# Missed Signals

## Missed Signals

A missed signal happens when a signal is sent by a thread before the other thread starts waiting on a condition. This is exemplified by the following code snippet. Missed signals are caused by using the wrong concurrency constructs. In the example below, a condition variable is used to coordinate between the **signaller** and the **waiter** thread. The condition is signaled at a time when no thread is waiting on it causing a missed signal.

In later sections, you'll learn that the way we are using the condition variable's `await` method is incorrect. The idiomatic way of using `await` is in a while loop with an associated boolean condition. For now, observe the possibility of losing signals between threads.

```
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.ReentrantLock;

class Demonstration {

    public static void main(String args[]) throws InterruptedException {
        MissedSignalExample.example();
    }
}

class MissedSignalExample {

    public static void example() throws InterruptedException {

        final ReentrantLock lock = new ReentrantLock();
        final Condition condition = lock.newCondition();

        Thread signaller = new Thread(new Runnable() {

```

```
20              public void run() {
21                  lock.lock();
22                  condition.signal();
23                  System.out.println("Sent signal");
24                  lock.unlock();
25              }
26          });
27
28          Thread waiter = new Thread(new Runnable() {
```

Missed Signal Example

The above code when ran, will never print the statement **`Program Exiting`** and execution would time out. Apart from refactoring the code to match the idiomatic usage of condition variables in a while loop, the other possible fix is to use a **semaphore** for signalling between the two threads as shown below

```
1  import java.util.concurrent.Semaphore;
2
3  class Demonstration {
4
5      public static void main(String args[]) throws InterruptedException {
6          FixedMissedSignalExample.example();
7      }
8  }
9
10 class FixedMissedSignalExample {
11
12     public static void example() throws InterruptedException {
13
14         final Semaphore semaphore = new Semaphore(1);
15
16         Thread signaller = new Thread(new Runnable() {
17
18             public void run() {
19                 semaphore.release();
20                 System.out.println("Sent signal");
21             }
22         });
23
24         Thread waiter = new Thread(new Runnable() {
```

```
24          Thread waiter = new Thread(new Runnable() {
25
26              public void run() {
27                  try {
28                      semaphore.acquire();
```

Fixed Missed Signal

Back

Reentrant Locks & Condition Variables

Next

Semaphore in Java

Completed

53% completed, meet the criteria and claim your course certificate!

Claim Certificate

Report an Issue

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