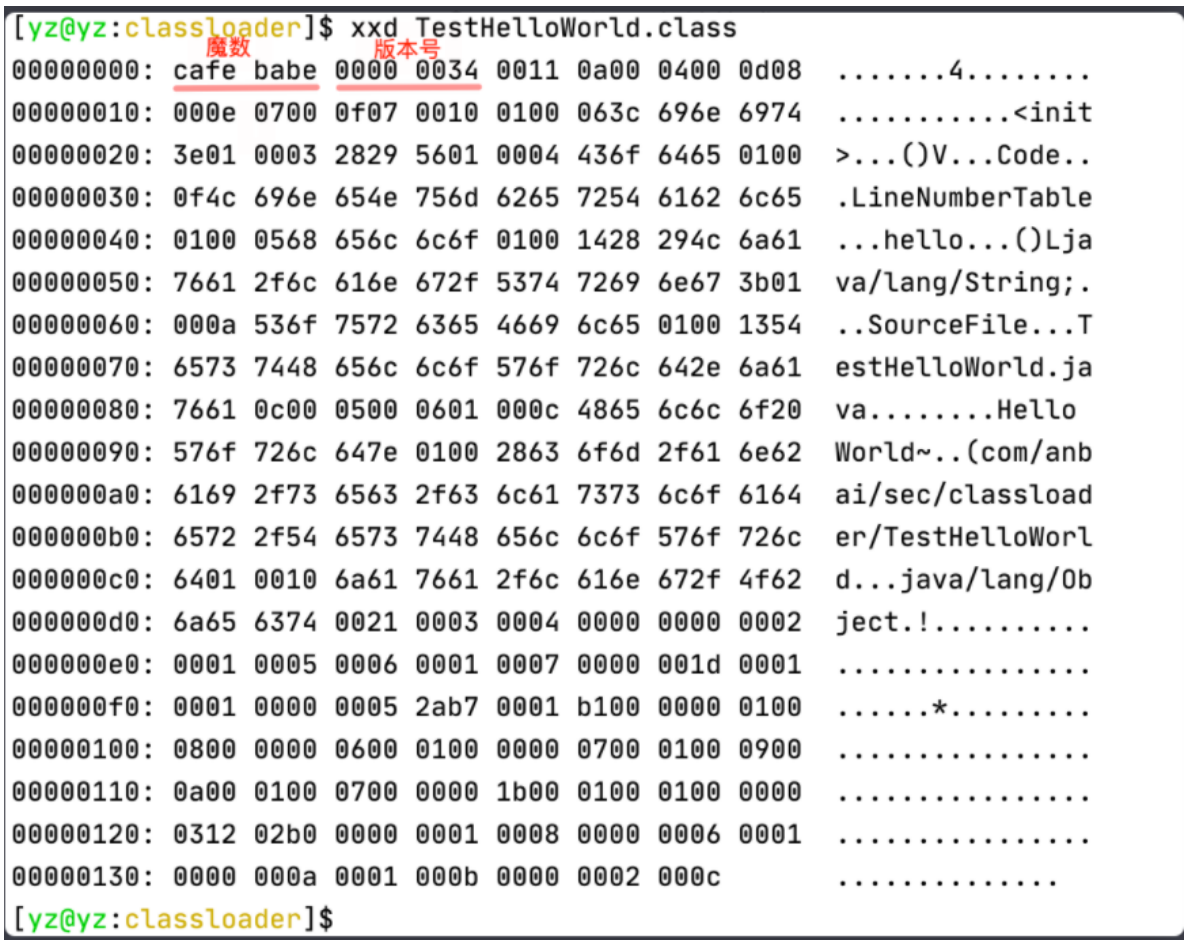
1. Java class文件格式

在JVM虚拟机规范第四章中规定了class文件必须是一个固定的结构，如下所示：

```
ClassFile {
    u4 magic;//魔数,是class文件的标识符
    u2 minor_version;//副版本号
    u2 major_version;//主版本号
    u2 constant_pool_count;//常量池中的数量
    cp_info constant_pool[constant_pool_count-1];
    //是一种表结构 cp_info表示的是常量池对象
    u2 access_flags;//某个类或者接口的访问权限及属性。
    u2 this_class;//当前类名称
    u2 super_class;//当前类的父类名称
    u2 interfaces_count;//表示的是当前类继承或实现的接口数。
    u2 interfaces[interfaces_count];//表示的是所有接口数组。
    u2 fields_count;//表示的是当前class中的成员变量个数。
    field_info fields[fields_count];//成员变量数组
    u2 methods_count;//当前类的成员方法数
    method_info methods[methods_count];//成员方法数组
    u2 attributes_count;//当前类的属性数
    attribute_info attributes[attributes_count];//属性数组
}
```

在JVM规范中`u1`、`u2`、`u4`分别表示的是1、2、4个字节的无符号数，可使用`java.io.DataInputStream`类中的对应方法：`readUnsignedByte`、`readUnsignedShort`、`readInt`方法读取。除此之外，表结构(`table`)由任意数量的可变长度的项组成，用于表示class中的复杂结构，如上述的：`cp_info`、`field_info`、`method_info`、`attribute_info`。

TestHelloWorld.class十六进制：



2. Magic（魔数）

魔数是class文件的标识符，固定值为`0xCAFEBABE`，JVM加载class文件时会先读取4字节（`u4 magic;`）的魔数信息校验是否是一个class文件。

3. Minor/Major Version（版本号）

class文件的版本号由两个`u2`组成（`u2 minor_version; u2 major_version;`），分别表示的是`minor_version`（副版本号）、`major_version`（主版本号），我们常说的`JDK1.8`、`Java9`等说的就是主版本号，如上图中的`TestHelloWorld.class`的版本号`0x34`即`JDK1.8`。

Java版本对应表：

JDK版本	十进制	十六进制	发布时间
JDK1.1	45	2D	1996-05
JDK1.2	46	2E	1998-12
JDK1.3	47	2F	2000-05
JDK1.4	48	30	2002-02
JDK1.5	49	31	2004-09
JDK1.6	50	32	2006-12
JDK1.7	51	33	2011-07
JDK1.8	52	34	2014-03
Java9	53	35	2017-09
Java10	54	36	2018-03
Java11	55	37	2018-09
Java12	56	38	2019-03
Java13	57	39	2019-09
Java14	58	3A	2020-03
Java15	59	3B	2020-09

4. constant_pool_count （常量池计数器）

`u2 constant_pool_count;`表示的是常量池中的数量，`constant_pool_count`的值等于常量池中的数量加1，需要特别注意的是`long`和`double`类型的常量池对象占用两个常量位。

5. constant_pool（常量池）

`cp_info constant_pool[constant_pool_count-1];`是一种表结构，`cp_info`表示的是常量池对象。

`cp_info`数据结构：

```
cp_info {
   u1 tag;
   u1 info[];
}
```

`u1 tag;`表示的是常量池中的存储类型

6. access_flags （访问标志）

`u2 access_flags;`，表示的是某个类或者接口的访问权限及属性。

标志名	十六进制值	描述
ACC_PUBLIC	0x0001	声明为public
ACC_FINAL	0x0010	声明为final
ACC_SUPER	0x0020	废弃/仅JDK1.0.2前使用
ACC_INTERFACE	0x0200	声明为接口
ACC_ABSTRACT	0x0400	声明为abstract
ACC_SYNTHETIC	0x1000	声明为synthetic，表示该class文件并非由Java源代码所生成
ACC_ANNOTATION	0x2000	标识注解类型
ACC_ENUM	0x4000	标识枚举类型

7. this_class（当前类名称）

`u2 this_class;` 表示的是当前class文件的类名所在常量池中的索引位置。

8. super_class（当前类的父类名称）

`u2 super_class;` 表示的是当前class文件的父类类名所在常量池中的索引位置。`java/lang/Object` 类的 `super_class` 的为0，其他任何类的 `super_class` 都必须是一个常量池中存在的索引位置。

9. interfaces_count（当前类继承或实现的接口数）

`u2 interfaces_count;` 表示的是当前类继承或实现的接口数。

10. interfaces[]（接口名称数组）

`u2 interfaces[interfaces_count];` 表示的是所有接口数组。

11. fields_count（当前类的成员变量数）

`u2 fields_count;` 表示的是当前class中的成员变量个数。

12. fields[]（成员变量数组）

`field_info fields[fields_count];` 表示的是当前类的所有成员变量，`field_info` 表示的是成员变量对象。

field_info数据结构：

```
copyfield_info {
    u2 access_flags;
    u2 name_index;
    u2 descriptor_index;
    u2 attributes_count;
    attribute_info attributes[attributes_count];
}
```

属性结构：

1. `u2 access_flags;` 表示的是成员变量的修饰符；
2. `u2 name_index;` 表示的是成员变量的名称；
3. `u2 descriptor_index;` 表示的是成员变量的描述符；
4. `u2 attributes_count;` 表示的是成员变量的属性数量；
5. `attribute_info attributes[attributes_count];` 表示的是成员变量的属性信息；

13. methods_count（当前类的成员方法数）

`u2 methods_count;` 表示的是当前class中的成员方法个数。

14. methods[]（成员方法数组）

`method_info methods[methods_count];` 表示的是当前class中的所有成员方法，`method_info` 表示的是成员方法对象。

method_info数据结构：

```
copymethod_info {
   u2 access_flags;
   u2 name_index;
   u2 descriptor_index;
   u2 attributes_count;
   attribute_info attributes[attributes_count];
}
```

属性结构：

1. `u2 access_flags;` 表示的是成员方法的修饰符；
2. `u2 name_index;` 表示的是成员方法的名称；
3. `u2 descriptor_index;` 表示的是成员方法的描述符；
4. `u2 attributes_count;` 表示的是成员方法的属性数量；
5. `attribute_info attributes[attributes_count];` 表示的是成员方法的属性信息；

15. attributes_count （当前类的属性数）

`u2 attributes_count;` 表示当前class文件属性表的成员个数。

16. attributes[]（属性数组）

`attribute_info attributes[attributes_count];` 表示的是当前class文件的所有属性，`attribute_info` 是一个非常复杂的数据结构，存储着各种属性信息。

`attribute_info` 数据结构：

```
copyattribute_info {
   u2 attribute_name_index;
   u4 attribute_length;
   u1 info[attribute_length];
}
```

`u2 attribute_name_index;`表示的是属性名称索引，读取`attribute_name_index`值所在常量池中的名称可以得到属性名称。

属性对象

属性表是动态的，新的JDK版本可能会添加新的属性值。每一种属性的数据结构都不相同，所以读取到属性名称后还需要根据属性的类型解析不同属性表中的值。比如`Code Attribute`中存储了类方法的异常表、字节码指令集、属性信息等重要信息。