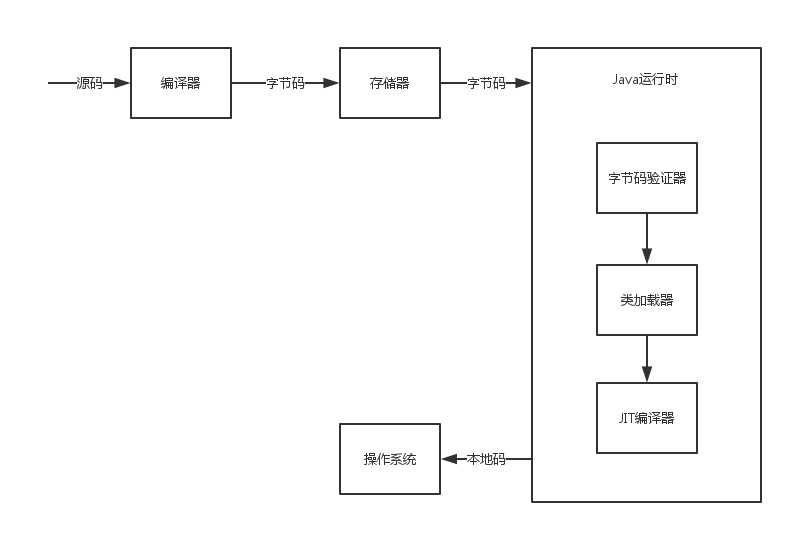
**Java Compilation Process**

Java是半解释型的语言。Java应用程序在JVM中执行。Java程序的执行不像纯解释型语言一样有一个中间的编译步骤。Java不会编译成操作系统在CPU上执行的本地码，而是一种介于两者之间运行在虚拟机上的字节码。让我们来看看源码转换为可执行代码以及程序如何执行。

**Figure 2.2. The Java Compile/Execute Path**



Java需要把每一个类放在它自己的源文件中（以类名和.java后缀组成）。这个基本的要求使得中型程序被分开在多个分开的源码文件中。当编译源码时，每一个类保存在它自己的文件中。类文件包含了这些字节码。Java编译器跟gcc/c++不同，事实上编译器编译某个类时，它所依赖的其他类将不被同时编译或者修改。这跟make类似，但不是完全类似。在编译好了所有的源文件之后，编译后的类文件的数目会跟源文件数目一样，这些类文件组成了java程序。类加载器会连同字节码验证器，这两个唯一的步骤区分了Java和类C/C++语言。

类加载器负责加载每一个类的字节码。Java给开发者提供了自由去编写自己的类加载器。类加载器可以加载任意位置的类，甚至是网络或者文档。N下面介绍类加载器如何加载一个类。

当JVM需要某个类时，loadClass(String name, boolean resolve);将被调用并传入类名。一旦找到包含这个类的字节码的文件，这个文件将读入类存并传递给defineClass 。如果类加载器没有找到这个类，它将委托父类加载器或者使用findSystemClass 从本地文件系统去加载。Java虚拟机规范对于什么时候进行以及如何进行字节码验证，但是通过简单的测试可以推断出defineClass完成了字节码的验证。验证器通过了4步验证法来验证字节码是否安全。在类验证和加载完成之后，就能够被运行时调用了。

原生的Java字节码允许用户很方便的反编译回源码。对于默认的编译行为，变量和方法都是可以被恢复的。用很多反编译工具，但是免费的做的好的是Jad。