Task 3

1. **What does the Thread.join() do? What happens if we remove it from the main method?**

The Thread.join() call is a blocking operation which makes the calling thread wait for the created thread to finish/die.
The calling thread does not wait for the created thread to die before resuming operation.

2. **Compare the real elapsed time using a stopwatch on your phone with your Java implementation. Does the elapsed time differ, why?**

The times do differ. I'm assuming because my solution isn't optimized at all and is lacking the typical observer design pattern regarding threads. There can be lots of factors that are in play, whether the operations used are blocking or non-blocking is also an important factor.

Final observations

How did you implement the task?

Task 1:
For this task I start out with the basic file that the teachers provided and go on from there. I make a second class that inherits from the Runnable class, ExternalThreadWorker, which is in charge of executing a new task on a new thread. This thread is started from the main function in TestProcessBuilder.

Task 2:
I made a class called StopWatch which has a main function and an implicit declaration of a thread method, which is being declared and run at the very start of the main function, finally being joined with the main thread.

Why did you solve it in the way you did it?

Task 1 and Task 2:
Easiest and cleanest solutions I could find

What difference in behaviour did you notice?

I don't quite understand what the question is referring to, compared to what?