Lesson 4 : Locks and Conditions

Monday, April 29, 2024 5:13 PM

We know that synchronized works if the method is invoked by same instance of object. If other object is used to call the syncronized method, both will get executed simultaneously as monitor locks work on same instance of object.
But what if we want to secure the method or block no matter it is invoked using some other object ?
This is realtime condition. To tackle this situation we have custom locks.

There are 4 types of locks :

1. Reentrant lock
2. ReadWrite lock
3. Stamped lock
4. Semaphore lock

These locks do not depend on object like the one we saw in syncronized methods.

Reenterent Lock : See code examples

ReadWrite Lock : Before learning about RWLock, we need to learn about shared lock and exclusive lock.

Suppose there are two threads and one resource(part of code which needs to get executed).
If t1 has put shared lock, t2 can also take shared lock and read the values
Exclusive lock can only be taken if there is no lock on that resource, and no lock can be taken until exclusive lock is released.

With shared lock you can only read. With exclusive lock you can read and write.
ReadLock/SharedLock : More than one thread can acquire the read lock.
WriteLock/ExclusiveLock : Only one thread can acquire the write lock.
See Code example.

When to use ReadWriteLock ?
Read are very high compared to write request.

Stamped Lock :
It supports Read/Write functionality like ReadWriteLock along with optimistic lock also
Optimistic lock --

It is technique of SQL database where it does not put a row lock and allows multiple writes to happen by using row version

ID	NAME	DESIGNATION	ROW VERSION (hidden column)
123	Zain	Student	1
456	Asus	Teacher	1

ROW VERSION is hidden column used internally, it increases by 1 when update happens on row

If two threads go for update of id 123, so both read row version, i.e 1.
Both set the new designation t1 sets ExStudent, t2 sets Admin. (Prepare variable phase)

Now when they go to update one will reach first to update and before updating it will check row version is same as of time of reading or not. It will update the version to 2 as soon as it updates.
Now the second thread will try updating but now row version is not same so it will again read the latest value and then try to update by matching row version.

Similar concept of stamp is used in StampedLock, where while locking a stamp is returned and while updating the stamp can be validate using lock.validate(stamp). See Code example for better clarity.

Semaphore Lock :
At time of creation of this lock, it accepts a integer parameter -> permits. This permit integer indicates the number of threads that can acquire this lock. See code examples
When is semaphore useful ?
Connection pool : suppose you have defined max connection pool of 5 so there you can use semaphore lock

INTER THREAD COMMUNICATION

Threads can communicate with each other using wait(), notify(). But that works in case of monitor lock. What about the custom locks like reenterant, semaphore etc. For that we have conditions.
await() = wait()
signal() = notify()
signalAll() = notifyAll()

See code example