Dt : 30/11/2020

Note:

=>The MainClass is loaded on to method_area first,then the remaining SubClasses will be loaded when they are in need for object creation.

===

faq:

define Method Frame?

=>The partition of Java Stack Area where the method is copied for execution is known as Method frame.

Note:

=>The method Frame will be destroyed automatically when the method execution completed.

--

Note:

=>Method_Area where the classes are loaded and the Heap_area where the objects are created.

---

faq:

define Object?

=>Object is a memory generated part of Heap Area related to a class holding the NonStatic members of the class.

Note:

what is Object?

=>Object is memory.(Storage)

where object is created?(Location)

=>Object is created part of Heap Area

what the object will hold?(Object components)

=>Object will hold the NonStatic members of the class.

=>After Object creation,the object will have the following:

(i)Object state

(ii)Object behaviour

(iii)Object Identification

(i)Object state:

=>The variable part of the object is known as Object State.

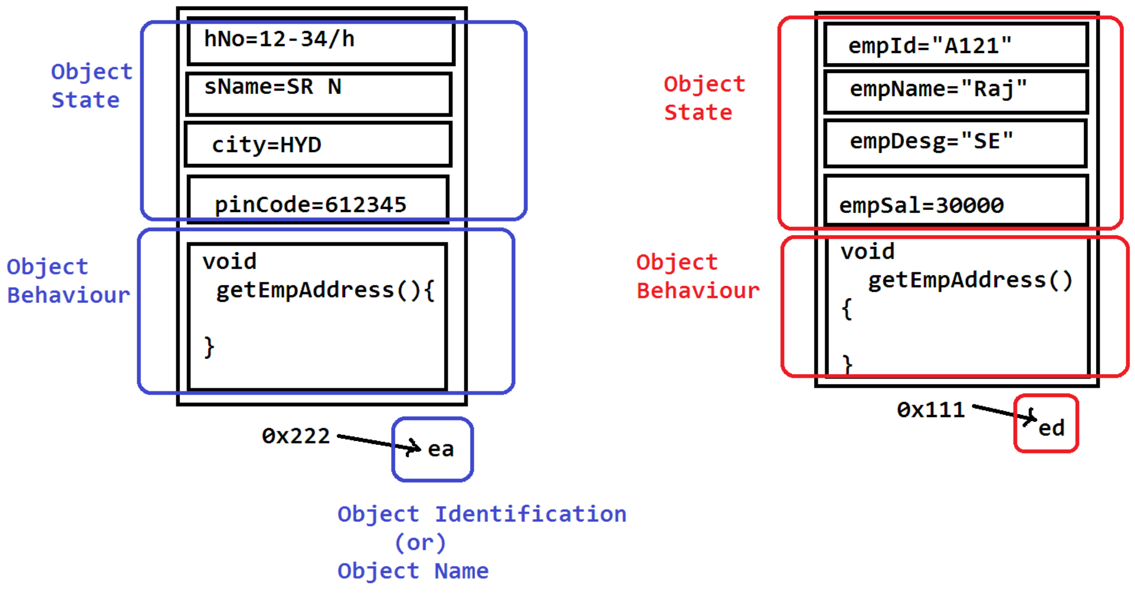
(ii)Object behaviour:

=>The method part of the object is known as Object behaviour.

(iii)Object Identification:

=>The variable which is holding object reference is known as

Object name or Object identication.



(c)Java Stack Area:

=>The memory block where the methods are executed is known Java Stack Area.

=>The main() method is the first method copied on to Java Stack Area and this main() method will call remaining methods for execution.

(d)PC Register Area:

=>Program Counter(PC) register will record the status of method execution in Java Stack Area,in this process every method will have its own Program Counter(PC) register.

=>All these Program Counters are opened in a separate memory block known as PC-Register Area.

=>These PC-Registers will be destroyed automatically when the method_frames are destroyed.

(d)Native Method Area:

=>The methods which are declared with native keyword part of JavaLib are known as Native Methods.

=>Native methods internally having c or c++ code.

=>when these Native methods are used part of application then they are separated and loaded on to separate memory block known as Native method Area.

=>Execution Engine will take the support of JNI(Java Native method Interface) for executing Native methods available in Native method Area.

=>while execution JNI uses Native Method Libraries.

faq:

why Native methods are available part of JavaLib?

=>Using Native methods the JavaApp can interact with the resources available outside the JVM.

3.Execution Engine:

=> Execution Engine is an 'executor' which starts the execution process from main() method available from Java Stack Area.

=>This Execution Engine internally having two translators:

(i)Interpreter

(ii)JIT Compiler

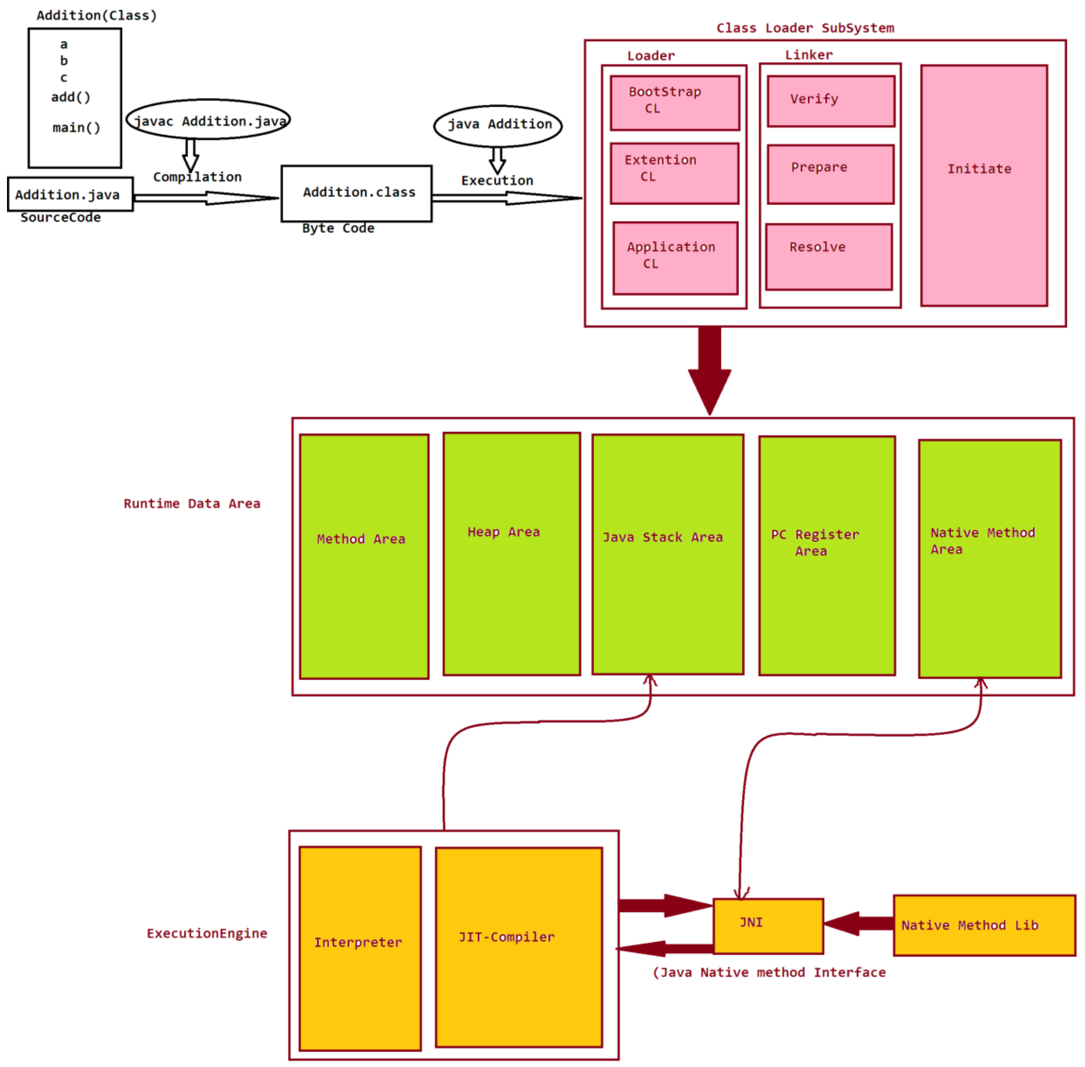
(i)Interpreter:

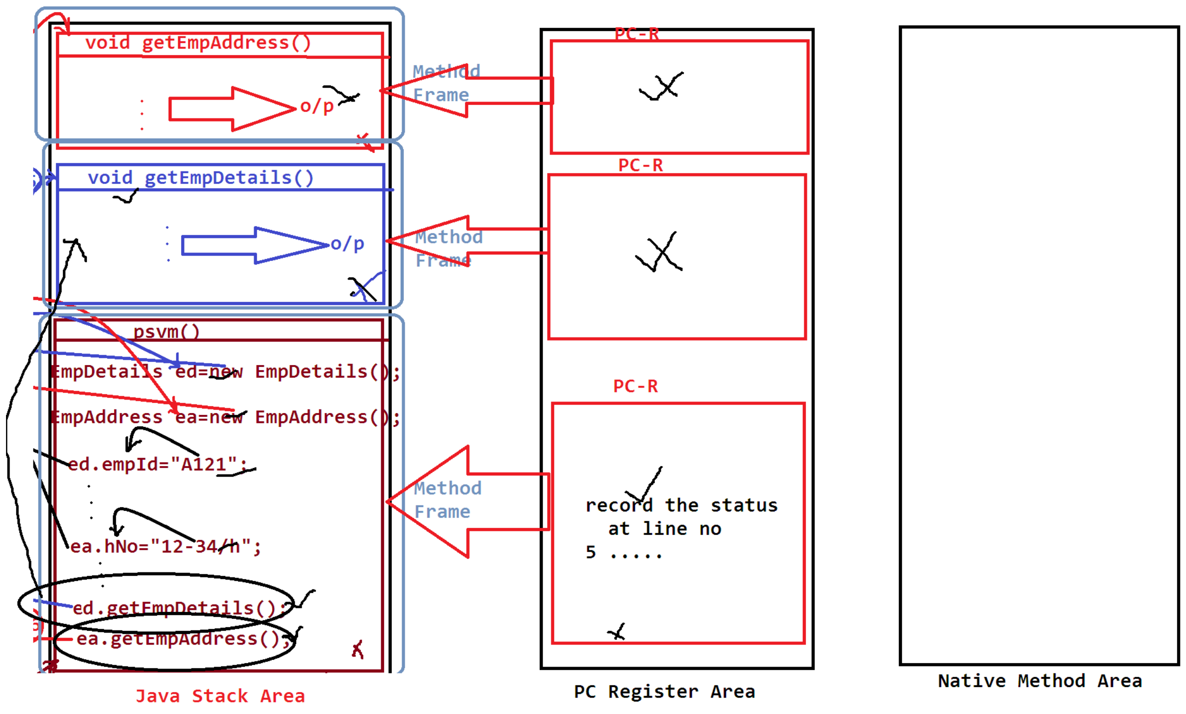
=>Interpreter will start the execution process and execute the Normal Instructions.

=>when Interpreter finds Stream Instructions(Multimedia Instructions) then the control is transferred to the 'JIT(Just-In-Time) Compiler'.

(ii)JIT Compiler:

=>JIT Compiler will execute Stream instructions or Multimedia Instructions.


Addition(Class)
a
b
c
add()
main()
javac Addition.java
Compilation
Addition.java
SourceCode
Addition.class
Byte Code
java Addition
Execution
Class Loader SubSystem
Loader
BootStrap CL
Extention CL
Application CL
Linker
Verify
Prepare
Resolve
Initiate
Runtime Data Area
Method Area
Heap Area
Java Stack Area
PC Register Area
Native Method Area
ExecutionEngine
Interpreter
JIT-Compiler
JNI
(Java Native method Interface
Native Method Lib


void getEmpAddress()
o/p
Method Frame
void getEmpDetails()
o/p
Method Frame
psvm()
EmpDetails ed=new EmpDetails();
EmpAddress ea=new EmpAddress();
ed.empId="A121";
ea.hNo="12-34/h";
ed.getEmpDetails();
ea.getEmpAddress();
Method Frame
Java Stack Area
PC-R
PC-R
PC-R
record the status at line no 5 .....
PC Register Area
Native Method Area