Thread Pool :

As we know how **context switching** causes slow performance . To overcome the performance drop , we use thread pools.

A **thread pool** is a group of pre-created worker threads that are ready to execute tasks..Although Thread pools **do not eliminate** context switching — but they **reduce it a LOT** by controlling the number of active threads.
Instead of creating a new thread for each task, you **reuse** existing threads.

Why do we need Thread Pools?

Creating a new thread is **expensive** (memory, CPU, context switching).

Imagine:
1000 tasks → 1000 new threads → system crash / lag
1000 tasks → 10 threads reused → stable + fast
Thread pools solve this problem.

If the created threads in the thread pools are busy with their tasks, the upcoming tasks will waits in the queue.

How Thread Pool Works

A fixed number of threads are created in advance
When you submit a task:

- If a thread is free → it runs the task immediately
- If all threads are busy → task waits in a queue
- When a thread finishes a task → it picks the next one

Creating Thread Pool in Java :
Java provides thread pools via the **ExecutorService**.

Fixed thread pool : As the name suggested , we have to pass the number of threads to created in the thread pool. In the thread pool at max 5 thread will be created no more and no less.

```
ExecutorService executor = Executors.newFixedThreadPool(5);

executor.submit(() -> {
   System.out.println("Task executed by: " +
Thread.currentThread().getName());
});

executor.shutdown();
```

Cached thread pool : (auto grows/shrinks) takes no parameter value, creates new thread only when needed , Reuses idle threads,Good for many short-lived tasks

ExecutorService executor = Executors.newCachedThreadPool**();**

Benefits of using thread pools :

>Better performance
  Threads are reused → less overhead

>Avoids too many threads
  Prevents memory and CPU overload

>Task queueing
  Tasks wait instead of crashing the system

> Easy management
  Shut down all threads cleanly

> Used heavily in Spring Boot
  Schedulers, async methods, web server request handling