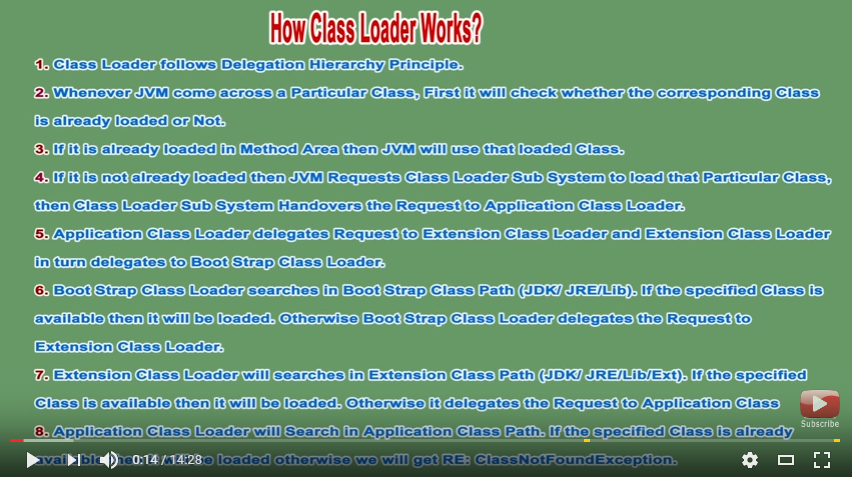
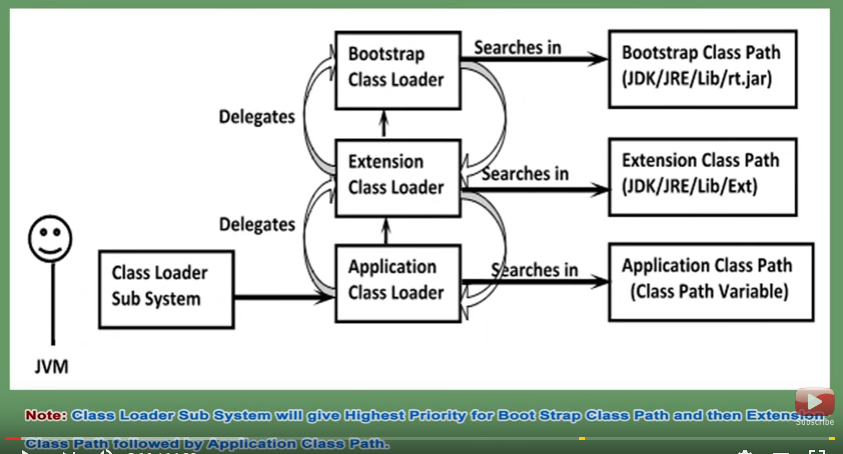
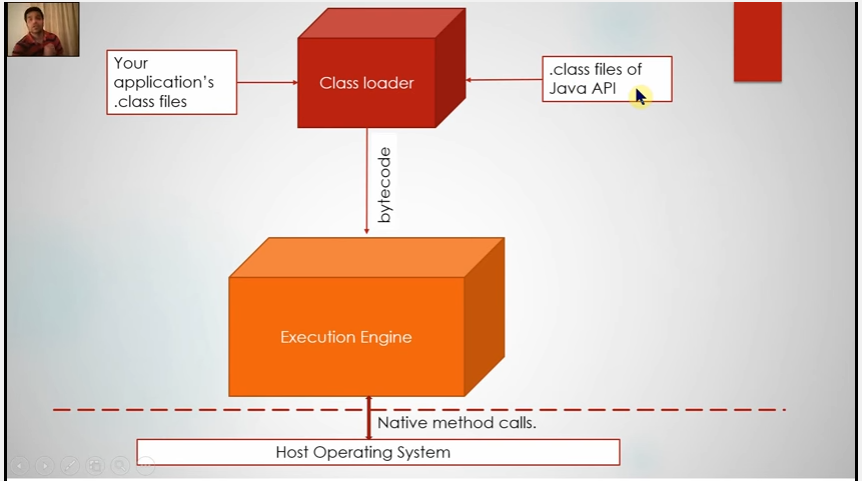
**What class loaders do**

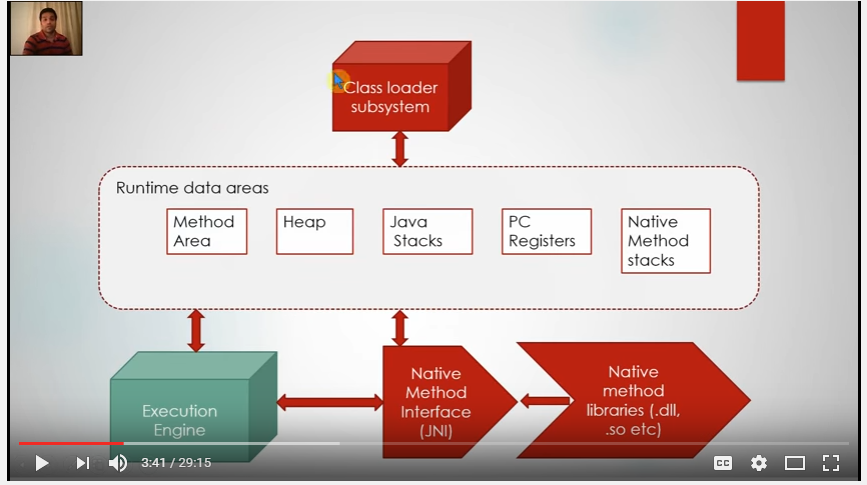
****

****

* Classes are introduced into java environment when they are referenced by name in a class which is already running.
* Class loaders will read all the .class files from jar or web socket

**JVM Architecture:**

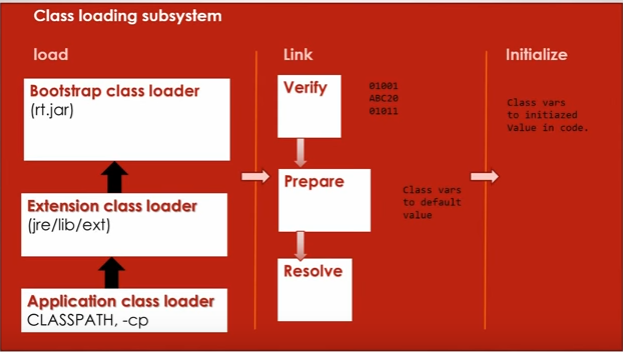
****

****

1. **Class Loaders:**

****

**Inside Class loaders:**

****

**Load Phase:**

Bootstrap class loader : it is responsible for loading java’s internal classes.(inside rt.jar).

Extension : responsible to loading additional class application jar that are present in jre/lib/ext.

Application class Loader : responsible for loading classes mentioned by class-path in env variables.

**Link Phase:**

**Verify:**

Verify bytecode compatibility / valid java bytecode.

**Prepare:**

Memory allocations for static variables are allocated in this phase. Here in this phase JVM will allocate if user has assigned some value. Default value for static variables.

**Resolve:**

All the symbolic references for the current class are resolved. Like references to other class /constant pools .these are changes to symbolic to actual references.

**Initialize:**

* Static initializers are run(static block).
* Actually value for static variable are set where it was set to default value in prepare phase.

Some random questions:

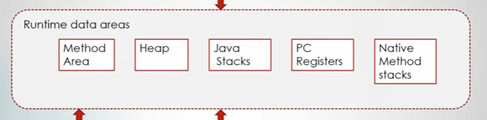
classNotFoundException:

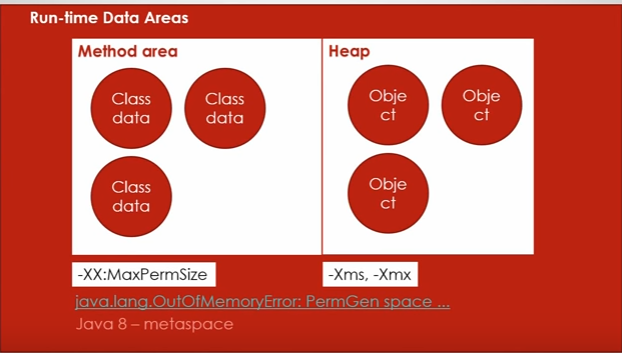
happens when class-loader fails to find bytecode corresponding to class we mentioned.

ClassDefNotFound:

This will occur when a class x is under resolve phase and x is referenced to y, and while searching for class Y if Y reference is not available.

1. **Run Time Data Areas:**





**Method Area (perm gen space) :**

Here Class data are stored like static variables, bytecode and constant pool.

-XX:MaxPermSize is 64 MB, u need to tune this if your application loads millions of classes, user needs to increase size accordingly.

If you don’t tune space your application will throw java.lang.OutOfMemoryError.PermGenSpace.

In Java 8 method area is called as metaSpace - it can grow based on your application(it can be limited).

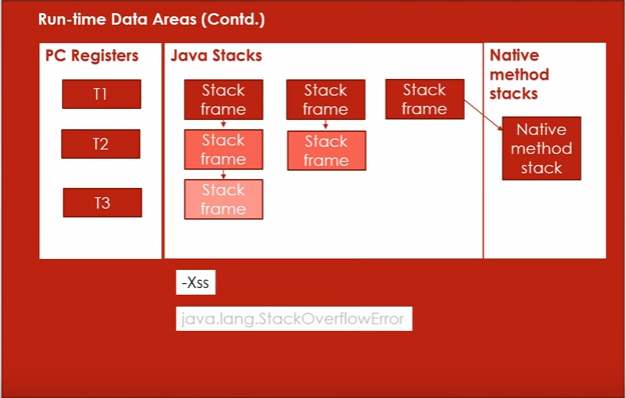
**Heap:**

It is the space where object data is stored.

Heap also needs tobe tuned for any application using parameters –Xms and –Xmx.

**Continued …**

**Below these areas are per thread.**

****

**PC Registers:**

Program counter is the pointer to next instruction to be executed per thread.

**Java Stacks:**

Per thread java stacks are created, for example T1 has 3 stack frames in stack as shown above.

**Native Method Stack:**

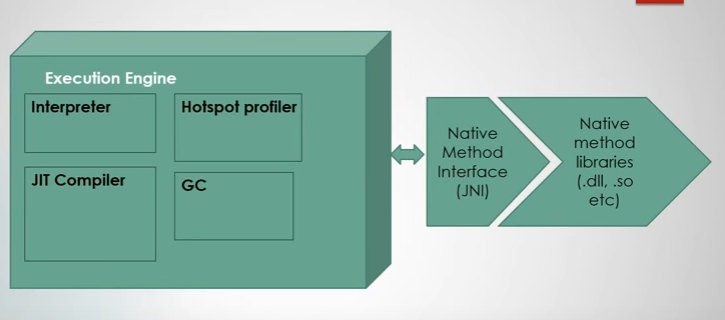
When Thread t1 calls any native method then native stack is created.

Note : Stacks Frames are per method invocation.

Java.lang.StackOverflowError : this exception is thrown when stack is full with stack frames and not popping out, May like be recursive method calling.

-Xss is the parameter to assign stack area size.

1. **Execution Engine**

****

**Interpreter : Takes bytecode instruction and finds what native operations to be done and execute that by using Native Method Interface(JNI).**

**Hotspot Profiler : it keeps on eye on the various bytecode frequently executed and inform JIT compiler to create hotspots, so that no more interpreter is required for these common bytecode/frequent bytecodes.**

**JIT Complier: it is responsible for improve the performance of system by the help of hotspot**