Dt : 22/8/2022

**imp*

JVM Internals with Execution flow of program:

=>JVM internally divided into the following partitions:

1.Class Loader SubSystem

2.Runtime Data Area

3.Execution Engine

1.Class Loader SubSystem:

=>Class Loader SubSystem will load class file(ByteCode) onto Runtime DataArea using Loader.

2.Runtime Data Area:

=>This Runtime DataArea internally divided into the following partitions:

(a)Method Area

(b)Heap Area

(c)Java Stack Area

(d)Pc Register Area

(e)Native method Area

(a)Method Area:

=>The memory location where the class is loaded is known as Method Area.

=>while Class loading static programming components will get the memory within the class.

=>Once main() method got the memory within the class,then it is automatically copied onto JavaStackArea to start the execution process.

Dt : 23/8/2022

(b)Heap Area:

=>The location where the objects are created is known as Heap Area.

(c)Java Stack Area:

=>The location where the methods are executed is known as Java Stack Area.

=>main() is the first method copied onto JavaStackArea to start the execution process.

=>main() method will call remaining methods for execution.

faq:

define Method Frame?

=>The partition of JavaStackArea where the method is copied for execution is known as Method Frame.

=>After method execution completed the Method Frame will be destroyed automatically.

(d)Pc Register Area:

=>Program Counter(PC) Registers will record the status of method execution in JavaStackArea.

=>Every method which is executing in JavaStackArea will have its own Program counter Register and all these PC-Registers are opened in a separate location known as PC-Register Area.

(e)Native method Area:

=>The methods which are declared with 'native' keyword in JavaLib are known as Native methods.

=>These native methods internally having c or c++ code.

=>when these native methods are used in application development, then the ClassLoader SubSytsem will load these native methods onto separate location known as Native method Area.

=>These Native methods are executed using JNI(Java Native method Interface) and this JNI internally uses Native method Libraries.

3.Execution Engine:

=>Execution Engine is an executor or processor of JVM and which starts the execution process from main() method available from JavaStackArea.

=>This execution engine internally having two translators:

(a)Interpreter

(b)JIT(Just-In-Time) Compiler

(a)Interpreter:

=>Interpreter will start the execution process and executes normal instructions.

=>when Interpreter finds Stream instructions or MultiMedia instructions then the execution control is transferred to JIT-Compiler.

(b)JIT(Just-In-Time) Compiler:

=>This JIT-Compiler will execute Stream instructions or MultiMedia instructions like Audio,Video,Image and Animation files.

---

faq:

why we use Interpreter in Execution process?

=>when we have Interpreter in execution process then we can accept the request in the middle of execution process and which is preferable for Server Application development.

Note:

=>when we have interpreter in execution process then we can call Java Lang as Inpterpreted language.

Summary:

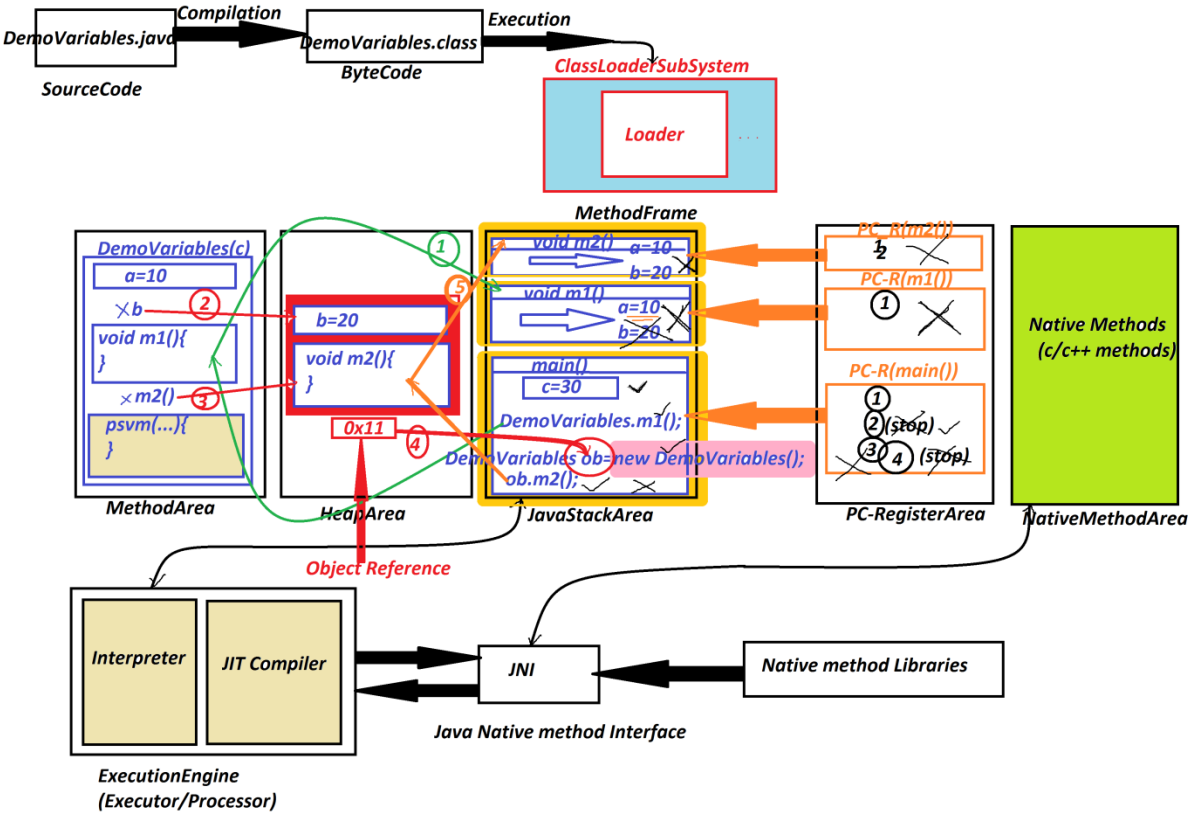
(i)Method Area where class is loaded

(ii)Heap Area where Object is created

(iii)Java Stack Area where methods are executed.

==========

Diagram:



==========