Briefing about **Elimination Backoff Stack:-**

We wish to allow threads with pushes and pops to coordinate and cancel out, but must avoid a situation in which a thread can make a sporting offer to more than one other thread. We do so by implementing the EliminationArray using coordination structures called exchangers, objects that allow exactly two threads (and no more) to rendezvous and exchange values.

we need a lock-free exchange, one in which threads spin rather than block, as we expect them to wait only for very short durations.

LockFreeExchanger:-

A LockFreeExchanger<T> object permits two threads to exchange values of type T. If thread A calls the object's exchange() method with argument a, and B calls the same object's exchange() method with argument b, then A's call will return value b and vice versa. On a high level, the exchanger works by having the first thread arrive to write its value, and spin until a second arrives. The second then detects that the first is waiting, reads its value, and signals the exchange. They each have now read the other's value, and can return.

An EliminationArray is implemented as an array of Exchanger objects of maximal size capacity.