[Home](http://stuffthathappens.com/blog/) • [About](http://stuffthathappens.com/blog/about/) • [Archives](http://stuffthathappens.com/blog/archives/) •

« [BPM: Not Even Comic-Worthy](http://stuffthathappens.com/blog/2007/10/10/bpm-not-even-comic-worthy/)

[You CAN Change Final Fields](http://stuffthathappens.com/blog/2007/10/13/you-can-change-final-fields/) »

[**On the Violation of Java Access Control**](http://stuffthathappens.com/blog/2007/10/11/on-the-violation-of-java-access-control/)

October 11th, 2007

Let’s write a simple class and show how easy it is to bypass normal Java language access checking. This means we can modify another class’s private data using reflection. Good times…

package pkgb;  
  
public class Caveman {  
 private String name;  
 private final int age = 10;  
  
 public String getName() {  
 return name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
}

Can you modify name, even thought it is private? Can you modify age, even though it is private **and** final?

### First, What Won’t Work…

You cannot simply create a new Caveman and set its name. This will not compile because name is private. Instead, let’s try reflection:

public class Main {  
 public static void main(String[] args) throws NoSuchFieldException {  
 Caveman caveman = new Caveman();  
  
 // note that getField(...) throws NoSuchFieldException  
 Field nameField = Caveman.class.getDeclaredField("name");  
  
 try {  
 nameField.set(caveman, "Opie");  
 } catch (IllegalAccessException e) {  
 System.out.println("Unable to set name to Opie.");  
 }  
  
 ...

Notice I use getDeclaredField() instead of getField(). If you try this, you will find you are *unable to set the value of the name*. It is private, after all.

### setAccessible(true) to the Rescue!

You need to call setAccessible(true) first:

nameField.setAccessible(true);  
 try {  
 nameField.set(caveman, "Anthony");  
 System.out.println("Successfully set name to " + caveman.getName());  
 } catch (IllegalAccessException e) {  
 // won't happen  
 }

Voila! It works. This is a Good Thing, because without this capability, tools like JPA (and many others) would be severely constrained.

### Can we Change the Age?

The age field is private and final. Can we use reflection to change it as well? Let’s try:

Field ageField = Caveman.class.getDeclaredField("age");  
 ageField.setAccessible(true);  
 try {  
 System.out.println("Setting age to 100...");  
 ageField.setInt(caveman, 100);  
  
 System.out.println("Age is now " + caveman.getAge());  
  
 System.out.println("Is it final? " +  
 Modifier.isFinal(ageField.getModifiers()));  
  
 } catch (IllegalAccessException e) {  
 System.out.println("Unable to set age.");  
 }

What does the code do?

**UPDATE: See** [**this post**](http://stuffthathappens.com/blog/2007/10/13/you-can-change-final-fields/) **for a correction!**

**~~It silently fails.~~**

~~The console prints this:~~

~~Setting age to 100...  
Age is now 10.  
Is it final? true~~

~~Yikes. As expected, you CANNOT change final fields. Phew! But surprisingly, attempting to set the value does not throw an exception. Instead, it silently ignores your request.~~

**~~If you need to use reflection to set field values, be sure you check the modifiers (final, etc…) before you do so.~~**

### Is this Secure?

If you are running under a SecurityManager, you might not be able to do any of this. You can verify this quite easily:

public static void main(String[] args) throws NoSuchFieldException {  
 System.setSecurityManager(new SecurityManager());  
  
 ...code just like before

Now, calling setAccessible(true) throws an exception:

Exception in thread "main" java.security.AccessControlException: access denied  
 (java.lang.reflect.ReflectPermission suppressAccessChecks)  
 at java.security.AccessControlContext.checkPermission  
 (AccessControlContext.java:323)  
 at java.security.AccessController.checkPermission  
 (AccessController.java:546)  
 etc...

So if you need to use these techniques in an app server that contains a security manager, you’ll need to edit a policy file and allow suppressAccessChecks. This isn’t my area of expertise, but I can point you to the right Google search: [policy file suppressAccessChecks](http://www.google.com/search?q=policy+file+suppressAccessChecks).

Posted on Thursday, October 11th, 2007 at 10:10 pm in [Java](http://stuffthathappens.com/blog/category/java/).



[Alex Miler](http://tech.puredanger.com) Says:

Or of course, you can just modify the byte code when the class is loaded so the field is no longer final….  Trivial to do with ASM.

[October 12th, 2007 at 9:25 am](#gjdgxs)



[Tim Vernum](http://blog.adjective.org/) Says:

I think you’re actually falling into an optimisation trap, not an access control trap.

Try calling:

System.out.println(”Age field is ” + ageField.get(caveman));

And you should find it is 100.

The reason your code prints 10 is most likely due to the compiler optimising the implementation of getAge(). Because age is final, and has a single fixed value (10), it is possible for the compiler to optimise the implementation of “getAge()” to “return 10;”

[October 13th, 2007 at 9:14 am](#30j0zll)



[It’s Just a Bunch of Stuff That Happens » Blog Archive » You CAN Change Final Fields](http://stuffthathappens.com/blog/2007/10/13/you-can-change-final-fields/) Says:

[...] On the Violation of Java Access Control Oct [...]

[October 13th, 2007 at 8:13 pm](#1fob9te)

Follow It’s Just a Bunch of Stuff That Happens via the [Entries](http://stuffthathappens.com/blog/feed/atom/) and [Comments](http://stuffthathappens.com/blog/comments/feed/atom/) Atom feeds.