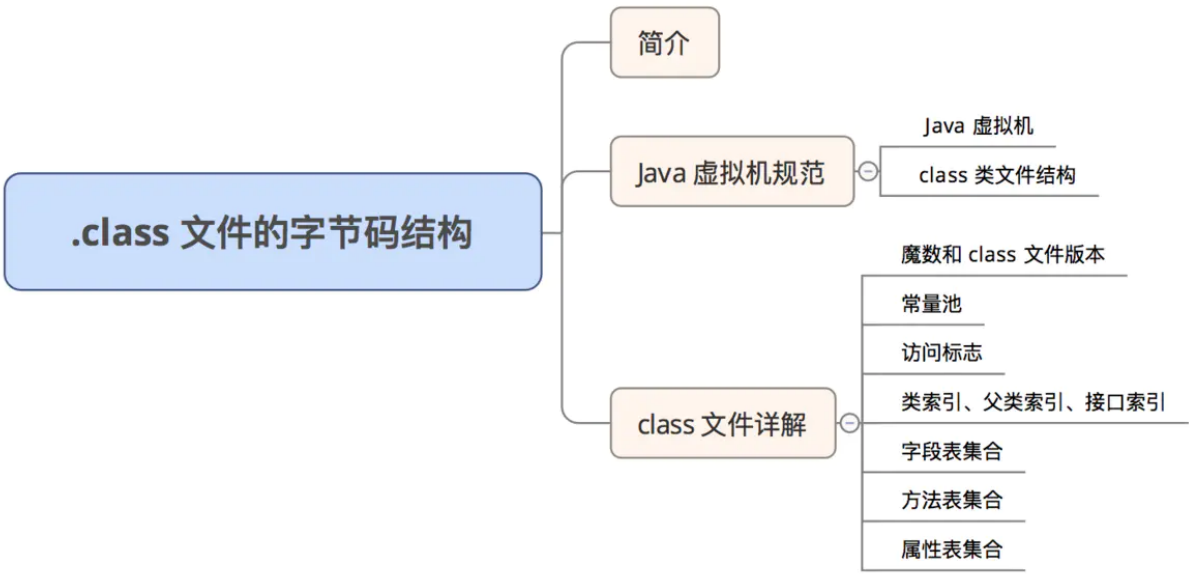
.class文件的字节码结构



一. 简介

写一个简单的 Demo.java 程序如下所示

```
package com.lijiankun24.classpractice;

public class Demo {

 private int m;

 public int inc() {
 return m + 1;
 }
}
```

使用 javac 命令编译 Demo.java 文件生成 Demo.class 文件

```
$ javac Demo.java
```

接着用文本编辑器打开生成的 Demo.class 文件，如下所示

```
cafe babe 0000 0034 0013 0a00 0400 0f09
0003 0010 0700 1107 0012 0100 016d 0100
0149 0100 063c 696e 6974 3e01 0003 2829
5601 0004 436f 6465 0100 0f4c 696e 654e
756d 6265 7254 6162 6c65 0100 0369 6e63
0100 0328 2949 0100 0a53 6f75 7263 6546
```

```
696c 6501 0009 4465 6d6f 2e6a 6176 610c
0007 0008 0c00 0500 0601 0004 4465 6d6f
0100 106a 6176 612f 6c61 6e67 2f4f 626a
6563 7400 2100 0300 0400 0000 0100 0200
0500 0600 0000 0200 0100 0700 0800 0100
0900 0000 1d00 0100 0100 0000 052a b700
01b1 0000 0001 000a 0000 0006 0001 0000
0001 0001 000b 000c 0001 0009 0000 001f
0002 0001 0000 0007 2ab4 0002 0460 ac00
0000 0100 0a00 0000 0600 0100 0000 0600
0100 0d00 0000 0200 0e
```

可以看到，该文件中是由十六进制符号组成的，这一段十六进制符号组成的长串是遵守 Java 虚拟机规范的

二. Java 虚拟机规范

在 Java 虚拟机规范中规定了 Java 虚拟机结构、Class 类文件结构、字节码指令等内容，可以参考 GitHub 上的《Java 虚拟机规范》

2.1 Java 虚拟机

1. 可以说 Java 虚拟机有两大特性：平台无关性和语言无关性，本篇文章主要介绍语言无关性的重要知识：.class 文件结构
2. Java 虚拟机就是一个虚拟的计算机，与真实的计算机一样，Java 虚拟机有自己完善的硬件体系，如处理器、堆栈、寄存器，还有相应的指令集系统。虚拟机与真实电脑的唯一区别就是：虚拟机的处理器、内存堆栈是用软件虚拟出来的，而真实的电脑的处理器、内存则是真真实实存在的
3. 在 Java 虚拟机规范中，介绍的 Java 虚拟机的整体架构、Java 虚拟机内存区域、垃圾回收、.class 文件结构、类加载机制和 Java 虚拟机指令集。在本篇文章中主要介绍 .class 文件结构，其他内容可以查阅相关书籍和 Java 虚拟机规范

2.2 class 类文件结构

.class 文件是一组以 8 位字节为基础单位的二进制流，各数据项目严格按照顺序紧凑地排列在 .class 文件中，中间没有添加任何分隔符，这使得整个 .class 文件中存储的内容几乎全都是程序需要的数据，没有空隙存在

1. .class 文件是以类似于 C 语言结构体的结构来存储数据的，其中存储的数据有两种：无符号数和表
2. 无符号数属于最基本的数据类型，以 u1、u2、u4、u8 分别代码 1 个字节、2 个字节、4 个字节和 8 个字节的无符号数，无符号数可以用来描述数字、索引引用、数量值或者按照 UTF-8 编码构成的字符串值

3. 表是一种复合数据结构，由无符号数或其他表构成，所有表都习惯性地以“info” 结尾

4. 在 .class 中有一个集合的概念。集合表示同一类数据项的集合，一般是由一个前置的计数器加若干个连续的同样类型的数据项组成，计数器表示此集合中数据项的个数，数据项是真正的数据内容

5. 整个 .class 文件本质上就是一张表，由下表所示的数据项构成

类型	名称	解释	数量
u4	magic	魔数	1
u2	minor_version	次版本号	1
u2	major_version	主版本号	1
u2	constant_pool_count	常量池常量个数	1
cp_info	constant_pool	常量池	constant_pool_count - 1
u2	access_flags	访问标记	1
u2	this_class	类索引	1
u2	super_class	父类索引	1
u2	interfaces_count	接口索引数量	1
u2	interfaces	接口内容	interfaces_count
u2	field_count	字段表字段数量	1
field_info	fields	字段表	field_count
u2	methods_count	方法表方法数量	1
method_info	methods	方法表	methods_count
u2	attributes_count	属性表属性数量	1
attribute_info	attributes	属性表	attributes_count

1. 上面的表其实可以划分为以下七个部分，.class 字节码文件包括：

- 魔数与class文件版本
- 常量池
- 访问标志
- 类索引、父类索引、接口索引
- 字段表集合
- 方法表集合

- 属性表集合

三. class 文件详解

我们通过 Demo.class 为例讲解 .class 文件的 7 个部分

3.1 魔数和 class 文件版本

3.1.1 概念介绍

在魔数和 class 文件版本中有如下四点需要介绍：

1. 魔数(Magic Number)：.class 文件的第 1 - 4 个字节，它唯一的作用就是确定这个文件是否是一个能被虚拟机接受的 class 文件，其固定值是：0xCAFEBABE（咖啡宝贝）。如果一个 class 文件的魔术不是 0xCAFEBABE，那么虚拟机将拒绝运行这个文件

2. 次版本号(minor version)：.class 文件的第 5 - 6 个字节，即编译生成该 .class 文件的 JDK 次版本号

3. 主版本号(major version)：.class 文件的第 7 - 8个字节，即编译生成该 .class 文件的 JDK 主版本号

4. Note：高版本的 JDK 能向下兼容低版本的 .class 文件，但不能运行新版本的 .class 文件。例如一个 .class 文件是使用 JDK 1.5 编译的，那么我们可以用 JDK 1.7 虚拟机运行它，但不能用 JDK 1.4 虚拟机运行它。各个版本的 SDK 的次版本号和主版本号如下表所示

JDK版本	次版本号	主版本号	十进制
JDK1.2	0000	002E	46
JDK1.3	0000	002F	47
JDK1.4	0000	0030	48
JDK1.5	0000	0031	49
JDK1.6	0000	0032	50
JDK1.7	0000	0033	51
JDK1.8	0000	0034	52

3.1.2 示例

在上面的 Demo.class 文件中，Magic Number：0xcafe babe，minor version：0x0000，major version：0x0034，可见我们是使用 JDK 1.8 编译生成的 Demo.class 文件

3.2 常量池

3.2.1 概念介绍

紧接着版本号之后的是常量池的入口，常量池可以理解为 class 文件之中的资源仓库，它是占用 class 文件空间最大的数据项之一。

常量池是一个集合，它由两部分组成：常量池计数器和常量池

1. 常量池计数器(constant_pool_count) 是一个 u2 的无符号数
2. 常量池(constant_pool)：紧跟在常量池计数器后面的内容就是该 .class 文件的常量池内容了，常量池中存放的数据一般分为两种类型：字面量和符号引用
    - 字面量：是指文本字符串、声明为 final 的常量值等
    - 符号引用： 是一个更偏向于编译原理方面的概念，主要包括三类常量：1). 类和接口的全限定名，2).字段的名称和描述符，3). 方法的名称和描述符
3. 在常量池中的常量共有 14 种类型，每个常量都是一个表，每一个表都有各自的组成结构。这 14 个常量有一个公共的特点，就是每个常量开始是一个用 u1 类型的无符号数表示的标志位（tag，取值见下表），表示此常量属于哪种常量类型

类型	标志	描述
CONSTANT_Utf_info	1	UTF-8编码的字符串
CONSTANT_Integer_info	3	整型字面量
CONSTANT_Float_info	4	浮点型字面量
CONSTANT_Long_info	5	长整型字面量
CONSTANT_Double_info	6	双精度浮点型字面量
CONSTANT_Class_info	7	累活接口的符号引用
CONSTANT_String_info	8	字符串类型字面量
CONSTANT_Fieldref_info	9	字段的符号引用
CONSTANT_Methodref_info	10	类中方法的符号引用
CONSTANT_InterfaceMethodref_info	11	接口中方法的符号引用
CONSTANT_NameAndType_info	12	字段或方法的部分符号引用
CONSTANT_MethodHandle_info	15	表示方法句柄
CONSTANT_MethodType_info	16	标识方法类型
CONSTANT_InvokeDynamic_info	18	表示一个动态方法调用点

3.2.2 示例

在上面的 Demo.class 文件中，常量池开始的偏移地址是：0x0008。

1. 首先是常量计数器（constant_pool_coun），数值是：0x0013，表示此 Demo.class 文件中共有 18 个常量

2. cp_info_constant_pool[1]：偏移地址是 0x000A，内容是：0x0A0004000F。0x0A 标志位表示是一个 CONSTANT_Methodref_info 常量，0x0004 是一个索引，指向常量池中第 4 个常量所表示的信息；0x000F 是一个索引，指向常量池第 15 个常量所表示的信息。CONSTANT_Methodref_info 常量的结构如下所示：

tag	index	index
u1	u2	u2
10	索引项：指向声明方法的类描述符 CONSTANT_Class_info	索引项：指向名称及类型描述符 CONSTANT_NameAndType

3. cp_info_constant_pool[2]：偏移地址是0x000F，内容是：0x0900030010，0x09 表示此常量是一个 CONSTANT_Fieldref_info 常量，0x0003 表示一个索引，指向常量池第 3 个常量所表示的信息；0x0010 是一个索引，表示指向常量池第 16 个常量所表示的信息。CONSTANT_Fieldref_info 常量的结构如下所示：

tag	index	index
u1	u2	u2
9	索引项：指向声明字段的类描述符 CONSTANT_Class_info	索引项：指向字段描述符 CONSTANT_NameAndType

4.cp_info_constant_pool[3]：偏移地址是0x0014，内容是：0x070011。0x07 标志位表示此常量 是一个 CONSTANT_Class_info 常量，索引 0x0011 指向常量池中第 17 个常量。CONSTANT_Class_info 常量的结构如下所示：

tag	index
u1	u2
7	索引项：指向全限定名常量项的索引

1. cp_info_constant_pool[4]：偏移地址是0x0017，内容是：0x070012。0x07 标志位表示此常量是一个 CONSTANT_Class_info 常量，索引 0x0012 指向常量池中第 18 个常量。

2. cp_info_constant_pool[5]：偏移地址是0x001A，内容是：0x0100016D。0x01 表示此常量是一个 CONSTANT_Utf8_info 常量，0x0001 表示 UTF-8 编码的字符串

占用的字节数；0x6D 表示 长度为 1 的 UTF-8 编码的字符串的内容: m。

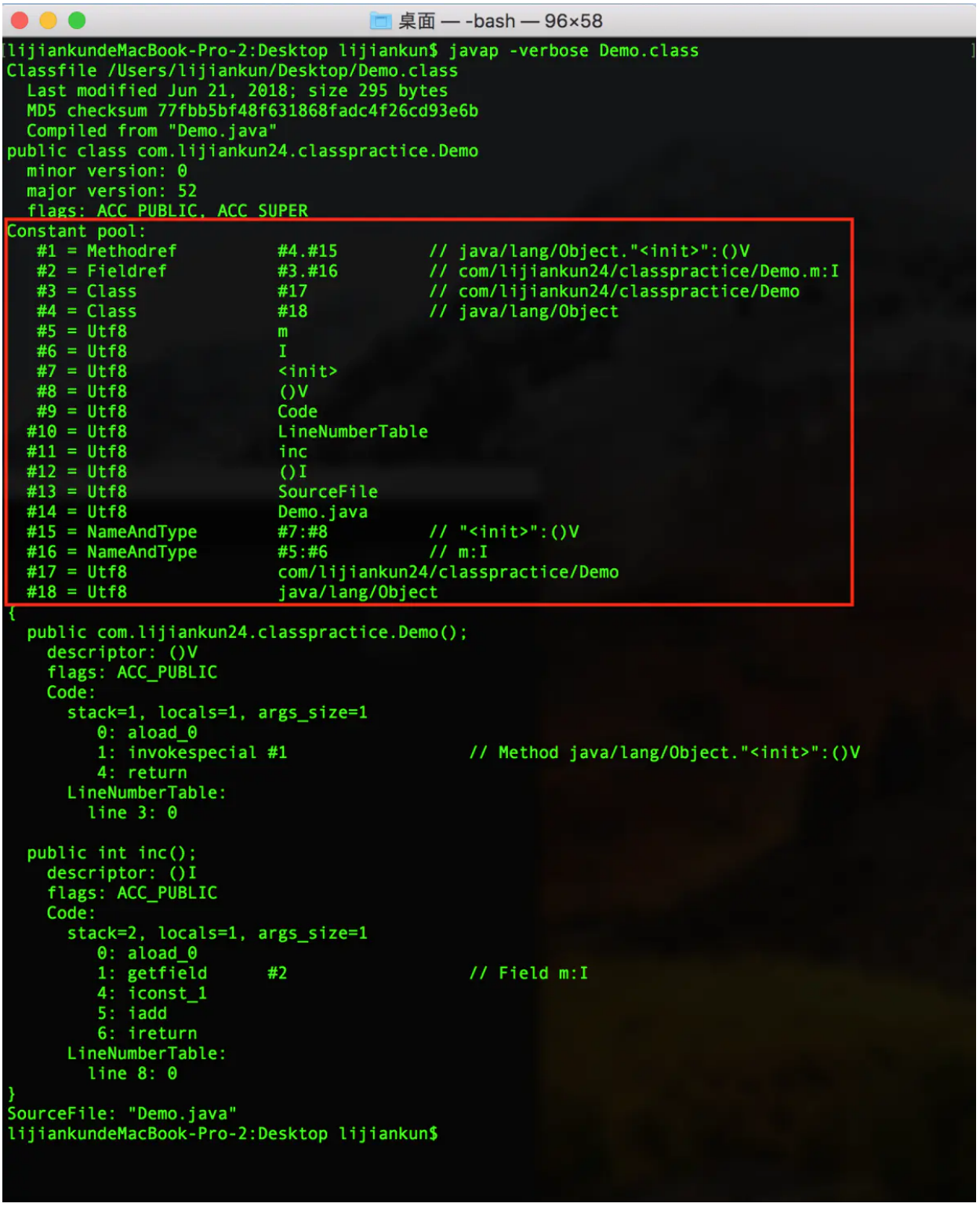
CONSTANT_Utf8_info 常量的结构如下所示：

tag	length	bytes
u1	u2	u1
1	Utf-8 编码的字符串占用的字节数	length 长度的 UTF-8 编码的字符串内容

3. cp_info_constant_pool[6]：偏移地址是0x001E，内容是：0x01000149。0x01 表示此常量是一个 CONSTANT_Utf8_info 常量，0x0001 表示 UTF-8 编码的字符串占用的字节数；0x49 表示长度为 1 的 UTF-8 编码的字符串的内容: I。

4. cp_info_constant_pool[7]：偏移地址是0x0022，内容是：0x0100063C696E69743E。0x01 表示此常量是一个 CONSTANT_Utf8_info 常量，0x0006 表示字符串长度为 6，0x3C696E69743E 表示长度为 6 的 UTF-8 编码的字符串的内容: <init>。

上面分析了 7 个常量，其余的常量也是类似的方法。根据第一个 u1 的标志位，就知道这个常量的类型和表结构，就可以知道这个常量的长度大小和代表的含义了。我们也可以通过 “javap -verbose” 命令查看 .class 文件的内容，如下图所示：



3.3 访问标志

3.3.1 概念介绍

常量池之后是 u2 类型的访问标志位（access_flags），这个访问标志位用于标识类或者接口层次的访问信息，包括：这个 Class 是类还是接口、是否定义为public类型、是否定义为abstract类型，如果是类的话，是否被 final 关键字修饰。具体的标志位以及标志的含义见下表

标志名称	标志值	含义
ACC_PUBLIC	0x0001	是否为public类型
ACC_FINAL	0x0010	是否被声明为final，只有类可设置
ACC_SUPER	0x0020	是否允许使用invokespecial字节码指令的新语义，invokespecial指令的语义在JDK1.2发生过改变，为了区别这条指令使用哪种语义，JDK1.2之后编译出来的类的这个标志都必须为真
ACC_INTERFACE	0x0200	标识这时一个接口
ACC_ABSTRACT	0x0400	是否为abstract类型，对于接口或抽象类来说，此标志值为真，其他类值为假
ACC_SYNTHETIC	0x1000	标识这个类并非由用户代码产生的
ACC_ANNOTATION	0x2000	标识这时一个注解
ACC_ENUM	0x4000	标识这时一个枚举

3.3.2 示例

在 Demo.class 文件中访问标志位是：0x0021。在上表中，我们并没有发现 00 21 的访问标志，这是因为在字节码文件中的访问标志，可以通过上表中多个访问标志通过或运算组成真正的访问标志。通过上表中的 ACC_SUPER 和 ACC_PUBLIC 就可以组合中 00 21 的访问标志了，也就是说该类的访问标志是 public 且允许使用 invokespecial 字节码指令的新语义的

3.4 类索引、父类索引、接口索引

3.4.1 概念介绍

在 .class 文件中由这三项数据来确定这个类的继承关系。

1. 类索引：u2 数据类型，用于确定这个类的全限定名。
2. 父类索引：u2 数据类型，用于确定这个类的父类的全限定名。
3. 接口索引：u2 数据类型的集合，用于描述类实现了哪些接口，这些被实现的接口将按照 implements 语句后的顺序从左至右排列在接口索引集合中。接口索引集合分为两部分，第一部分表示接口计数器(interfaces_count)，是一个 u2 类型的数据，第二部分是接口索引表表示接口信息，紧跟在接口计数器之后。若一个类实现的接口为 0，则接口计数器的值为 0，接口索引表不占用任何字节。

3.4.2 示例

在此 Demo.class 文件中，类索引、父类索引、接口索引分别如下：

1. 类索引：偏移地址是 0x00B3，内容是 0x0003，表示其指向了常量池中第 3 个常量 CONSTANT_Class_info，第 3 个常量索引指向第 17 个常量，第 17 个常量是一个 UTF-8 编码的字符串，其值是：com/lijiankun24/classpractice/Demo，表示此类的全限定名

2. 父类索引：偏移地址是 0x00B5，内容是 0x0004，其指向了常量池中第 4 个常量 CONSTANT_Class_info，第 4 个常量索引指向第 18 个常量，第 18 个常量的值是：java/lang/Object，表示父类的全限定名

3. 接口索引：偏移地址是 0x00B7，内容是 0x0000。因为 Demo 类没有实现任何接口，所以接口索引的计数器是 0，表示没有接口索引。

3.5 字段表集合

3.5.1 概念介绍

字段表集合用于描述接口或类中声明的变量。这里说的字段包括类级变量（static 修饰）和对象级变量（没有用 static 修饰），但不包括方法中声明的局部变量。

字段表集合包括两部分：字段计数器和字段表，字段计数器表示有多少个字段，字段表的每个字段用一个名为 field_info 的表来表示，field_info 表的数据结构如下所示：

类型	名称	数量	含义
u2	access_flags	1	字段访问标识
u2	name_index	1	字段名称索引项
u2	descriptor_index	1	字段描述符索引项
u2	attributes_count	1	属性表计数器
attribute_info	attributes	attribute_count	属性表

字段表都包含的固定数据项目到 descriptor_index 为止就结束了，不过在 descriptor_index 之后跟随着一个属性表集合用于存储一些额外的信息，字段都可以在属性表中描述零至多项的额外信息。在字段描述符之后，一般会有该字段的属性表集合，属性表集合有两部分，第一部分是属性计数器，第二部分是属性表。

在字段表集合中不会列出父类的字段，但是有可能会有一些 Java 代码中没有声明的字段，比如在内部类中为了保持对外部类的访问性，会自动添加指向外部类实例的字段

3.5.2 示例

在 Demo.class 文件中的字段表集合的偏移地址是：0x00B9，内容是：0x 0001 0002 0005 0006 0000.

1. 0x0001 表示字段计数器是 1，表示只有 1 个字段

2. field_info_fields[0]：偏移地址是 0x00BB，内容是 0x0002 0005 0006 0000，根据字段表的结构来分析这段数据

    - 0x0002 表示该字段的访问标识，0002 表示是 private 的

- 0x0005 表示该字段的名称索引项，指向常量池中的第 5 个常量，第 5 个常量是一个 UTF-8 的字符串 m
- 0x0006 表示该字段的描述符索引项，指向常量池中的第 6 个常量，第 6 个常量是一个 UTF-8 的字符串 I，I 描述符表示是一个 int 类型的字段
- 0x0000 表示 m 字段的属性表集合，属性表集合计数器是 0，表示此字段没有额外的属性信息。

3.6 方法表集合

3.6.1 概念介绍

在字段表之后紧跟着方法表集合，方法表表示类或接口中的方法信息。

方法表集合和上述的字段表集合几乎完全一样，最开始的 2 个字节表示一个方法计数器，在方法计数器之后，才是真正的方法数据项。方法表中的每个方法都用一个 method_info 表示，其数据结构如下：

类型	名称	数量	含义
u2	access_flags	1	方法访问标识
u2	name_index	1	方法名称索引项
u2	descriptor_index	1	方法描述符索引项
u2	attributes_count	1	属性表计数器
attribute_info	attributes	attribute_count	属性表

在方法表结构中，我们可以看到方法的访问标志位、名称索引、描述符索引、属性表集合，方法中的代码在编译之后，放到方法属性表集合中的一个名为 “code” 的属性里面

3.6.2 示例

1. 在 Demo.class 中方法表集合的偏移地址是：0x00C3，方法表集合计数器是 0x0002， 表示此方法表集合中有两个方法表数据项。可能有人会有疑问，Demo.java 中我们只写了一个方法，为什么在方法表中会有两个方法呢？因为编译器会自动添加实例构造器 <init> 方法
2. method_info_methods[0]：偏移地址是：0x00C5，内容是：0x00 0100 0700 0800 0100 0900 0000 1d00 0100 0100 0000 052a b700 01b1 0000 0001 000a 0000 0006 0001 0000 0003。

- 0x0001：access_flags 表示ACC_PUBLIC，即表示该方法是 public 的
- 0x0007：name_index 表示方法名称索引，指向常量池中的第 7 个常量，第 7 个常量是一个 UTF-8 字符串，值是：<init>
- 0x0008：descriptor_index 表方法描述符索引项，指向常量池中的第 8 个常量，是一个 UTF-8 字符串，值是：()V
- 0x0001：表示此方法的属性表集合计数器，有 1 个属性
- 0x0009：表示此属性的 attribute_name_index，指向常量池中的第 9 UTF-8 常量：Code，说明此属性是方法的字节码描述 Code 属性

表 6-15　Code 属性表的结构

类　型	名　称	数　量
u2	attribute_name_index	1
u4	attribute_length	1
u2	max_stack	1
u2	max_locals	1
u4	code_length	1
u1	code	code_length
u2	exception_table_length	1
exception_info	exception_table	exception_table_length
u2	attributes_count	1
attribute_info	attributes	attributes_count

那么我们就依次按照上表的结构分析此实例构造器的字节码内容，至于具体的字节码含义会在后面的文章中分析介绍

- 0x0009：上面已经介绍过，表示此 Code 属性的名称索引，其值就是“Code” 字符串
- 0x0000 001d：attribute_length 表示属性长度为 29 个字节
- 0x0001：max_stack 表示操作数栈的最大深度是 1
- 0x0001：max_locals 表示局部变量表的最大长度是 1
- 0x0000 0005：code_length 表示字节码指令长度是 5，共有 5 个字节码指令
- 0x2ab7 0001 b1：这 5 个 u1 数据，表示 3 个字节码指令，0x2a = aload_0，0xb7 = invokespecial，0x0001 = 表示一个指向常量池的索引，是 invokespecial 指令的参数，0xb1 = return 表示从当前方法返回
- 0x0000 exception_table_length=0，异常表集合长度为 0

- 0x0001：attributes_count=1(Code属性表内部还含有1个属性表)
- 0x000a：指向常量池中的第十个常量：LineNumberTable，

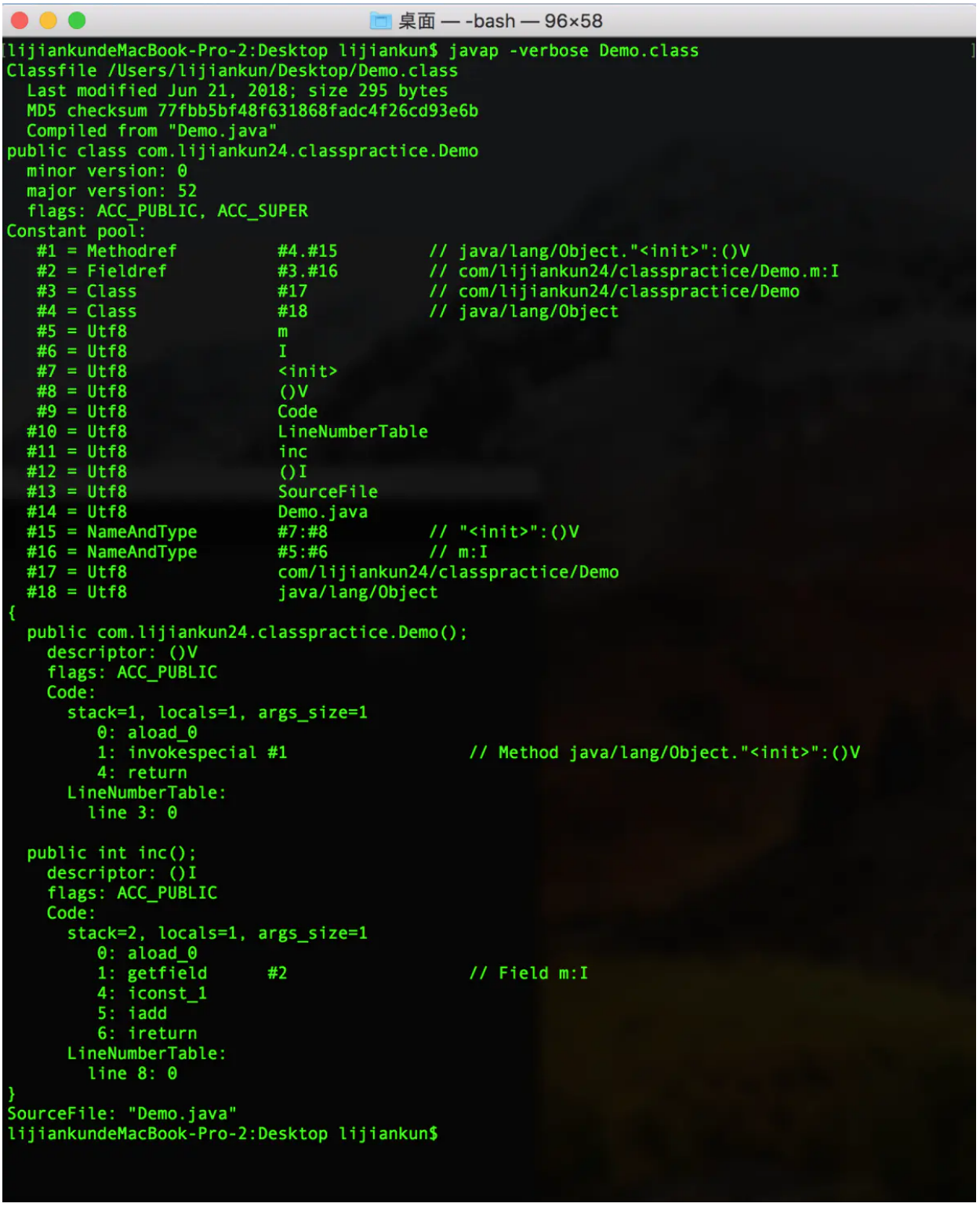
LineNumberTable 属性结构如下图所示，内容是：0000 0006 0001 0000 0003

LineNumberTable属性结构

类型	名称	数量
u2	attribute_name_index	1
u4	attribute_length	1
u2	line_number_table_length	1
line_number_info	line_number_table	line_number_table_length

- 0x0000 0006：attribute_length 表示属性长度为 6
- 0x0001：line_number_table_length，表示后面的 line_number_info 表有 1 个，line_number_info表包括了 start_pc 和 line_number 两个 u2 类型的数据项，前者是字节码行号，后者是 java 源码行号：start_pc：00 00，end_pc：00 03

3. method_info_methods[1]：上面我们分析了第一个方法：实例构造器方法，分析流程就是上面这样，方法表有固定的结构，其中包含一些固定的信息，包括操作数栈最大深度、局部变量表最大长度、以及很重要的 Code 属性，在 Code 属性中包含 java 方法编译生成的字节码指令，如果想快速的浏览方法表集合的内容，也可以使用 "javap -verbose Demo.class" 指令查看，如下图所示



3.7 属性表集合

3.7.1 概念介绍

1. 在 class 文件、字段表、方法表都可以携带自己的属性表集合，用以描述某些场景专有的信息。
2. 属性表的格式是相对固定的，包括三部分内容：
    - 一个 u2 的 attribute_name_index 指向常量池中的一个 UTF-8 字符串常量表示一个属性名称

- 一个 u4 的数据类型表示 attribute_length 表示该属性值的字节长度
- 该长度的属性值信息，结构如下图所示：

表 6-14 属性表结构

类 型	名 称	数 量
u2	attribute_name_index	1
u4	attribute_length	1
u1	info	attribute_length

对于属性表的限制来说相对较宽松，任何人实现的编译器都可以向属性表中写入自定义的属性值信息，Java 虚拟机对于它自己不认识的属性值则会忽略掉。

1. 在 Java 7 虚拟机规范中已经预定义了 21 项属性

3.7.2 示例

1. Demo.class 中属性表的偏移地址是：0x011D，内容是 0x00 0100 0D00 0000 0200 0E
2. 0x0001 表示此属性表集合的计数器是1，有 1 个属性
3. attribute_info_attributes[0]：偏移地址是：0x011F，内容是 0x00 0D00 0000 0200 0E
   - 0x000D：指向常量池中的第 13 个 Utf-8 常量：SourceFile，SourceFile 属性用于记录生成这个 Class 文件的源码文件名称，其结构如下图所示：

表 6-21 SourceFile 属性结构

类 型	名 称	数 量
u2	attribute_name_index	1
u4	attribute_length	2
u2	sourcefile_index	1

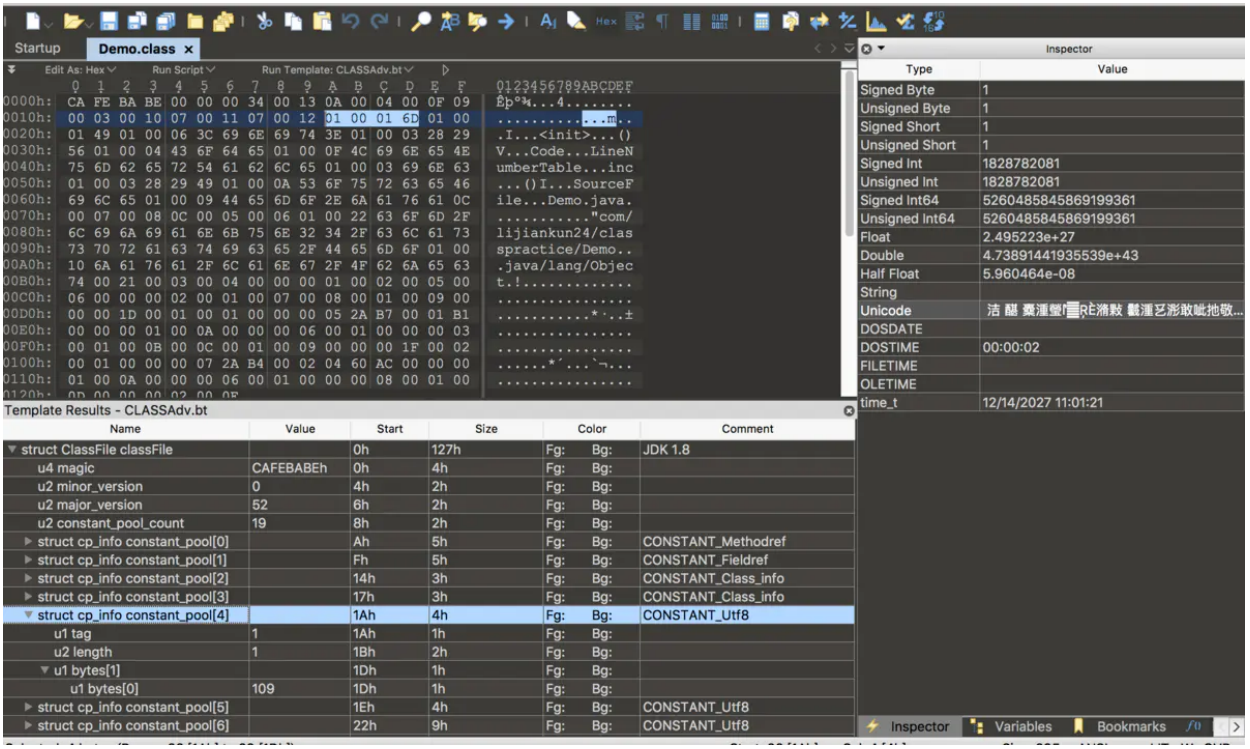
   - 0x0000 0002：attribute_length 属性长度是 2
   - 00 0E：sourcefile_index 指向常量池中第 14 个常量 Demo.java

3.8 010 Editor

分析 .class 文件结构是比较枯燥无聊的，但是如果可以看懂 .class 文件结构的内容，并且理解其中的含义，知道 .class 文件结构中 Code 属性中字节码指令的执行过程，对我们的 Java 能力提升还是比较大的。

分析 .class 文件结构，我们可以使用 "javap -verbose Demo.class" 指令查看，我们也可以使用 010 Editor 软件分析，可以方便的查看各个数据项的地址偏移量、数据项内容。比

如，我们想查看第 4 个常量池的内容，如下图所示



写在最后，这篇文章分析了 .class 文件的结构，知道了其本质是以 8 位字节为单位存储的二进制流文件

1个字节 = 2个16进制字符