使用HashMap，如果key是自定义的类，就必须重写hashCode（）和equals方法。

hashCode（）和equals（）都继承于object，在object类中的定义为：

```
public boolean equals(Object obj){
    return (this == obj);
}
public native int hashCode();
```

1.hashcode（）和equals（）是在哪里被用到的？什么用的？

hashmap基于散列函数，以数组和链表的方式实现的。

而对于每一个对象，通过其hashCode（）方法可为其生成一个整形值（散列码），该整形值被处理后，将会作为数组下标，存放该对象所对应的Entry。

equals（）方法则是在hashmap中插入值或查询时会使用到。当HashMap中插入值或者查询值对应的散列码和数组中的散列码相等时，则会通过equals方法比较key值是否相等。

2.为什么要重新hashCode（）和equals（）方法？

HashMap中，如果要比较key是否相等，要同时使用这两个函数，如果不重写，equals和hashCode（）方法默认比对象存储的地址，两个一样的对象地址不同，但是他们的key应该是相同的，如果不重写，就会判断为不同

没有重写

```
public static void main(String[] args) {

Map<PhoneNumber, String> map =new  HashMap<PhoneNumber, String>();
PhoneNumber phoneNumber1=new PhoneNumber();
phoneNumber1.setPhoneNumber(111);
phoneNumber1.setPrefix(111);
PhoneNumber phoneNumber2=new PhoneNumber();
phoneNumber2.setPhoneNumber(222);
phoneNumber2.setPrefix(222);
map.put(phoneNumber1, "111");
map.put(phoneNumber2, "222");

System.out.println(map.get(phoneNumber1));
System.out.println(map.get(phoneNumber2));

PhoneNumber phoneNumber3=new PhoneNumber();
//参数内容和phoneNumber2一样
phoneNumber3.setPhoneNumber(222);
phoneNumber3.setPrefix(222);
System.out.println(map.get(phoneNumber3));

 }
```

输出结果

```
111
222
null
```

重写了

在PhoneNumber 类中重写equals()和hashCode()方法；

```
@Override
public boolean equals(Object o)
{
    if(this == o)
    {
        return true;
    }
    if(!(o instanceof PhoneNumber))
    {
        return false;
    }
    PhoneNumber pn = (PhoneNumber)o;
    return pn.prefix == prefix && pn.phoneNumber == phoneNumber;
}
```

```
@Override
public int hashCode()
{
    int result = 17;
    result = 31 * result + prefix;
    result = 31 * result + phoneNumber;
    return result;
}
```

输出结果

```
111
222
222
```