Why do I need to override the equals and hashCode methods in Java?

Collections such as HashMap and HashSet use the hashcode value of an object to determine how the object should be stored in the collection, and the hashcode is used again to help locate the object in the collection.

Hashing retrieval is a two-step process.

1. Find the right bucket (using hashCode())
2. Search the bucket for the right element (using equals() )

here is a small example why we should overrride equals() and hashcode().Consider an Employee class which has two fields age and name.

public class Employee {

String name;

int age;

public Employee(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

@Override

public boolean equals(Object obj) {

if (obj == this)

return true;

if (!(obj instanceof Employee))

return false;

Employee employee = (Employee) obj;

return employee.getAge() == this.getAge()

&& employee.getName() == this.getName();

}

// commented

/\* @Override

public int hashCode() {

int result=17;

result=31\*result+age;

result=31\*result+(name!=null ? name.hashCode():0);

return result;

}

\*/

}

Now create a class, insert Employee object to a HashSet and test whether that object is present or not.

public class ClientTest {

public static void main(String[] args) {

Employee employee = new Employee("rajeev", 24);

Employee employee1 = new Employee("rajeev", 25);

Employee employee2 = new Employee("rajeev", 24);

HashSet<Employee> employees = new HashSet<Employee>();

employees.add(employee);

System.out.println(employees.contains(employee2));

System.out.println("employee.hashCode(): " + employee.hashCode()

+ " employee2.hashCode():" + employee2.hashCode());

}

}

It will print

false

employee.hashCode(): 321755204 employee2.hashCode():375890482

Now uncomment hashcode() method , execute the same and the output would be

true

employee.hashCode(): -938387308 employee2.hashCode():-938387308

Now can you see why if two objects are considered equal, their hashcodes must also be equal? Otherwise, you'd never be able to find the object since the default hashcode method in class Object virtually always comes up with a unique number for each object, even if the equals() method is overridden in such a way that two or more objects are considered equal. It doesn't matter how equal the objects are if their hashcodes don't reflect that. So one more time: If two objects are equal, their hashcodes must be equal as well.