Homework 3: Java shared memory performance races

Testing Environment:
Project was done in SEASnet using
- Java version 1.8.0_112

Implementation
Synchronized
The instructors gave the implementation for Synchronized. It is reliable, given that it uses synchronized keyword.

Unsynchronized
Have the same implementation as Synchronized without the Synchronized keyword for the swap() function. Because of this, this class often encounters deadlocks. As a matter of fact, deadlock can occur with as low as two threads. It does work faster than Synchronized, but it is not very reliable.

GetNSet
Instead of using byte array to store value, we are using Atomic Integer Array, which is thread-safe. This implementation is in fact faster than Synchronized although slower than Unsynchronized. However, it is more reliable than Unsynchronized.

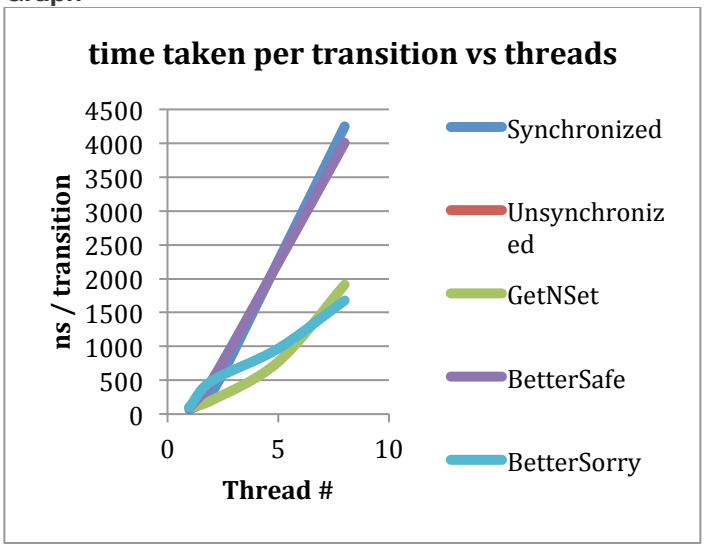
BetterSafe
For BetterSafe, I used lock for the implementation. The idea is to limit resource access to prevent race condition in dangerous blocks. This implementation is also reliable. However, it is slower than GetNSet implementation. It also did not do very well in low threads compared to Synchronization. However, as the number of threads increase, its performance also becomes better compared to Synchronized.

BetterSorry
For BetterSorry's implementation, I also used Atomic Integer. However, it is only used when incrementing and decrementing values. This implementation becomes better than GetNSet as the number of threads increases. It also has better reliability than Unsynchronized. This is because unlike Unsynchronized, this at least guards the resources during increment and decrement to avoid race condition. When the thread becomes higher, the probability of failure increases.

Results

Thread#	Sync	Unsync	Get	BeSafe	BeSorry
1	64	71	93	94	104
2	309		203	496	489
5	2263		771	2237	978
8	4249		1911	4006	1677

Graph



The graph above shows the performance of each implementation. As can be seen, some implementations may perform better compared to another when the number of threads is increased.

If we are looking only at performance, BetterSorry might be the best choice as it is the fastest while having pretty good reliability. However, if we want 100% reliability, BetterSafe would be the better choice, as it has better performance than Synchronized while also retaining 100% reliability.