Deadlock

A deadlock occurs when two or more threads (or processes) are waiting indefinitely for resources that the other thread is holding, leading to a situation where none of them can proceed. A classic scenario involves two threads trying to acquire two locks in different orders:

* Thread 1 locks LockA and waits for LockB.
* Thread 2 locks LockB and waits for LockA.
* Neither thread can proceed because they are each holding the resource the other needs.

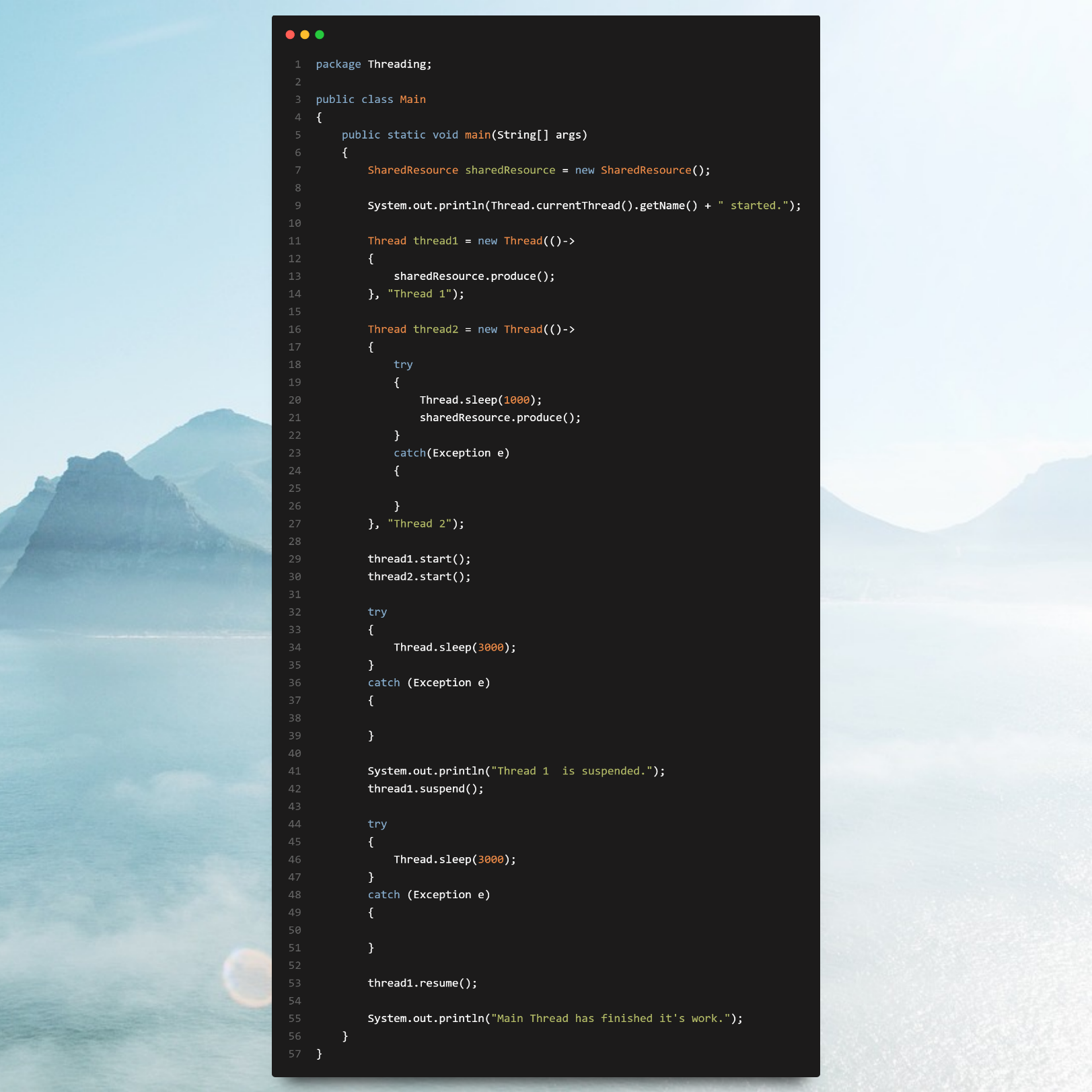
Why stop, resume, suspended methods are deprecated?

Stop: It terminates the thread abruptly, No lock release, No resource clean up happens.

Suspend: Puts the thread on hold temporarily. No Lock is release too.

Resume: Used to resume the execution of suspended thread.





Race Condition

A race condition is a broader concept where the outcome of a program depends on the timing or sequence of uncontrollable events, such as thread execution order or process scheduling. If two or more operations execute out of the expected sequence, a race condition may lead to incorrect behaviour or bugs.

Imagine a system where two threads are responsible for withdrawing money and checking the balance from a shared bank account. If thread A checks the balance and finds sufficient funds, and thread B withdraws the money immediately afterward, thread A may still proceed to withdraw, leading to an overdraft due to a race condition.

Data race

A data race occurs when two or more threads or processes access shared data concurrently, and at least one of them is modifying the data. If proper synchronization mechanisms (like locks, semaphores, or atomic operations) are not used to coordinate these accesses, the result becomes unpredictable. Data races often lead to incorrect program behaviour.