# ThreadPoolExecutor – Java Thread Pool Example

[**Java thread**](https://www.journaldev.com/1079/multithreading-in-java)**pool** manages the pool of worker threads, it contains a queue that keeps tasks waiting to get executed. We can use ThreadPoolExecutor to create thread pool in Java.

Java thread pool manages the collection of Runnable threads. The worker threads execute Runnable threads from the queue. **java.util.concurrent.Executors** provide factory and support methods for **java.util.concurrent.Executor** interface to create the thread pool in java.

Executors is a utility class that also provides useful methods to work with ExecutorService, ScheduledExecutorService, ThreadFactory, and Callable classes through various factory methods.

Let’s write a simple program to explain it’s working.

First, we need to have a Runnable class, named WorkerThread.java

WorkerThread.java

**ExecutorService Example**

Here is the test program class SimpleThreadPool.java, where we are creating fixed thread pool from **Executors framework**.

SimpleThreadPool.java

In the above program, we are creating a fixed size thread pool of 5 worker threads. Then we are submitting 10 jobs to this pool, since the pool size is 5, it will start working on 5 jobs and other jobs will be in wait state, as soon as one of the job is finished, another job from the wait queue will be picked up by worker thread and get’s executed.

The output confirms that there are five threads in the pool named from “pool-1-thread-1” to “pool-1-thread-5” and they are responsible to execute the submitted tasks to the pool.

**ThreadPoolExecutor Example**

**Executors** class provide simple implementation of **ExecutorService** using **ThreadPoolExecutor** but ThreadPoolExecutor provides much more feature than that. We can specify the number of threads that will be alive when we create ThreadPoolExecutor instance and we can limit the size of thread pool and create our own **RejectedExecutionHandler** implementation to handle the jobs that can’t fit in the worker queue.

Here is our custom implementation of RejectedExecutionHandler interface.

RejectedExecutionHandlerImpl.java

ThreadPoolExecutor provides several methods using which we can find out the current state of the executor, pool size, active thread count and task count. So I have a monitor thread that will print the executor information at a certain time interval.

MyMonitorThread.java

Here is the thread pool implementation example using **ThreadPoolExecutor**.

WorkerPool.java

Notice that while initializing the ThreadPoolExecutor, we are keeping initial pool size as 2, maximum pool size to 4 and work queue size as 2. So if there are 4 running tasks and more tasks are submitted, the work queue will hold only 2 of them and the rest of them will be handled by RejectedExecutionHandlerImpl.

output

Notice the change in active, completed and total completed task count of the executor. We can invoke shutdown() method to finish execution of all the submitted tasks and terminate the thread pool.