class文件描述:

java语言实际上和语言无关，从Java程序到jRuby, Groovy以及JPython等语言可以看出，java虚拟机不和任何语言绑定，而是只和class文件关联，class文件中包含了java虚拟机指令集和符号表以及其他若干辅助信息。

class文件是一组以8位字节为基础单位的二进制流，当遇到8位字节以上单位的数据时，则会按照高位在前的方式分割成若干个8位字节进行存储

class文件的结构:

1. class文件由无符号数和表构成
2. 无符号数：以u1, u2, u4, u8分别表示1个字节，2个字节，4个字节和8个字节无符号数，可以用来描述数字、索引引用、数量值、按照UTF-8编码构成的字符串值
3. 表：以多个无符号或者其他表作为数据构成的复杂数据类型，所有表都习惯性的以_info结尾

class文件格式:

数据描述	数据类型	备注
magic	u4	魔数，标识是 0xCAFEBABE
minor_version	u2	小版本号
major_version	u2	大版本号
constant_pool_count	u2	常量池大小，
constant_pool[constant_pool_count-1]	cp_info	常量池信息
access_flag	u2	访问标识: pub protect
this_class	u2	类索引
super_class	u2	父类索引
interface_count	u2	接口个数
interface[interface_count]	u2	接口类索引信
fields_count	u2	字段表
fields[fields_count]	field_info	字段表信息
methods_count	u2	方法数
methods[methods_count]	method_info	方法信息
attributes_count	u2	属性个数
attibutes[attributes_count]	attribute_info	属性表信息

如下示例：

```
package com.asheng.jvm.data.struts;

/**
```

```
 * @author asheng
 * @since 2019/4/15
 */
public interface Work {

    void work();

    int getWorkHours();
}


package com.asheng.jvm.data.struts;

/**
 * @author asheng
 * @since 2019/4/15
 */
public class Programmer implements Work {

    private int workHours;

    public Programmer(int workHours) {
        this.workHours = workHours;
    }


    @Override
    public void work() {
        System.out.println("作为一名开发人员，我们规定的工作时间是: " +
workHours +"，但实际我们的工作时间是：996");
    }

    @Override
    public int getWorkHours() {
```

```
        return workHours;
    }
}
```

对Programmer.class文件做分析，使用Sublime打开。

```
cafe babe 0000 0034 0038 0a00 0d00 1e09
000c 001f 0900 2000 2107 0022 0a00 0400
1e08 0023 0a00 0400 240a 0004 0025 0800
260a 0004 0027 0a00 2800 2907 002a 0700
2b07 002c 0100 0977 6f72 6b48 6f75 7273
0100 0149 0100 063c 696e 6974 3e01 0004
2849 2956 0100 0443 6f64 6501 000f 4c69
6e65 4e75 6d62 6572 5461 626c 6501 0012
4c6f 6361 6c56 6172 6961 626c 6554 6162
6c65 0100 0474 6869 7301 0027 4c63 6f6d
2f61 7368 656e 672f 6a76 6d2f 6461 7461
2f73 7472 7574 732f 5072 6f67 7261 6d6d
6572 3b01 0004 776f 726b 0100 0328 2956
0100 0c67 6574 576f 726b 486f 7572 7301
0003 2829 4901 000a 536f 7572 6365 4669
6c65 0100 0f50 726f 6772 616d 6d65 722e
6a61 7661 0c00 1100 190c 000f 0010 0700
2d0c 002e 002f 0100 176a 6176 612f 6c61
6e67 2f53 7472 696e 6742 7569 6c64 6572
0100 3be4 bd9c e4b8 bae4 b880 e590 8de5
bc80 e58f 91e4 baba e591 98ef bc8c e688
91e4 bbac e8a7 84e5 ae9a e79a 84e5 b7a5
e4bd 9ce6 97b6 e997 b4e6 98af 3a20 0c00
3000 310c 0030 0032 0100 2aef bc8c e4bd
86e5 ae9e e999 85e6 8891 e4bb ace7 9a84
e5b7 a5e4 bd9c e697 b6e9 97b4 e698 afef
bc9a 3939 360c 0033 0034 0700 350c 0036
0037 0100 2563 6f6d 2f61 7368 656e 672f
6a76 6d2f 6461 7461 2f73 7472 7574 732f
5072 6f67 7261 6d6d 6572 0100 106a 6176
612f 6c61 6e67 2f4f 626a 6563 7401 001f
636f 6d2f 6173 6865 6e67 2f6a 766d 2f64
6174 612f 7374 7275 7473 2f57 6f72 6b01
0010 6a61 7661 2f6c 616e 672f 5379 7374
656d 0100 036f 7574 0100 154c 6a61 7661
2f69 6f2f 5072 696e 7453 7472 6561 6d3b
0100 0661 7070 656e 6401 002d 284c 6a61
7661 2f6c 616e 672f 5374 7269 6e67 3b29
4c6a 6176 612f 6c61 6e67 2f53 7472 696e
6742 7569 6c64 6572 3b01 001c 2849 294c
6a61 7661 2f6c 616e 672f 5374 7269 6e67
```

```
D:\code\jvm-study\target\classes\com\asheng\jvm\data\struts>javap -v Programmer.class
Classfile /D:/code/jvm-study/target/classes/com/asheng/jvm/data/struts/Programmer.class
  Last modified 2019-4-15; size 1025 bytes
  MD5 checksum 816a9c8bf4050f655c116652a808cae6
  Compiled from "Programmer.java"
public class com.asheng.jvm.data.struts.Programmer implements com.asheng.jvm.data.struts.Work
  minor version: 0
  major version: 52
  flags: ACC_PUBLIC, ACC_SUPER
Constant pool:
   #1 = Methodref          #13.#30        // java/lang/Object."<init>":()V
   #2 = Fieldref           #12.#31        // com/asheng/jvm/data/struts/Programmer.workHours:I
   #3 = Fieldref           #32.#33        // java/lang/System.out:Ljava/io/PrintStream;
   #4 = Class              #34            // java/lang/StringBuilder
   #5 = Methodref          #4.#30         // java/lang/StringBuilder."<init>":()V
   #6 = String             #35            // 作为一名开发人员，我们规定的工作时间是：
   #7 = Methodref          #4.#36         // java/lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/StringBuilder;
   #8 = Methodref          #4.#37         // java/lang/StringBuilder.append:(I)Ljava/lang/StringBuilder;
   #9 = String             #38            // ，但实际我们的工作时间是：996
  #10 = Methodref          #4.#39         // java/lang/StringBuilder.toString:()Ljava/lang/String;
  #11 = Methodref          #40.#41        // java/io/PrintStream.println:(Ljava/lang/String;)V
  #12 = Class              #42            // com/asheng/jvm/data/struts/Programmer
  #13 = Class              #43            // java/lang/Object
  #14 = Class              #44            // com/asheng/jvm/data/struts/Work
  #15 = Utf8               workHours
  #16 = Utf8               I
  #17 = Utf8               <init>
```

```
  #17 = Utf8               <init>
  #18 = Utf8               (I)V
  #19 = Utf8               Code
  #20 = Utf8               LineNumberTable
  #21 = Utf8               LocalVariableTable
  #22 = Utf8               this
  #23 = Utf8               Lcom/asheng/jvm/data/struts/Programmer;
  #24 = Utf8               work
  #25 = Utf8               ()V
  #26 = Utf8               getWorkHours
  #27 = Utf8               ()I
  #28 = Utf8               SourceFile
  #29 = Utf8               Programmer.java
  #30 = NameAndType        #17:#25        // "<init>":()V
  #31 = NameAndType        #15:#16        // workHours:I
  #32 = Class              #45            // java/lang/System
  #33 = NameAndType        #46:#47        // out:Ljava/io/PrintStream;
  #34 = Utf8               java/lang/StringBuilder
  #35 = Utf8               作为一名开发人员，我们规定的工作时间是：
  #36 = NameAndType        #48:#49        // append:(Ljava/lang/String;)Ljava/lang/StringBuilder;
  #37 = NameAndType        #48:#50        // append:(I)Ljava/lang/StringBuilder;
  #38 = Utf8               ，但实际我们的工作时间是：996
  #39 = NameAndType        #51:#52        // toString:()Ljava/lang/String;
  #40 = Class              #53            // java/io/PrintStream
  #41 = NameAndType        #54:#55        // println:(Ljava/lang/String;)V
  #42 = Utf8               com/asheng/jvm/data/struts/Programmer
  #43 = Utf8               java/lang/Object
```

从以上信息可以看出，常量池中主要存放两大类常量：字面量以及符号引用

1. 字面量指的是：文本字符串、被声明为final的常量值

2. 符号引用：

(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