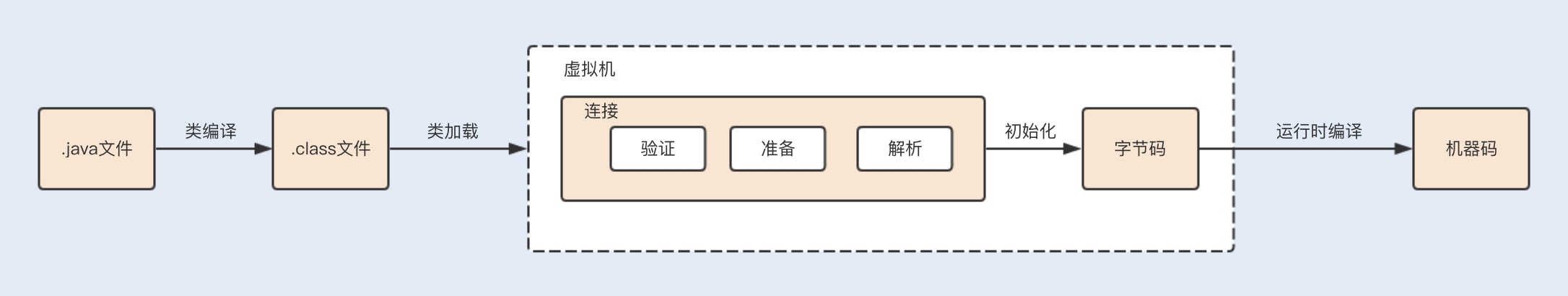
# Java类编译加载执行过程

东方奇骥

1. Java类编译加载总体示意图



1. .java 文件被编译成 .class 文件的过程，这个编译我们一般称为**前端编译**。Java 的编译和运行过程非常复杂，除了前端编译，还有运行时编译。由于机器无法直接运行 Java 生成的字节码，所以在运行时，JIT 或解释器会将字节码转换成机器码，这个过程就叫**运行时编译**。
2. 编译后的.class字节码文件主要包括常量池和方法表集合这两部分。

常量池主要记录的是类文件中出现的字面量以及符号引用。**字面常量包括字符串常量（例如 String str=“abc”，其中"abc"就是常量），声明为 final 的属性以及一些基本类型（例如，范围在 -127-128 之间的整型）的属性**。

**符号引用**包括类和接口的全限定名、类引用、方法引用以及成员变量引用（例如 String str=“abc”，其中 str 就是成员变量引用）等。

**方法表集合**中主要包含一些方法的字节码、方法访问权限（public、protect、prviate 等）、方法名索引（与常量池中的方法引用对应）、描述符索引、JVM 执行指令以及属性集合等。

1. 类加载

在加载类的时候，JVM 会先加载 .class 文件，通过类加载器将字节码文件加载到内存中。而在 **.class 文件中除了有类的版本、字段、方法和接口等描述信息外，还有一项信息是常量池 (Constant Pool Table)**，**用于存放编译期间生成的各种字面量和符号引用**。

在类加载后， .class类文件中的常量池信息以及类信息会被保存到 JVM 内存的方法区中。

1. 链接

1）验证：验证类符合 Java 规范和 JVM 规范，在保证符合规范的前提下，避免危害虚拟机安全。

2）准备：为类的静态变量分配内存，初始化为系统的初始值。对于 final static 修饰的变量，直接赋值为用户的定义值。例如，private final static int value=123，会在准备阶段分配内存，并初始化值为 123，而如果是 private static int value=123，这个阶段 value 的值仍然为 0。

3）解析：将符号引用转为直接引用的过程。在编译时，Java 类并不知道所引用的类的实际地址，因此只能使用符号引用来代替。类结构文件的常量池中存储了符号引用，包括类和接口的全限定名、类引用、方法引用以及成员变量引用等。如果要使用这些类和方法，就需要把它们转化为 JVM 可以直接获取的内存地址或指针，即直接引用。

1. 初始化

类初始化阶段是类加载过程的最后阶段，在这个阶段中，JVM 首先将执行构造器 <clinit> 方法，编译器会在将 .java 文件编译成 .class 文件时，收集所有类初始化代码，包括静态变量赋值语句、静态代码块、静态方法，收集在一起成为 <clinit>() 方法。

JVM 会保证 <clinit>() 方法的线程安全，保证同一时间只有一个线程执行。

这也是为什么private static final Singletone instance = new Singletone()

能实现单例模式的原因。

JVM 在初始化执行代码时，如果实例化一个新对象，会调用 <init> 方法对实例变量进行初始化，并执行对应的构造方法内的代码。

1. 即时编译

初始化完成后，类在调用执行过程中，执行引擎会把字节码转为机器码，然后在操作系统中才能执行。在字节码转换为机器码的过程中，虚拟机中还存在着一道编译，那就是即时编译。