

Re: Rules for constructors

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2004-03/2425.html>

From: Alex Hunsley (lard_at_tardis.ed.ac.molar.uk)

Date: 03/17/04

Date: Wed, 17 Mar 2004 16:28:25 +0000

Alex Hunsley wrote:

> *I just found quite an annoying thing some code I am maintaining...*

>

> *Here is the essence of what happened:*

> *(this is a runnable self-contained example)*

[snip actual code]

>

> *You'd expect the code to print out memberVariable=7, but it actually
> prints out memberVariable=0.*

> *This is because the constructor of B calls super(param1, param2), and by
> stepping through the code in eclipse I could see that the java runtime
> processes the call to setUpSomeThings() in super(..) *before* it gets
> around to settings up the member variables in class B! Therefore, once
> the setUpSomeThings in the subclass (B) has been called by the
> constructor for A, the initialisation of the member variables in class B
> s constructor happens, which sets the member variable to 0 and wipes out
> the pervious value of 7.*

>

> *Anyway, what exactly is the moral of this tale?*

>

> *I suspect it's either*

> *a) always check your class is initialised,*

> *as per*

> <http://www.javaworld.com/javaworld/jw-12-1998/jw-12-securityrules.html>,

> *but this is a bit security-paranoid*

>

> *b) never call non-final methods from your constructor*

> *(or else they will be overridden causing chaos as I have experienced
> today)*

> *c) overridden methods should not access member variables that are
> from this class – should only access superclass member variables*

>

> *any other suggestions?*

> *What do people take a generally good rules for constructors?*

> *I already practice the rule that you shouldn't do any populating or,*

> *worse, realising of GUI items from a constructor.*

>
> alex
>

Sorry, I've just realised my code contains non-empty constructors which aren't used and may distract from the issue at hand. Here is simpler code that also shows the problem:

```
public class B extends A {
    private int memberVariable = 0;

    public B() {
        super();

        System.out.println("memberVariable = "
            + memberVariable);
    }

    protected void setUpSomeThings() {
        memberVariable = 7;
    }

    public static void main(String[] args) {
        B bInstance = new B();
    }
}
```

```
class A {

    public A() {
        setUpSomeThings();
    }

    protected void setUpSomeThings() {
        // some code in here
    }
}
```

alex