Javassist 源码阅读

1 什么是 Java 字节码？

当我们编写完一个 .java 文件并使用 javac 编译后，会生成一个 .class 文件，这个文件包含了 Java 字节码（bytecode），可以在任何安装了 JVM（Java 虚拟机）的设备上运行。这一编译过程是与平台无关的，因此无论在什么操作系统上编译，同样的 .java 文件会生成相同的 .class 文件。这也是 Java “一次编译，到处运行”特性的核心所在，极大地提升了代码的可移植性。

在此过程中，JVM 起到了关键作用，它将底层硬件平台进行抽象，为上层 Java 程序提供了统一的虚拟架构。Java 字节码就是运行在这一虚拟架构上的机器指令，通常每条指令为一个字节（即“字节码”）。一个 .class 文件中包含了类运行所需的字节码指令，JVM 会将这些指令转化为目标平台的机器代码，使程序能够在不同硬件架构上执行。为了适应不同的硬件，Java 字节码主要通过栈来运算，从而避免直接依赖硬件的寄存器配置。

2 什么是 Javassist？

Javassist（Java Programming Assistant）是一个开源的 Java 字节码操作库，主要用于在运行时动态生成、编辑和操作 Java 类。它提供了一个简单易用的 API，让开发者能够通过类似 Java 源代码的方式来操作字节码，而无需深入了解 JVM 字节码的复杂细节。这使得 Javassist 特别适合于开发需要在运行时修改类行为的应用程序，例如 AOP（面向切面编程）、代理类生成、代码注入、性能监控、日志记录等场景。

Javassist 的主要功能包括：

1. **动态生成类和方法**：可以在运行时创建新类或方法，无需提前在源码中定义。
2. **修改现有类**：支持在加载类之前修改已有类的字节码，例如插入新的方法、修改方法体等。
3. **简洁的 API**：提供了面向高层的 API，通过类似源代码的方式编写字节码操作，降低了学习成本。
4. **字节码操作**：可以直接操作 Java 字节码，使得动态代理、增强代码等操作更加灵活。

Javassist 由 JBOSS 组织维护，通常被集成在 Java 框架。

3 .class 文件如何记录信息

.class 文件是 Java 编译器将 Java 源代码（.java 文件）编译后的输出文件，它包含了 Java 虚拟机（JVM）能够直接执行的字节码。.class 文件记录的信息包括：

魔数：每个.class 文件的前四个字节是魔数 0xCAFEBABE，用于标识文件格式，确保文件是一个有效的 Java 类文件。

版本号：紧跟魔数后面的是两个字节，表示.class 文件的次版本号和主版本号，帮助 JVM 识别编译时的 Java 版本。

常量池：存储类、方法名、字符串字面量等，提供指向各种数据的引用池，方便字节码中使用。常量池是.class 文件中最复杂的部分。

访问标志：用于标志类的访问权限和其他属性，比如是否为 public、abstract 或 final 等。

类信息：包括当前类和父类的名称索引，指向常量池中的对应项。

接口信息：记录该类实现的接口列表，用于支持多态和接口调用。

字段表：包含类的字段信息（如属性的名称、类型和访问权限）。

方法表：包含类的方法信息，包括方法名、参数类型、返回类型和字节码等。每个方法都有自己对应的字节码，这就是 JVM 运行的核心内容。

属性表：记录附加信息，比如 SourceFile 属性表示源代码文件的名称，LineNumberTable 记录代码行号信息等，有助于调试和反编译。

这些信息使.class 文件能够被 JVM 加载、解析并执行，从而实现 Java 程序的跨平台能力。

4 ClassFileWriter 类

以下为 javassist 手册中的例子：

```
package sample;

public class Test {

    public int value;

    public long value2;
```

```
    public Test() {
    super();
    }
    public one() {
    return 1;
    }
}
```

创建以上的类时需要用到以下功能：

1.创建字节码对象，写入头部信息

2.添加常量

3.添加字段/成员变量

4.添加方法

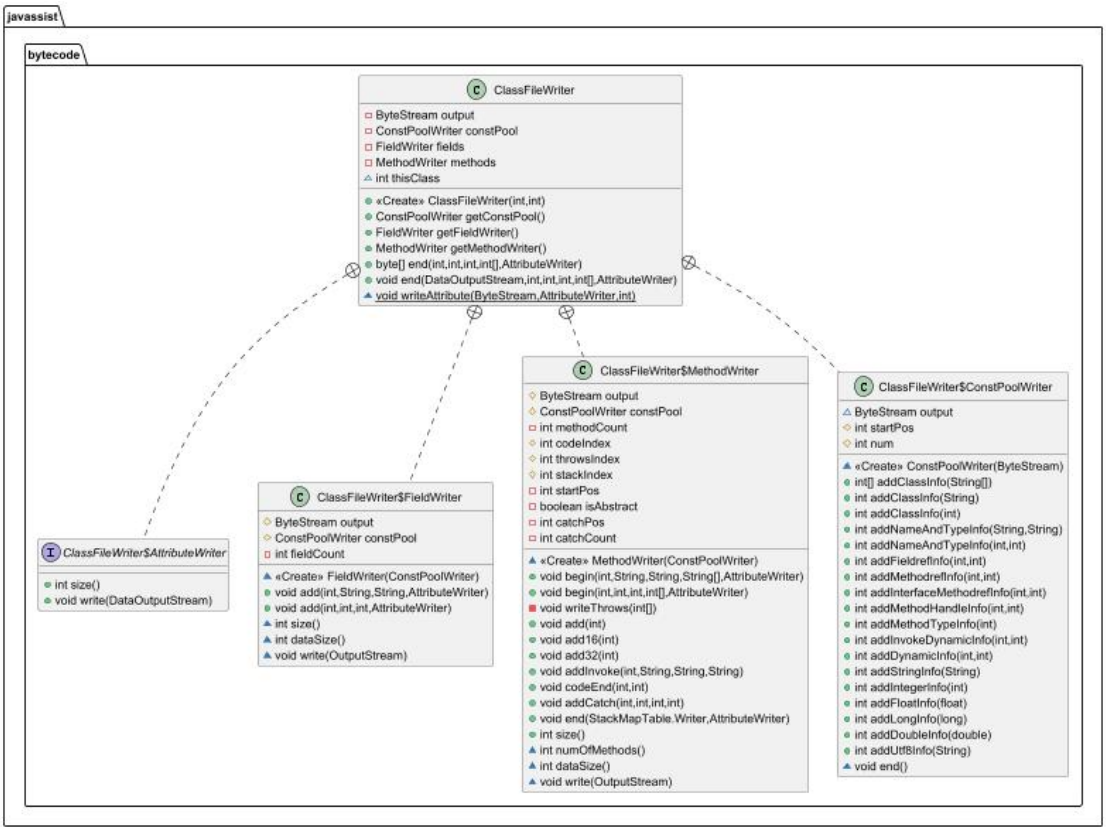
5.更新附加属性

需要以下的类来实现其功能：

类	属性	方法
类文件编写器	.class 的当前类、父类、常量池数据、成员数据、方法数据、输出流	填写初始数据、填写常量池、填写字段表、填写方法表、传递表项至外界
常量编写器	常量输出流、写入的起始位置、已写入的常量个数	Java 规范中 14 种项目类型各自的添加方法、常量池数据写回至输出流
字段编写器	字段输出流、字段的常量池、已写入的字段个数	添加一个字段、输出字段个数、输出字段占用空间大小、写回字段表
方法编写器	方法输出流、方法的常量池、已写入的方法数、写入的起始位置、代码段索引、例外程序索引、栈索引、抽象方法标志位、catch 子句位置、catch 子句数目	填入方法头、添加字节指令、添加 catch 子句、添加 invoke 调用、汇总 code 信息、输出方法数、输出方法占用空间、 写 throw 信息、写回方法表

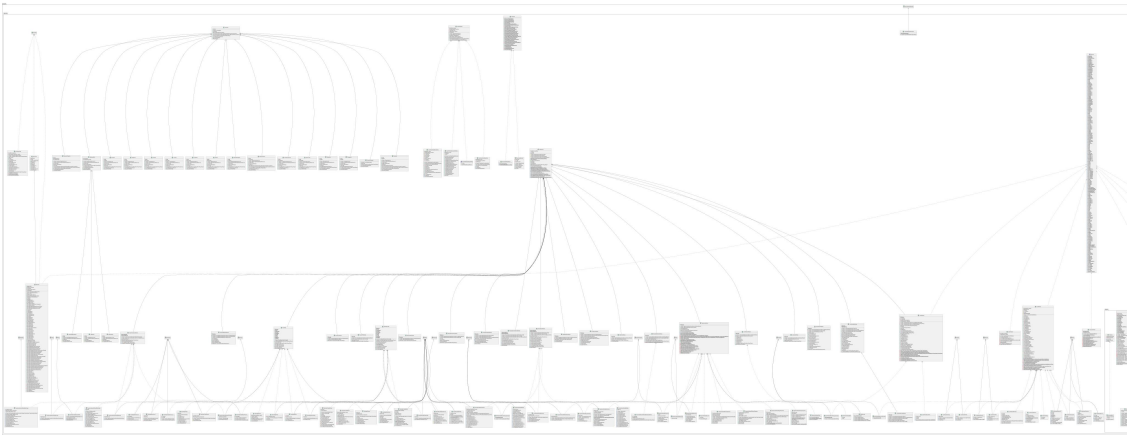
以上功能可通过 ClassFileWriter 类中所提供的 API 实现。

在 ClassFileWriter 中,提供了 ConstPoolWriter,FieldWriter 和 MethodWriter 三个类以实现上述功能。而填写文件头的初始信息可以通过 ClassFileWriter 的构造方法完成。



可以看到，AttributeWriter 接口将更新附加属性这一功能抽象出来，其余三个 Writer 类通过实现该接口获得更新属性的功能。这简化了代码的复杂度，体现了继承和复用的思想。

5 bytecode 模块



可以看出，Bytecode 类是 ByteVector 的子类，而 FloatInfo, Utf8Info 和 StringInfo 等数据类型的信息则继承自 ConstInfo 这个父类,将数据类型的宽度等信息作 为常量存储在.class 文件中，为跨平台的特点提供支持。ClassFile，即类文件类主要依赖于 ConstInfo, FieldInfo, MethodInfo 以及 AttributeInfo 等类。