





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.

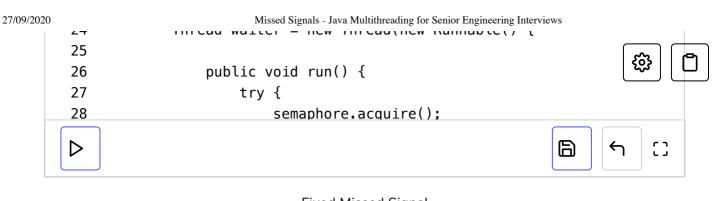
```
import java.util.concurrent.locks.Condition;
1
2
    import java.util.concurrent.locks.ReentrantLock;
3
   class Demonstration {
 5
        public static void main(String args[]) throws InterruptedException {
6
            MissedSignalExample.example();
        }
8
    }
9
10
11
    class MissedSignalExample {
12
13
        public static void example() throws InterruptedException {
14
15
            final ReentrantLock lock = new ReentrantLock();
16
            final Condition condition = lock.newCondition();
17
18
            Thread signaller = new Thread(new Runnable() {
```

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

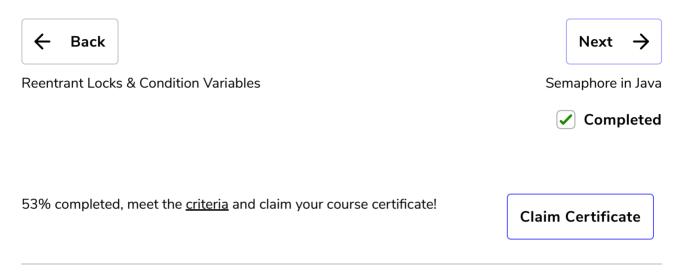
Missed Signal Example

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

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



Fixed Missed Signal



! Report an Issue

? Ask a Question

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