YSC3248: Parallel, Concurrent and Distributed Programming

Programming Assignment 3

In this assignment, you will be implementing a synchronisation primitive called **Counting Block**. A counting block helps to differentiate between *passive* and *active* threads, structuring a concurrent execution in stages. It allows to block passive threads until all active threads give permission for the passive threads to proceed. Count block is initialised with a number n that corresponds to how many active threads that it is going to account for. When an active method is ready for the passive threads to run, it calls `countDown()`, which decrements the counter. Each passive thread calls `await()`, blocking itself until the counter reaches zero. This programming assignment consists of two sub-tasks:

1. Provide a `CountBlock` implementation and tests for it, exercising it on multiple scenarios with many passive and active threads.
2. Provide a `ReusableCountBlock` implementation, which "restarts" the counter after it has reached zero, allowing for multi-stage interactions. Implement tests for it.

The template repository for this assignment is available on Canvas.
Once you clone the template repository, you'll be ready to go! The files you need to modify are:

- `CountBlock.scala`
- `CountBlockTests.scala`
- `ReusableCountBlock.scala`
- `ReusableCountBlockTests.scala`

As before, portion of this assignment's total score is reserved for better testing. Passing basic functionality tests, invoked in `*Tests.scala` files, will reward you with some points; however you are also expected to write any additional tests you deem necessary.

To hand in your code for this assignment, submit the link to the tagged release in your repository on GitHub. Feel free to add additional comments about your implementation and discovery to the `README.md` file in the root of the project.

Here are some tips to get you started on the assignment:

- Start by making sure you understand how the example test in `CountBlockTests` is supposed to work.