Visualizing the Java Monitor
Object Coordination Example

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.
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”

Critical Section

4: Running thread

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

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

We’ll assume the queue is initially empty

Critical Section

Enter monitor object
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();
```

`T_1` 

while(isEmpty())
wait();

Critical Section

SimpleBoundedBlockingQueue
Visual Analysis of SimpleBoundedBlockingQueue

Critical Section

while(isEmpty())
    wait();

Calling \textit{wait()} atomically releases the monitor lock & puts the calling thread to sleep

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

Critical Section

Acquire lock

T2

SimpleBoundedBlockingQueue

Block on monitor condition

T1

new Thread() -> {
  for(int i = 0; i < 10; i++)
    mQueue.put(Integer.toString(i));
}.start();
The queue is not full (since it is initially empty), so continue past the guard.
Visual Analysis of SimpleBoundedBlockingQueue

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

Critical Section

```java
mList.add(msg);
notifyAll();
```

SimpleBoundedBlockingQueue

Block on monitor condition

T1

T2

T3
Visual Analysis of SimpleBoundedBlockingQueue

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

```
Critical Section
mList.add(msg);
notifyAll();
```

```
SimpleBoundedBlockingQueue
```

```
Block on monitor condition
```

```
T2
```

```
T1
```

```
T3
```

```
``
Critical Section

SimpleBoundedBlockingQueue

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

Unblock on monitor condition

$mList.add(msg);$
notifyAll();

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

```java
SimpleBoundedBlockingQueue

for (int i = 0; i < 10; i++)
    mQueue.put(Integer.toString(i));
```

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

Unblock on monitor condition

Critical Section

Exit monitor object

T₁

T₂
Visual Analysis of SimpleBoundedBlockingQueue

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

Critical Section

Unblock on monitor condition
Visual Analysis of SimpleBoundedBlockingQueue

```java
SimpleBoundedBlockingQueue

Acquire lock

Critical Section

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

while (isEmpty())
    wait();

The queue is no longer empty, so continue past the guard

Critical Section
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.
Critical Section

\[
\text{new Thread}() \rightarrow \{ \\
\quad \text{while (true)} \\
\quad \quad \text{System.out.println} \\
\quad \quad \quad (\text{mQueue.take}()); \\
\quad \}.\text{start}(); \\
\text{return mList.poll}(); \\
\text{notifyAll}(); \\
\]

Visual Analysis of SimpleBoundedBlockingQueue
Visual Analysis of SimpleBoundedBlockingQueue

```
SimpleBoundedBlockingQueue

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

Leave monitor object

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