链滴

Java 对象头

作者：614756773

原文链接：https://ld246.com/article/1584950083257

来源网站：链滴

以下的内容都是基于 32 位 JDK，或者 64 位 JDK 并且开启了指针压缩
如果使用 64 位 JDK 并且没有开启指针压缩，那么 MarkWord 和 Klass 都会变为 8 字节

一个对象在内存中的布局

对象在内存中的布局可以分为两种情况，一种是普通对象，一种是数组对象

在 jvm 中所有的对象大小都是 8 字节的整数倍，所以对象会有一个对其填充数据

数组对象的对象头除了 MarkWord、Klass 指针外还有一个 4 字节的 length 用来表示数组的大小

在 32 位的系统中 MarkWord 、Klass 都是 4 字节

但是在 64 位的系统中 MarkWord 、Klass 都占用 8 字节

原文链接：Java 对象头

若在 64 为的系统中开启了指针压缩 +UseCompressedOops 则 MarkWord 占 8 字节，Klass 占 4 节

例子：

```
// 64位系统，开启了指针压缩
class Student{
    private String name;
    private boolean boy;
}

Student s = new Student();

现在我们可以计算出s的大小为：
12(对象头大小) + 4(name的引用长度) + 1(boolean为基本类型只有1字节) + 7(padding)= 24
```

Mark Word

锁状态	25bit		4bit	1bit	2bit
	23bit	2bit		是否偏向锁	锁标志位
无锁态	对象的hashCode		分代年龄	0	01
轻量级锁	指向栈中锁记录的指针				00
重量级锁	指向互斥量(重量级锁)的指针				10
GC标记	空				11
偏向锁	线程ID	Epoch	分代年龄	1	01

计算对象大小

基于上面的知识，我们可以使用 unsafe 来计算一个对象的大小，代码：

```
public class ObjectSizeUtil {

    private static Map<Class, Long> map = new HashMap<>(16);

    static {
        map.put(byte.class, 1L);
        map.put(char.class, 1L);
        map.put(boolean.class, 1L);
        map.put(short.class, 2L);
        map.put(int.class, 4L);
        map.put(float.class, 4L);
        map.put(double.class, 8L);
        map.put(long.class, 8L);
    }

    /**
     * 获取对象的大小，单位字节
```

```
     * 不会递归计算对象的大小
     */
    public static long sizeOf(Object o) {
        Unsafe unsafe = reflectionGetUnsafe();
        Field[] fields = o.getClass().getDeclaredFields();
        long size;

        long position = 0;
        Field lastField = fields[fields.length - 1];
        for (Field field : fields) {
            // 静态变量不属于对象的大小，不计算
            if (Modifier.isStatic(field.getModifiers())) continue;
            long l = unsafe.objectFieldOffset(field);
            if (l >= position) {
                position = l;
                lastField = field;
            }
        }

        Class lastFieldType = lastField.getType();
        if (!lastFieldType.isPrimitive()) {
            size = position + 4;
        } else {
            size = position + primitiveSize(lastFieldType);
        }
        return padding(size);
    }

    /**
     * 将结果对齐，因为在jvm中对象大小都是8字节的倍数
     * @return 对象的大小，单位字节
     */
    private static long padding(long size) {
        if (size % 8 == 0) {
            return size;
        }
        return (size / 8 + 1) * 8;
    }

    private static long primitiveSize(Class lastFieldClass) {
        return map.get(lastFieldClass);
    }

    /**
     * 使用反射的方式获取Unsafe
     */
    private static Unsafe reflectionGetUnsafe() {
        try {
            Field field = Unsafe.class.getDeclaredField("theUnsafe");
            field.setAccessible(true);
            return (Unsafe) field.get(null);
        } catch (NoSuchFieldException | IllegalAccessException e) {
            e.printStackTrace();
        }
```

```
            throw new IllegalArgumentException("未知错误");
        }
    }
```

我这份代码只能计算当前对象的大小，不会去递归计算引用的对象，而且没有包含计算数组的情况，**少应该加上 String 成员变量大小的递归计算的。。。我忘了**

当然，如果要计算数组对象大小也很简单咯，在外部遍历数组，多次调用该方法即可

计算原理

使用 unsafe 获取最后一个字段的偏移位置，然后再加上最后一个字段的大小就等于对象的大小