Calvin Chan
304144970
CS131 HW 4

The BetterSafe Implementation is faster than Synchronized because the Synchronized state runs the entire swap method in sequence. There can be no interleaving of the threads that run the swap method whatsoever. However, the BetterSafe Implementation uses the AtomicIntegerArray class to implement synchronization on a lower level. Through volatile puts and gets on individual array elements, instead of the entire swap method, BetterSafe provides better performance than Synchronized. Both BetterSafe and Synchronized are data-race free because the protect against deadlock and ensure happens-before relationships between threads.

BetterSorry is even faster than Better Safe because it does not check for race conditions between two threads that may get and set particular elements in the array. As such, one thread may call get on an array element, and before that thread can set it, another thread sets that element. When the prior thread sets the element, the second thread's write will be lost. As such, BetterSorry is not data-race free. BetterSorry only protects against deadlock and prevents concurrent operations on a single data element.

**Summary of models**

**Synchronized:** Data-race free, but slow. 100% reliability
**Unsynchronized:** Not Data-race free. Can be faster than Synchronized, but not reliable. Deadlocks when there is a high number of swaps
**GetNSet:** Not Data-race free. Can be faster than Synchronized, but not reliable. More reliable than Unsynchronized in that it prevents concurrent operations on individual data elements, but still has thread races. Deadlocks when there is a high number of swaps.
**BetterSafe:** Data-race free. Faster than Synchronized and is 100% reliable.
**BetterSorry:** Not Data-race free. Similar to GetNSet, it prevents concurrent operations on individual data elements, but still has thread races. However, it eliminates deadlock, making it superior to GetNSet.

The test results below show the test cases used to demonstrate DRF (or lack thereof) in each model.

**Test Results** (Used with MAXVAL = 6 and ARRAY = [5 6 3 0 3]

**Synchronized 8 threads 1000000 swaps**
```
Threads average 3220.26 ns/transition
Threads average 2796.12 ns/transition
Threads average 2264.24 ns/transition
Threads average 1869.98 ns/transition
Threads average 1510.89 ns/transition
```

**Synchronized 16 threads 1000000 swaps**
```
Threads average 6006.97 ns/transition
Threads average 5183.15 ns/transition
Threads average 5142.49 ns/transition
Threads average 5441.88 ns/transition
Threads average 6148.22 ns/transition
```

**Unsynchronized 8 threads 1000000 swaps**

```
deadlock
deadlock
deadlock
deadlock
deadlock
```

**Unsynchronized 8 threads 1000 swaps**

```
Threads average 72735.0 ns/transition
sum mismatch (17 != 15)
Threads average 70675.6 ns/transition
sum mismatch (17 != 20)
deadlock
Threads average 78509.4 ns/transition
sum mismatch (17 != 21)
Threads average 79078.2 ns/transition
```

**Unsynchronized 16 threads 1000 swaps**

```
Threads average 190033 ns/transition
sum mismatch (17 != 15)
Threads average 140576 ns/transition
sum mismatch (17 != 19)
Threads average 159914 ns/transition
sum mismatch (17 != 19)
Threads average 152847 ns/transition
sum mismatch (17 != 15)
Threads average 155227 ns/transition
sum mismatch (17 != 15)
```

**GetNSet 8 threads 1000000 swaps**

```
deadlock
deadlock
deadlock
deadlock
deadlock
```

**GetNSet 8 threads 1000 swaps**

```
Threads average 96544.2 ns/transition
sum mismatch (17 != 10)
Threads average 107976 ns/transition
sum mismatch (17 != 12)
Threads average 89930.2 ns/transition
sum mismatch (17 != 10)
Threads average 91667.0 ns/transition
sum mismatch (17 != 16)
Threads average 91106.1 ns/transition
```

```
sum mismatch (17 != 14)
```

**GetNSet 16 threads 1000 swaps**

```
Threads average 233658 ns/transition
sum mismatch (17 != 16)
Threads average 198441 ns/transition
sum mismatch (17 != 19)
Threads average 239045 ns/transition
sum mismatch (17 != 24)
Threads average 204615 ns/transition
sum mismatch (17 != 12)
Threads average 202241 ns/transition
sum mismatch (17 != 19)
```

**BetterSafe 8 threads 1000000 swaps**

```
Threads average 1459.58 ns/transition
Threads average 967.500 ns/transition
Threads average 1563.42 ns/transition
Threads average 984.096 ns/transition
Threads average 1267.27 ns/transition
```

**BetterSafe 16 threads 10000000 swaps**

```
Threads average 2956.69 ns/transition
Threads average 2690.66 ns/transition
Threads average 2799.38 ns/transition
Threads average 3318.16 ns/transition
Threads average 2839.19 ns/transition
```

**BetterSorry 8 threads 1000000 swaps**

```
Threads average 1526.84 ns/transition
output too large (9 != 6)
Threads average 1643.86 ns/transition
output too large (9 != 6)
Threads average 1544.02 ns/transition
output too large (10 != 6)
Threads average 1526.34 ns/transition
output too large (8 != 6)
```

**BetterSorry 16 threads 1000000 swaps**

```
Threads average 1523.45 ns/transition
output too large (10 != 6)
Threads average 3099.20 ns/transition
output too large (13 != 6)
Threads average 2809.38 ns/transition
output too large (8 != 6)
Threads average 2472.13 ns/transition
output too large (9 != 6)
Threads average 3461.93 ns/transition
output too large (9 != 6)
```