**Equality, Hashcode, Equals**

In Java, the equals() and hashCode() methods are fundamental for comparing objects and managing collections like HashMap and HashSet. Here’s a brief overview:

equals()**Method**

* **Purpose**: Determines if two objects are considered equal.
* **Default Behavior**: The default implementation in the Object class checks if two references point to the same object (i.e., ==).
* **Override**: To compare objects based on their data, you need to override this method.
* **Contract**:
  + **Reflexive**: a.equals(a) should return true.
  + **Symmetric**: a.equals(b) should return true if and only if b.equals(a) returns true.
  + **Transitive**: If a.equals(b) and b.equals(c), then a.equals(c) should return true.
  + **Consistent**: Multiple invocations of a.equals(b) should consistently return true or false, provided no information used in the comparison is modified.
  + **Non-null**: a.equals(null) should return false.

hashCode()**Method**

* **Purpose**: Provides an integer representation of the object, used in hashing-based collections.
* **Default Behavior**: The default implementation in the Object class returns a unique integer for each object.
* **Override**: When equals() is overridden, hashCode() must also be overridden to maintain the general contract.

A hash code is an integer value that is associated with each object in Java. Its main purpose is to facilitate hashing in hash tables, which are used by data structures like HashMap

n Java, the hashCode() method returns a 32-bit number (int) for any object. Comparing two numbers is much faster than comparing two objects using the equals() method, especially if that method considers many fields. If our program compares objects, this is much simpler to do using a hash code.

to define object equality and to enable the effective use of objects in hash-based data structures such as HashMap, HashSet, etc

* **Contract**:
  + If two objects are equal according to equals(), they must have the same hash code.
  + If two objects are not equal, they can still have the same hash code, but it’s not required.

**Example**

Here’s a simple example to illustrate:

**Java**

public class Person {

private String name;

private int id;

public Person(String name, int id) {

this.name = name;

this.id = id;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Person person = (Person) obj;

return id == person.id && name.equals(person.name);

}

@Override

public int hashCode() {

return Objects.hash(name, id);

}

}

[In this example, the equals() method checks if two Person objects have the same name and id, while the hashCode() method ensures that equal objects have the same hash code](https://www.geeksforgeeks.org/equals-hashcode-methods-java/)