How to Compare Two Objects in Java

**Java Object** class is the super class of all the Java classes. All Java classes implements the Object class by default. The Java Object class provides the two important methods to **compare two**[**objects in Java**](https://www.javatpoint.com/object-and-class-in-java), i.e. **equals()** and **hashCode()** method. In this section, we will learn how **equals()** and **hashCode()** method works. Along with this, we will also learn **how to compare two objects in Java** with proper examples.

Java provides the two methods of the Object class to compare the objects are as follows:

* Java equals() Method
* Java hashCode() Method

Java equals() Method

The **equals()** method of the Object class compare the equality of two objects. The two objects will be equal if they share the same memory address.

**Syntax:**

1. **public** **boolean** equals(Object obj)

The method parses a reference object as a parameter. It returns true if the objects are equal, else returns false.

It is also possible that an object is equal to another given object, then the equals() method follow the **equivalence relation** to compare the objects.

* **Reflexive:** If **x** is a non-null reference, the calling of **x.equals(x)** must return true.
* **Symmetric:** If the two non-null references are **x** and **y, x.equals(y)** will return **true** if and only if **y.equals(x)** return **true**.
* **Transitive:** If the three non-null references are **x, y**, and **z, x.equals(z)** will also return **true** if **x.equals(y**) and **y.equals(z)** both returns **true**.
* **Consistent:** If the two non-null references are **x** and **y**, the multiple calling of **x.equals(y)** constantly returns either true or false. It does not provide any information used in the comparison.
* For any non-null reference **x, x.equals(null)** returns false.

AD

In short, for any non-null reference say **x** and **y**, it returns true if and only if both references refer to the same object.

**Remember:** When we override the equals() method, it is necessary to override the hashCode() method. Overriding follow the convention for the hashCode() method that states, the equal object must have equal hash code.

Example of equals() method

In the following example, we have created [constructor](https://www.javatpoint.com/java-constructor) of the **Double** and [**Long** class](https://www.javatpoint.com/java-long) and passes corresponding values, as an argument that stored in their objects, respectively.

After that, in the first println statement, we have invoked equals() method and parse an object y as a parameter that compares the object x and y. It returns **false** because x holds the double value and y holds the long value that is not equal.

Similarly, in the second println statement, we have invoked equals() method and parse the same value as in the constructor of the [Double class](https://www.javatpoint.com/java-double). It returns **true** because the object of double class i.e. x holds the same value as we have passed in the equals() method.

**ObjectComparisonExample.java**

1. **public** **class** ObjectComparisonExample
2. {
3. **public** **static** **void** main(String[] args)
4. {
5. //creating constructor of the Double class
6. Double x = **new** Double(123.45555);
7. //creating constructor of the Long class
8. Long y = **new** Long(9887544);
9. //invoking the equals() method
10. System.out.println("Objects are not equal, hence it returns " + x.equals(y));
11. System.out.println("Objects are equal, hence it returns " + x.equals(123.45555));
12. }
13. }

**Output:**

Objects are not equal, hence it returns false

Objects are equal, hence it returns true

Difference Between == Operator and equals() Method

In [Java](https://www.javatpoint.com/java-tutorial), the **==** operator compares that two references are identical or not. Whereas the **equals()** method compares two objects.

Objects are **equal** when they have the same state (usually comparing variables). Objects are **identical** when they share the class identity.

For example, the expression **obj1==obj2** tests the identity, not equality. While the expression **obj1.equals(obj2)** compares equality.

Java hashCode() Method

In Java, hash code is a 32-bit signed integer value. It is a unique id provided by [JVM](https://www.javatpoint.com/jvm-java-virtual-machine) to Java object. Each Java object is associated with the hash code. The hash code is managed by a hash-based data structure, such as HashTable, HashSet, etc.

**Remember:** When we override the equals() method, it is necessary to override the hashCode() method, also.

**Syntax:**

1. **public** **int** hashCode()

It returns a randomly generated hash code value of the object that is unique for each instance. The randomly generated value might change during the several executions of the program.

The general contract for hashCode is:

* When it is invoked more than once during the execution of an application, the hashCode() method will consistently return the same hash code (integer value). Note that the object should not be modified.
* If the two objects are equal according to the equals() method, then invoking the hashCode() method on these two objects must produce the same integer value.
* It is not necessary that if the two objects are unequal according to equals() method, then invoking the hashCode() method on these two objects may produce distinct integer value. It means that it can produce the same hash code for both objects.

Example of hashCode() method

In the following example, we have created two classes **Employee.java** and **HashCodeExample.java**.

AD

In the Employee class, we have defined two fields regno of type int and name of type String. After that, we have created a constructor of the Employee class and passes these two fields as a parameter.

To perform the comparison of objects, we have created a separate class named **HashCodeExample**. In this class, we have created two instances of the Employee class i.e. **emp1** and **emp2**. After that, we have invoked the hashCode() method using objects. We have stored the hash code value in the variable **a** and **b**, respectively.

**Employee.java**

1. **public** **class** Employee
2. {
3. **private** **int** regno;
4. **private** String name;
5. //constructor of Employee class
6. **public** Employee(**int** regno, String name)
7. {
8. **this**.name = name;
9. **this**.regno = regno;
10. }
11. **public** **int** getRegno()
12. {
13. **return** regno;
14. }
15. **public** **void** setRegno(**int** Regno)
16. {
17. **this**.regno = regno;
18. }
19. **public** String getName()
20. {
21. **return** name;
22. }
23. **public** **void** setName(String name)
24. {
25. **this**.name = name;
26. }
27. }

**HashCodeExample.java**

1. **public** **class** HashcodeExample
2. {
3. **public** **static** **void** main(String[] args)
4. {
5. //creating two instances of the Employee class
6. Employee emp1 = **new** Employee(918, "Maria");
7. Employee emp2 = **new** Employee(918, "Maria");
8. //invoking hashCode() method
9. **int** a=emp1.hashCode();
10. **int** b=emp2.hashCode();
11. System.out.println("hashcode of emp1 = " + a);
12. System.out.println("hashcode of emp2 = " + b);
13. System.out.println("Comparing objects emp1 and emp2 = " + emp1.equals(emp2));
14. }
15. }

**Output:**

hashcode of emp1 = 2398801145

hashcode of emp2 = 1852349007

Comparing objects emp1 and emp2 = false

AD

Overriding equals() Method

We can override the equals() method in the following way if we want to provide own implementation.

1. //overriding equals() method
2. @Override
3. **public** **boolean** equals(Object obj)
4. {
5. **if** (obj == **null**)
6. **return** **false**;
7. **if** (obj == **this**)
8. **return** **true**;
9. **return** **this**.getRegno() == ((Employee) obj). getRegno();
10. }

The above code snippet shows that two employees will be equal if they are stored in the same memory address or they have the same regno. When we run the program (HashCodeExample.java) with the above code snippet, we get the following output.

**Output:**

hashcode of emp1 = 2032578917

hashcode of emp2 = 1531485190

Comparing objects emp1 and emp2 = true