Object #hashCode（）方法，其中@see java.lang.Object#equals(java.lang.Object)；@see java.lang.System#identityHashCode；说明equals，hashCode，identityHashCode之间存在着一定联系。

Object#hashCode()

```
/**
 * Returns a hash code value for the object. This method is
 * supported for the benefit of hash tables such as those provided by
 * {@link java.util.HashMap}.
 * <p>
 * The general contract of {@code hashCode} is:
 * <ul>
 * <li>Whenever it is invoked on the same object more than once during
 * an execution of a Java application, the {@code hashCode} method
 * must consistently return the same integer, provided no information
 * used in {@code equals} comparisons on the object is modified.
 * This integer need not remain consistent from one execution of an
 * application to another execution of the same application.
 * <li>If two objects are equal according to the {@code equals(Object)}
 * method, then calling the {@code hashCode} method on each of
 * the two objects must produce the same integer result.
 * <li>It is <em>not</em> required that if two objects are unequal
 * according to the {@link java.lang.Object#equals(java.lang.Object)}
 * method, then calling the {@code hashCode} method on each of the
 * two objects must produce distinct integer results. However, the
 * programmer should be aware that producing distinct integer results
 * for unequal objects may improve the performance of hash tables.
 * </ul>
 * <p>
 * As much as is reasonably practical, the hashCode method defined by
 * class {@code Object} does return distinct integers for distinct
 * objects. (This is typically implemented by converting the internal
 * address of the object into an integer, but this implementation
 * technique is not required by the
 * Java&trade; programming language.)
 *
 * @return a hash code value for this object.
 * @see java.lang.Object#equals(java.lang.Object)
 * @see java.lang.System#identityHashCode
```

```
 */
public native int hashCode();
```

System#identityHashCode(Object x);

```
/**
 * Returns the same hash code for the given object as
 * would be returned by the default method hashCode(),
 * whether or not the given object's class overrides
 * hashCode().
 * The hash code for the null reference is zero.
 *
 * @param x object for which the hashCode is to be calculated
 * @return the hashCode
 * @since JDK1.1
 */
public static native int identityHashCode(Object x);
```

identityHashCode(Object x)和hashCode（）

identityHashCode(Object x)和hashCode（）都是native方法，native方法请参考本地方法；

以java8为例,JDK中代码样式可以看出，两个java方法使用的是同一个c++的方法JVM_IHashCode()，所以结果是一样的。

Object#hashCode

```
static JNINativeMethod methods[] = {
 {"hashCode", "()I", (void *)&JVM_IHashCode},
 {"wait", "(J)V", (void *)&JVM_MonitorWait},
 {"notify", "()V", (void *)&JVM_MonitorNotify},
 {"notifyAll", "()V", (void *)&JVM_MonitorNotifyAll},
 {"clone", "()Ljava/lang/Object;", (void *)&JVM_Clone},
};

```

System#identityHashCode

```
JNIEXPORT jint JNICALL
Java_java_lang_System_identityHashCode(JNIEnv *env, jobject this, jobject x)
{
 return JVM_IHashCode(env, x);
}
```

接下还是一段代码的展示equal，“==”,hashCode(),identityHashCode()的区别，选取Object类，原始类型boolean，int，原始类型包装类Boolean类,Integer，以及String类进行说明。

Object

```
1
2  /**********************************************************************
***
3   * @Title: EqualsDome
4   * @Package com.base
5   * @Description: 实验比较equals
6   * @author shimingda
7   * @date 2020/1/9
8   * @version V1.0
9
**********************************************************************
*/
10  public class EqualsDome
11  {
12   public static void main(String[] args)
13   {
14   Object a1="a";
15   Object a2="a";
16   Object b1=new Object();
17   Object b2=new Object();
18
19   System.out.println("a1==a2:"+(a1==a2));
20   System.out.println("b1==b2:"+(b1==b2));
21
22   System.out.println("a1.equals(a2):"+(b1.equals(b2)));
23   System.out.println("b1.equals(b2):"+(b1.equals(b2)));
24
25   System.out.println("a1.hashCode() is : "+a1.hashCode()+" System.identit
yHashCode(a1) is : "+ System.identityHashCode(a1));
26   System.out.println("a2.hashCode() is : "+a2.hashCode()+" System.identit
yHashCode(a2) is : "+ System.identityHashCode(a2));
27   System.out.println("b1.hashCode() is : "+b1.hashCode()+" System.identit
yHashCode(b1) is : "+ System.identityHashCode(b1));
28   System.out.println("b2.hashCode() is : "+b2.hashCode()+" System.identit
yHashCode(b2) is : "+ System.identityHashCode(b2));
29   }
30  }
```

```
----------------------------------------------------
a1==a2:true
b1==b2:false
a1.equals(a2):false
b1.equals(b2):false
a1.hashCode() is : 97 System.identityHashCode(a1) is : 1163157884
a2.hashCode() is : 97 System.identityHashCode(a2) is : 1163157884
b1.hashCode() is : 1956725890 System.identityHashCode(b1) is : 1956725890
b2.hashCode() is : 356573597 System.identityHashCode(b2) is : 356573597
```

int和Integer

```


/**********************************************************************
 * @Title: EqualsDome
 * @Package com.base
 * @Description: 实验比较equals
 * @author shimingda
 * @date 2020/1/9
  * @version V1.0
  **********************************************************************/
public class EqualsDome
{
 public static void main(String[] args)
 {
 int a1=130;
 int a2=130;
 Integer b1=130;
 Integer b2=130;
 Integer c1=new Integer(130);
 Integer c2=new Integer(130);


 System.out.println("a1==a2:"+(a1==a2));
 System.out.println("a1==b1:"+(a1==b1));
 System.out.println("b1==b2:"+(b1==b2));
 System.out.println("bi==c1:"+(b1==c1));
 System.out.println("c1==c2:"+(c1==c2));
```

```
29
30   System.out.println("b1.equals(b2):"+(b1.equals(b2)));
31   System.out.println("bi.equals(c1):"+(b1.equals(c1)));
32   System.out.println("c1.equals(c2):"+(c1.equals(c2)));
33
34   System.out.println("System.identityHashCode(a1) is : "+ System.identity
HashCode(a1));
35   System.out.println("System.identityHashCode(a2) is : "+ System.identity
HashCode(a2));
36   System.out.println("b1.hashCode() is : "+b1.hashCode()+" System.identit
yHashCode(b1) is : "+ System.identityHashCode(b1));
37   System.out.println("b2.hashCode() is : "+b2.hashCode()+" System.identit
yHashCode(b2) is : "+ System.identityHashCode(b2));
38   System.out.println("c1.hashCode() is : "+c1.hashCode()+" System.identit
yHashCode(c1) is : "+ System.identityHashCode(c1));
39   System.out.println("c2.hashCode() is : "+c2.hashCode()+" System.identit
yHashCode(c2) is : "+ System.identityHashCode(c2));
40   }
41  }
42  ------------------------------------------------------------
43  结果:
44  a1==a2:true
45  a1==b1:true
46  b1==b2:false
47  bi==c1:false
48  c1==c2:false
49  b1.equals(b2):true
50  bi.equals(c1):true
51  c1.equals(c2):true
52  System.identityHashCode(a1) is : 1163157884
53  System.identityHashCode(a2) is : 1956725890
54  b1.hashCode() is : 130 System.identityHashCode(b1) is : 356573597
55  b2.hashCode() is : 130 System.identityHashCode(b2) is : 1735600054
56  c1.hashCode() is : 130 System.identityHashCode(c1) is : 21685669
57  c2.hashCode() is : 130 System.identityHashCode(c2) is : 2133927002
```

boolean和Boolean

```
1  package com.base;
2
3  import sun.applet.Main;
4
```

```java
/**********************************************************************
 * @Title: EqualsDome
 * @Package com.base
 * @Description: 实验比较equals
 * @author shimingda
 * @date 2020/1/9
 * @version V1.0
 **********************************************************************/
public class EqualsDome
{
 public static void main(String[] args)
 {
 boolean a1=true;
 boolean a2=true;

 Boolean b1=true;
 Boolean b2=true;
 Boolean c1=new Boolean(true);
 Boolean c2=new Boolean(true);

 System.out.println("a1==a2:"+(a1==a2));
 System.out.println("a1==b1:"+(a1==b1));
 System.out.println("b1==b2:"+(b1==b2));
 System.out.println("bi==c1:"+(b1==c1));
 System.out.println("c1==c2:"+(c1==c2));

 System.out.println("b1.equals(b2):"+(b1.equals(b2)));
 System.out.println("bi.equals(c1):"+(b1.equals(c1)));
 System.out.println("c1.equals(c2):"+(c1.equals(c2)));

 System.out.println("System.identityHashCode(a1) is : "+ System.identityHashCode(a1));
 System.out.println("System.identityHashCode(a2) is : "+ System.identityHashCode(a2));
 System.out.println("b1.hashCode() is : "+b1.hashCode()+" System.identityHashCode(b1) is : "+ System.identityHashCode(b1));
 System.out.println("b2.hashCode() is : "+b2.hashCode()+" System.identityHashCode(b2) is : "+ System.identityHashCode(b2));
 System.out.println("c1.hashCode() is : "+c1.hashCode()+" System.identityHashCode(c1) is : "+ System.identityHashCode(c1));
```

```
 System.out.println("c2.hashCode() is : "+c2.hashCode()+" System.identityHashCode(c2) is : "+ System.identityHashCode(c2));
  }
}
-----------------------------------------
a1==a2:true
a1==b1:true
b1==b2:true
bi==c1:false
c1==c2:false
b1.equals(b2):true
bi.equals(c1):true
c1.equals(c2):true
System.identityHashCode(a1) is : 1163157884
System.identityHashCode(a2) is : 1163157884
b1.hashCode() is : 1231 System.identityHashCode(b1) is : 1163157884
b2.hashCode() is : 1231 System.identityHashCode(b2) is : 1163157884
c1.hashCode() is : 1231 System.identityHashCode(c1) is : 1956725890
c2.hashCode() is : 1231 System.identityHashCode(c2) is : 356573597

```

String

```

/**********************************************************************
 * @Title: EqualsDome
 * @Package com.base
 * @Description: 实验比较equals
 * @author shimingda
 * @date 2020/1/9
 * @version V1.0

**********************************************************************/
public class EqualsDome
{
 public static void main(String[] args)
 {
 String a1="a";
 String a2="a";
 String b1=new String("a");
```

```
17   String b2=new String("a");
18
19
20   System.out.println("a1==a2:"+(a1==a2));
21   System.out.println("a1==b1:"+(a1==b1));
22   System.out.println("b1==b2:"+(b1==b2));
23
24   System.out.println("a1.equals(a2):"+(b1.equals(b2)));
25   System.out.println("ai.equals(b1):"+(b1.equals(b1)));
26   System.out.println("b1.equals(b2):"+(b1.equals(b2)));
27
28   System.out.println("a1.hashCode() is : "+a1.hashCode()+" System.identit
yHashCode(a1) is : "+ System.identityHashCode(a1));
29   System.out.println("a2.hashCode() is : "+a2.hashCode()+" System.identit
yHashCode(a2) is : "+ System.identityHashCode(a2));
30   System.out.println("b1.hashCode() is : "+b1.hashCode()+" System.identit
yHashCode(b1) is : "+ System.identityHashCode(b1));
31   System.out.println("b2.hashCode() is : "+b2.hashCode()+" System.identit
yHashCode(b2) is : "+ System.identityHashCode(b2));
32
33   }
34 }
35 ----------------------------------------------------------------------
36 a1==a2:true
37 a1==b1:false
38 b1==b2:false
39 a1.equals(a2):true
40 ai.equals(b1):true
41 b1.equals(b2):true
42 a1.hashCode() is : 97 System.identityHashCode(a1) is : 1163157884
43 a2.hashCode() is : 97 System.identityHashCode(a2) is : 1163157884
44 b1.hashCode() is : 97 System.identityHashCode(b1) is : 1956725890
45 b2.hashCode() is : 97 System.identityHashCode(b2) is : 356573597
```

对于以上实验进行总结

1.当一个类没有重写Object类的hashCode()方法时，它的hashCode和identityHashCode是一致的

2.当一个类重写了Object类的hashCode()方法时，它的hashCode则有重写的实现逻辑决定，此时的hashCode值一般就不再和对象本身的内部地址有相应的哈希关系了

3.identityHashCode值相等，"=="结果一定是true;但是“==”为true时，identityHashCode不一定相等，样例int大于127时。

4..identityHashCode值相等，equal()不一定是true;

5..hashCode值相等时，"=="结果不一定是true