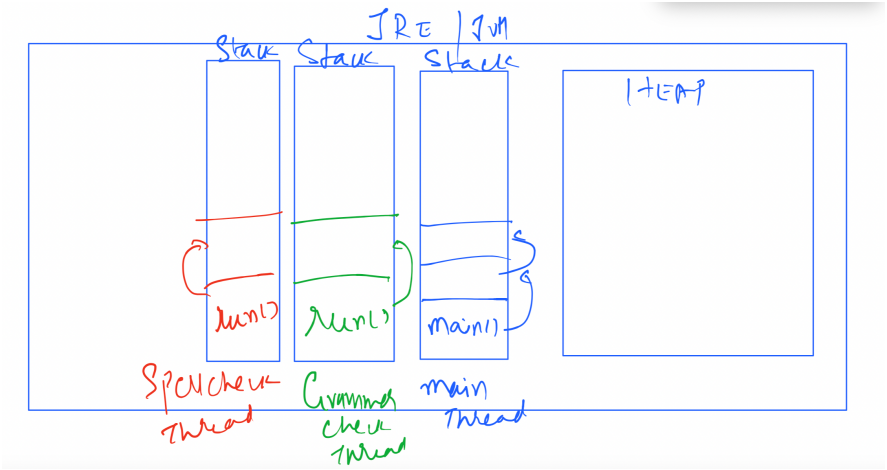
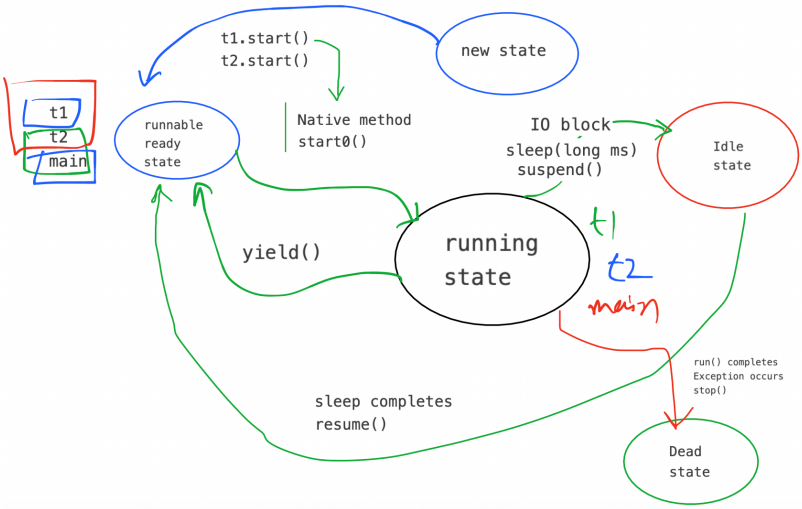
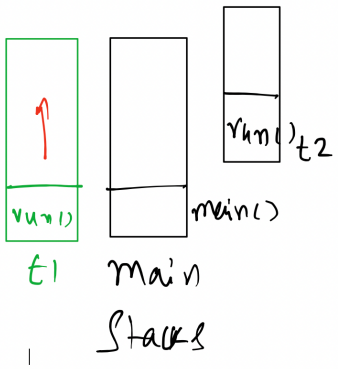
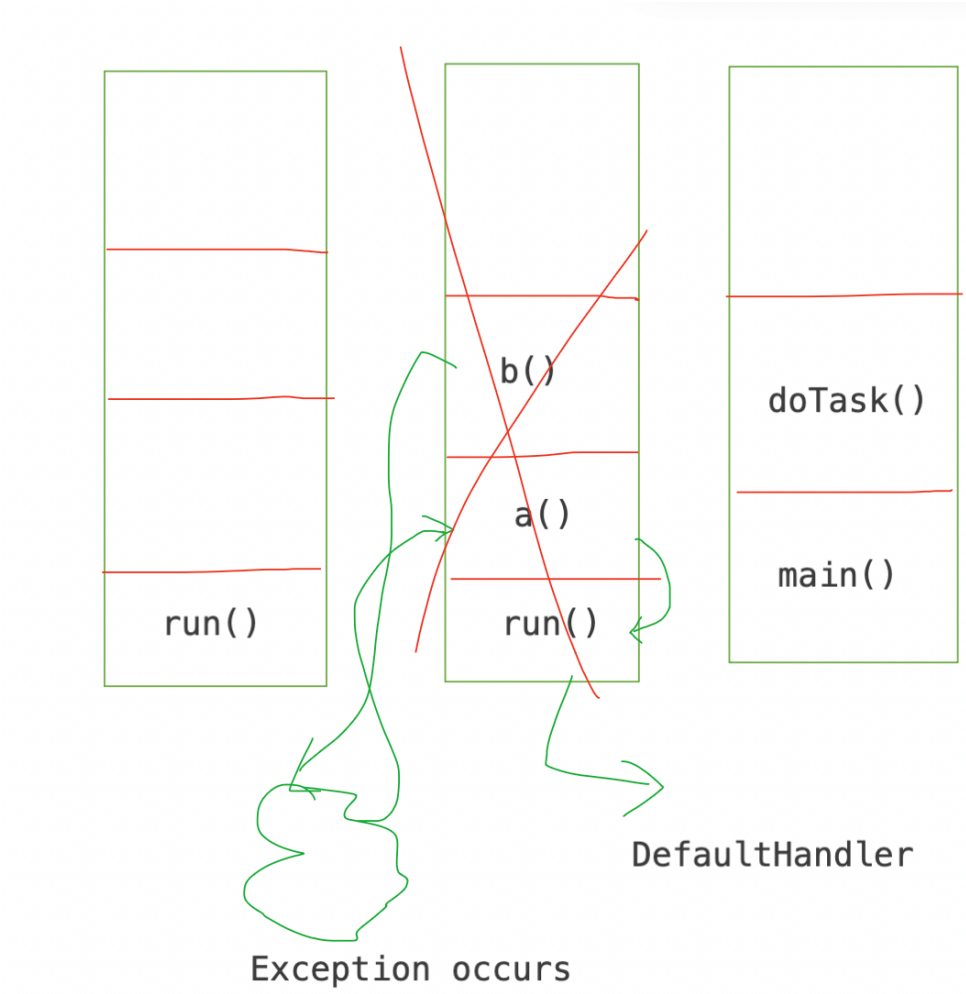


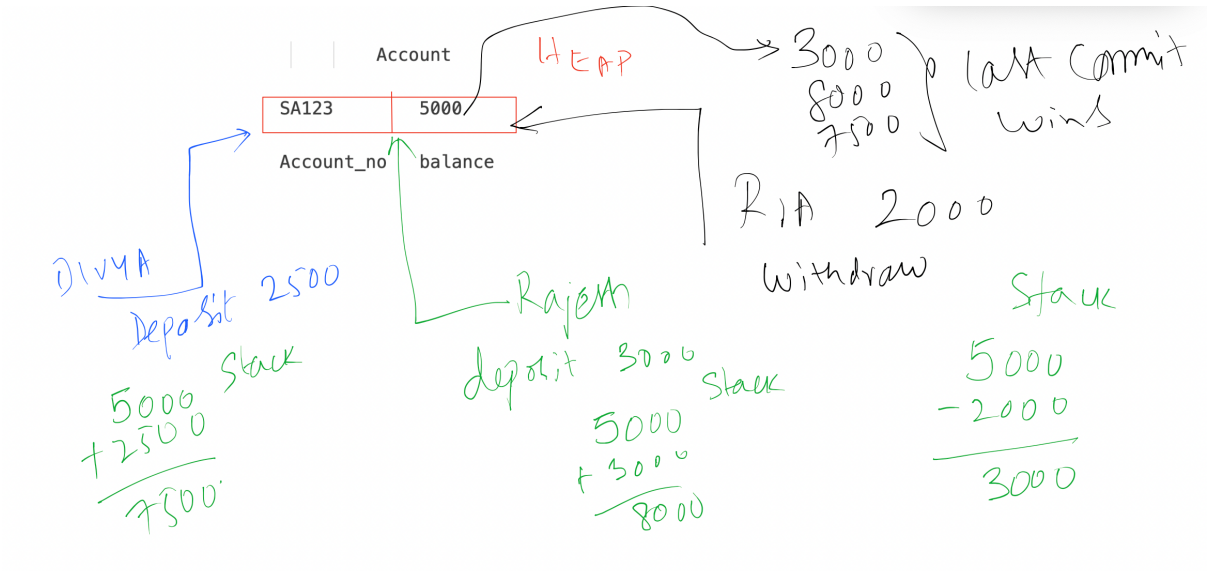

```
SpellCheckThread t1 = new SpellCheckThread();
GrammerCheckThread t2 = new GrammerCheckThread();
```



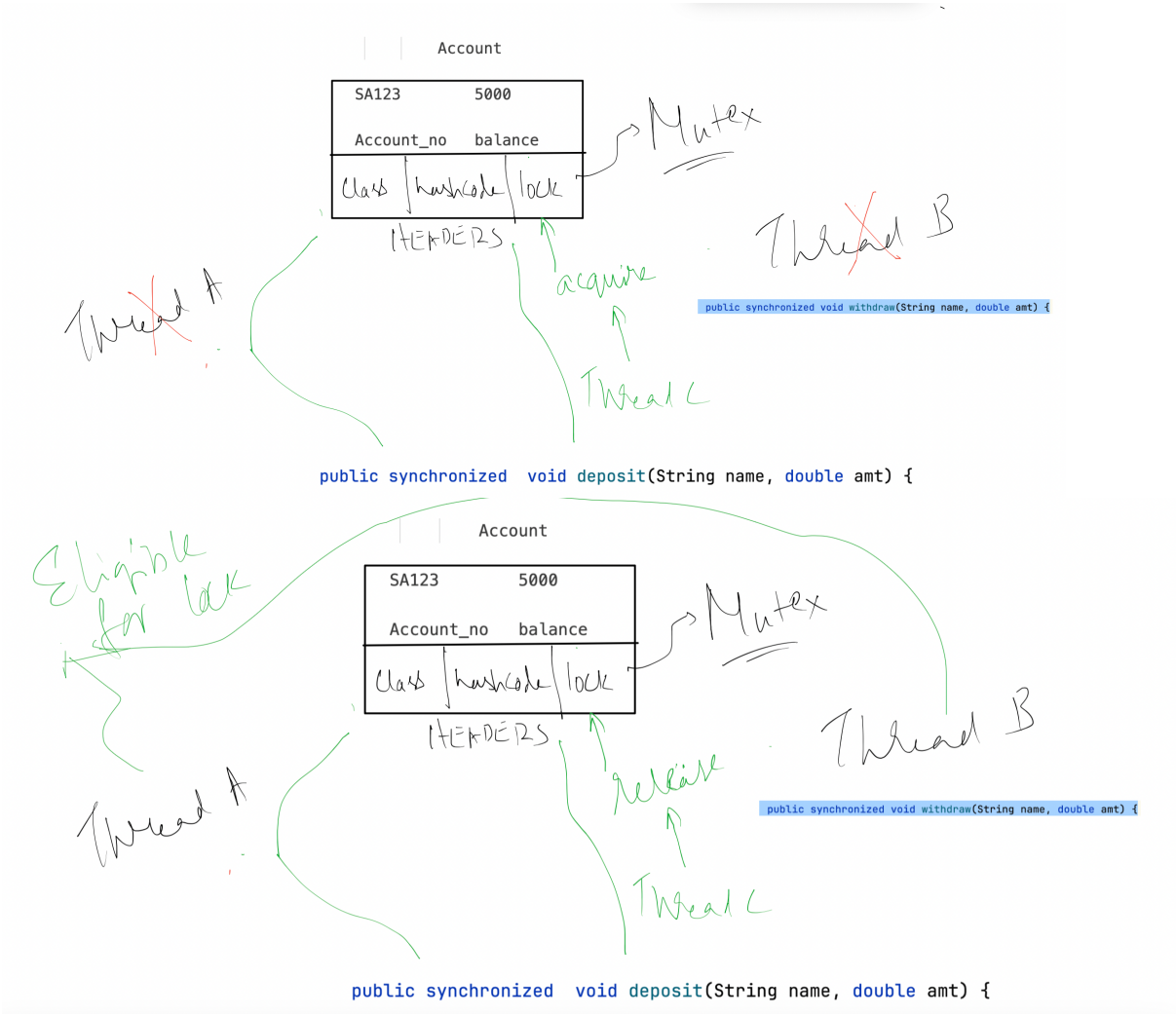




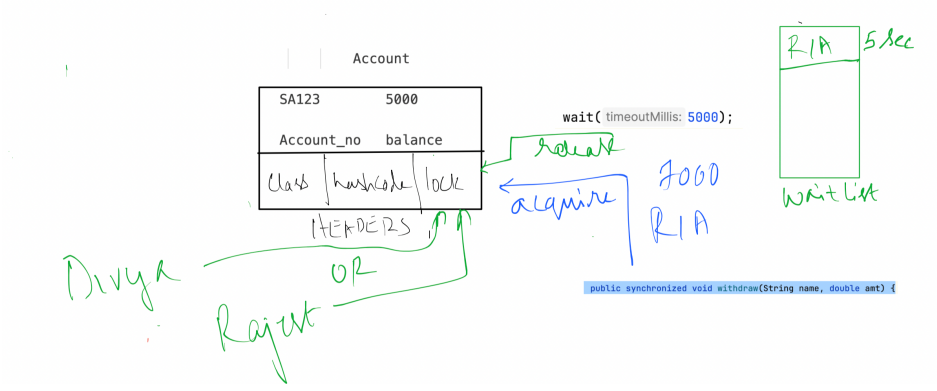
Thread Safety:
Instance variables are not Thread Safe.



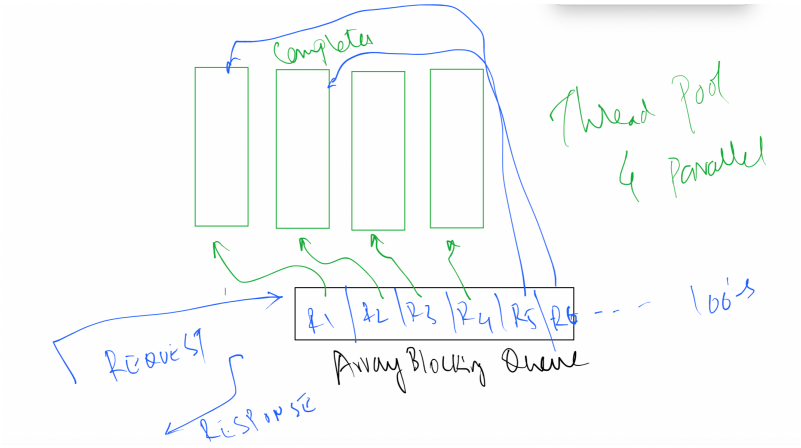
Thread C acquires the lock to execute a critical section, Thread B and Thread A has to wait for Thread C to release the lock before they execute deposit () or withdraw ()
But they are free to access getBalance() → not critical because not marked as synchronized



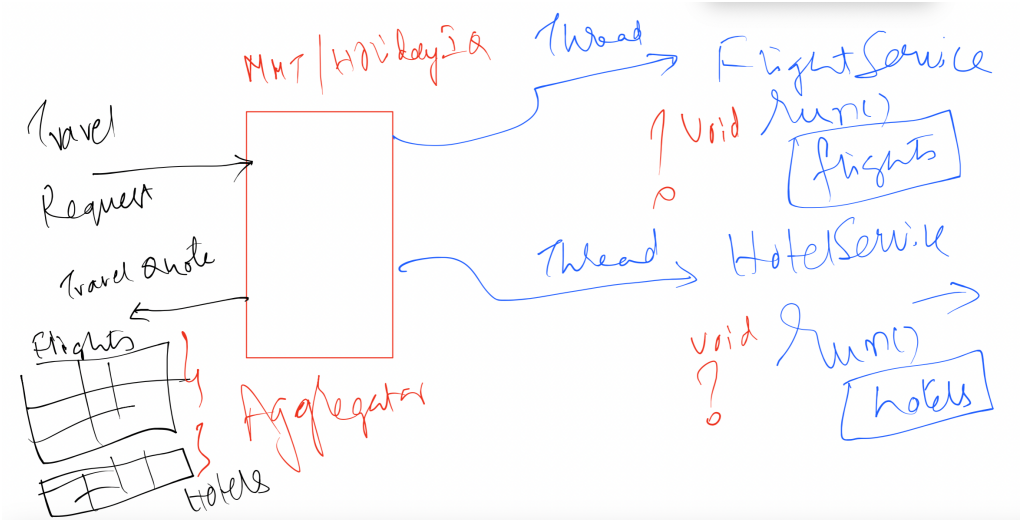
Inter Thread Communication:



Thread Pools

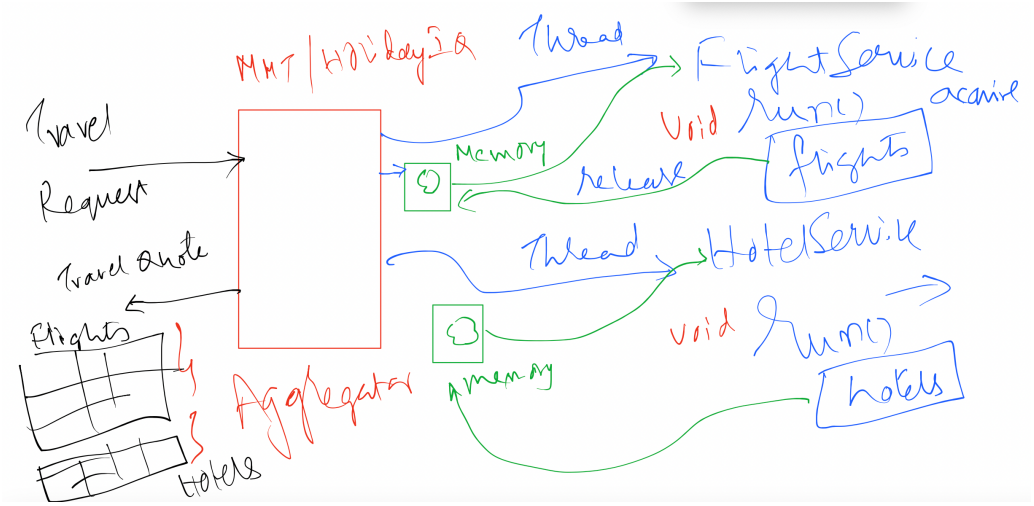


Runnable public void run() {} can't return the values



One solution:
Aggregator can create object and share with FlightService, another object with HotelService.
Services are going to acquire lock on the shared objects, once data is fetched it will update the shared objects and release the lock

Problem is MMT/HolidayIQ aggregators has to keep looking at the shared memory for lock and if services has updated the data [long pooling]


MMT / Holidays
Thread
FlightService
acquire
Void run()
Memory
release
flights
Travel
Request
Travel quote
Thread
HotelServic
Flights
Void run()
Aggregator
memory
hotels
Hotels