初始化过程

1. java的对象都是通过new指令开始，首先会根据指令参数【操作数】在常量池中定位到一个类的符号引用
2. 如果没有定位到这个符号引用，那么这个类就没有被加载，就需要jvm进行类的加载
3. 符号引用解析完成后，jvm会为对象在对上创建分配内存[hotspot虚拟机实现Java对象分为三个部分：对象头、实例字段、和对齐填充字段]，这个时候对象的大小已经被定义
4. 对象分配完后，jvm会将该内存（除了对象头区域外）进行零值初始化。这就是结束为什么java的属性字段无需显示初始化就能被使用，而方法的局部变量却必须要显示初始化以后才能访问
5. 然后调用对象的构造函数，调用过程会一直上溯到object类

对象的组成部分

1.对象头

（1）对象头用于存储对象自身运行时数据，对象头包括了对象的hash码，对象的gc分代年龄、锁状态标志、线程持有的锁、偏向线程id、偏向时间戳，官方称为【Mark Word】

（2）另一部分是类型指针，即对象指向它的类元数据指针，虚拟机通过这个指针来确定这个对象是哪个类的实例。然而，并不是查找对象的元数据一定要通过对象本身，比如数组对象的对象头中必须保存记录数据组长度的数据，因此数组元数据中无法确定数组的大小，但是从普通对象的元数据信息可以确定对象的大小

2.实例数据

本部分存储的是对象真正有效的信息，存储着自身定义的和从父类集成下来的实例字段。字段存储顺序会受虚拟机的分配策略和字段在java源码中定义的顺序影响

3. 对其填充

因为Java对象的大小必须是8的倍数，因此使用对其模型进行填充，非必须

所以如何却行一个对象的大小就专业那个可以确定了

普通对象：8字节的对象头 + 4 / 8字节的对象指针 + 数据区 （包括父类）+ padding（8的倍数）

数组对象：8字节的对象头 + 4 / 8字节的对象指针 + 4 字节素组长度 + 数据区 + padding（8的倍数）
其中4/8字节的对象指针主要是独享是否开启了指针压缩，jdk1.6以后默认开启，因此jdk1.8默认是4

```
private static class ObjcectA {
    String a;
    int b;
    byte c;
    byte d;
    int e;
    Object f;
    byte g;
}
```

size = 8 + 4 + 4 （a）+ 4 (b) + 1(c) + 1(d) + 2 (padding) +4 (e) + 4 (f point) + 1 (g) + 7 (padding)

对象引用
（1）句柄引用，句柄池会在堆上分配一块内存，切每个句柄包括两个部分，一个是指向堆上的是对象的指针，一个是指向对象类型的指针，引用指向句柄
（2）直接指针访问，指针直接指向堆中的对象，堆中的对象会包括对象类型数据
优缺点：
句柄引用，当对象实例发生改变的时候，只需要需改掉句柄对实例对象物理地址的引用即可，修改比较方便与快捷
缺点，但是确定是对象的分配比较频繁，因此获取对象是一个频繁的动作，使用直接指针访问，这样的性能会更好，因此hot spot使用的方式是直接指针引用