# Interrupting Threads

## Interrupted Exception

You'll often come across this exception being thrown from functions. When a thread wait()-s or sleep()-s then one way for it to give up waiting/sleeping is to be interrupted. If a thread is interrupted while waiting/sleeping, it'll wake up and immediately throw Interrupted exception.

The thread class exposes the `interrupt()` method which can be used to interrupt a thread that is blocked in a `sleep()` or `wait()` call. Note that invoking the interrupt method only sets a flag that is polled periodically by sleep or wait to know the current thread has been interrupted and an interrupted exception should be thrown.

Below is an example, where a thread is initially made to sleep for an hour but then interrupted by the main thread.

```
class Demonstration {

    public static void main(String args[]) throw
        InterruptExample.example();
    }
}

class InterruptExample {

    static public void example() throws Interrup
```

```
11
12          final Thread sleepyThread = new Thread(r
13
14              public void run() {
15                  try {
16                      System.out.println("I am toc
17                      Thread.sleep(1000 * 60 * 60)
18                  } catch (InterruptedException ie
19                      System.out.println("The inte
20                      Thread.currentThread().inter
21                      System.out.println("Oh somec
22                      System.out.println("The inte
23
24                  }
25              }
26          });
27
28          sleepyThread.start();
```

Take a minute to go through the output of the above program. Observe the following:

- Once the interrupted exception is thrown, the interrupt status/flag is cleared as the output of line-19 shows.
- On line-20 we again interrupt the thread and no exception is thrown. This is to emphasize that merely calling the interrupt method isn't responsible for throwing the interrupted exception. Rather the implementation should periodically check for the interrupt status and take appropriate action.
- On line 22 we print the interrupt status for the thread, which is set to true because of line 20.
- Note that there are two methods to check for the interrupt status of a thread. One is the static method `Thread.interrupted()` and the other is `Thread.currentThread().isInterrupted()`. The important

other is `Thread.currentThread().isInterrupted()`. The important difference between the two is that the static method would return the interrupt status and also clear it at the same time. On line 22 we deliberately call the object method first followed by the static method. If we reverse the ordering of the two method calls on line 22, the output for the line would be *true* and *false*, instead of *true* and *true*.



Back
Wait & Notify

Next
Volatile

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/interrupting-threads__multithreading-in-java__java-multithreading-for-senior-engineering-interviews)