Java Interview Questions

- equals(Object) vs == operator
- Background of equals(Object) & hashCode() methods
- Contract between equals(Object) & hashCode() methods
- Overriding of equals(Object) & hashCode() methods
- Usage of equals(Object) & hashCode() in HashMap & Hashtable.

equals(Object) vs == operator

In Java, both equals(-) method & “==” operator are used to check equality of objects. But, here are some of the differences between these two:

- Main difference is that, one is method (equals(-)) and other is operator (==).
- We can use == operator for reference comparison (address comparison) and .equals(-) method for content comparison.
- In simple words, == operator checks, if both objects are pointing to the same memory location, whereas .equals(-) method evaluates to the comparison of values (fields) in the objects.
- If a class does not override the equals(-) method, then by default it uses equals(Object o) method of the closest parent class that has overridden this method.

Background of equals(Object) & hashCode()

equals(Object) & hashCode() methods have been defined in java.lang.Object class, which is parent class for all java classes. For this reason, all java objects inherit a default implementation of these methods.

equals(-) method

The equality can be compared in two ways:

Shallow Comparison: The default implementation of equals method is defined in java.lang.Object class which simply checks if two objects are references (say a and b) to the same memory location. i.e, It checks if a == b. Such comparison based on references is known as shallow comparison.

Background of equals(Object) & hashCode()

Deep Comparison: Suppose a class provides its own implementation of equals(-) method in order to compare the objects of that class w.r.t state of the objects. That means, fields (data members) of objects are to be compared one with another. Such comparison based on fields is known as deep comparison.

This method checks if some other Object passed to it as an argument is equal to the Object on which it is invoked.

Some principles of equals(-) method of Object class :
If some other object is equal to a given object, then it follows these rules:

Reflexive : for any reference value a, a.equals(a) should return true.

Background of equals(Object) & hashCode()

Symmetric : for any reference values a and b, if a.equals(b) should return true then b.equals(a) must return true.

Transitive : for any reference values a, b, and c, if a.equals(b) returns true and b.equals(c) returns true, then a.equals(c) should return true.

Consistent : for any reference values a and b, multiple invocations of a.equals(b) consistently return true or consistently return false, provided no information used in equals comparisons on the object is modified.
Note: For any non-null reference value a, a.equals(null) should return false.

Background of equals(Object) & hashCode()

hashCode() method

- A hashCode is an integer value that represents the state of the current object.
- If you write a custom type then you are responsible for creating a hashCode implementation that will represent the state of the current instance.
- Hashcode value is mostly used in hashing based collections like HashMap, HashSet and HashTable etc...
- hashCode() method must be overridden in every class which overrides equals(-) method.

Background of equals(Object) & hashCode()

Default implementation :

```
public class Object {

public boolean equals(Object obj) {
        return (this == obj);
}

@HotSpotIntrinsicCandidate
public native int hashCode(); // This method returns the hash code value for the object on which this method is invoked.
:
:
}
```

Contract between equals(Object) & hashCode()

- Objects which are equal by .equals(Object) MUST have the same hashCode.
- Two different objects are NOT equal by .equals(Object) method may return similar hashCode
- The hashCode() method should return the same integer value for the same object for each calling of this method unless the value stored in the object is modified.
- If you only override the hashCode() method, it returns a proper hashCode but the equality fails while comparing two similar objects with same content.
- If you only override the equals method, if a.equals(b) is true, it means the hashCode of a and b must be the same, but that does not happen since you did not override the hashCode method.
- **Note :** hashCode() method of Object class always returns a new hashCode for each object.

Overriding of equals(Object) & hashCode() methods

- If you don't override hashcode() then the default implementation in Object class will be used by collections. This implementation gives different values for different objects, even if they are equal according to the equals() method.
- Some collections, like HashSet, HashMap or HashTable use the hash code to store its data and to retrieve it. If you don't implement hashcode() and equals() in a consistent manner, then they will not function properly.
- Override both equals(-) and hashCode() methods
- Recommended to consider the required fields while overriding both methods
- Consider eclipse generated source code for equals, hashCode and other methods

Use of equals(Object) & hashCode()

In java.util.HashMap<Key, Value>,

- hashCode() is used to calculate the bucket and therefore calculate the index.
- equals(-) method is used to check that 2 objects are equal or not.
- java.util.HashMap uses equals(-) to compare the keys, whether the keys are equal or not. If equals(-) method return true then the keys are equal otherwise not equal.
- equals(-) & hashCode() methods are executed on Key in the java.util.HashMap<Key, Value>

Download the source code from GIT

Please download the source code from below GIT repository URL :

Enter this URL on browser - https://github.com/ChandraKodam5/java
Click on **Code** Drop-down → Click on **Download ZIP** → Open eclipse IDE

Goto **File** → **Import** → Existing projects into workspace → **Next** → click on **checkbox** (**select archive file**) → click on **Browse** → choose the downloaded **EqualsHashcodeContract** ZIP file.

OR

Type the below command in git bash
$ git clone https://github.com/ChandraKodam5/java.git
Goto **File** → **Import** → Existing projects into workspace → **Next** → click on **checkbox** (**select archive file**) → click on **Browse** → choose the downloaded **EqualsHashcodeContract** ZIP file.

Thank You

GIT Repository :

https://github.com/ChandraKodam5/java