# Why should you override equals() and hashcode() methods in your class in Java

For example you created a class Student and now you want to compare two objects that have same content using equals()

```
Student s1 = new Student(10,"Rohan");
Student s2= new Student(10,"Rohan");

s1.equals(s2)    //returns false
```

why false?

Because the defualt implementation of the equals() method does only reference comparison.

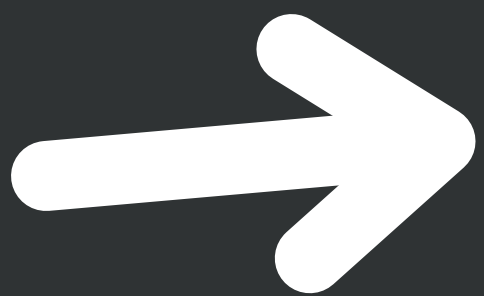
So if  the two reference variables are pointing to the same object then only this method will return true.

```
Student s1 = new Student(10,"Rohan");
 Student s2 = s1;

s1.equals(s2);    //return true
```

So by overriding equals(), we write the logic to compare two objects.

then why do we need to override hashcode()?



Hash based collections like Hashset, Hashtable, Hashmap uses the hashcode() of the objects to put and get them to/from a proper bucket

And this hashcode() method is implemented based on the implementation of the equals().

So if we override equals() method and don't override hashcode() method then these hash based collections will not work properly with our objects.

**There are some rules that need to be taken care of while overriding hashcode() method**

what are they?

1. Whenever it(hashcode) is invoked on the same object more than once during an execution of a Java application, the hashCode method must consistently return the same integer, provided no information used in 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.

2. If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.

3. It is not required that if two objects are unequal according to the equals(java.lang.Object) method, then calling the hashCode method on each of the two objects must produce distinct integer results.

Thanks for reading...