Thread Pool

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class Task implements Runnable {

private int taskId;

public Task(int taskId) {

this.taskId = taskId;

}

@Override

public void run() {

System.out.println("Task " + taskId + " is being processed by " +

Thread.currentThread().getName());

}

}

// Another class

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class ThreadPoolExample {

public static void main(String[] args) {

// Create a thread pool with 3 threads

ExecutorService executorService = Executors.newFixedThreadPool(3);

// Submit tasks to the pool

for (int i = 1; i <= 5; i++) {

executorService.submit(new Task(i));

}

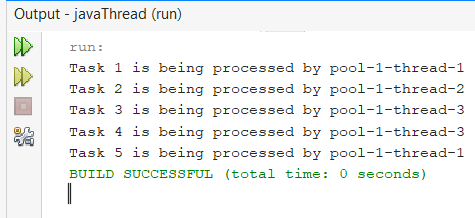
// Shutdown the thread pool

executorService.shutdown();

}

}

**Output:**



**Explanation:**

• ExecutorService manages a pool of threads. In this example, we use Executors.newFixedThreadPool(3) to create a pool with 3 threads.

• We submit 5 tasks, and the thread pool executes them concurrently using the available threads.

• The shutdown() method ensures that the pool stops accepting new tasks and gracefully terminates the threads.