Executor framework is awesome feature to use threads like a kid.

When we want to run 100 tasks async, then we have to create threads inside a for loop, and java create 100 threads, which corresponds to 100 threads in the OS.

1 java thread = 1 OS thread

Instead of this java provides us a pool of threads and we can assign the tasks to the pool and the threads in the pool will be automatically allotted for the tasks and reuses the threads again and again for further tasks instead of creating new OS threads.

Consider a thread pool of 10 threads will be working and allotted to 100 tasks, they’ll be used again and again to finish 100 tasks with 10 threads.

You can think, this makes the 90 tasks little bit wait time, right?

even if we create 100 threads like traditional way, all threads won’t be created at same time, it will be based of the count of cores of the CPU, that means there is a waiting time there.

comparatively, the pool concept is faster.

What should we have as the pool size? (best practice)

* If the tasks are CPU intensive task, then pool size should be number of processors size.
  + Runtime.getRuntime().availableProcessors() -> gives us the count of processors of the CPU system.
* If the tasks are IO intensive, then pol size can be high, our wish.

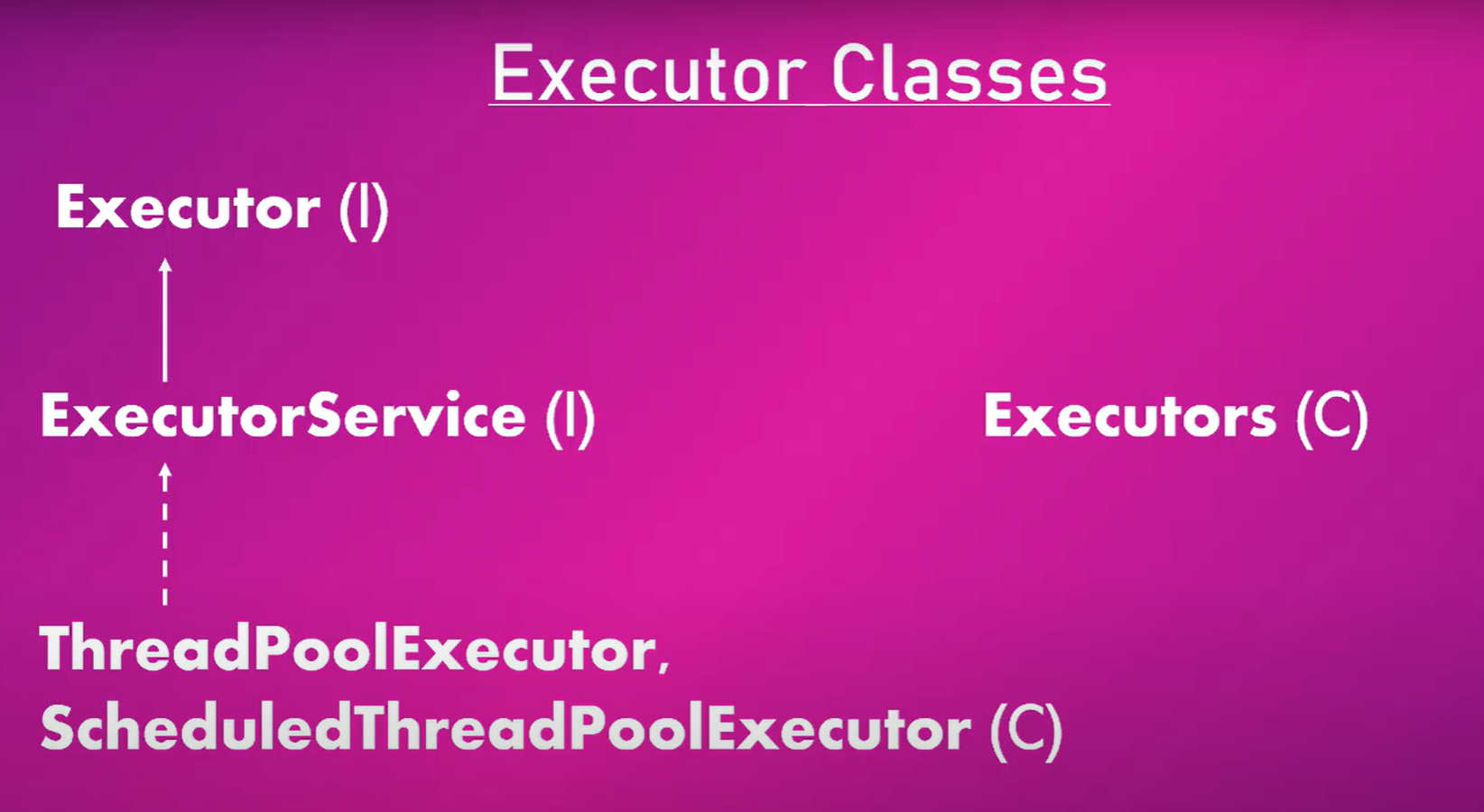
Advantages of Executor framework:

* Separates task and thread creation.
  + We will take care of the task and logic, and executor framework will take care of the thread creation and handling.
* Provides pooling support.
* Support to get return value from a thread using Callable and Future interfaces.

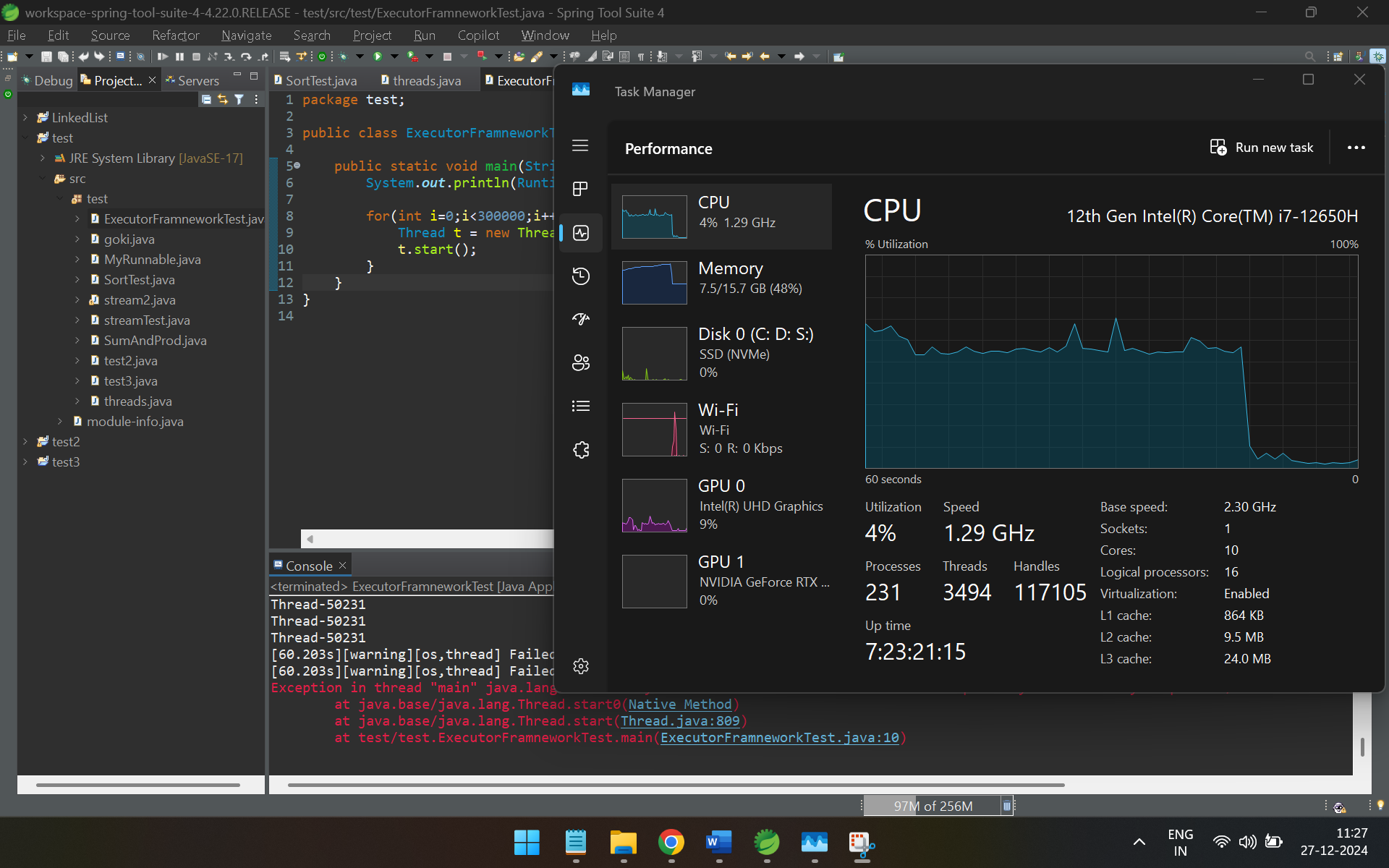
Types of thread pools:

1. FixedThreadPool
2. ChachedThreadPool
3. ScheduledThreadPool
4. SingleThreadedExecutor

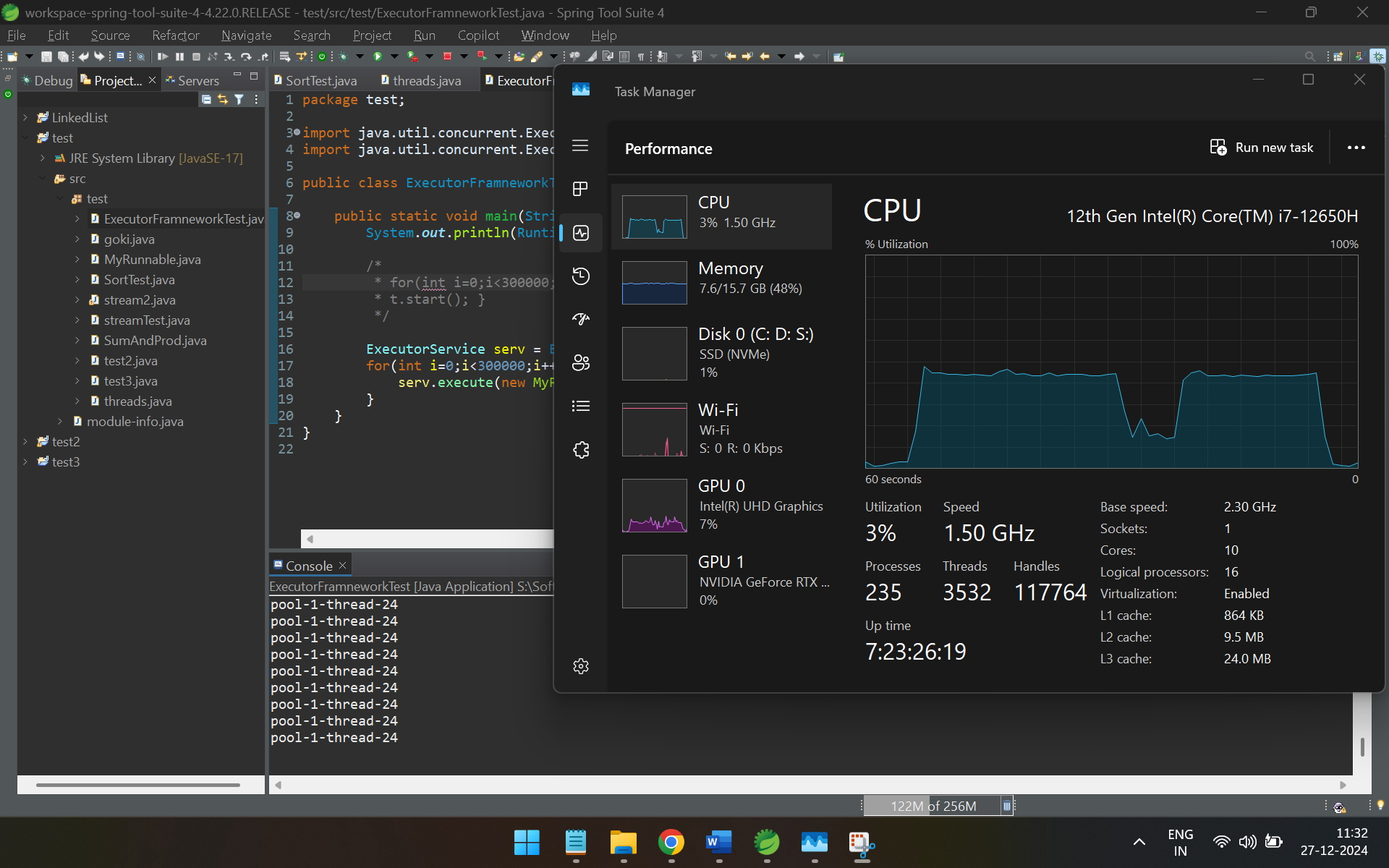
Executors is an utility class used for creating ThreadPoolExecutors, with many options, and user friendly, and we receive it in Executor service interface.



Running a traditional thread program with 10000 threads created in a for loop.



And using executor:





Reference: <https://www.youtube.com/watch?v=Tju58ioqRcA>

Reference: <https://www.youtube.com/watch?v=r-YnSDhN1ao>