**第8章 虚拟机字节码执行引擎**

字节码执行引擎是Java虚拟机最核心的组成部分之一。在Java虚拟机规范中制定了虚拟机字节码执行引擎的**概念模型**，这个概念模型成为**各种虚拟机执行引擎的统一外观（Facade）**。

在不同虚拟机实现里面，执行引擎在执行Java代码的时候可能会有解释执行（通过解释器执行）和编译执行（通过即时编译器产生本地代码执行）两种选择，也可能两者兼备，甚至还可能会包含几个不同级别的编译器执行引擎。但从外观上看起来，所有的Java虚拟机的执行引擎都是一致的：输入的是字节码文件，处理过程是字节码解析的等效过程，输出的是执行结果。本章主要从概念模型的角度来讲解虚拟机的**方法调用**和**字节码执行**。

运行时栈帧结构

**栈帧（Stack Frame）**是用于支持虚拟机进行**方法调用**和**方法执行**的数据结构，它是虚拟机运行时数据区中的虚拟机栈（VM Stack）的栈元素。每一个栈帧存储了每一个方法的**局部变量表**、**操作数栈**、**动态连接**和**方法返回地址**等信息。

每一个**方法从调用开始至执行完成**的过程，都对应着一个**栈帧**在虚拟机栈里从**入栈到出栈**的过程。

在编译程序代码时，每一个栈帧（方法）中需要多大的局部变量表、多深的操作数栈都已经完全确定，并写入到对应方法表的Code属性中。因此一个栈帧需要分配多少内存，不会受到程序运行期变量数据的影响，而仅仅取决于具体的虚拟机实现。

一个线程中的方法调用链可能会很长，所以会有很多方法都同时处于执行状态。而对于执行引擎来说，在活动线程中，只有位于栈顶的栈帧才是有效的，称为**当前栈帧**（Current Stack Frame），与这个栈帧相关联的方法称为**当前方法**（Current Method）。执行引擎运行的所有字节码指令都只针对当前栈帧进行操作。概念模型的栈帧结构如下所示：

活动线程（当前线程） 线程n 线程n-1

**当前栈帧**

局部变量表

Local Variable Table

**……**

操作数栈

Operand Stack

动态连接

Dynamic Linking

返回地址

Return Address

**……**

**……**

**栈帧2**

**栈帧1**

栈帧的概念结构