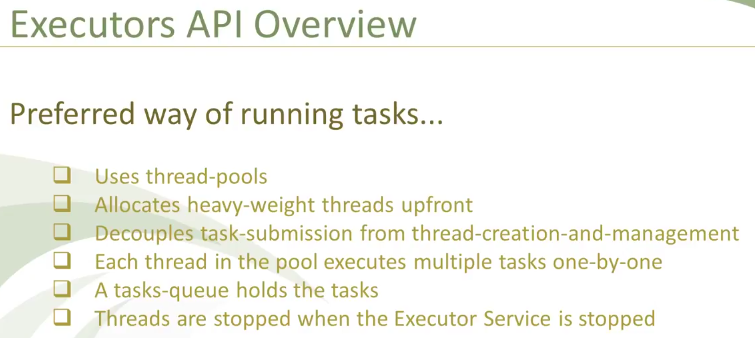
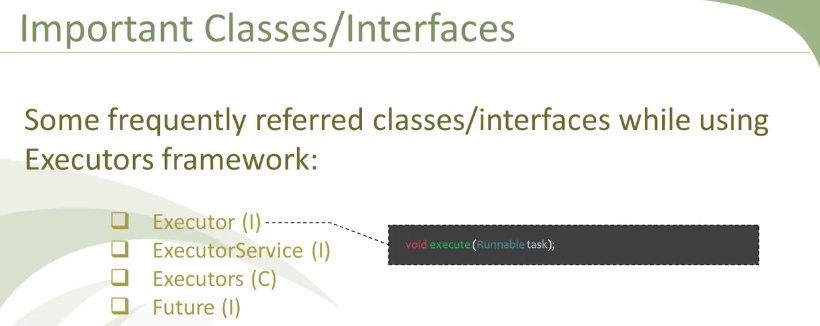


**Executor service:**

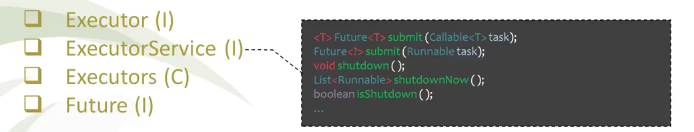
Write code for defining the task to run it in different task. We should provide this task to the executor service. Executor Service automatically creates thread and

run our code in that thread, without having us to create the thread. Thread creation has CPU overhead. It uses thread-pool internally. It is recycled and reused.



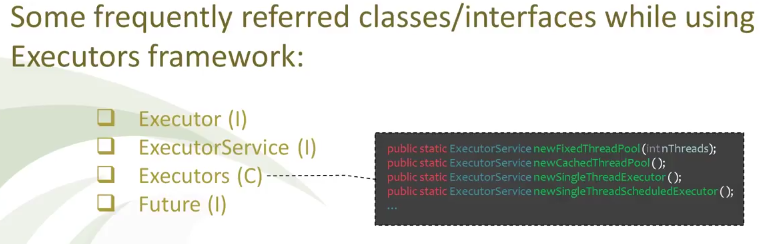


Executor decouples the task submission step from thread creation and running steps.



It provides life cycle. Initialization service and destruction phase. We call a task by callable interface when we want that task to return the value. Runnable does not return anything.

Using shutdown, we shut the task. No new other task can be submitted.

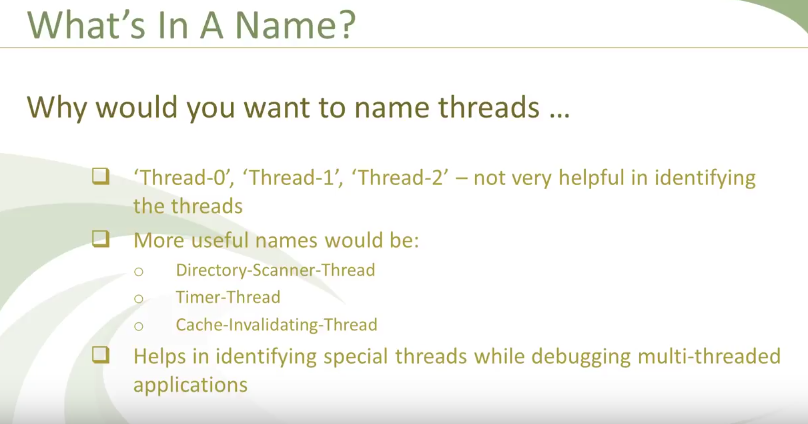


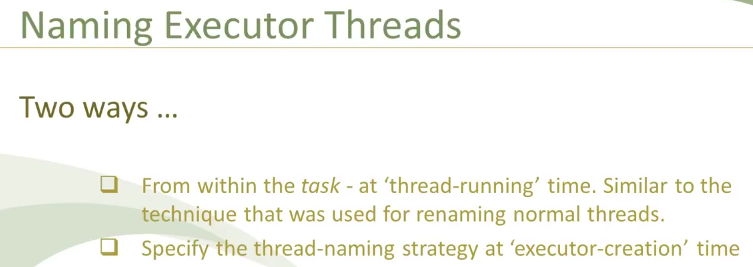
Factory class that constructs and returns various kind of instances of executor service with commonly useful configuration settings.

1. Public static ExecutorService newFixedThreadPool (int thead)-🡪 Only n number of threads can be executed
2. Public static ExecutorService newCachedThreadPool ();
3. Public static ExecutorService newSingleThreadExecutor (); ---One by One from queue. But can never execute more than one task simultaneously
4. Public static ExecutorService newSingleThreadScheduledExecutor ();-🡪Whenever we submit a callable for execution the ExecutorService returns an instance of Future. We can use get method of future to check the status of task.





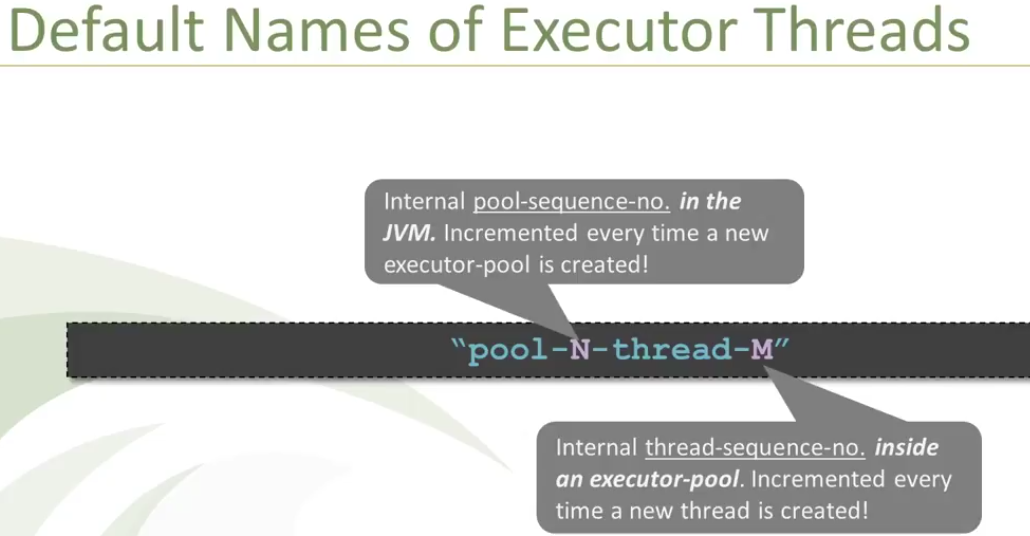




1st Way: 

2nd Way: 

Naming a thread at executor creation time:





In Fixed Thread-Pool executor the number of thread is fixed. If the number of submitted task exceeds the number of threads created in the fixed thread pool. The extra thread will have to wait in the thread queue till the time some thread finishes its execution and becomes available to execute another task from queue. At any point of time only fixed number of threads will be executing.

Submitting a task after shutdown will raise exception

**In cached Thread Pool** as many number of threads are created as the number of threads submitted to execute concurrently. So that no task will have to wait for execution. No concept of wait queue.

If a task is completed it will be reused for executing a new task.

Difference between fixed and cached thread pool executor is in fixed thread pool the number of thread is fixed. If the number of submitted task exceeds the number of fixed thread then extra task will have to wait in the queue till the thread finishes executing the previous task.

However, in cached Thread Pool there is no concept of waiting queue. Threads in cached thread pool will be reused when a new task is submitted.

**Single Thread Executor:**