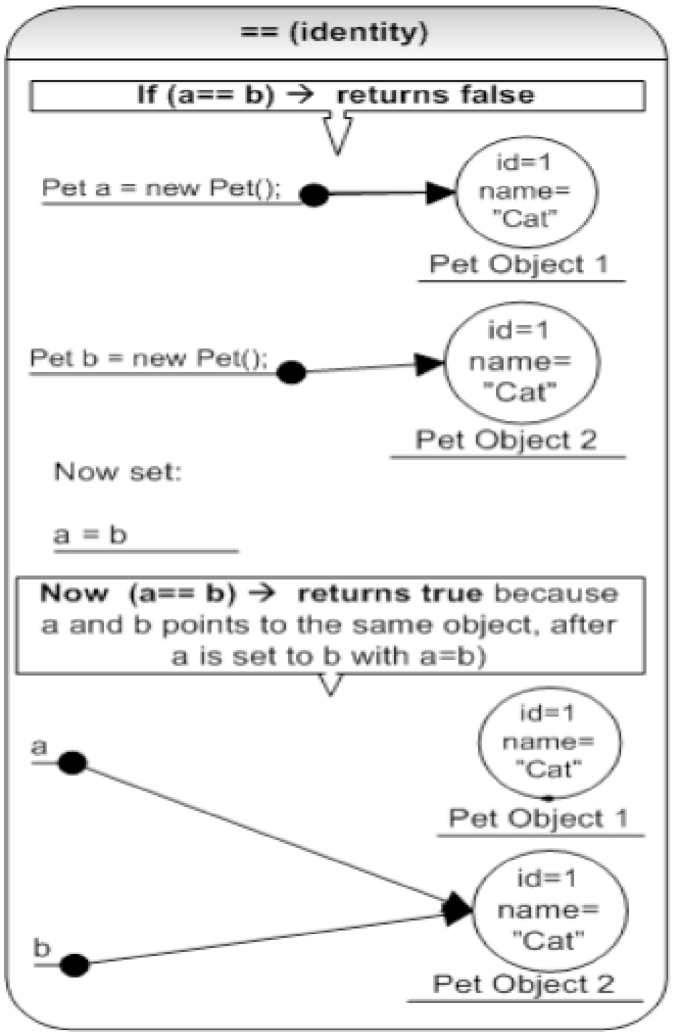
♦♥ Object equals Vs == and pass by reference Vs value

Posted on August 16, 2014 by Arulkumaran Kumaraswamipillai — 4 Comments ↓

Q1. What is the difference between "==" and equals(..) method when comparing 2 objects?
A1.It is important to understand the difference between identity (i.e. ==) comparison, which is a shallow comparison that compares only the object references, and the equals( ) comparison, which is a **deeper comparison** that compares the object attributes. The diagram below explains the difference between the two. There are some exceptional conditions when using primitives, String objects, and enums.

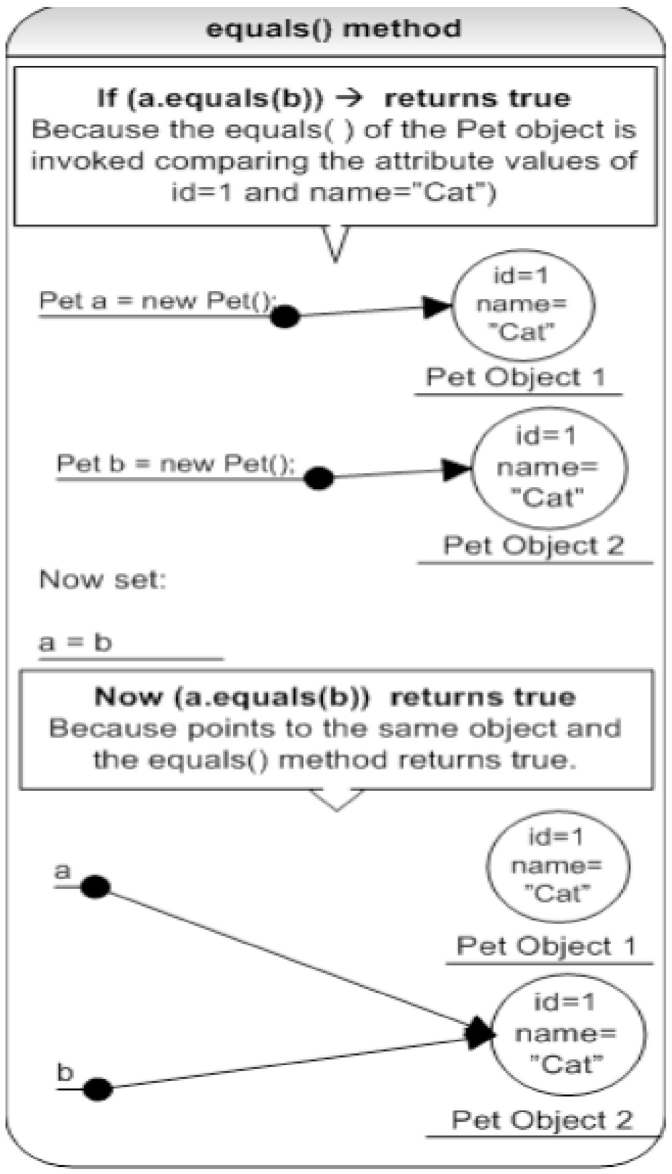
If the equals(..) method is not overridden, then the Object class's default implementation is invoked, which only compares the object references. Invoking the equals(..) method of the Object class is equivalent to making a shallow comparison with "==". This is why it is imperative to override the **equals( )** method, and the **hashCode( )** methods in your custom classes like Pet. The Java API objects like String, and the wrapper classes like Integer, Double, Float, etc override the equals(..), hashCode( ), and the toString(..) methods. These methods are meant to be overridden as discussed in detail at "5 Java Object class methods interview questions & answers"



Java identity ==

As a Java Architect

Java architecture & design concepts interview Q&As with diagrams | What should


equals() method
If (a.equals(b)) → returns true
Because the equals( ) of the Pet object is invoked comparing the attribute values of id=1 and name="Cat")
Pet a = new Pet();
id=1 name= "Cat"
Pet Object 1
Pet b = new Pet();
id=1 name= "Cat"
Pet Object 2
Now set:
a = b
Now (a.equals(b)) returns true
Because points to the same object and the equals() method returns true.
a
id=1 name= "Cat"
Pet Object 1
b
id=1 name= "Cat"
Pet Object 2

Deeper equals( ) compares values by invoking equals( ) method

If you were to implement the equals(…) and hashCode(…) methods:

```
public final class Pet {
    int id;
    String name;

    @Override
    public boolean equals(Object that){
        if ( this == that ) return true;

        if ( ! (that instanceof Pet) ){
            return false;
        }

       Pet pet = (Pet)that;
```

be a typical Java EE architecture?

Senior Java developers must have a good handle on

open all | close all

- ⊞ Best Practice (6)
- ⊞ Coding (26)
- ⊞ Concurrency (6)
- ⊞ Design Concepts (7)
- ⊞ Design Patterns (11)
- ⊞ Exception Handling (3
- ⊞ Java Debugging (21)
- ⊞ Judging Experience I
- ⊞ Low Latency (7)
- ⊞ Memory Managemen
- ⊞ Performance (13)
- ⊞ QoS (8)
- ⊞ Scalability (4)
- ⊞ SDLC (6)
- ⊞ Security (13)
- ⊞ Transaction Managem

80+ step by step Java Tutorials

open all | close all

- ⊞ Setting up Tutorial (6)
- ⊞ Tutorial - Diagnosis (2
- ⊞ Akka Tutorial (9)
- ⊞ Core Java Tutorials (2
- ⊞ Hadoop & Spark Tuto
- ⊞ JEE Tutorials (19)
- ⊞ Scala Tutorials (1)

```
        return  this.id == pet.id  && this != nul
    }

    @Override
    public int hashCode( ) {
        int hash = 9;
        hash = (31 * hash) + id;
        hash = (31 * hash) + (null == name ? 0 :
        return hash;
    }
}
```

You can learn more at “**Sorting objects in Java interview Q&As**”

Q2. What happens when you run the following code?

```
Boolean b1  = new Boolean(false);
Boolean b2  = Boolean.FALSE;

 if(b1 == b2) {
      System.out.println("Equal");
 }
 else{
      System.out.println("Not Equal");
 }
```

A2. Prints “Not Equal”.

The == is a shallow comparison that only compares the references. The references are not equal. If you want to print “Equal”, perform a deeper comparison as shown below, which compares the values.

```

If (b1.equals(b2)){
    System.out.println("Equal");//gets printed
}
else {
    System.out.println("Not Equal");
}
```

Or, you need to take advantage of the **flyweight design pattern** that reuses objects.

```
Boolean b1 = Boolean.valueOf("false"); // create
Boolean b2 = Boolean.FALSE; //points to the same
```

Spring & HIbernate Tu
Tools Tutorials (19)
Other Tutorials (45)

Preparing for Java written & coding tests

open all | close all
♦ Complete the given
Can you write code? (
Converting from A to I
Designing your classe
Java Data Structures
Passing the unit tests
What is wrong with th
Writing Code Home A
Written Test Core Jav
Written Test JEE (1)

How good are your...to go places?

open all | close all
Career Making Know-
Job Hunting & Resum

```

```

or

```
Boolean b1 = Boolean.valueOf("false"); // create
Boolean b2 = Boolean.valueOf("false"); //points t

```

Q3. Can you discuss the output of the following code?

```

public class PrimitiveAndObjectEquals {

    public static void main(String[ ] args) {
        int a = 5;
        int b = 5;

        Integer c = new Integer(5);
        Integer d = new Integer(5);

        if (a == b) {  //Line 1
            System.out.println("primitives a and
        }

        if (c == d) {  //Line 2
            System.out.println("Objects c and d
        }

        if (c.equals(d)) {  //Line 3
            System.out.println("Objects c and d
        }

        if (a == d) { //Line 4
           System.out
               .println("Primitive a and Object
        }
    }
}
```

A3. **Output is:**

```

primitives a and b are ==
Objects c and d are equals( )
Primitive a and Object d are == due to auto unbox

```

1) Line 1 is printed as both a and b are primitive data types, and primitives are compared with == as they don't have an equals() method.

2) Line 2 will not get printed as they are comparing the object references (shallow comparison). Line 3 will get printed as they are comparing the actual values (deep er comparison).

3) Line 4 is printed because the object reference “d” is auto-unboxed to a primitive int value and then compared with the primitive reference “a”. This also illustrates a hidden chance of a **NullPointerException** being thrown if the reference “d” were to be null.

Q4. Can you discuss the output of the following code?

```
public class EnumEquals {

  public enum Action {START, STOP, CONTINUE}

  private static Action action = Action.STOP;

  public static void main(String[ ] args) {

      if(Action.STOP == action){
          System.out.println("Enumurations can b
      }

      if(Action.STOP.equals(action)){
          System.out.println("Enumurations can b
      }
    }
}

```

A4. Output is:

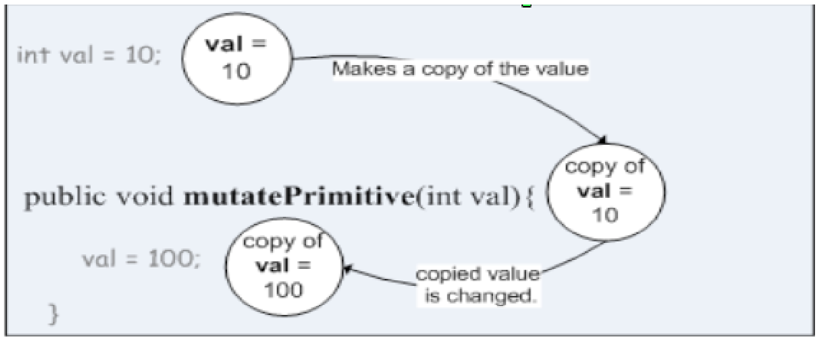
```

Enumurations can be compared to ==.
Enumurations can be compared to equals( ) also.

```

The best practice is to use the referential == for enums.

Q5. Explain the statement Java is always pass by value?
A5. Other languages use pass-by-reference or pass-by-pointer. But in Java, no matter what type of argument (i.e. a primitive variable or an object reference) you pass, the corresponding parameter will get a copy of that data, which is exactly how pass-by-value (i.e. copy-by-value) works. Even though the definition is quite straight forward, the way the

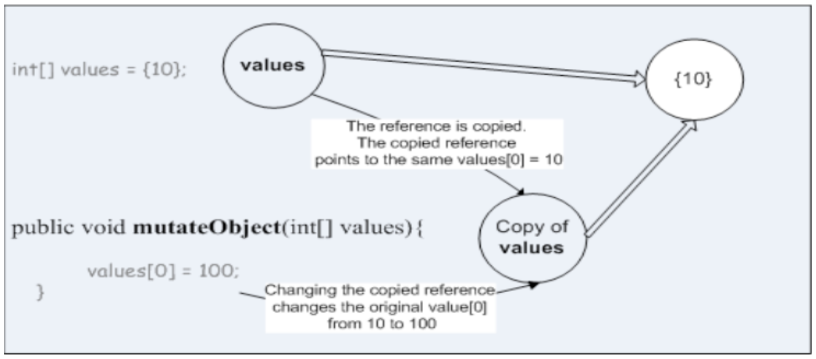
primitives and object references behave when passed by value, will be different.

For example, If the passed in argument was a primitive value like int, char, etc, the passed in primitive value is copied to the method parameter. Modifying the copied parameter will not modify the original primitive value.



pass-by-value primitive variables like int, long, etc

On the contrary, if the passed in argument was an object reference, the passed in reference is copied to the method parameter. The copied reference will still be pointing to the same object. So if you modify the object value through the copied reference, the original object will be modified.


int[] values = {10};
values
{10}
The reference is copied. The copied reference points to the same values[0] = 10
public void mutateObject(int[] values){
values[0] = 100;
}
Copy of values
Changing the copied reference changes the original value[0] from 10 to 100

pass by value for objects like int[], Pet, Car, etc

Q6. The value of Point p before the following method calls is (10,20). What will be the value of Point p after executing the following method calls?

Scenario 1:

```java
static void mutatePoint(Point p) {
      p.x =  50;
      p.y=100;
}

```

Scenario 2:

```java
static void mutatePoint(Point p) {
      p = new Point(50,100);
}
```

A6.

Scenario 1:

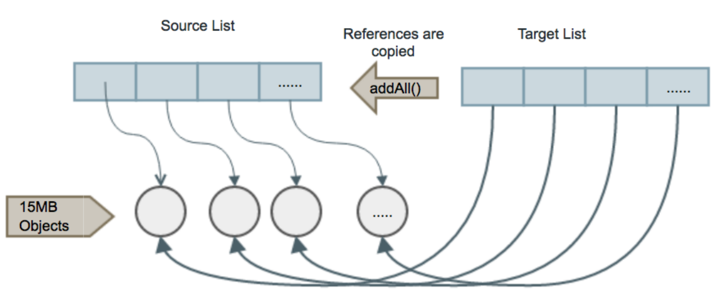
Point p = (50,100), as the copied reference will still be pointing and modifying the original Point (10,2 0) object through the mutatePoint( ) method.

Scenario 2:

Point p = (10,20), as the copied reference will be creating and pointing to the newly created Point (50, 100) object.

Q7. If there is a source array list with 15MB of data, and then you create a new target empty array list and copy the source to target with target.addAll(source). How much memory will be consumed after invoking the addAll(…) method?

A7. The memory will still be 15MB because of "**pass-by-value**" where new objects are not created when addAll(…) is invoked. Only the references are copied, but the copied references will still be pointing to the source list objects. For example,



Java is pass by value

```java

package com.test;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class PassByReference {


	public static void main(String[] args) {

		List<String> source = Arrays.asLis

		List<String> target = new ArrayLis

		target.addAll(source);

		//memory will still be 15MB, why?

		System.out.println(source.get(0) =

		//This is because source and targe
		//No new objects are created by ad
		//Only the references are copied
	}
}


```

Popular Posts

♦ 11 Spring boot interview questions & answers

857 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

825 views

18 Java scenarios based interview Questions and Answers

447 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

401 views

♦ 7 Java debugging interview questions & answers

311 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

302 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

292 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

286 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

263 views

8 Git Source control system interview questions & answers

215 views

Bio **Latest Posts**

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.

‹ ♦ 5 Java Object class methods interview questions & answers

01: ♦ ♥ 17 Java overview interview questions and answers ›

Posted in Objects

Tags: Core Java FAQs, Java/JEE FAQs, Novice FAQs

4 comments on "♦♥ Object equals Vs == and pass by reference Vs value"

Rishi Raj says:
February 22, 2016 at 4:10 pm

Hi ArulKumaran,
hashCode() and equals() are overridden in Integer. Also hashCode() returns same value for equal-valued Integer objects, and equals() compares "value" member of Integer objects. Then, why c == d does not come out to be true? Where and why is it established that c is not going to be equal to d – in hashCode() or in equals()?

Reply

Arulkumaran Kumaraswamipillai says:
February 22, 2016 at 8:13 pm

Hi Rishi,

If you create c & d as shown below

```
Integer c = new Integer(5);
Integer d = new Integer(5);
```

Then c & d are NOT "==", but equals().

If you create it as shown below then:

Integer c = Integer.valueOf(5);
Integer d = Integer.valueOf(5);

Both “==” and equals() will return true. This is because the c & d will be pointing to the same single object. The references are ==, and so are the values (i.e. equals()).

Reply

Antonio Gil says:
February 20, 2016 at 9:25 am

I think your tutorials are great. Love your approach to java.

Reply

Arulkumaran Kumaraswamipillai says:
February 20, 2016 at 10:37 am

Thanks.

Reply

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from "Blah blah" to "Wow"? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer