1. 概述

HotSpot是基于C++实现的，C++是面向对象的语言，本身具备面向对象基本特征，所以在Java中的对象表示，最简单做法是为每一个java类生成一个C++类与之对应。

但是HotSpot JVM是设计了一个OOP-Klass Model.这里的OOP指的是普通对象指针。

（在JVM中，用OOP—Class模型来表示java对象）

1. OOP-Klass模型
   1. Mark word Klass

Klass简单来说是Java类在HotSpot中的C++对等体，用来描述Java类

Klass主要有两个功能：

1. 实现语言层面的Java类
2. 实现Java对象的分发功能
   1. OOP

Klass是在class文件在加载过程创建的，OOP则是在Java程序运行过程中new对象创建的。

一个oop对象包含以下几个部分：

1.对象头：

：主要存储对象运行时记录信息，如hashcode，GC分代年龄，锁状态标志，线程ID，时间戳等

元数据指针：指向方法区的instanceKlass实例

2.实例数据

**在JVM中，使用了OOP-KLASS模型来表示java对象，即：**

**1.jvm在加载class时，会创建instanceKlass，表示其元数据，包括常量池、字段、方法等，存放在方法区；instanceKlass是jvm中的数据结构；**

**2.在new一个对象时，jvm创建instanceOopDesc，来表示这个对象，存放在堆区，其引用，存放在栈区；它用来表示对象的实例信息，看起来像个指针实际上是藏在指针里的对象；instanceOopDesc对应java中的对象实例；**

**3.HotSpot并不把instanceKlass暴露给Java，而会另外创建对应的instanceOopDesc来表示java.lang.Class对象，并将后者称为前者的“Java镜像”，klass持有指向oop引用(\_java\_mirror便是该instanceKlass对Class对象的引用)；**

**4.要注意，new操作返回的instanceOopDesc类型指针指向instanceKlass，而instanceKlass指向了对应的类型的Class实例的instanceOopDesc；有点绕，简单说，就是Person实例——>Person的instanceKlass——>Person的Class。**

**instanceOopDesc，只包含数据信息，它包含三部分：**

**1. 对象头，也叫Mark Word，主要存储对象运行时记录信息，如hashcode, GC分代年龄，锁状态标志，线程ID，时间戳等;**

**2. 元数据指针，即指向方法区的instanceKlass实例 （虚拟机通过这个指针来群定这个对象是哪个类的实例。）**

**3. 实例数据;**

**4. 另外，如果是数组对象，还多了一个数组长度**