Interrupting Threads

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.

```
1   class Demonstration {
2
3       public static void main(Str
4           InterruptExample.exampl
5       }
6   }
7
8   class InterruptExample {
9
10      static public void example
11
12          final Thread sleepyThre
13
14              public void run()
15                  try {
16                      System.out
17                      Thread.sle
18                  } catch (Inter
19                      System.out
20                      Thread.cur
```

```
 21                      System.out
 22                      System.out
 23
 24                  }
 25              }
 26          });
 27
 28      sleepyThread.start();
 29
 30      System.out.println("Ab
 31      sleepyThread.interrupt
```

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 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*.