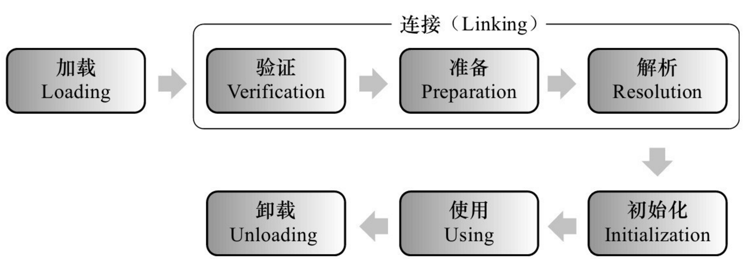
类加载流程：



# **加载**

根据一个类的全限定名(如cn.edu.hdu.test.HelloWorld.class)来读取此类的二进制字节流到JVM内部;

将二进制字节流所代表的静态存储结构转换为方法区的运行时数据结构

在内存中生成一个与目标类型对应的java.lang.Class对象；作为访问入口

**验证**——保证加载的类定义二进制字节流符合虚拟机的要求，不会危害虚拟机

1、文件格式验证:是否符合class文件格式规范

2、元数据验证：语义分析，保证没有语法错误

3、字节码验证

4、符号引用验证

注：验证阶段重要，但不是必须的。可通过阐述-Xverifynone 来关闭验证。

**准备**

正式为类变量（static）分配内存（栈内存），并设置默认值（类型的默认值）。类变量的内存都分配在方法区中——栈

1、实例变量分配在堆中，在对象实例化时分配

2、类变量在（类）初始化阶段才会执行

3、final的static变量是在准备阶段赋值的。

**解析**

将常量池中的符号引用替换为直接引用。

7类符号引用：类、接口、字段、类方法、接口方法、方法类型、方法句柄、调用点限定符。解析主要针对这7类进行。

**类初始化**

真正开始执行类中定义的java程序。初始化阶段是执行类构造器<clinit>()方法的过程。

类构造器<clinit>()方法

类构造器<clinit>()方法 由 静态变量赋值+静态语句块 构成。<clinit>()是由编译器生成的。

实例构造器<init>()

父类<clinit>()——》子类<clinit>()——》<init>()

父类中定义的静态语句块要优先于子类的变量赋值操作。

<clinit>()方法对于类或者接口来说并不是必需的，如果一个类中没有静态语句块，也没有对变量的赋值操作，那么编译器可以不为这个类生产<clinit>()方法。

接口中不能使用静态语句块，但仍然有变量初始化的赋值操作，因此接口与类一样都会生成<clinit>()方法。但接口与类不同的是，执行接口的<clinit>()方法不需要先执行父接口的<clinit>()方法。只有当父接口中定义的变量使用时，父接口才会初始化。另外，接口的实现类在初始化时也一样不会执行接口的<clinit>()方法。

虚拟机会保证一个类的<clinit>()方法在多线程环境中被正确的加锁、同步，如果多个线程同时去初始化一个类，那么只会有一个线程去执行这个类的<clinit>()方法，其他线程都需要阻塞等待，直到活动线程执行<clinit>()方法完毕。如果在一个类的<clinit>()方法中有耗时很长的操作，就可能造成多个线程阻塞，在实际应用中这种阻塞往往是隐藏的

虚拟机会保证<clinit>()只有一个线程执行，且只会执行一次。

同一个类加载器下，一个类只会被初始化一次

虚拟机规范严格规定了有且只有5中情况）必须对类进行“初始化”：

1. 遇到new,getstatic,putstatic,invokestatic这4个字节码指令时，如果类没有进行过初始化，则需要先触发其初始化。生成这4条指令的最常见的Java代码场景是：使用new关键字实例化对象的时候、读取或设置一个类的静态字段（被final修饰、已在编译器把结果放入常量池的静态字段除外）的时候，以及调用一个类的静态方法的时候。
2. 使用java.lang.reflect包的方法对类进行反射调用的时候，如果类没有进行过初始化，则需要先触发其初始化。
3. 当初始化一个类的时候，如果发现其父类还没有进行过初始化，则需要先触发其父类的初始化。
4. 当虚拟机启动时，用户需要指定一个要执行的主类（包含main()方法的那个类），虚拟机会先初始化这个主类。
5. 当使用jdk1.7动态语言支持时，如果一个java.lang.invoke.MethodHandle实例最后的解析结果REF\_getstatic,REF\_putstatic,REF\_invokeStatic的方法句柄，并且这个方法句柄所对应的类没有进行初始化，则需要先出触发其初始化。

不触发初始化的情况：

1. 类数组定义

Student[] students = new Student[10];

1. 常量在编译阶段会存入调用类的常量池中。使用常量，本质上没有直接引用到定义常量的类，因此不会触发定义常量的类初始化。

实例构造器<init>() ——1、先初始化成员变量。2、非静态语句块。3、构造方法

类构造器<clinit>() ——1、静态变量。2、静态语句块

# **类加载器种类**

启动类加载器，Bootstrap ClassLoader，加载JAVA\_HOME\lib，或者被-Xbootclasspath参数限定的类  
扩展类加载器，Extension ClassLoader，加载\lib\ext，或者被java.ext.dirs系统变量指定的类  
应用程序类加载器，Application ClassLoader，加载ClassPath中的类库  
自定义类加载器，通过继承ClassLoader实现，一般是加载我们的自定义类

双亲委派

所谓双亲委派是指每次收到类加载请求时，先将请求委派给父类加载器完成（所有加载请求最终会委派到顶层的Bootstrap ClassLoader加载器中），如果父类加载器无法完成这个加载（该加载器的搜索范围中没有找到对应的类），子类尝试自己加载。