hashCode和equal和==

设计hashCode()时最重要的因素就是：无论何时，对同一个对象调用hashCode()都应该产生同样的值。如果在讲一个对象用put()添加进HashMap时产生一个hashCdoe值，而用get()取出时却产生了另一个hashCode值，那么就无法获取该对象了。所以如果你的hashCode方法依赖于对象中易变的数据，用户就要当心了，因为此数据发生变化时，hashCode()方法就会生成一个不同的散列码。

equals方法：

如果我们需要通过比较属性来判断引用的对象是否相等，就需要重写equals方法。

equals方法是Object类中定义的方法，Object类是所有类的父类，因此我们可以重写equals方法。

equals方法的判断流程一般是：

1. 判断是否是引用同一个对象，如果是，返回true, 否则继续
2. 判断传入的引用是否为NULL, 如果是，返回false
3. 判断传入的引用的类型和调用equals方法的对象类型是否一致，如果不一致，返回false
4. 将传入的对象强转
5. 比较相关属性的值，返回结果

```
import java.util.Objects;
public class A {
    int i;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        A a = (A) o;
        return i == a.i;
    }
    @Override
    public int hashCode() {
        return Objects.hash(i);
    }
}
```

在设计hashCode方法和equals方法的时候，如果对象中的数据易变，则最好在equals方法和hashCode方法中不要依赖于该字段。

可以直接根据hashcode值判断两个对象是否相等吗？肯定是不可以的，因为不同的对象可能会生成相同的hashcode值。虽然不能根据hashcode值判断两个对象是否相等，但是可以直接根据hashcode值判断两个对象不等，如果两个对象的hashcode值不等，则必定是两个不同的对象。如果要判断两个对象是否真正相等，必须通过equals方法。

也就是说对于两个对象，如果调用equals方法得到的结果为true，则两个对象的hashcode值必定相等；

- 如果equals方法得到的结果为false，则两个对象的hashcode值不一定不同；
- 如果两个对象的hashcode值不等，则equals方法得到的结果必定为false；
- 如果两个对象的hashcode值相等，则equals方法得到的结果未知。

在设计hashCode方法和equals方法的时候，如果对象中的数据易变，则最好在equals方法和hashCode方法中不要依赖于该字段。

equals() 和 "==" 有什么区别？

equals()方法用来比较的是两个对象的内容是否相等，由于所有的类都是继承自java.lang.Object类的，所以适用于所有对象，如果没有对该方法进行覆盖的话，调用的仍然是Object类中的方法，而Object中的equals方法返回的却是==的判断；

"==" 比较的是变量(栈)内存中存放的对象的(堆)内存地址，用来判断两个对象的地址是否相同，即是否是指相同一个对象。

下面根据"类是否覆盖equals()方法"，将它分为2类。

- 若某个类没有覆盖equals()方法，当它的通过equals()比较两个对象时，实际上是比较两个对象是不是同一个对象。这时，等价于通过“==”去比较这两个对象。
- 我们可以覆盖类的equals()方法，来让equals()通过其它方式比较两个对象是否相等。通常的做法是：若两个对象的内容相等，则equals()方法返回true；否则，返回fasle。