

Re: Why Generics?

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2005-05/msg02488.html>

- *From:* "Chris Uppal" <chris.uppal@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 25 May 2005 18:21:26 +0100
-

Phillip Lord wrote:

> Chris> It was always possible, the new enums don't generate code
> Chris> that couldn't be expressed in "normal" Java. Supporting
> Chris> switch would require a fairly unpleasant amount of wordy
> Chris> duplication, though. You'd have to have a public static
> Chris> final int field corresponding the the 'ordinal' of each enum
> Chris> object, which is obviously error prone unless the compiler
> Chris> does it for you.
>
> I think, actually, you are wrong and it doesn't work. The problem is
> that with switch case, the value of the cases must be a int or char
> definable at compile time.

No, there's no problem except wordy, error-prone, repetition.

Suppose that we want an enumeration of the three primary colours (this probably wouldn't compile but you'll be able to see what I mean, I hope):

```
public final class Colour
{
    public static final int
    __RED_ORDINAL = 0,
    __GREEN_ORDINAL = 1,
    __BLUE_ORDINAL = 2;

    public static final Colour
    RED = new Colour("RED", __RED_ORDINAL),
    GREEN = new Colour("RED", __GREEN_ORDINAL),
    BLUE = new Colour("BLUE", __BLUE_ORDINAL);

    private final static Colour s_values[] =
    { RED, BLUE, GREEN };

    private final int m_ordinal;
    private final String m_name;

    private Colour(String name, int ordinal)
    {
```

Re: Why Generics?

```
m_name = name;
m_ordinal = ordinal;
}

public int ordinal() { return m_ordinal; }
public String toString() { return m_name; }

public static Colour[]
values()
{
return (Colour[])(s_values.clone());
}
}
```

and then a case statement can be written like:

```
public boolean
isPutrid(Colour colour)
{
switch (colour.ordinal())
{
case Colour.__RED_ORDINAL:
case Colour.__BLUE_ORDINAL:
return false;
case Colour.__GREEN_ORDINAL:
return true;
default: // WTF ??
return false;
}
}
```

Which is almost exactly what the Java compiler does at the bytecode level. Of course it would be a pain to maintain and use (and not allow people to /ab/use) the __XXX_ORDINAL numbers in parallel to the "real" enumeration objects — that's what I meant when I called it wordy and error-prone, but it isn't at all impossible.

Note that the implementation of switch is a /real/ switch (a tableswitch bytecode instruction) which can be implemented as a pure jump table. There is no cascade of if-thens, nor is there any hashing going on.

Actually, the bytecode generated by javac uses one more level of indirection. The ordinal number is used to lookup an integer value in a synthetic int[] array, and then the value from the array is used in the generated switch statement. I'm not sure why it does that. Possibly it's to allow switch statements to continue to work if the enum definition is changed after it has been compiled, though I don't really see how that would work..

-- chris

- *Follow-Ups:*
 - ◆ **Re: Why Generics?**
 - ◇ *From:* Phillip Lord

- *References:*
 - ◆ **Why Generics?**
 - ◇ *From:* David Blickstein
 - ◆ **Re: Why Generics?**
 - ◇ *From:* Eric Sosman
 - ◆ **Re: Why Generics?**
 - ◇ *From:* Chris Uppal
 - ◆ **Re: Why Generics?**
 - ◇ *From:* Mike Schilling
 - ◆ **Re: Why Generics?**
 - ◇ *From:* Chris Uppal
 - ◆ **Re: Why Generics?**
 - ◇ *From:* David Blickstein
 - ◆ **Re: Why Generics?**
 - ◇ *From:* David Blickstein
 - ◆ **Re: Why Generics?**
 - ◇ *From:* Dale King
 - ◆ **Re: Why Generics?**
 - ◇ *From:* Chris Uppal
 - ◆ **Re: Why Generics?**
 - ◇ *From:* Phillip Lord

- Prev by Date: **Re: Why Generics?**
- Next by Date: **Re: peer to peer messaging**
- Previous by thread: **Re: Why Generics?**
- Next by thread: **Re: Why Generics?**
- Index(es):
 - ◆ **Date**
 - ◆ **Thread**