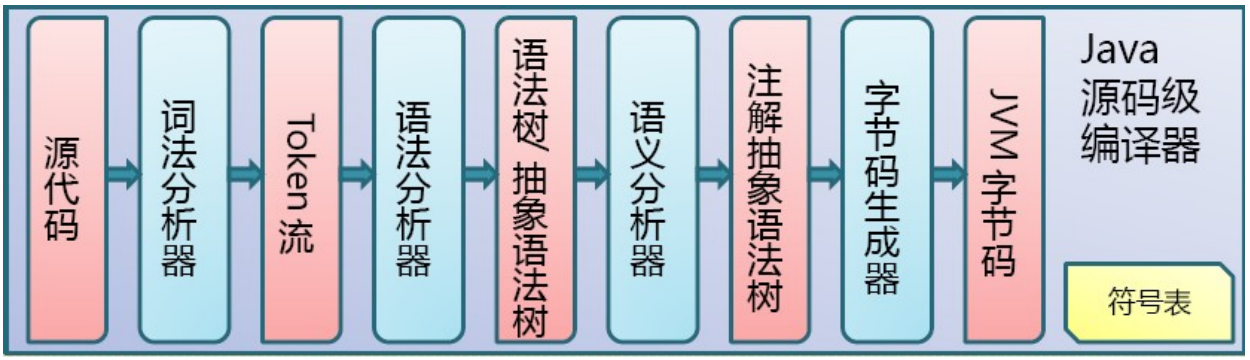
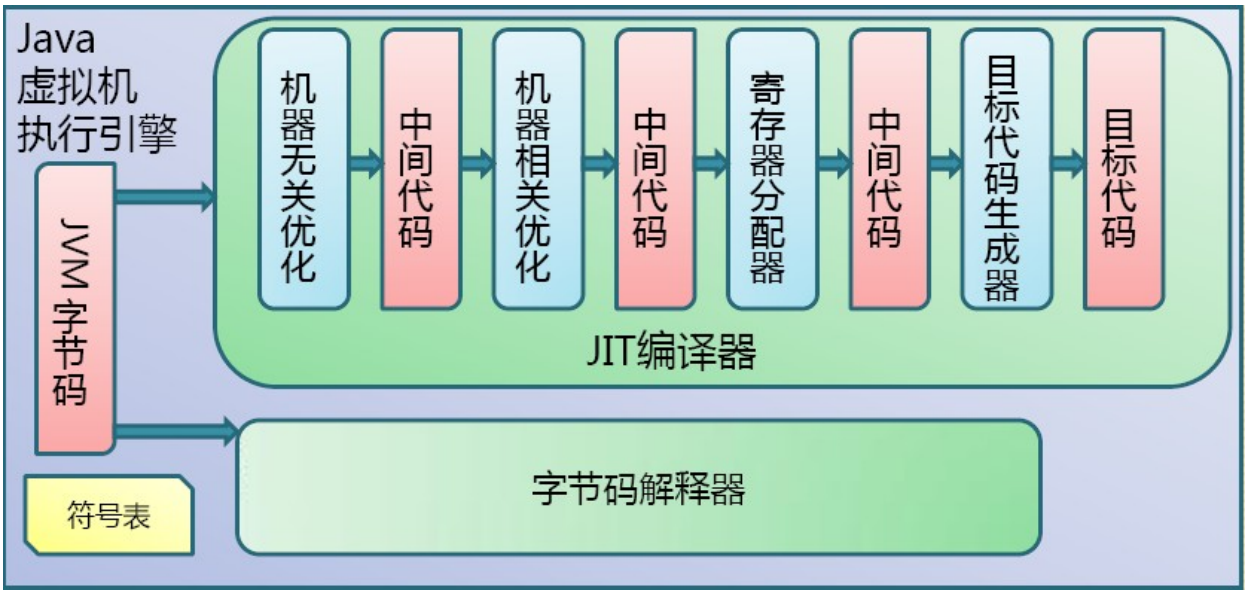
编译与运行：

编译是由Java源码编译器来完成。**编译过程**是将java源文件编译成字节码（jvm可执行代码，即.class文件）的过程，在这个过程中不与内存打交道的，在这个过程中编译器会进行语法的分析，如果语法不正确就会报错。ps. JVM 在编译阶段会把引用常量的代码替换成具体的常量值。



字节码的执行是由JVM执行引擎来完成。**运行过程**是指jvm装载字节码文件并解释执行（转化为机器码）。这个过程才创立内存布局（xf.应该在类加载的准备阶段），执行java程序。



java字节码的执行有两种方式：

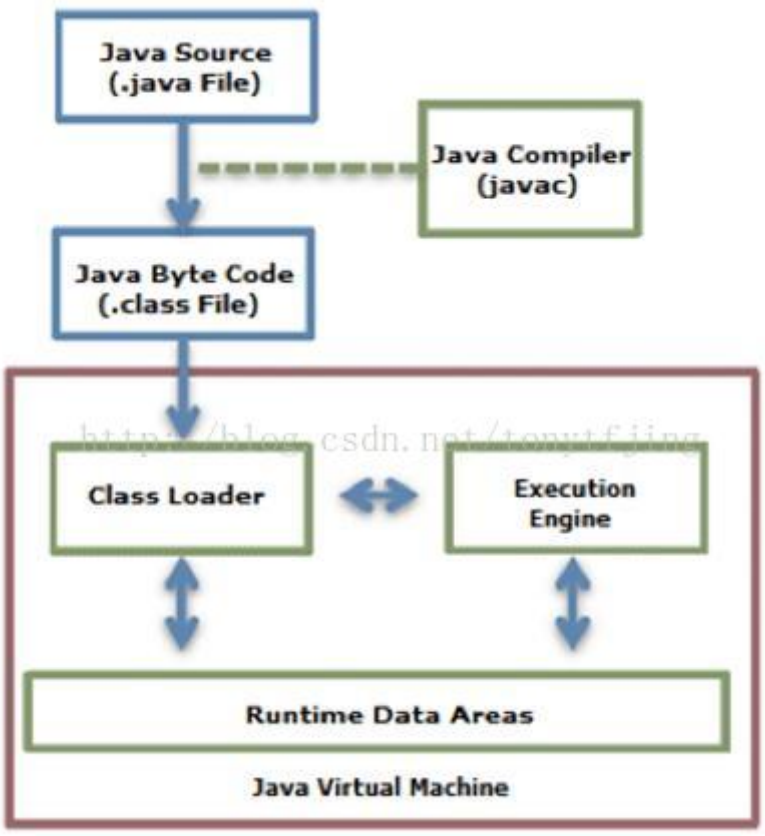
（1）即时编译方式（运行时编译）：编译器先将字节编译成机器码，然后再执行该机器码；

（2）解释执行方式：解释器通过每次解释并执行一小段代码来完成java字节码程序的所有操作。

编译与解释共存：

即时编译器（JIT compiler，just-in-time compiler）是一个把Java的字节码转换成可以直接发送给处理器的指令（机器码）的程序。通过解释器逐行解释执行的方式比较慢，而且有些方法和代码块被重复调用（即热点代码），便引进JIT编译器，属于运行时编译。当JIT编译器完成第一次编译时，会将字节码对应的机器码保存下来（以method为翻译单位，主要编译重复性代码），下次便直接使用。故java编译和解释共存。

ps.JDK9引进AOT。RTSJ，继javac之后执行AOT二次编译，生成静态的目标平台码。3需要程序员手工指定。



Java代码编译和执行的整个过程包含了以下三个重要的机制：

- **Java源码编译机制**

源码编译由以下三个过程组成：分析和输入到符号表，注解处理，语义分析和生成class文件。

最后生成的class文件由以下部分组成：

- 结构信息：包括class文件格式版本号及各部分的数量与大小的信息
- 元数据　：对应于Java源码中声明与**常量的信息**。包含类/继承的超类/实现的接口的**声明信息、域**与**方法声明信息和常量池。**
- 方法信息：对应Java源码中语句和表达式对应的信息。包含字节码、异常处理器表、求值栈与局部变量区大小、求值栈的类型记录、调试符号信息。

- **类加载机制**

类加载是通过ClassLoader及其子类来完成的。

- **类执行机制**

JVM是基于栈的体系结构来执行class字节码的。线程创建后，会产生对应的程序计数器（PC）和栈（Stack），程序计数器存放下一条要执行的指令在方法内的偏移量，栈中存放一个个栈帧，每个栈帧对应着每个方法的每次调用。

局部变量区用于存放方法中的局部变量和参数；操作数栈中用于存放方法执行过程中产生的中间结果。