76% completed

Search Course

Multithreading in Java
- Atomic Assignments
- Thready Safety & Synchronized
- Wait & Notify
- Interrupting Threads
- Volatile
- Reentrant Locks & Condition Variables
- Missed Signals
- Semaphore in Java
- Spurious Wakeups
- Atomic Classes
- More on Atomics
- Non-blocking Synchronization
- Miscellaneous Topics

Java's Memory Model

Interview Practice Problems

Practice Mock Interview →

- More on Atomics
- Non-blocking Synchronization
- Miscellaneous Topics

Java's Memory Model

Interview Practice Problems

Practice Mock Interview →

Semaphore in Java

Explanation

Java's semaphore can be `release()-ed` or `acquire()-d` for signalling amongst threads. However the important call out when using semaphores is to make sure **that the permits acquired should equal permits returned**. Take a look at the following example, where a runtime exception causes a deadlock.

Java — Press ⇧ + ↵ to interact — main.java — Saved

```java
import java.util.concurrent.Semaphore;

class Demonstration {

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

class IncorrectSemaphoreExample {

    public static void example() throws InterruptedException {
```

Run

Incorrect Use of Semaphore

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

class IncorrectSemaphoreExample {

    public static void example() throws InterruptedException {
```

Run

Incorrect Use of Semaphore

The above code when run would time out and show that one of the threads threw an exception. The code is never able to release the semaphore causing the other thread to block forever. Whenever using locks or semaphores, remember to unlock or release the semaphore in a **finally** block. The corrected version appears below.

Java — Press ⇧ + ↵ to interact — main.java — Saved

```java
import java.util.concurrent.Semaphore;

class Demonstration {

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

class CorrectSemaphoreExample {

    public static void example() throws InterruptedException {
```

Run

Running the above code will print the `Exiting Program` statement.

← Back lesson — Missed Signals

✓ Completed   Next → — Spurious Wakeups