问题产生

- 观察下面程序的输出结果

```
public class IntegerAndInt {
    public static void main(String[] args) {
        Integer a = 1;
        int b = 1;
        Integer c = Integer.valueOf(1);
        Integer d = new Integer(1);
        System.out.println(a == b);//true
        System.out.println(a == c);//true
        System.out.println(c == d);//false
        System.out.println(b == d);//true
    }
}
```

为何他们的值如上述所示呢？

先介绍一下java中Integer的对象池就会恍然大悟

- java 在编译代码的时候会把 `integer a = 1`、 `integer d = new integer (1)` 翻译成 `integer x = integer.valueof (1)` 所以关键在于 `valueof ()` 函数

1. `valueof()` 源码

```
public static Integer valueOf(int i) {
    if (i >= IntegerCache.low && i <= IntegerCache.high)
        return IntegerCache.cache[i + (-IntegerCache.low)];
    return new Integer(i);
}
```

2. `IntegerCache` (缓存池数组)

```
private static class IntegerCache {
    static final int low = -128;
    static final int high;
    static final Integer cache[];

    static {
        // high value may be configured by property
        int h = 127;
        String integerCacheHighPropValue =
            sun.misc.VM.getSavedProperty("java.lang.Integer.IntegerCache.high");
        if (integerCacheHighPropValue != null) {
            try {
                int i = parseInt(integerCacheHighPropValue);
                i = Math.max(i, 127);
                // Maximum array size is Integer.MAX_VALUE
```

```
                h = Math.min(i, Integer.MAX_VALUE - (-low) -1);
            } catch( NumberFormatException nfe) {
                // If the property cannot be parsed into an int, ignore it.
            }
        }
        high = h;

        cache = new Integer[(high - low) + 1];
        int j = low;
        for(int k = 0; k < cache.length; k++)
            cache[k] = new Integer(j++);

        // range [-128, 127] must be interned (JLS7 5.1.7)
        assert IntegerCache.high >= 127;
    }

    private IntegerCache() {}
}
```

- 对象池原理

看到上方cache缓存数组，在static静态块中，而Integer类又在java.lang包（jdk目录中）下，在java编译时就加载了类，所以静态块中的数组在编译的时候就创建了，从命名可以就可以看出这个数组是作为缓存数组，保存着从[-128,127]的Integer对象。

- 创建对象过程

只要 Integer 对象的值在 [-128,127] 范围内，都是从这个对象池中取(同一对象)。所以只要是这个范围内的 Integer 对象，只要值相同，就是同一个对象。那么 == 的结果，就是 true。超过了这个范围，则会 new 新的 Integer 对象，尽管值相同，但是已经是不同的对象了。

- **这下在来看最开始程序的结果就可以很容易理解了**

```
public class IntegerAndInt {
    public static void main(String[] args) {
        Integer a = 1;//在对象池中获取
        int b = 1;//定义基本数据类型
        Integer c = Integer.valueOf(1);//对象池中获取
        Integer d = new Integer(1);//新建的对象
        System.out.println(a == b);//true
        System.out.println(a == c);//true
        System.out.println(c == d);//false
        System.out.println(b == d);//true
    }
}
```

1. `a==b`，`b==d`输出结果为`true`，因为 `Integer` 和 `int` 比都会自动拆箱（jdk1.5 以上）

2. `a==c` 输出结果为 `true`,因为a,c为对象池中的相同对象，如果a,c不在[-128,127]中，结果则为false，因为从源码可以看到会新`new`一个对象
3. c==d `输出结果为`false`，和`String`一样，在内存中的对象不一样，输出结果为`false`.