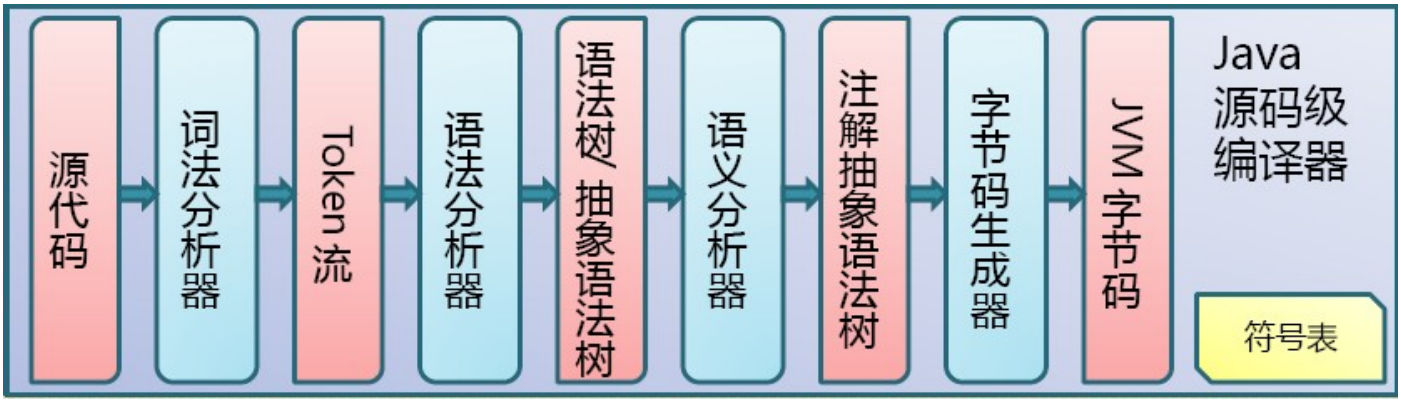
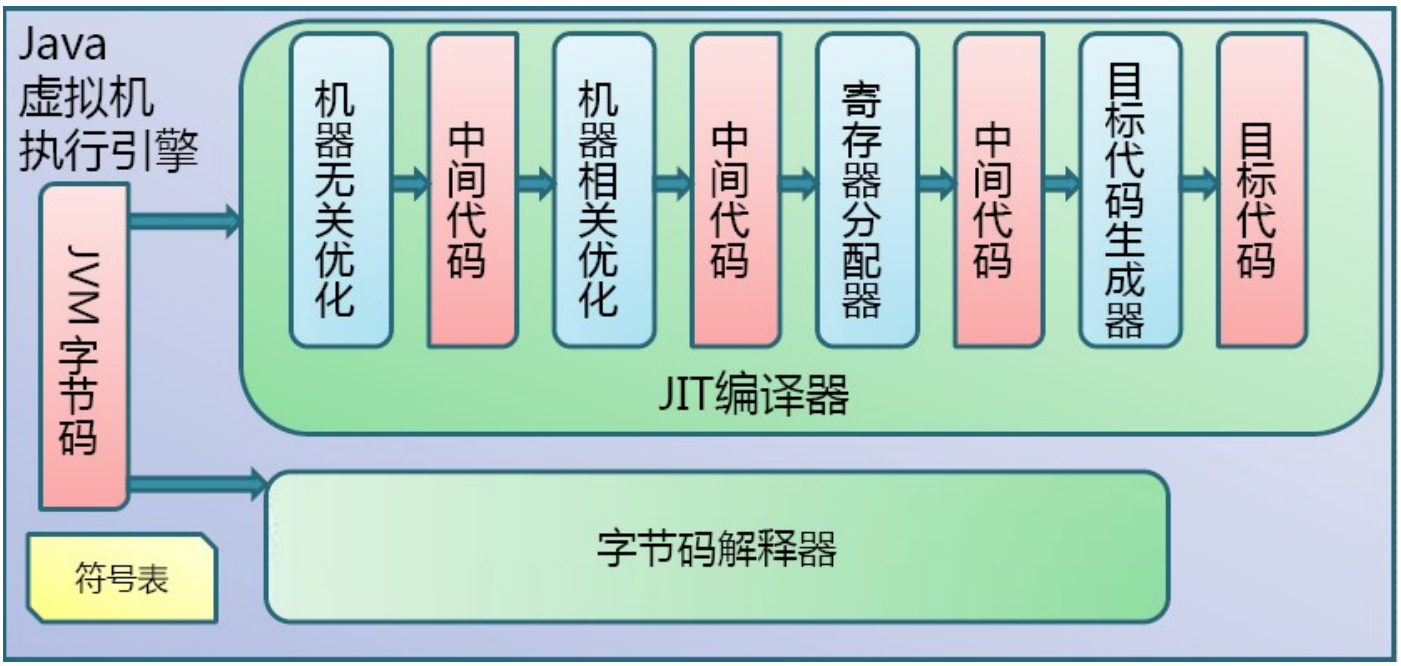
Java 代碼編譯和執行的整個過程

wiki.jikexueyuan.com/project/java-vm/java-debug.html

Java 代碼編譯是由 Java 源碼編譯器來完成，流程圖如下所示：



Java 字節碼的執行是由 JVM 執行引擎來完成，流程圖如下所示：



Java 代碼編譯和執行的整個過程包含了以下三個重要的機制：

- Java 源碼編譯機制
- 類加載機制
- 類執行機制

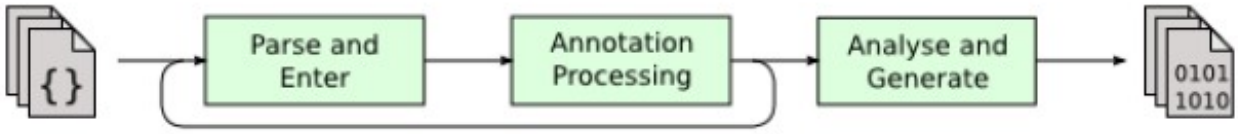
Java 源碼編譯機制

Java 源碼編譯由以下三個過程組成：

- 分析和輸入到符號表
- 註解處理

- 語義分析和生成 class 文件

流程圖如下所示：

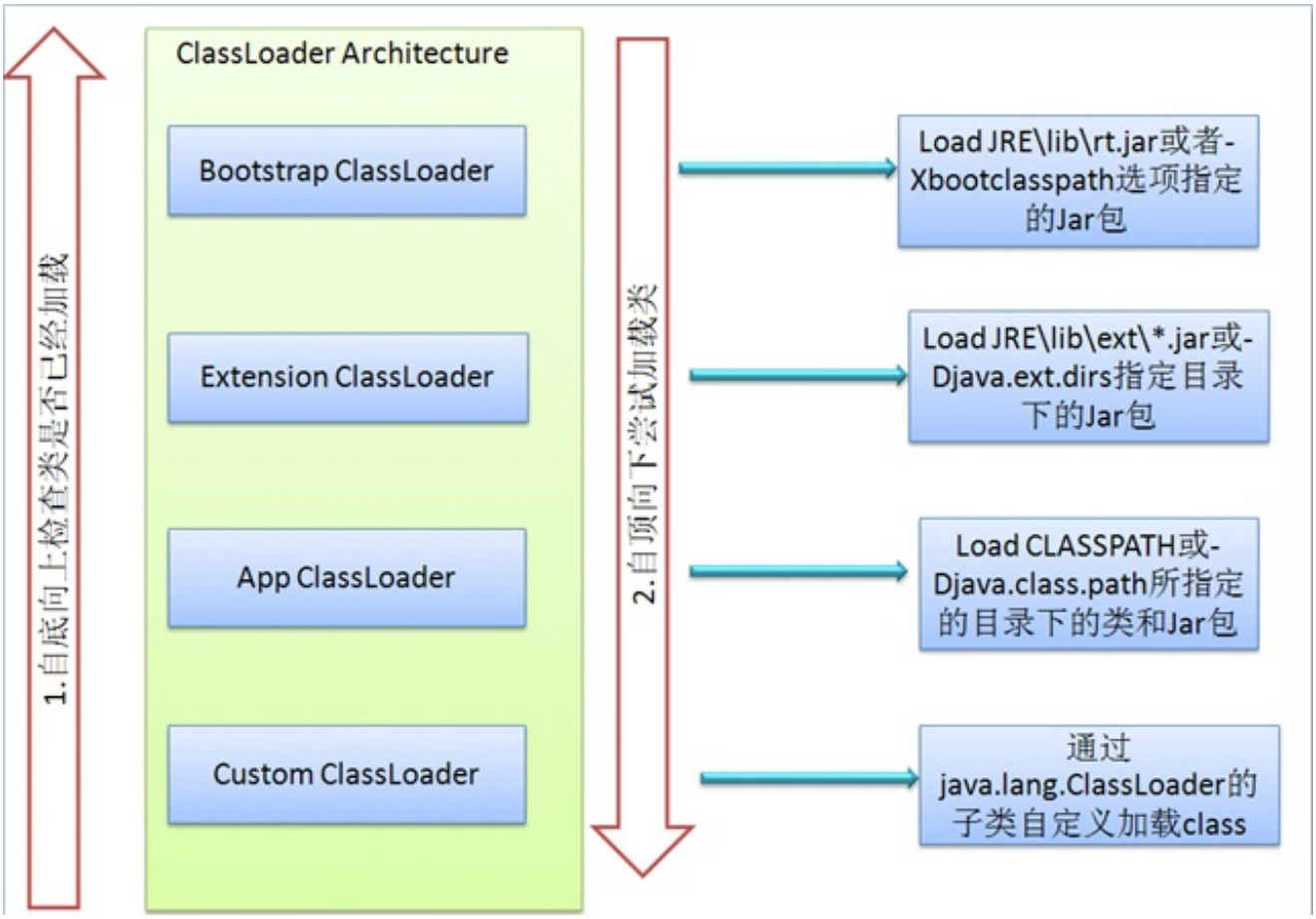


最後生成的 class 文件由以下部分組成：

- 結構信息。包括 class 文件格式版本號及各部分的數量與大小的信息。
- 元數據。對應於 Java 源碼中聲明與常量的信息。包含類/繼承的超類/實現的接口的聲明信息、域與方法聲明信息和常量池。
- 方法信息。對應 Java 源碼中語句和表達式對應的信息。包含字節碼、異常處理器表、求值棧與局部變量區大小、求值棧的類型記錄、調試符號信息。

類加載機制

JVM 的類加載是通過 ClassLoader 及其子類來完成的，類的層次關係和加載順序可以由下圖來描述：



1）Bootstrap ClassLoader

負責加載`$JAVA_HOME`中`jre/lib/rt.jar`裡所有的 class，由 C++ 實現，不是 ClassLoader 子類。

2）Extension ClassLoader

負責加載Java平台中擴展功能的一些 jar 包，包括`$JAVA_HOME`中`jre/lib/*.jar`或`-Djava.ext.dirs`指定目錄下的 jar 包。

3）App ClassLoader

負責記載 classpath 中指定的 jar 包及目錄中 class。

4）Custom ClassLoader

屬於應用程序根據自身需要自定義的 ClassLoader，如 Tomcat、jboss 都會根據 J2EE 規範自行實現 ClassLoader。

加載過程中會先檢查類是否被已加載，檢查順序是自底向上，從 Custom ClassLoader 到 BootStrap ClassLoader 逐層檢查，只要某個 Classloader 已加載就視為已加載此類，保證此類只所有 ClassLoade r加載一次。而加載的順序是自頂向下，也就是由上層來逐層嘗試加載此類。

類執行機制

JVM 是基於棧的體系結構來執行 class 字節碼的。線程創建後，都會產生程序計數器（PC）和棧（Stack），程序計數器存放下一條要執行的指令在方法內的偏移量，棧中存放一個個棧幀，每個棧幀對應著每個方法的每次調用，而棧幀又是有局部變量區和操作數棧兩部分組成，局部變量區用於存放方法中的局部變量和參數，操作數棧中用於存放方法執行過程中產生的中間結果。棧的結構如下圖所示：