Overriding equals method in Java

Consider the following Java program:

```
class Complex {
    private double re, im;

    public Complex(double re, double im) {
        this.re = re;
        this.im = im;
    }
}

// Driver class to test the Complex class
public class Main {
    public static void main(String[] args) {
        Complex c1 = new Complex(10, 15);
        Complex c2 = new Complex(10, 15);
        if (c1 == c2) {
            System.out.println("Equal ");
        } else {
            System.out.println("Not Equal ");
        }
    }
}
```

Output:

```
Not Equal
```

The reason for printing “Not Equal” is simple: when we compare c1 and c2, it is checked whether both c1 and c2 refer to same object or not (object variables are always references in Java). c1 and c2 refer to two different objects, hence the value (c1 == c2) is false. If we create another reference say c3 like following, then (c1 == c3) will give true.

```
Complex c3 = c1;  // (c3 == c1) will be true
```

So, how do we check for equality of values inside the objects? All classes in Java inherit from the Object class, directly or indirectly (See point 1 of this). The Object class has some basic methods like clone(), toString(), equals(),.. etc. We can override the equals method in our class to check whether two objects have same data or not.

```
class Complex {

    private double re, im;

    public Complex(double re, double im) {
        this.re = re;
        this.im = im;
    }

    // Overriding equals() to compare two Complex objects
    @Override
    public boolean equals(Object o) {

        // If the object is compared with itself then return true
        if (o == this) {
            return true;
        }

        /* Check if o is an instance of Complex or not
          "null instanceof [type]" also returns false */
        if (!(o instanceof Complex)) {
            return false;
        }

        // typecast o to Complex so that we can compare data members
        Complex c = (Complex) o;

        // Compare the data members and return accordingly
        return Double.compare(re, c.re) == 0
                && Double.compare(im, c.im) == 0;
    }
}

// Driver class to test the Complex class
public class Main {

    public static void main(String[] args) {
        Complex c1 = new Complex(10, 15);
        Complex c2 = new Complex(10, 15);
        if (c1.equals(c2)) {
            System.out.println("Equal ");
        } else {
            System.out.println("Not Equal ");
        }
    }
}
```

Output:


```
Equal
```

As a side note, when we override equals(), it is recommended to also override the hashCode() method. If we don't do so, equal objects may get different hash-values; and hash based collections, including HashMap, HashSet, and Hashtable do not work properly (see this for more details). We will be coverig more about hashCode() in a separate post.

References:
Effective Java Second Edition

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

1 Top IT MNCs hiring freshers - CTC:25k-5Lac/Month Exp:3-15+
2 Upload Your Resume Today Start Your Job Search With Over 3 Lacs

Recommended Posts:

Exception Handling with Method Overriding in Java
Vector equals() Method in Java
Set equals() method in Java with Examples
ConcurrentLinkedDeque equals() method in Java with Example
LongBuffer equals() method in Java
MathContext equals() Method in Java
TreeSet equals() method in Java with Example
CopyOnWriteArraySet equals() method in Java
CharBuffer equals() method in Java
StringWriter equals() method in Java with Example
Map equals() method in Java with Examples
LinkedBlockingDeque equals() method in Java with Example
IdentityHashMap equals() Method in Java
ConcurrentSkipListSet equals() method in Java
Calendar.equals() method in Java