Q1. Components of JDK

↳ JDK stands for Java Development Kit is a Software Development environment which is used to develop java applications and applets.

JDK physically exists.

Path: new C/programfiles/Java/JDK1.8

JDK → Development tools + JRE - Java Runtime Environment.

Basically contains
1. Java Development tools
2. Java API DOCS
3. Java Archive - (rt.jar)
4. Java virtual machine

JRE = 3 + 4

rt.jar - all core java API available in this file.

Q2. Diff. between JDK, JVM & JRE

	JVM	JRE.
JDK Superset	Subset of JRE	Subset of JDK
- Abstract machine		
- virtual No physical existance		physically exists.
	JVM ↓	- Set of software tools used for developing java Applications
	*Class loader	
	*Runtime Data areas	
- Required to develope java programme	* Execution Area.	- Minimum Software required to run java programm
JDK ↓ JRE + Development tools.		JRE ↓ Java API (rt.jar) + JVM.
JDK has inbuilt JRE.		JRE can be downloded Seperately

Q3. Role of JVM in java? & How does the JVM execute java code?

→ JVM bridge between written java program and executable code across different platforms.
→ JVM - heart of java platform independence.
→ JVM complex system intricately manages code execution, memory and system resources.

→ JVM has following main components.

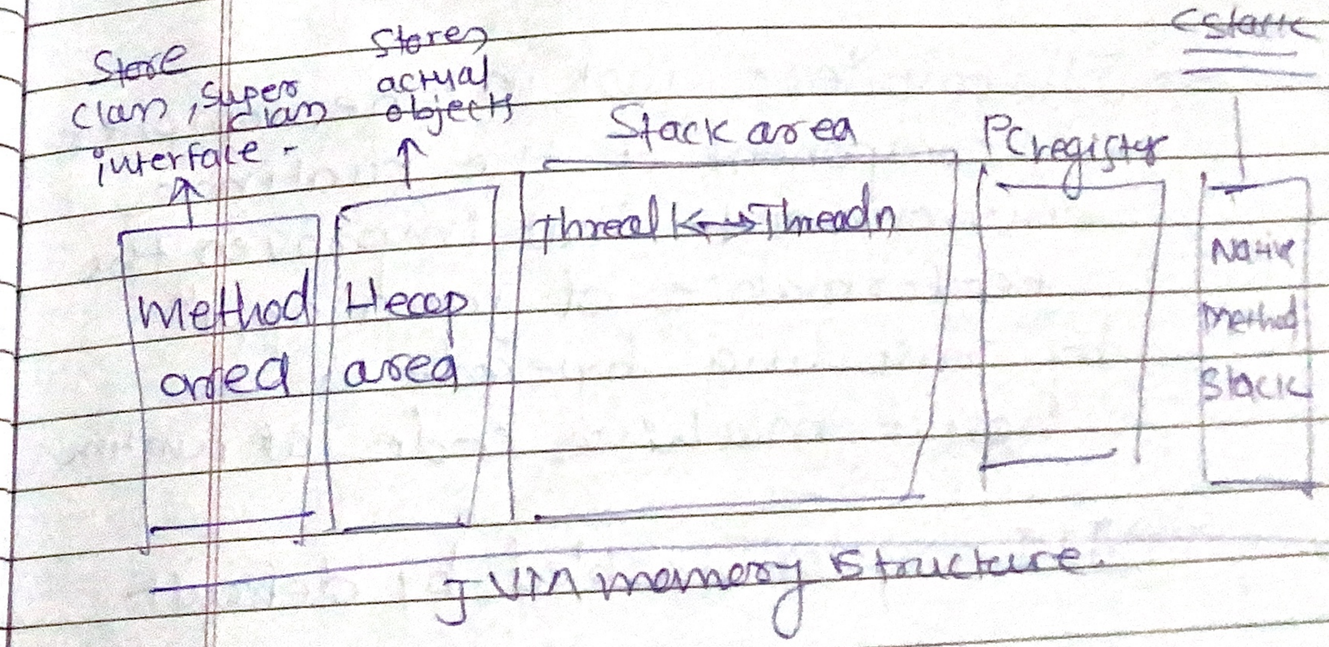
1) Class loader
2) Run time Data area
3) Execution area.

↑ After compilation (javac)
JVM - class loader try to locate file.class file and then JVM uses JIT compiler or interpreter interprete your code line by line and print the output.

4) Memory Management System of JVM.

→ Memory mgmt. is process of allocation and de-allocation of object, called memory mgmt.
→ Java does it automatically via garbage collector.
↳ so additional requirement of memory mgmt. logic.

→ JVM creates various Run time Data areas. → The memory areas destroyed when JVM exits, where as data areas are destroyed when the thread exit.



JVM memory structure.

Q5 what is JIT compiler and it's role in JVM.? what is bytecod. and why it is important for java?

At compile time

Sourcecode.java → Compiler → Bytecode

Native Machine code ← JIT Compiler ← Bytecode

At Run Time

→ JIT Compiler - just in time compiler is a component of the runtime environment that improves the performance of java appl^{n} by compiling bytecode to native machine code at runtim

→ JIT compiler enabled by default

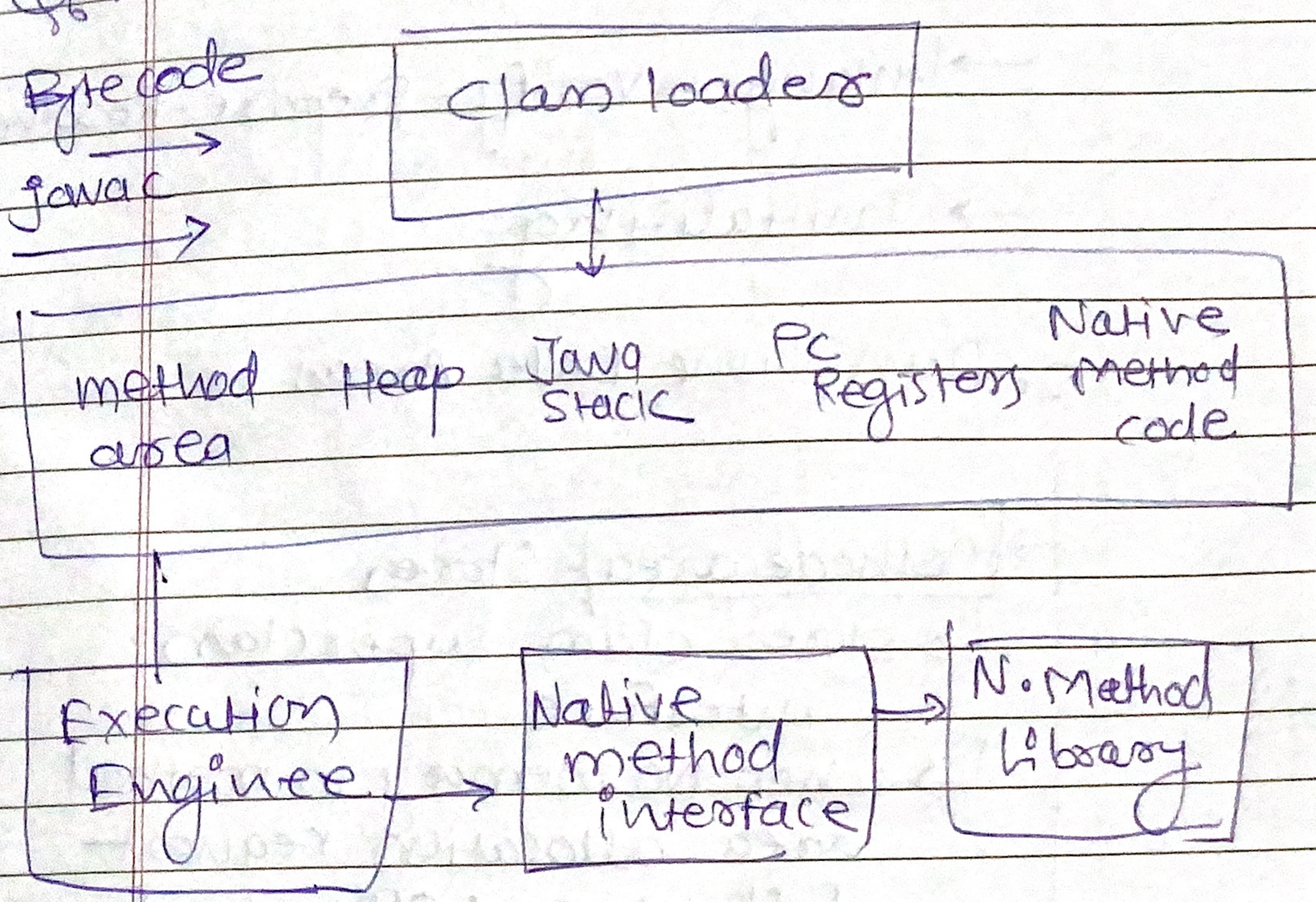
→ Bytecode → Object oriented Assembly language code designed for JVM

→ .class file is Bytecode which can be executed on any OS with JRE and No need for sourcecode file of such java application.

→ Byte code helps java achieve both portability & security.

Qf. Architecture of JVM.

Bytecode → javac →



JVM is divided into 3 main Subsystemes.

1. Class loaders

↳ dynamically loads java class files from file system network etc into JVM.

→ loading ——→ Bootstrap class loader
——→ Extention class
——→ Appl^n

→ linking → verify - Prepare - Resolv

→ Initializing

2. Run time Data Areas

→ Methode area - Stores
↳ stores class superclass interface
↳ when No memory in method area allocation request fail → Out Of Memory Error.

→ Heap
↳ stores actual objects

→ Stack - stores frames
↳ StackOverflow error when computation is larger

→ PC register - stores address of Instruction

→ Native method Stack - holds native method info.

3. Execution Engine

Interpreter — Read & execute bytecode instructions line by line

JIT — used to ↑ efficiency of interpreter

Garbage collector — destroyes unreferenced objects

Q.7 How does java achieve platform independence through JVM

→ java Application when compile create a bytecode file with extention '.class' this '.class' file can be run on any device that has JRE installed.

→ JRE has JVM.

→ This means you can write a java program on any operating System and can run on a different platform without making any changes with help of java virtual machine.

→ JVM has JIT which compile the bytecode to native machine code during run time

Q8 Significance of classloader and process of Garbage Collection.

→

→ classloader responsible for loading java classes dynamically to JVM during runtime

→ classloader - part of JRE.

→ heirarchy model function in classloade

1st - Application Classloader ←
↓
Extension ←
↓
Bootstrap ←

→ JVM controls Garbage collection

* Object is eligible for garbage collection when no live thread can access it
* Program reachable objects are not selected by garbage collector for deletion.
* heap is part of memory with object and only part associated with Garbage collector and GC works to free up heap memory.