

Basics - Implement Runnable

```
Runnable runnable = () -> { / code
here / }
Thread thread = new
Thread(runnable);
thread.start();
```

Basics - Executor

`shutdown()` Waits for currently running tasks to finish

`shutdownNow()` Interrupts all running tasks and shut the executor down immediately

`awaitTermination(5, TimeUnit.SECONDS)` Waits a certain amount of time for termination of currently running tasks. After a max of 45 seconds, the executor finally shuts down by interrupting all tasks

Callables Callable are like runnables but return a value.

Futures

```
ExecutorService executor =
Executors.newSingleThreadExecutor();
executor.submit(() -> { / Code /
});
```

Avoiding Deadlocks

Ensure all threads use same order If all threads try to get the locks in the same order, a deadlock will not occur

Timeout / Retry Put a timeout with a retry mechanism when obtaining locks

Wait Can only be called from within synchronized block (must have the lock to call). Once wait is called, all locks are released

Notify

NotifyAll

Deadlocks occur when multiple threads need the same locks but obtain them in a different order



By **gnowakow**

cheatography.com/gnowakow/

Not published yet.

Last updated 21st July, 2017.

Page 1 of 1.

Sponsored by **Readability-Score.com**

Measure your website readability!

<https://readability-score.com>