**早期（编译器）优化**

**概述**

Java语言的“编译期”是一段不确定的操作过程，可能会有三种不同的操作

(1)可能是指前端编译器把\*.java文件转变成\*.class文件的过程

(2)也可能是指虚拟机的后端运行编译器(JIT编译器)把字节码转变成机器码的过程

(3)还可能是指使用静态提前编译器(AOT编译器)直接把\*.java文件编译成本地机器代码的过程

虚拟机设计团队把对性能的优化集中到了后端的即时编译器(JIT编译器)中，这样可以让那些不是由Javac产生的Class文件也同样能享受到编译器所带来的好处。

但是Javac做了许多针对Java语言编码过程的优化措施来改善程序员的编码风格和提高编码效率，相当多的新生的Java语法特性，都是靠编译器的“语法糖“来实现，而不是依赖于虚拟机的底层改进来支持。Java中即时编译器在运行期的优化过程对于程序运行来说更重要，而前端编译器在编译期的优化过程对于程序编码来说关系更加密切。

**Javac编译器**

该编译器本身就是由java语言编写的程序。Javac编译器编译过程可以分为以下三个步骤：

1.解析和填充符号表过程

2.插入式注解处理器的注解处理过程

3.分析与字节码生成

**1.解析和填充字节码**

解析步骤包括了经典程序编译原理中的词法分析和语法分析两个过程

**(1)词法、语法分析**

词法分析是将源代码的字符流转变为标记(Token)集合，单个字符是程序编写过程的最小元素，而标记则是编译过程的最小元素，关键字、变量名、字面量、运算符都可以标记。例如：

”Int a=b+2”这句代码包含了6个标记，分别是int、a、=、b、+、2，int不可再拆分。

语法分析是根据Token序列构造出抽象语法树的过程，抽象语法树是一种用来描述程序代码结构的树形表示方式，语法树的每一个节点都代表着程序代码中的一个语法结构，例如包、类型、修饰符、运算符甚至代码注释都可以是一个语法结构

**(2)填充符号表**

完成了语法分析和词法分析之后，下一步就是填充符号表的过程。符号表是由一组符号地址和符号信息构成的表格，可以将它想象成哈希表中K-V值对的形式

**2.注解处理器**

可以将注解处理器看做是一组编译器的插件，在这些插件里面，可以读取、修改、添加抽象语法树中的任意元素。如果这些插件在处理注解期间对语法树进行了修改，编译器将回到解析及填充符号表的过程重新处理，直到所有插入式注解处理器度没有对语法树进行修改为止，每一次循环称为一个Round

**3.语义分析与字节码生成**

语法分析之后，编译器获得了程序代码的抽象语法树表示，语法树能表示一个结构正确的源程序的抽象，但无法保证源程序是符合逻辑的。语义分析的主要任务是对结构上正确的源程序进行上下文有关性质的审查，如进行类型审查。语义分析过程分为标注检查和数据及控制流分析

**(1)标注检查**

标注检查步骤检查的内容包括诸如变量使用前是否已经被声明、变量与赋值之间的数据类型是否能够匹配。在标注检查步骤中，有一个重要的动作称为常量折叠，比如有下列语句：

int a=1+2;在语法树中依然可以看到字面量1,2以及操作符，但是经过常量折叠之后，该语句将会被折叠成为字面量3.  
**(2)数据及控制流分析**

数据及控制流分析是对程序上下文逻辑更进一步的验证，它可以检查出诸如局部变量在使用前是否有赋值、方法的每条路径是否都有返回值、是否所有的受查问题都被正常处理了等问题。

**(3)解语法糖**

语法糖，也称糖衣语法，指的是计算机语言中添加的某种语法，这种语法对语言的功能并没有影响，但是更方便程序员使用，使用语法可以增强程序的可读性，从而减少程序代码出错的可能性。

Java中最常用的语法糖主要包括泛型、变长参数、自动装箱、自动拆箱等，虚拟机在运行时并不支持这些语法，它们只是在编译阶段会被还原到简单的基础语法结构，该过程称为解语法糖。

**(4)字节码生成**

字节码生成阶段将各个步骤生成的信息(语法树、符号表)转化成字节码写到磁盘中，并且编译器还进行了少量的代码添加和转换工作。