Notes - Executor Service

ExecutorService -

In the previous example we executed the copyTask using separate threads, but there is a critical point to consider here, Thread creation is a costly activity as it includes creating a separate execution context, stack etc.. Hence we should refrain from creating too many threads. And also creating a thread for each task is not a good idea, instead we can create a pool of threads and effectively utilise them in executing all the task. This could be achieved using ExecutorService in Java. Use the execute method of the ExecutorService to submit a Runnable task, if a thread is available in the pool then it assigns this task to the thread otherwise the task is added to the blocking queue and is kept till a thread is available.

Creating a ThreadPool -

Below statement creates a thread pool of size 5.

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

Submitting a Runnable task -

We can submit a task for execution using the execute method.

`executor.execute( runnableTaskInstance );`

e.g.

`executor.execute(new CopyTask(sourceFile1, destFile1));`

Modified version of previous example to use ExecutorService -

```
public class Main {

   public static void main(String[] args) throws IOException {
        String sourceFile1 = "a.txt";
        String sourceFile2 = "b.txt";

        String destFile1 = "c.txt";
        String destFile2 = "d.txt";

        // Creates a fixed thread pool of size 5.

        ExecutorService executor = Executors.newFixedThreadPool(5);

        // Assume you are submitting 100 copy tasks,
        // then executor service uses a fixed thread
        // pool of size 5 to execute them.

        executor.execute(new CopyTask(sourceFile1, destFile1));
        executor.execute(new CopyTask(sourceFile2, destFile2));
    }
}
```