

## Re: Why doesn't Integer have a setValue(int i), and is there a way of changing its value.

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2003-12/2601.html>

---

**From:** bagbourne (noway\_at\_noway.com)

**Date:** 12/20/03

Date: Sat, 20 Dec 2003 16:03:57 +0000 (UTC)

natG wrote:

> "Bent C Dalager" <bcd@pvv.ntnu.no> wrote in message  
> news:brsjhp\$t6r\$I@tyfon.itea.ntnu.no...  
>  
>>In article <vu3hd8iad8546c@corp.supernews.com>,  
>>Alex Hunsley <lard@tardis.ed.ac.molar.uk> wrote:  
>>  
>>>natG wrote:  
>>>  
>>>  
>>>>Can you please explain "safer"? Do I practice SAFE Java<g>?  
>>>>Seriously, safer in what respect?  
>>  
>>Two cases spring immediately to mind.  
>>  
>>If you use instances of the class as keys in a Map, or you have them  
>>in a Collection that you expect to always be sorted, then immutability  
>>guarantees that your sorting won't inadvertantly get screwed up by  
>>someone calling a setter at inopportune times. With immutable keys,  
>>the only thing that can mess with your sorting is when the  
>>Collection's setters are called. The alternative to immutability would  
>>be for the class to fire events whenever you changed something and the  
>>Collections classes to be sensitive to such events and be able to  
>>auto-resort themselves, but this gets real messy real quick.  
>  
>  
> Question: Suppose:  
> Integer a = new Integer(10);  
> Integer b = new Integer(20);  
> Integer c = new Integer(30);  
> TreeMap tm = new TreeMap();  
> tm.put(a);  
> tm.put(b);  
> tm.put(c);  
> And then:

comp.lang.java.programmer: Re: Why doesn't Integer have a setValue(int i), and is there a way of changing its value

- > *b = new Integer(60);*
- > *Now, I assume that the second Integer added to the Map, will return an int value*
- > *of 60.*
- > *If so, what happens to the sort at that point?*

You have a fundamental misunderstanding. I'm surprised nobody picked up on this. You sent a reference to an Integer object (stored in variable named "b") to TreeMap.put(). It stored that reference.

Then you create a *\*totally new instance\** of an Integer with 60 in it, and put a reference to that object into variable b.

That has nothing to do with the reference that you previously put into the TreeMap.

If it helps, think of object variables as smart pointers that cannot be corrupted to point to anything else other than what they were declared as.

When you assign using "=", you just change the variable to *\*reference\** that new object.