

Re: Lahman, how ya doing?

## Re: Lahman, how ya doing?

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.object/2005-05/msg00204.html>

---

- *From:* [glhansen@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:glhansen@xxxxxxxxxxxxxxxxxxxxxxxx) (Gregory L. Hansen)
  - *Date:* Mon, 16 May 2005 15:09:15 +0000 (UTC)
- 

In article <[daMhe.6627\\$4d6.2704@trndny04](mailto:daMhe.6627$4d6.2704@trndny04)>, H. S. Lahman <[hsl@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:hsl@xxxxxxxxxxxxxxxxxxxxxxxx)> wrote:  
>Responding to Hansen...

[...]

>> That is what I was angling for, and I think I've resolved that in the  
>> other message. So, send data in an event when the value needed is what  
>> existed when the event was sent, and not when it is consumed. In my  
>> specific application, an event queue is hardly needed since the events are  
>> consumed immediately, so one way or the other is all good.  
>  
>OK. I talked about this in the other message today, so I won't add  
>anything here.

Yeah, I think we can consolidate all of this in the other message.

>  
>>>>So what happens if the queue empties before...  
>>>>  
>>>>  
>>>>Things are more complicated in a truly asynchronous system where the  
>>>>queue may have to wait for an external event. (Or for events generated  
>>>>by a real- or scaled-time timer.) But then the push just needs to  
>>>>restart the queue operation if the event count is exactly 1 after the  
>>>>push. Since you are using simulation ticks rather than scaling, this  
>>>>  
>>>>  
>>>>Oh, okay. I suppose somewhere in the bowels of the queue it will be  
>>>>looping until it finds a quit event directed to queue?  
>>>>  
>>>>Not exactly. There can be times when the queue is completely comatose.  
>>>>  
>>>>  
>> But something must be looping or the program will hit the return 0 at the  
>> end of main(), and quit.  
>  
>As long as we are talking about generalities... B-)

Re: Lahman, how ya doing?

Re: Lahman, how ya doing?

>

>There is nothing that requires a program to execute instructions  
>continuously (though stopping it may require explicitly inserting a HALT  
>instruction). Events can be external to the application. For example,  
>memory resident programs are designed to be dormant almost all of the  
>time waiting for somebody to invoke a callback to push an event via RPC  
>or whatever. That's precisely because one doesn't want a memory  
>resident program to be constantly sucking down CPU cycles for a polling  
>loop.

Okay, that is more general than I'd thought of. But it makes sense.

>

>You've talked about the application actually running multiple  
>simulations. Between those simulations you want the queue to be  
>quiescent because you don't want a polling loop to be stealing cycles  
>from the UI or whatever is being done between simulations. IOW, between  
>simulations the event queue is just another object in the application  
>that isn't being accessed. So the default behavior in a reusable queue  
>should be that it doesn't do anything at all when it isn't needed ---  
>which is no different than any other object in the application.

After thinking about it a little, I like the way you did that in your  
sample event queue code.

I've read of differences of opinion on whether programming philosophies,  
like OOP versus structured, can be mixed. It seems that few people think  
they can or should. But I can imagine an OOP simulation included in a  
very structured analysis program that, in an algorithmic fashion, sends  
some parameters to Controller, pushes an event on to the queue to get that  
going, and then gets a data set to analyze.

>> Well, I did see the results of a survey of programmers about the  
>> advantages of OOP, and code reuse was at the bottom. It was still an  
>> advantage to be included in a list, but it was the least of the  
>> advantages. Some shops that have tried to make generic and eminently  
>> reuseable libraries have found it so time-consuming that it was just  
>> abandoned.

>

>So the programmers that were surveyed write their own String or Array  
>classes for each application? Good luck. Everybody gets reuse of  
>computing space objects through commercial libraries. That's because  
>they are very narrowly defined data holders. So is an event queue; it's  
>just a smart stack.

And the algorithmic programmer gets code reuse with stdio.h and math.h,  
too. I don't think that's what they were talking about as an OOP  
advantage.

>

>As far as object reuse in general is concerned, I agree it never lived

Re: Lahman, how ya doing?

Re: Lahman, how ya doing?

>up to the hubris of the '70s and early '80s. The problem is that one  
>can provide quite different valid syntaxes for the same semantics. So  
>when an object moves to a new reuse context the new client may be  
>expecting to use a different access syntax (interface) than the one  
>provided with the object for the original client. So even though client  
>and service are very clear about the semantics of the service, they  
>can't talk to one another.

I thought it would have more to do with tailoring something for a specific purpose versus building a Swiss army knife. You created your own queue in the sample code, and it would probably take about half a page to manage the pointers and so on. Compare that with the STL queue structure.

>  
>Basically there are three solutions. One can modify the new client to  
>use the access interface provided with the service rather than the one  
>it wants to use. That means touching the implementation of the client,  
>which is a major no-no in OO development. The second possibility is to  
>add the interface that the new client wants to see to the service. That  
>results in rapid increase in the complexity of the interface, much of  
>which is redundant. That redundancy hurts if one uses the interface to  
>"tailor" the object (e.g., provide a burdened cost from a base cost and  
>burden rate) because it requires double edits if the tailoring changes,  
>which is fragile.  
>  
>The third solution is to provide "glue" code between the interface the  
>reuse client wants and the interface the service object provides (e.g.,  
>a Facade pattern).

Glue? Is that also called a wrapper?

—

"Tell me, Dr. Einstein, at what time does Boston arrive at this train?"

.

---

• *Follow-Ups:*

- ◆ **Re: Lahman, how ya doing?**  
◇ From: H. S. Lahman

• *References:*

- ◆ **Re: Lahman, how ya doing?**  
◇ From: H. S. Lahman
- ◆ **Re: Lahman, how ya doing?**  
◇ From: Gregory L. Hansen
- ◆ **Re: Lahman, how ya doing?**  
◇ From: H. S. Lahman

• Prev by Date: **Re: Some Help Please!**

• Next by Date: **Re: A Java Brainteaser – a Static Factory method Narrative**

Re: Lahman, how ya doing?

Re: Lahman, how ya doing?

- Previous by thread: ***Re: Lahman, how ya doing?***
- Next by thread: ***Re: Lahman, how ya doing?***
- Index(es):
  - ◆ ***Date***
  - ◆ ***Thread***