Exercise: Threading

In these exercises, you will work with C# threads to create software with concurrency.

Exercise 1:
Create a new C# console application project and add a new class "*HelloWriter*" to the project.
HelloWriter shall have a name (you can use a property or assign a name in the constructor).
Create a *SayHello()* method, which loops 1000 times and in the loop outputs:

"Hello from <name> #<number of times run>"

e.g. "Hello from writer A #394".

Instantiate two *HelloWriters*, with different names.
Create two threads and make one thread run the *SayHello()* method in one of the *HelloWriters* and make the other thread run the *SayHello()* method in the other *HelloWriter*.

Run your program a couple of times and observe the output.

Exercise 2:
Change your program, so the number of times to loop in the *SayHello()* method is configurable.
There are multiple ways to do this:

1. Create a *numberOfTimesToLoop* property on the *HelloWriter*.
2. Pass the number of times to loop to the *HelloWriter* constructor.
3. Pass the number of times to loop in the *Start()* method on the thread.

Try implementing all three ways (one at the time).
Discuss with another student: Are there any benefits or drawbacks of the different methods above?

Exercise 3:
Make your HelloWriters sleep between each output. One should sleep 200 ms and one should sleep 500 ms.

Exercise 4:
Make your main thread (the one staring the other threads) write "Hello from main" after starting the threads.
Observe the console output. When do you see the message from the main thread?

Exercise 5:
Change your code, so the main thread does not output "Hello from main" until both your HelloWriter threads have finished.

Exercise 6:
Add a new thread, the *NeverEndingStoryThread*, to your program. The thread shall run an endless loop, which writes "Never ending story" every 5 seconds. Run your program. What happens?

Exercise 7:
Change the *NeverEndingStoryThread* to be a background thread. What happens?

Exercise 8:
Change the *NeverEndingStoryThread* back to being a foreground thread.
Add code, so the *NeverEndingStoryThread* is stopped gracefully after your two HelloWriter threads have finished.
How long does it take for the program to stop?

Advanced exercise 1:
Your mission, *should you choose to accept it*, is to create a chat program running in a console window.
Talking to your self is no fun (what's the point in always being right, right..?), so when your program starts, it shall connect to a chat program on one of your fellow students computers.
The text you enter shall be sent to the other computer and written to the console.
Likewise, text entered on the other computer shall be sent to your computer and written to the console.

Hint: You will probably need a *TCPListener* and *TCPClient* for your application. And multiple threads of course.