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 and Futures	Callable are like runnables but return a value.

```
ExecutorService executor = Executors.newSingleTreadExecutor();
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.