**Hashing**

* **Hashing** is a process of converting an object into integer form by using the method hashCode().
* Its necessary to write hashCode() method properly for better performance of HashMap.

**Java equals():**

* it is by default for reference comparison. If we want, we can override it for content comparison.

If two objects are equal by == operator then these objects always equal by .equals() i.e r1==r2 is true , then r1.equals(r2) is always true.

If two objects are not equal by == operator then we can’t conclude anything about **equals(),** it may returns true of false. i.e r1==r2 is false, then **r1.equals(r2)** may returns true or false. And we can’t expect exactly.

Object class defined equals() method like this:

public boolean equals(Object obj) {

return (this == obj);

}

According to java documentation of equals() method, any implementation should adhere to following principles.

* For any object x,**x.equals(x)**should return true.
* Multiple invocations of **x.equals(y)** should return same result, unless any of the object properties is modified that is being used in the equals() method implementation.
* Object class **equals()** method implementation returns true only when both the references are pointing to same object.
* The problem is caused by the un-overridden method "hashCode()". The contract between **equals**() and **hashCode**() is:
  1. **If two objects are equal, then they must have the same hash code.**
  2. **If two objects have the same hash code, they may or may not be equal.**

**Java hashCode()**

Java Object **hashCode()** is a native method and returns the integer hash code value of the object. The general contract of hashCode() method is:

* Multiple invocations of hashCode() should return the same integer value, unless the object property is modified that is being used in the equals() method.
* An object hash code value can change in multiple executions of the same application.
* Two equivalent object must have same hascode.
* If two objects are equal according to equals() method, then their hash code must be same.
* If two objects are unequal according to equals() method, their hash code are not required to be different. Their hash code value may or may-not be equal.

**HASHMAP**

**Working:**

**Implementation:**

**Syntax:**

**When to use:**

**Drawbacks:**

**HASHTABLE**

**Working:**

**Implementation:**

**Syntax:**

**When to use:**

**Drawbacks:**

**HASHTREE**

**Working:**

**Implementation:**

**Syntax:**

**When to use:**

**Drawbacks:**

**HASHSET**

**Working:**

**Implementation:**

**Syntax:**

**When to use:**

**Drawbacks:**

* **treeset vs hashset**