To test the models I ran them each 30 times under 3 sets of inputs:
8 threads, 100,000 swaps, and 20 elements (4x10, 4x20, 4x30, 4x40, 4x50)
16 threads, 1,000,000 swaps, and 20 elements (4x10, 4x20, 4x30, 4x40, 4x50)
32 threads, 1,000,000 swaps, and 20 elements (20x100)

Synchronized was by far the slowest method but also had 100% reliability. This model is obviously DRF since it uses the synchronized keyword for the swap function, preventing any other thread from accessing the method while it is in use. For the 3 test cases its average times were: 5000ns, 9500ns, 10800ns

Unsynchronized was the fastest method but also horribly unreliable. This method relies on concurrent simple loads and stores on the shared data and is highly vulnerable to data races. Its times were: 3000ns, 1500ns, 2600ns with a sum difference of 100/600, 3000/6000, 5000/2000,

GetNSet in the middle of the field in terms of speed and reliability. Although the individual array accesses are atomic, the load, modify, and store process caused unreliable results. Somewhat However when more processors were introduced this model seemed to stabilize. it Its times were: 3500ns, 2400ns, 5000ns with sum differences of 300/600, 1500/600, 500/2000

BetterSafe had moderately good speed with 100% reliability. This model avoided DRF by using synchronized statements for each array element, which allowed it to perform faster than Synchronized. Its times were: 4000ns, 3200ns, 6900ns.

BetterSorry had the second best speed and but poor reliability. By using atomic integers instead of arrays, this method was able to operate faster than GetNSet by sacficing stability. At lower processor counts this method excelled. Its times were: 2900ns, 1600ns, 3200ns with sum differences of 150/600, 3000/600, 1000/2000