1. Design a shared counter in Java.

Input:

Number of threads (n) and

Maximum count in each thread (N)

Output:

Display the value of each counter in parallel, as the value gets incremented in each counter.
All the counters should reset to zero when at least one of them reaches maximum count N.

2. Dining Philosophers' problem is stated as follows.

There are five philosophers (P1 – P5) sitting around a circular table, spending their lives doing only eating and thinking. They have five plates, one in front of each person. They have five forks also, fork 1 is to the right side of philosopher 1 and to the left of philosopher 2. It can be used by either of them. Similarly, fork 2 is shared between philosopher 2 and 3.. fork 5 between philosopher 5 and philosopher 1. A philosopher needs both the left and right forks to eat.

After eating, she puts both of them down and then they can be picked by another philosopher.

The eating and thinking of philosophers can be simulated using a programme, with 'Philosopher' as a class, and each philosopher running a separate thread.

In each thread, do this : pick up both (left and right) forks, keep eating for a while (put some time lapse statement), and put the forks back (so that others could pick it up and use it for eating).

Then wait for some time (do some thinking), and then again try to pick up both the forks and eat. A fork can be picked up only if it is free, so you need to synchronize between the threads.

There is a potential problem with this : If all five philosophers pick up the fork on their left side first simultaneously, all of them would be waiting for the other fork (the one on their right side) to become free. Since everyone is waiting with one fork hoping that they would get the other one soon, nobody puts the fork back, so this results in an eternal circular wait.

See how this problem can be resolved.

You may read more about this problem and see the solution at
https://www.baeldung.com/java-dining-philoshophers

Try to understand the issues involved, and how each issue is solved. Then write your own code.