

Re: Design Problem Aggregation

Source: <http://coding.derkeiler.com/Archive/General/comp.object/2004-05/0720.html>

From: Robert C. Martin (unclebob_at_objectmentor.com)

Date: 05/11/04

Date: Tue, 11 May 2004 12:52:33 -0500

On 9 May 2004 15:54:24 -0700, merlin2769@hotmail.com (Merlin) wrote:

>Hi
>
>Imagine the following classes (A class diagram will help)
>
>BASE, A, B, C, D, E, F, G.
>
>
>A, B, C, D, G inherit from BASE.
>
>E, F inherit from D.
>
>Class E has a member (aggregation by value) of type A.
>Class F has a member (aggregation by value) of type B.
>
>
>Class G has a member that is container that can accept objects of type
>D. As D is the base class of E and F we can add to the container
>objects of type E or F. I have made the container type safe in this
>way.
>
>
>I wanted G to be a collection of objects of type A or B but never C so
>I introduced an abstract class D and made the container of that type
>so it would only accept objects of base type D. However, although this
>looks ok, I am not happy with the extra work it has created.
>As I need to access the interface to A and B, I need to repeat all
>that interface in E and F. A and B have many member functions and I
>dont want to rewrite all that interface in E and F and delegate the
>calls to the aggregate.

In effect E and F are Decorators that allow insertion into G. (See the Decorator pattern in the GOF book).

Another way to cut at this would be:

```
+->|Gpart|
| * A
```

```
|| |Base|  
| +--+--+ A  
| \ \ |  
| +----+----+----+----+  
| | \ \ |  
+--+|G| |A| |B| |C|
```

Thus, both A and B derive from a separate interface named Gpart. G contains a list of Gparts. When you pull a Gpart out of the list you then cast it to a Base and use it naturally.

Th