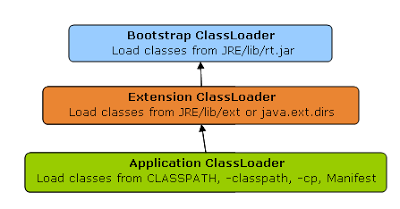
# Java Class Loader

Java Class Loader是Java虚拟机重要组成部分，它们用来载入Java类和接口。一般情况下，Java开发者并不需要定制一个Class Loader。

Class Loader分为三类[1]。如图1所示：



1. ClassLoader类型
2. **Bootstrap Class Loader** – 装载JDK内部类，如rt.jar和其他核心类java.lang.\*包类。
3. **扩展Class Loader** – 装载JDK扩展类, 通常是JRE的lib/ext文件夹。Java 9之后，**扩展Class Loader已经被重新命名为Platform Class Loader替换**[3][4]。
4. **系统（应用）Class Loader** –从系统类路径装载类, 可以在启动程序时通过-cp或者-classpath设置。

Java 9之后，具体涉及Class Loader的改动可参考[5]。

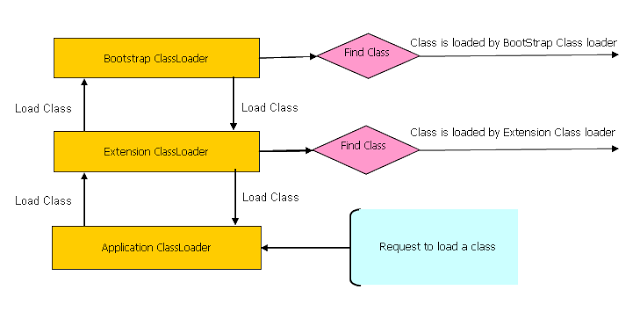
类装载的时机包括两种情况：

1. **当执行新的字节码时；**
2. **当字节码静态引用类时。**

Class Loader执行装载时遵从如下规则，如图2：

1. **监测类是否已经装载；**
2. **如果没有状态，请求父Class Loader装载；**
3. **如果父Class Loader不能装载，尝试使用当前Class Loader。**

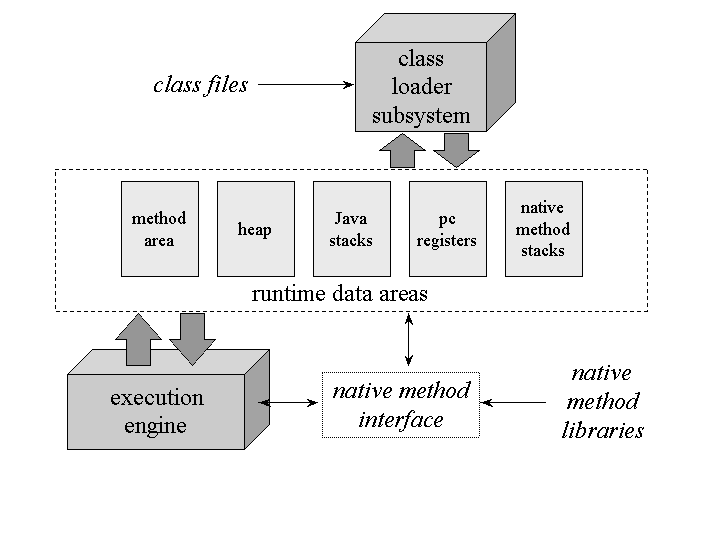
类装载分为静态和动态方式，new操作符是静态状态，动态装载指使用Class.forName()调用class loader方法实现的装载技术。**loadClass方法装载类但不进行初始化，而Class.forName在装载后会初始化Class对象（如其中的静态变量）[2]**。



1. 装载类步骤

具体装载步骤如下所示：

* **Loading**: 根据具体名字查找、导入类二进制数据，由二进制数据创建一个类或者接口。
* **Linking**: 执行验证、准备、和解析（可选）。
* **Verification**: 保证导入类型的正确性。
* **Preparation**: 为类变量分配内存并初始化为默认对象。
* **Resolution**: 将符号引用由类型转换为直接引用。
* **Initialization**: 触发Java代码将类变量初始化为适当的起始值。



1. JVM虚拟机架构

前面介绍了Class Loader工作机制，在Class Loader将class文件装载到内存后，后续的工作交由执行引擎执行对应的代码。图3是Java虚拟机架构。我们后续会继续分析执行引擎如何工作。

1. <https://javatutorial.net/java-class-loaders-explained>
2. <https://javabeat.net/class-forname-classloader-loadclass-difference/>
3. <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/ClassLoader.html>
4. <https://stackoverflow.com/questions/46494112/classloaders-hierarchy-in-java-9?noredirect=1&lq=1>
5. <https://docs.oracle.com/javase/9/migrate/toc.htm#JSMIG-GUID-A78CC891-701D-4549-AA4E-B8DD90228B4B>