TDT4165 Programming Language - Autumn 2021
Alexander Høyby
Bjørn Are Odden

Scala Project Delivery 1

Task 1:

A - Nothing to comment

B - Nothing to comment

C - Nothing to comment

D - BigInt is 8 bytes whereas Int is 4 bytes. This results in a bigger range and more memory usage, but prevents overflow when doing arithmetic with bigger numbers.

Task 2:

A - Nothing to comment

B - We had to modify the incr counter method, as it would always print the same "correct" output. We did this by adding looping to actually see that the results varied between runs. (Which is what expected beforehand) Probably due to the small amount of computation needed, as the thread would be finished before the main thread is able to start the next thread.

C - The increase method was made atomic using the synchronized call as described in the provided material. This ensures that one, and only one, thread can execute the subsequent code block at any given time, and the function is now thread safe.

D - A deadlock occurs when two processes are waiting for a resource that is locked by the the other process.

An example:

```
Thread 1 (T1) wants resource 1 (R1) and resource 2 (R2).
Thread 2 (T2) also wants R1 and R2.
T1 locks R1 but gets interrupted by T2 locking R2.
Now both T1 and T2 are waiting for R2 and R1 respectively -> We have a deadlock.
```

Preventing deadlocks can be done by eliminating any of the 4 deadlock conditions:

1. Mutual exclusion - Each resource is either currently allocated to exactly one process or it is available. (Two processes cannot simultaneously control the same resource or be in their critical section).
2. Hold and Wait - processes currently holding resources can request new resources

3. No preemption - Once a process holds a resource, it cannot be taken away by another process or the kernel.
4. Circular wait - Each process is waiting to obtain a resource which is held by another process.