

## Base Language - Feature #7000

### Replace old-style synchronization primitives with java.util.concurrent classes

12/20/2022 02:58 AM - Igor Skornyakov

<b>Status:</b>	New	<b>Start date:</b>	
<b>Priority:</b>	Normal	<b>Due date:</b>	
<b>Assignee:</b>		<b>% Done:</b>	0%
<b>Category:</b>		<b>Estimated time:</b>	0.00 hour
<b>Target version:</b>		<b>vendor_id:</b>	GCD
<b>billable:</b>	No		
<b>Description</b>			

#### History

##### #1 - 12/20/2022 03:22 AM - Igor Skornyakov

FWD code contains many places where old-style synchronization primitives (synchronized blocks, wait/notify) are used. I think that it makes sense to re-work such places using `java.util.concurrent` classes. This will not only potentially reduce contention but will make the program logic cleaner.

For example, we can use `ReentrantReadWriteLock` instead of synchronized blocks for a more fine-grained synchronization. Using classes from `java.util.concurrent.atomic` allows us to eliminate synchronization on access to a single variable. Using concurrent collections simplifies working with shared maps, sets, lists, and queues. Using barriers and executors instead of code based on wait/notify greatly simplifies the program logic. In many situations using executors also allows re-using the threads instead of starting new ones which is more expensive.

##### #2 - 12/20/2022 06:24 AM - Greg Shah

- Start date deleted (12/20/2022)

Do you have specific areas seen in profiling where these old-style primitives are heavily involved?

##### #3 - 12/20/2022 07:47 AM - Igor Skornyakov

Greg Shah wrote:

Do you have specific areas seen in profiling where these old-style primitives are heavily involved?

It is difficult to see this in a profiler because we need to simulate a heavy load to see the monitors' activity in a situation of high contention. However, I've seen a significant improvement with `java.util.concurrent` in my previous projects where it was relatively easy to create an automated stress test and see the application behavior under a high load.