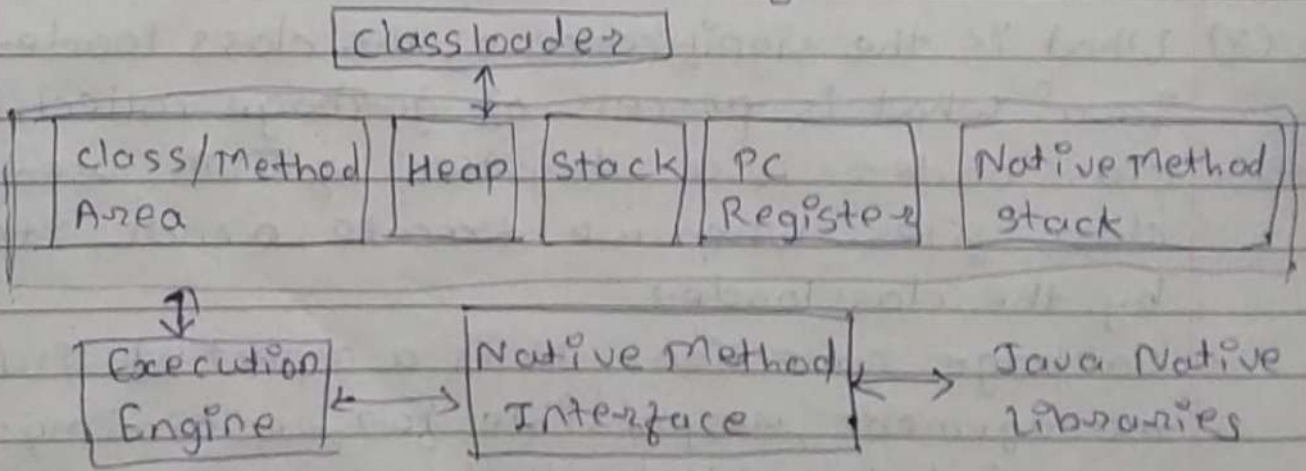
1) Method area - stores class structure, superclass name, interface name & constructors.
2) Heap Area - Stores actual objects & user can control the heap if needed.
- when new keyword used JVM creates instance for object in the heap while the reference of that object stored in stacks.
- when heap is full, the garbage is collected.
3) Stack Area. - It is allocated per thread used to store data.
4) PC Registers - Each thread has a Program Counter register associated with it. It
- It stores return address or native pointer.
- Also contains address of JVM instructions currently being executed.
5) Native method Stack - Stack of native code written in a language other than Java.

Q.(5) What are JIT compiler & its role in the JVM? What is a bytecode & why is it important for Java?

→ - The Just-In-Time (JIT) compiler is a component of the runtime environment that improves the performance of Java applications by compiling bytecodes to native machine code at runtime.
- It is enabled by default. When a method has been compiled, the JVM called the compiled code of that method directly instead of interpreting it.

- Byte code: It can be defined as an intermediate code generated by the compiler after the compilation of source code (Java program).
- It makes Java a platform-independent language.
- i.e. Byte code is computer object code that is converted into binary machine code for computer's hardware processor.

Q.(6) Describe the architecture of JVM.



1) Classloader - Subsystem of JVM used to load class files. A Java program is 1st loaded by it when we run it.
Built-in classloaders - Bootstrap, Extension, system/API

2) Class (Method) Area - Stores pre-class structures.
3) Heap - It is runtime data area stores objects.
4) Stack - Allocated per thread stores data.
5) PC Register - stores address of JVM m/c instruction currently being executed.
6) Method stack - contains native methods.
7) Execution Engine - contains a virtual processor, Interpreter, Just-In-Time (JIT) compiler.
8) Java Native Interface (JNI) - It is a framework which provides an interface to communicate with another application written in another language.

Q.(7) How does Java archieve platform independence through the JVM?

→ - Java compiler generates a bytecode after compiling a java program which then converted into machine code by an interpreter which is JVM.
- So this bytecode can be run on any device with JVM making Java as platform independent language.

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

→ - classloader is a subsystem of JVM used to load class files. When we execute a code, its loaded by the classloader.
- Garbage collection - It is a aspect of JVM memory management, responsible for reclaiming memory occupied by objects that are no longer in use.
- It automatically identifies & removes unreferenced objects, preventing memory leaks & optimizing memory usage.