【Java基础】关于`equals()`方法和`hashCode()`方法

关于`equals()`方法

在未被重写情况下，equals()方法用于两个对象内存地址的比较。equals是object类的方法，可重写做特殊化的比较，比如String类型的比较。

关于`hashCode()`方法

同样是object类的方法。默认返回对象内存地址计算出的int值。由于重写算法的原因，两个***不同对象的hashcode值可能相同***。

hashcode有什么用

用于支持hash表，如HashSet，HashMap。判断key值是否相等时首先判断key的hash值，如果hashcode()返回不同值，则认为是不同key。

HashMap底层原理

HashMap在创建时分配一个一定大小的`bucket数组`，每个bucket数组按照`链表结构`储存值，HashMap执行put(key, value)操作时，在`bucket数组`中寻找`hashCode()`值对应的backet位置。如果此位置没有值，则放入当前值。否则依次遍历bucket链表调用`equals()`方法比较是否相等，相等则覆盖，否则加入链表末尾。

同理，HashMap在执行`get(key)`方法是先调用`hashCode()`方法找到相对的bucket再遍历bucket链表调用`equals()`方法比较key值。

另外，如果当前为map分配的空间已满，程序会自动另外继续分配当前两倍大小的空间。

HashMap的工作原理解析

为什么重写`equals()`方法时要重写`hashCode()`方法

如果值重写equals()方法，尽管equals方法返回true，显示是同一个对象，在其作为key值操作map时，由于其hashcode()值不等，也会被当作不同的对象做不同的key

重写hashCode() 方法及 equals() 方法的代码实现

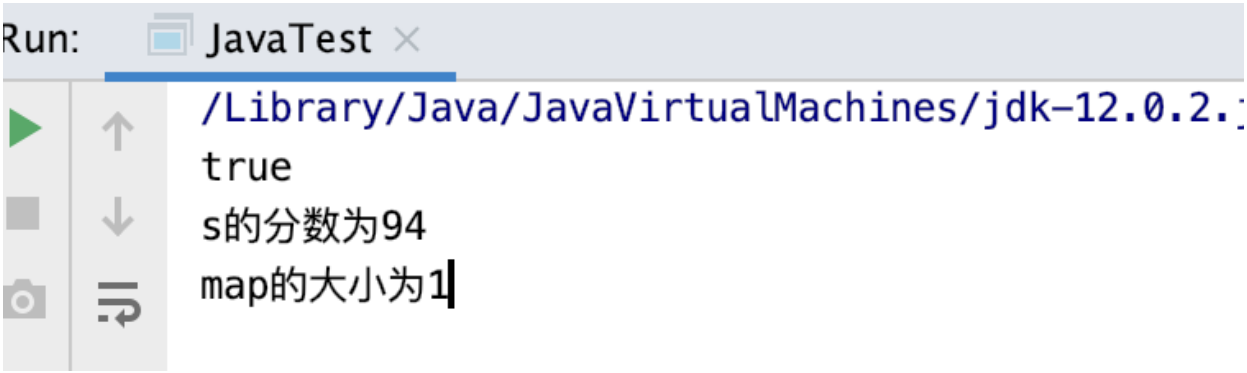
```
public class JavaTest {
    public static void main(String[] args) {
        Student s = new Student(1);
        Student s2 = new Student(1);
        System.out.println(s.equals(s2));
        Map<Student, Integer> map = new HashMap<>();
        map.put(s, 93);
        map.put(s2, 94);
        System.out.println("s的分数为" + map.get(s));
        System.out.println("map的大小为" + map.size());
    }
}

class Student{
    int sid;
    int name;
    public Student(int id){
        this.sid = id;
    }

    @Override
    public boolean equals(Object obj) {
        if(obj != null && obj instanceof Student){
            int oid = ((Student) obj).sid;
            return oid == sid;
        }
        return false;
    }

    @Override
    public int hashCode() {
        int hash = 17;
        hash = hash * 31 + sid;
        return hash;
    }
}
```

输出:


Run:
JavaTest
/Library/Java/JavaVirtualMachines/jdk-12.0.2.
true
s的分数为94
map的大小为1