Deep, Shallow and Lazy Copy with Java Examples

Difficulty Level : Medium • Last Updated : 13 Feb, 2018

In object-oriented programming, object copying is creating a copy of an existing object, the resulting object is called an object copy or simply copy of the original object.There are several ways to copy an object, most commonly by a copy constructor or cloning.

We can define Cloning as **"create a copy of object"** Shallow, deep and lazy copy is related to cloning process these are actually the ways for creating copy object.

Shallow Copy

- Whenever we use default implementation of clone method we get shallow copy of object means it creates new instance and copies all the field of object to that new instance and returns it as object type, we need to explicitly cast it back to our original object. This is shallow copy of the object.
- clone() method of the object class support shallow copy of the object. If the object contains primitive as well as nonprimitive or reference type variable in shallow copy, the cloned object also refers to the same object to which the original object refers as only the object references gets copied and not the referred objects themselves.

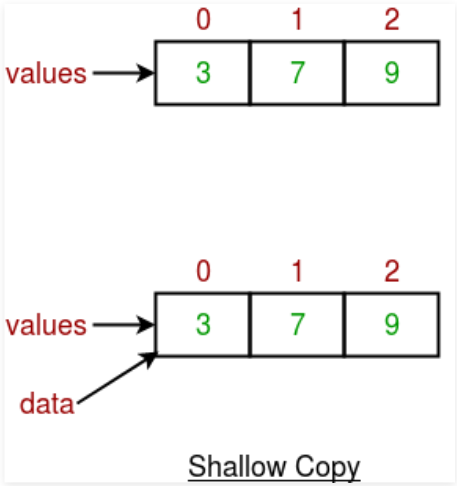
```
//code illustrating shallow copy
public class Ex {

    private int[] data;

    // makes a shallow copy of values
    public Ex(int[] values) {
        data = values;
    }

    public void showData() {
        System.out.println( Arrays.toString(data) );
    }
}
```


The above code shows shallow copying. data simply refers to the same array as vals.



Shallow Copy

This can lead to unpleasant side effects if the elements of values are changed via some other reference.

```
public class UsesEx{

    public static void main(String[] args) {
        int[] vals = {3, 7, 9};
        Ex e = new Ex(vals);
        e.showData(); // prints out [3, 7, 9]
        vals[0] = 13;
        e.showData(); // prints out [13, 7, 9]

        // Very confusing, because we didn't
        // intentionally change anything about
        // the object e refers to.
    }
}
```

```
Output 1 : [3, 7, 9]
Output 2 : [13, 7, 9]
```


Deep Copy

- Whenever we need own copy not to use default implementation we cal
  copy of the object we need to implement according to our need.
- So for deep copy we need to ensure all the member class also impleme
  the clone() method of the object class.

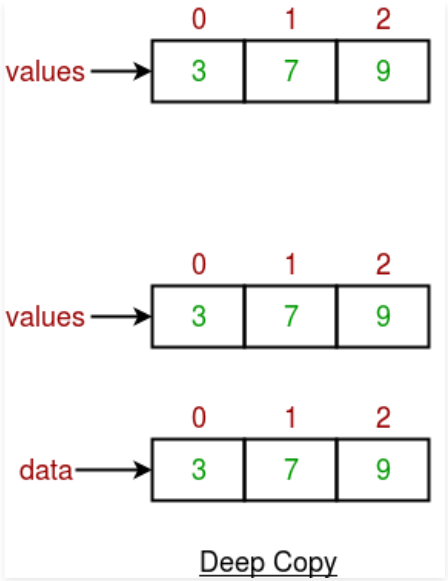
A deep copy means actually creating a new array and copying over the values.

```
// Code explaining deep copy
public class Ex {

    private int[] data;

    // altered to make a deep copy of values
    public Ex(int[] values) {
        data = new int[values.length];
        for (int i = 0; i < data.length; i++) {
            data[i] = values[i];
        }
    }

    public void showData() {
        System.out.println(Arrays.toString(data));
    }
}
```



Deep Copy


```java
public class UsesEx{

    public static void main(String[] args) {
        int[] vals = {3, 7, 9};
        Ex e = new Ex(vals);
        e.showData(); // prints out [3, 7, 9]
        vals[0] = 13;
        e.showData(); // prints out [3, 7, 9]

       // changes in array values will not be
       // shown in data values.
    }
}
```

```
Output 1 : [3, 7, 9]
```

Changes to the array vals will not result in changes to the array data.

when to use what

There is no hard and fast rule defined for selecting between shallow copy
keep in mind that if an object has only primitive fields, then obviously we s
has references to other objects, then based on the requirement, shallow c
references are not updated then there is no point to initiate a deep copy.

Lazy Copy

A lazy copy can be defined as a combination of both shallow copy and deep copy. The mechanism follows a simple approach – at the initial state, shallow copy approach is used. A counter is also used to keep a track on how many objects share the data. When the program wants to modify the original object, it checks whether the object is shared or not. If the object is shared, then the deep copy mechanism is initiated.

Summary

In shallow copy, only fields of primitive data type are copied while the objects references are not copied. Deep copy involves the copy of primitive data type as well as objet references. There is no hard and fast rule as to when to do shallow copy and when to do a deep copy. Lazy copy is a combination of both of these approaches.

This article is contributed by **Abhishek Gupta**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Attention reader! Don't stop learning now. Get hold of all the important **Java Foundation** and Collections concepts

RECOMMENDED ARTICLES

Page : 1 2 3

01 **Java Program to Show Shallow Cloning and Deep Cloning**
12, Jan 21

02 **Collections copy() method in Java with**

05 **Java Program to Copy and Paste an image in OpenCV**
13, Dec 20

03 **Copy Elements of One Java Vector to Another Vector in Java**
18, Jan 21

04 **How to Copy and Add all List Elements to an Empty ArrayList in Java?**
03, Dec 20

07 **Array Co**
01, Feb 16

08 **Differen to Anoth**
23, Oct 20

Previous

Find sum of all right leaves in a given Binary Tree

Next

Make all combinations of size k

GeeksforGeeks

Current difficulty : Medium

Easy | Norma

Article Tags : Java

Practice Tags : Java

Improve Article | Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

name • 5 months ago

incomplete garbage

Reply • Share ›

Guest ➚ name • 5 months ago

Your response certainly is... If you have actual details of what you think needs
please do so. If you just want to spew garbage as you literally did without add
information, please refrain from posting.

Reply • Share ›

ai1 Tutorial • 7 months ago

Shallow copy returns Object as type and which needs to explicitly cast back to original object

Reply • Share ›

Shravya • a year ago

In deep copy we just take a new array and copy all the elements there..right?

Reply • Share ›

Jut • 2 years ago • edited

If i need to make a shallow and deep class and test them, would i need a separate testing class for each or will a toString() and equals() do it for me?
(im just starting to learn inheretance, so please bear with me)

edit
after re-reading this a few times and doing some of my own research, i figured it out.

Reply • Share ›

shadab shaikh • 2 years ago

Nice Explanation

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 2(

feedback@geeksforgeeks.org

Company

About Us
Careers
Privacy Policy
Contact Us

Learn

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

Practice

Courses
Company-wise
Topic-wise
How to begin?

Contribute

Write an Article
Write Interview Experience
Internships
Videos