Object class and its method :
----
* Object is a predefined class available in java.lang pacakge.

* It is the super class for all the classes we have in java.

* Since It is a super class so we can override the non final methods of Object class in any class in java.

* It plays a major role in object creation because whenever we create an object in java, the control must reach to object class first.

* Object class has provided total 11 non static methods.

public native int hashCode() :
----
* Predefined non static method of Object class.

* It is used to find out (OR calculate) the hashcode value of an object.

* This object hashCode value is useful in the Hashtable data structure to find out the bucket location.

* **hashCode() method is never used for Object compariosn, for comparing of two objects we should use equals(Object obj) method of Object class.**

* **There is a contract b/w hashCode() & equals(Object obj) which describes that :**

If two objects are having same hash code then both the objects may be same OR may not be same but If two Object are same based on the equals(Object obj) method then their hash code must be same.

Program :

```java
package com.ravi.object_class_methods;

class Employee
{
}

public class HashCodeDemo1
{
    public static void main(String[] args)
    {
        Employee e1 = new Employee();
        Employee e2 = new Employee();

        System.out.println(e1==e2);
        System.out.println(e1.equals(e2));
        System.out.println(e1.hashCode());
        System.out.println(e2.hashCode());
    }

}

package com.ravi.elc;

public class HashcodeDemo2
{
    public static void main(String[] args)
    {
        Integer i = 65;
        String s = "A";

        System.out.println(i.hashCode());
        System.out.println(s.hashCode());


        String s1 = "Java";
        String s2 = new String("Java");
        System.out.println(s1==s2);
        System.out.println(s1.equals(s2)); //true
        System.out.println(s1.hashCode());
        System.out.println(s2.hashCode());

    }

}
```

public boolean equals(Object obj)
----
* Predefined method of Object class.

* It is used to compare two objects based on the memory address OR Memory reference bacause internally It is using == operator only as shown in the program.

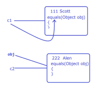
```java
package com.equals;

class Customer
{
    private int id;
    private String name;

    public Customer(int id, String name)
    {
        super();
        this.id = id;
        this.name = name;
    }
}

public class EqualsDemo1
{
    public static void main(String[] args)
    {
        Customer c1 = new Customer(111, "Scott");
        Customer c2 = new Customer(111, "Scott");

        IO.println(c1==c2); //false
        IO.println(c1.equals(c2)); //false
    }

}
```

Note : here equals(Object obj) method is providing false because Object class equals method internally using == operator only.

IF WE WANT TO COMPARE THESE CUSTOMER OBJECTS BASED ON THE **CONTENT (NOT BASED ON THE MEMORY ADDRESS) THEN WE SHOULD OVERRIDE EQUALS METHOD FROM OBJECT CLASS.**

Diagram :

```java
                                    c2
@Override
public boolean equals(Object obj)
{
  //Retrieving the first object data
   int id1 = this.id;
   String name1 = this.name;
```



```java
c1.equals(c2);
```

* As we know there is a contract b/w hash code and equals method, It is strongly recommended to override both the methods in the BLC class otherwise It is the voilation of contract.

```java
package com.equals;

class Customer
{
    private int id;
    private String name;

    public Customer(int id, String name)
    {
        super();
        this.id = id;
        this.name = name;
    }

    @Override
    public int hashCode()
    {
        return this.id;
    }



    //Overriding equals(Object obj) method for content comparison
    @Override
    public boolean equals(Object obj)
    {
        //Retrieving the first object data
        int id1 = this.id;
        String name1 = this.name;

        //Retrieving the second object data
        Customer c2 = (Customer) obj;

        int id2 = c2.id;
        String name2 = c2.name;

        if(id1==id2 && name1.equals(name2))
        {
            return true;
        }
        else
        {
            return false;
        }

    }


}

public class EqualsDemo1
{
    public static void main(String[] args)
    {
        Customer c1 = new Customer(111, "Scott");
        Customer c2 = new Customer(111, "Scott");

        IO.println(c1==c2); //false
        IO.println(c1.equals(c2)); //true
        IO.print(c1.hashCode()+" : "+c2.hashCode());
    }

}
```

Overriding hashCode() and equals() method for Product object content compariosn :

```java
package com.equals;

class Product
{
    private int id;
    private String name;

    public Product(int id, String name) {
        super();
        this.id = id;
        this.name = name;
    }

    @Override
    public int hashCode()
    {
        return this.id;
    }

    @Override
    public boolean equals(Object obj) //obj = s1
    {
        if(obj instanceof Product)
        {
          Product p2 = (Product) obj;

          if(this.id == p2.id && this.name.equals(p2.name))
          {
            return true;
          }
          else
          {
            return false;
          }
        }
        else
        {
            IO.println("Objects are not comparable object");
            return false;
        }
    }
}

public class EqualsDemo2
{
    public static void main(String[] args)
    {
       Product p1 = new Product(111, "Camera");
       Product p2 = new Product(111, "Camera");

       IO.println(p1.equals(p2));
       IO.println(p1.hashCode() +" : "+p2.hashCode());

       IO.println("......................");

       Product p3 = new Product(222, "Mobile");
       Student s1 = new Student(111,"Ravi");
       IO.println(p3.equals(s1));  //two different objects

    }

}

class Student
{
    int id;
    String name;

    public Student(int id, String name) {
        super();
        this.id = id;
        this.name = name;
    }


}
```