

Re: Demoting and Promoting Serialised Objects

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2005-07/msg00373.html>

- *From:* "Boudewijn Dijkstra" <usenet@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 5 Jul 2005 18:19:48 +0200
-

"Roedy Green" <look-on@xxxxxxxxxxxxxxxxxxxxxx> schreef in bericht news:6kqkc1pisre6rts11roqhks9m9oq0f6t27@xxxxxxxxxx

> I see a general problem I call demoting and promoting objects.
>
> A Dog gets promoted to a Dalmatian object by gaining extra
> Dalmatian-specific fields.
>
> A Dalmatian gets demoted to Dog object by stripping off its Dalmatian
> specific fields.
>
> The object itself is unchanged. You create a new modified object and
> discard the old one.
>
>
> Why would you do such a thing?
>
> 1. to strip and object down for export, serialisation, or RMI
> transport.

Isn't this where readObject and writeObject would come in handy?

> 2. to strip an object of confidential information before handing [it]
> on.
>
> 3. to avoid having to give confidential code used to process, edit or
> construct your objects when all they will do is view them.
>
> The way you did the equivalent thing in COBOL was a MOVE CORRESPONDING
> that copied field by field. In Java traditionally you do the same
> thing writing code guaranteed to stop working the instant any of the
> class structure changes.
>
> How might you handle this more elegantly?
>
> 1. a class parser then generates code to do the move-corresponding.
>
> 2. Java gain some feature to easily create copy constructor so you
> could say:
>

Re: Demoting and Promoting Serialised Objects

> something like:
>
> /**
> * cloning constructor
> * Makes a Dog out of any Dog object including a Dalmatian object.
> * The new object will have only the Dog fields and methods, even if
> * cast back to the original type.
> */

You cannot 'cast back to the original type', because it was lost in the conversion object.

> Dog (Dog) { Magic.copy(); }
>
> Where copy is some magic JNI or new Java feature.

Why not reflection?

> The assembler code theoretically could be just a block byte move of
> dog's object size bytes --- extremely fast compared with field by field
> copy.

Field by field copy bytecode could be automagically optimized into just that.

.

- *Follow-Ups:*

- ◆ [**Re: Demoting and Promoting Serialised Objects**](#)
 ◇ From: Roedy Green

- *References:*

- ◆ [**Demoting and Promoting Serialised Objects**](#)
 ◇ From: Roedy Green
- Prev by Date: [**Re: getting a RuntimeException in Vector.add\(\)**](#)
- Next by Date: [**Re: import magic**](#)
- Previous by thread: [**Demoting and Promoting Serialised Objects**](#)
- Next by thread: [**Re: Demoting and Promoting Serialised Objects**](#)
- Index(es):
 - ◆ [**Date**](#)
 - ◆ [**Thread**](#)