Java Monitor Objects: Coordination Example Visualization

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

- Learn how to fix a buggy concurrent Java program using Java’s wait & notify mechanisms, which provide coordination.

- Visualize how Java monitor objects can be used to ensure mutual exclusion & coordination between threads running in a concurrent program.

1. Enter monitor object
2. Acquire lock
3. wait()
4. notifyAll()
5. Release lock
6. Leave monitor object
Visual Analysis of the SimpleBlockingBounded Queue Example
Visual Analysis of SimpleBoundedBlockingQueue

1. Enter monitor object
2. Acquire lock
3. wait()
4. notifyAll()
5. Release lock
6. Leave monitor object

Critical Section

Visual Analysis of SimpleBoundedBlockingQueue

1. Enter monitor object
2. Acquire lock
3. wait()
4. notifyAll()
5. Release lock
6. Leave monitor object

Queue of threads blocked on the monitor lock’s “entrance queue”
Queue of threads waiting on the monitor condition’s “wait queue”

See en.wikipedia.org/wiki/Monitor_(synchronization)#Implicit_condition_variable_monitors
Visual Analysis of SimpleBoundedBlockingQueue

We’ll assume the queue is initially empty
Visual Analysis of SimpleBoundedBlockingQueue

```java
new Thread(() -> {
    while (true)
        System.out.println(mQueue.take());
}).start();
```
new Thread(() -> {
    while (true)
        System.out.println
            (mQueue.take());
}).start();
Visual Analysis of SimpleBoundedBlockingQueue

Calling wait() atomically releases the monitor lock & puts the calling thread to sleep
Visual Analysis of SimpleBoundedBlockingQueue

```java
SimpleBoundedBlockingQueue

new Thread(() -> {
    while (true)
        System.out.println
            (mQueue.take());
}).start();
```
Visual Analysis of SimpleBoundedBlockingQueue

```
new Thread(() -> {
    for(int i = 0; i < 10; i++)
        mQueue.put(Integer.toString(i));
}).start();
```

We’ll assume the queue is not full
Critical Section

Acquire lock

new Thread(() -> {
    for (int i = 0; i < 10; i++)
        mQueue.put(Integer.toString(i));
}).start();

Block on monitor condition

SimpleBoundedBlockingQueue
Visual Analysis of SimpleBoundedBlockingQueue

Critical Section

while (isFull())
    wait();

The queue is not full (since it is initially empty), so continue past the guard

new Thread(() -> {
    for (int i = 0; i < 10; i++)
        mQueue.put(Integer.toString(i));
}).start();

Block on monitor condition
new Thread(() -> {
    for (int i = 0; i < 10; i++)
        mQueue.put(Integer.toString(i));
}).start();

mList.add(msg);
notifyAll();
Visual Analysis of SimpleBoundedBlockingQueue

new Thread(()->{
    for(int i = 0; i < 10; i++)
        mQueue.put(Integer.toString(i));
}).start();

mList.add(msg);
notifyAll();

Block on
monitor condition

T1

T2

Critical
Section
Visual Analysis of SimpleBoundedBlockingQueue

Critical Section

Thread $T_1$ wakes up, but can’t get lock

SimpleBoundedBlockingQueue

Unblock on monitor condition

new Thread(() -> {
    for(int i = 0;
        i < 10; i++)
        mQueue.put(Integer.toString(i));
}).start();

$T_2$

mList.add(msg);
notifyAll();

$T_1$
Visual Analysis of SimpleBoundedBlockingQueue

```
new Thread(()
    -> {
        for(int i = 0;
            i < 10; i++)
            mQueue.put(Integer.toString(i));
    }).start();
```
Visual Analysis of SimpleBoundedBlockingQueue

```java
new Thread(() -> {
    for(int i = 0; i < 10; i++)
        mQueue.put(Integer.toString(i));
}).start();
```
Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue

Unblock on monitor condition

new Thread(() -> {
    while (true)
        System.out.println (mQueue.take());
}).start();
Visual Analysis of SimpleBoundedBlockingQueue

```
new Thread(() -> {
    while (true)
        System.out.println(mQueue.take());
}).start();
```
Visual Analysis of SimpleBoundedBlockingQueue

```
new Thread(() -> {
    while(true)
    System.out.println(mQueue.take());
}).start();
```

while(isEmpty())
wait();

The queue is no longer empty, so continue past the guard
Visual Analysis of SimpleBoundedBlockingQueue

Calling `notifyAll()` before removing/returning the front item in the queue is ok since the monitor lock is held & only one method can be in the monitor object.
Visual Analysis of SimpleBoundedBlockingQueue

Critical Section

```
notifyAll();
return mList.poll();
```

```
new Thread(() -> {
    while(true)
        System.out.println(mQueue.take());
}).start();
```
Visual Analysis of SimpleBoundedBlockingQueue

```java
SimpleBoundedBlockingQueue

new Thread(() -> {
    while (true)
        System.out.println(mQueue.take());
}).start();
```

Critical Section

Release lock

T1
Visual Analysis of SimpleBoundedBlockingQueue

new Thread(() -> {
    while (true)
        System.out.println(mQueue.take());
}).start();
End of Java Monitor Object: Coordination Example Visualization