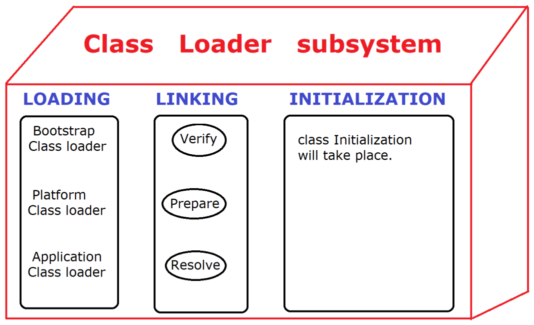
**Class Loader subsystem contains 3 major parts :**
**-----------------------------------------------------------**
**1) LOADING**
**2) LINKING**
**3) INITIALIZATION**



**LOADING :**
**-------------**
* It provides different class loaders to load the required .class file into JVM memory.
* It uses an algorithm **"Delegation Hierarchy Algorithm"** to load the .class files.
* In order to load the .class files into JVM memory, it uses the following class loaders :

**1) Bootstrap OR Primordial Class Loader:**
**--------------------------------------------------**
* It is the **built-in (predefined)** class loader of JVM. It is mainly used to load all the predfined .class files (Java API classe) into JVM memory.
* Among all the class loaders (Bootstrap, Platform and Application), Bootstrap class loader is having highest priority because It is the super class of Platform class loader.

* Bootstrap class loader loads all the predefined .class files from the following path :
  C:\\Program files\java\JDK 25\lib \ jrt-fs.jar

**2) Extension OR Platform Class Loader :**
**-------------------------------------------------**
Before java 9V, this class loader was known as **Extension class loader** because It was loading the .class file from the following path.

C -> Program files -> Java -> JDK -> lib -> ext (extension) -> 3rd Party jar file. [ojdbc17.jar is required to connect java application to Oracle Database]

It has highest priority than Application class loader.

From Java 9V onwards we have a new concept called **JPMS** (Java Platform Module System) so, instead of using jar file we can directly use module. [module is internally using jar file concept only]

Example : java.base, java.compiler and so on.

Command to create the jar file:

jar cf FileName.jar FileName.class        [*.class]  Example : jar cf NIT.jar Alpha.class    [OR    *.class]

[If we want to compile more than one java source file at a time then the command is :  javac *.java]

**Application OR System class loader :**
**--------------------------------------------**
It is responsible to load all user defined .class file into JVM memory.

It has the lowest priority because it is the sub class Platform class loader.

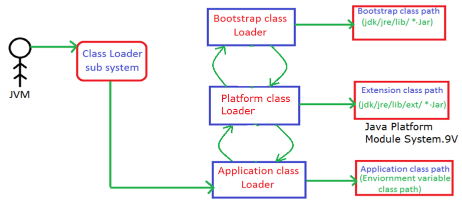
It loads the .class file from class path level or environment variable.

Note :-
------
If all the class loaders are failed to load the .class file into JVM memory then we will get a Runtime exception i.e **java.lang.ClassNotFoundException.**
---------------------------------------------------------------------------------------------------------------
**How Delegation Hierarchy algorithm works internally ?**
**------------------------------------------------------------**
Whenever JVM makes a request to class loader subsystem to load the required .class file into JVM memory, first of all, class loader subsystem makes a request to Application class loader, Application class loader will **delegate** (by pass) the request to the Platform class loader, Platform class loader will also delegate the request to Bootstrap class loader.

Bootstrap class loader will load the .class file from lib folder(jrt-fs.jar) and then by pass the request back to Platform class loader, Platform class loader will load the .class file from ext folder(*.jar [3rd party jar file]) OR JPMS (Java Platform Module System) and by pass the request back to Application class loader, It will load the .class file from environment variable into JVM memory.

Note : java.lang.Object is the first class to be loaded into JVM Memory.

The class loader sub system follows Delegation Hierarchy Algorithm


JVM
Class Loader sub system
Bootstrap class Loader
Platform class Loader
Application class Loader
Bootstrap class path (jdk/jre/lib/ *.Jar)
Extension class path (jdk/jre/lib/ext/ *.Jar)
Java Platform Module System.9V
Application class path (Enviornment variable class path)