

Visualizing the Java Monitor Object Coordination Example



Douglas C. Schmidt

d.schmidt@vanderbilt.edu

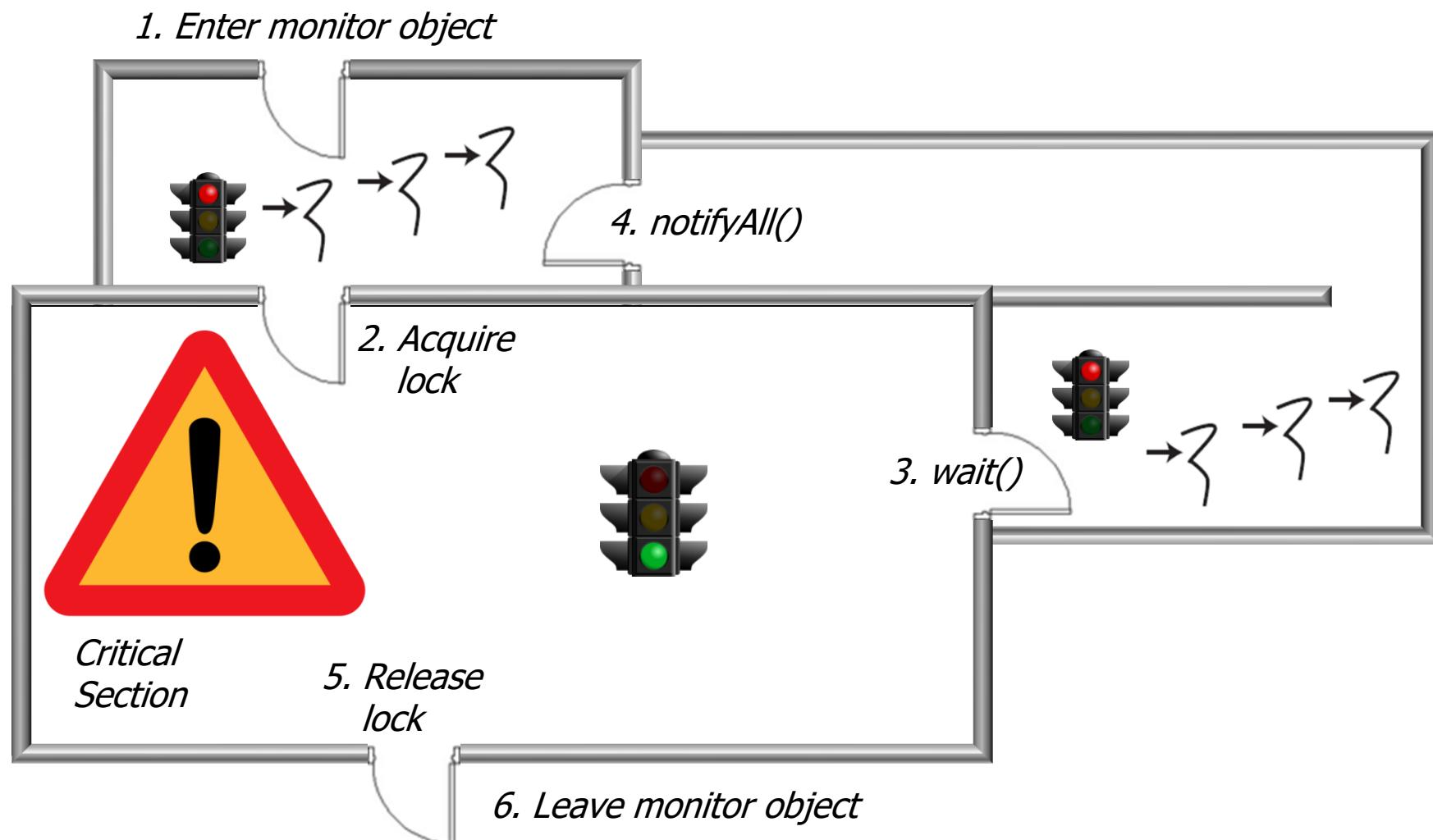
www.dre.vanderbilt.edu/~schmidt

Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA



Learning Objectives in this Part of the Lesson

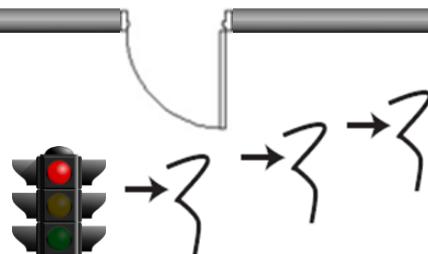
- Learn how to fix a buggy concurrent Java program using Java's wait & notify mechanisms, which provide *coordination*
- Visualize how Java monitor objects can be used to ensure mutual exclusion & coordination between threads running in a concurrent program



Visual Analysis of the SimpleBlockingBounded Queue Example

Visual Analysis of SimpleBoundedBlockingQueue

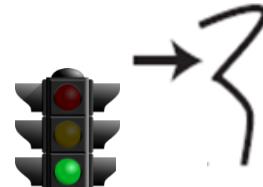
1. Enter monitor object



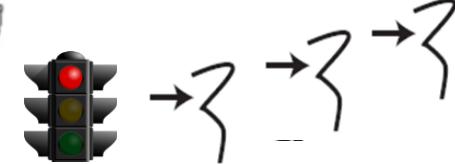
SimpleBoundedBlockingQueue

4. notifyAll()

2. Acquire lock



3. wait()



4: Running thread

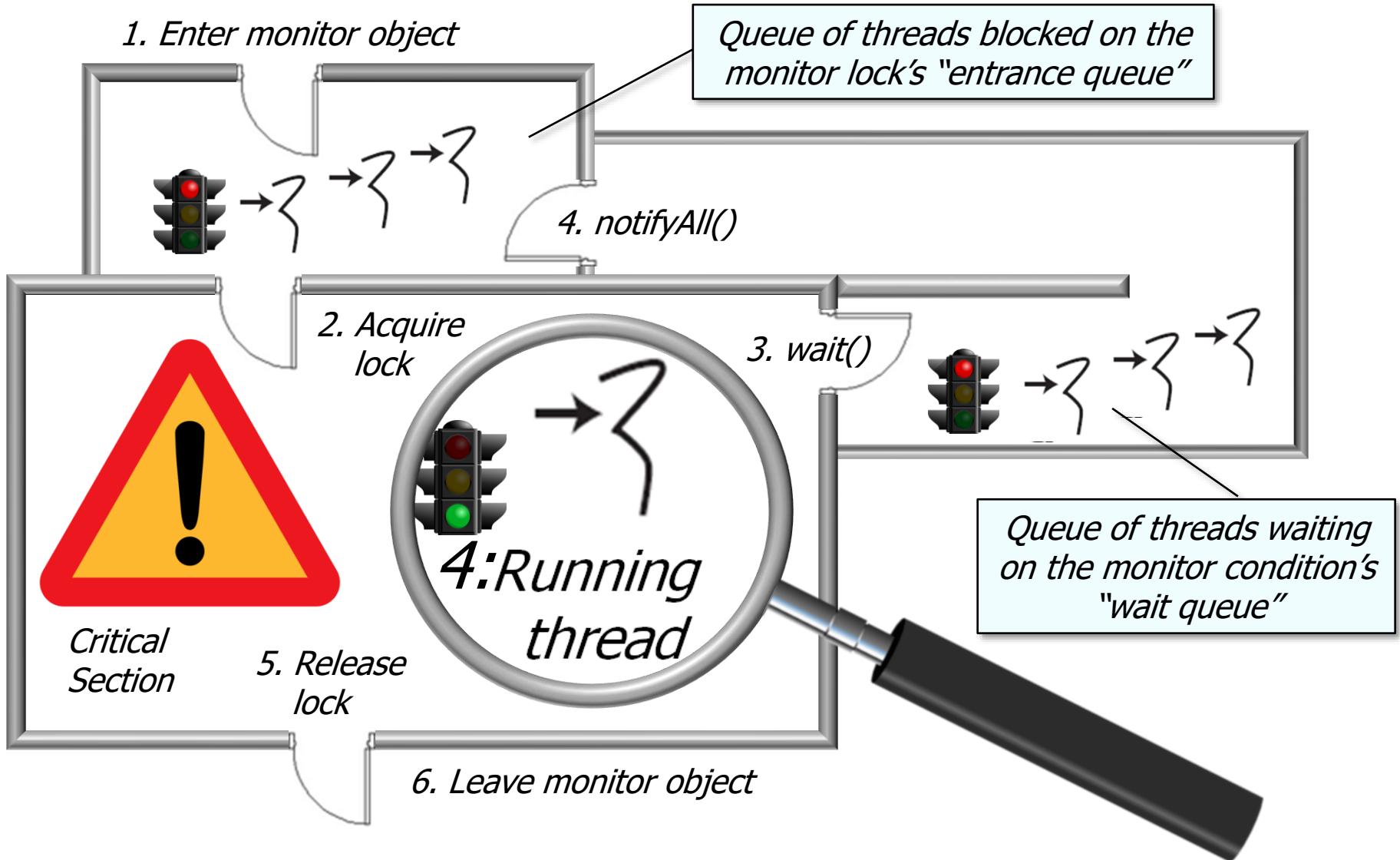
Critical Section

5. Release lock

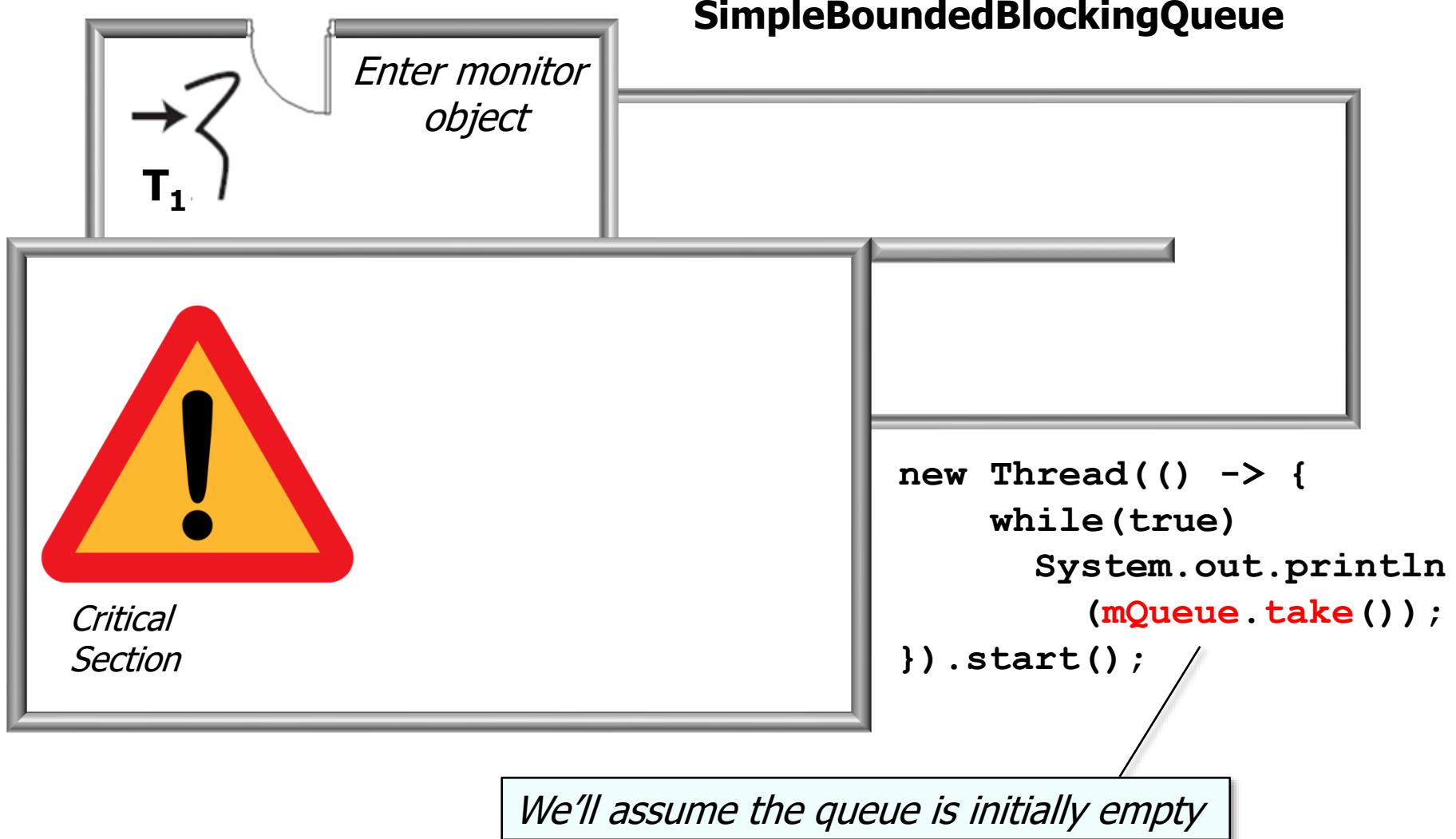
6. Leave monitor object

See github.com/douglascraigschmidt/POSA/tree/master/ex/M3/Queues/SimpleBoundedBlockingQueue

Visual Analysis of SimpleBoundedBlockingQueue

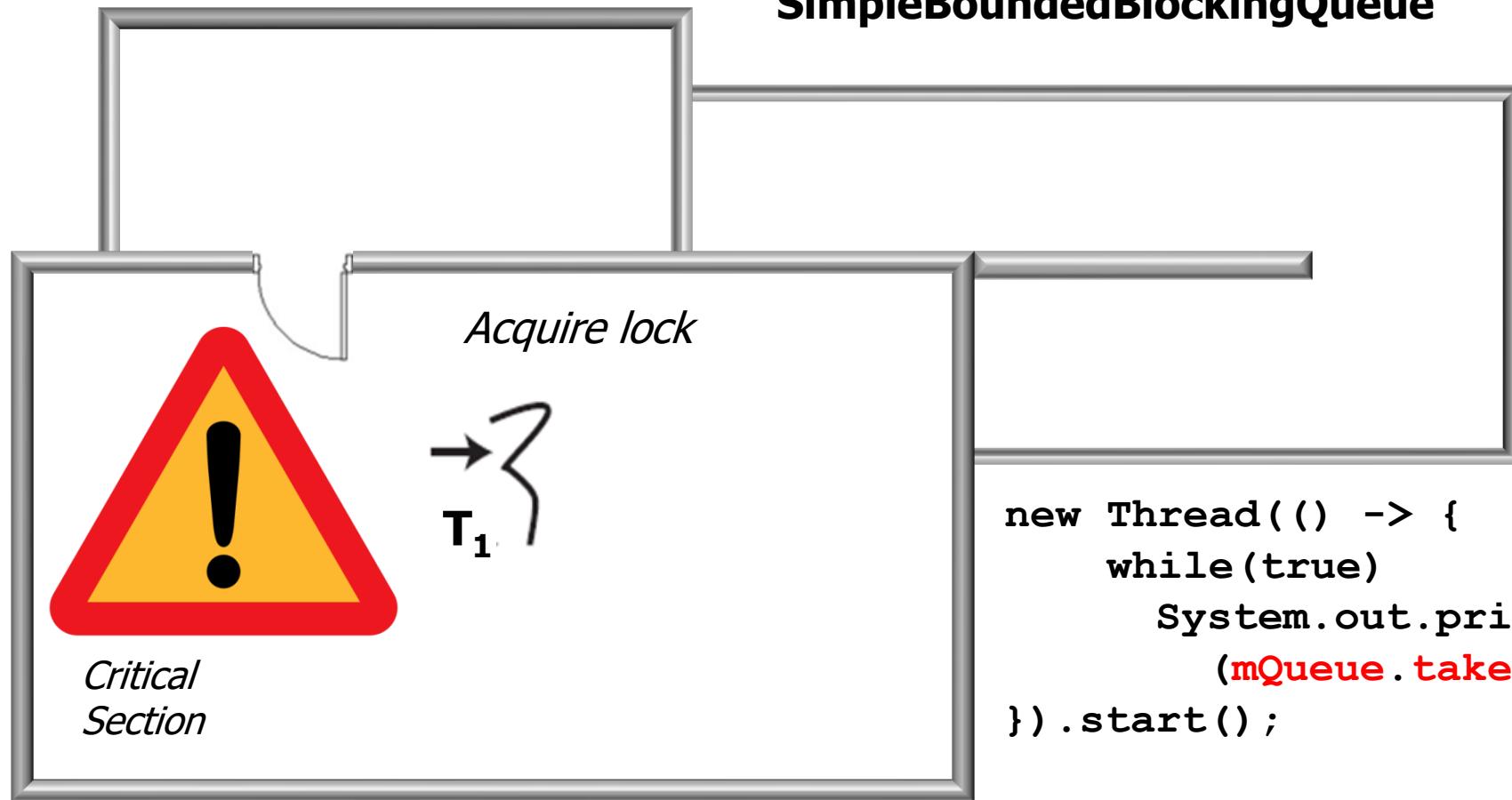


Visual Analysis of SimpleBoundedBlockingQueue



Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



→
T₁



*Critical
Section*

```
while(isEmpty())
    wait();
```

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

Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



→
T₁



Critical
Section

while (isEmpty())
wait();

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

Calling wait() atomically releases the monitor lock & puts the calling thread to sleep

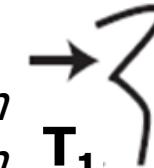
Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



*Critical
Section*

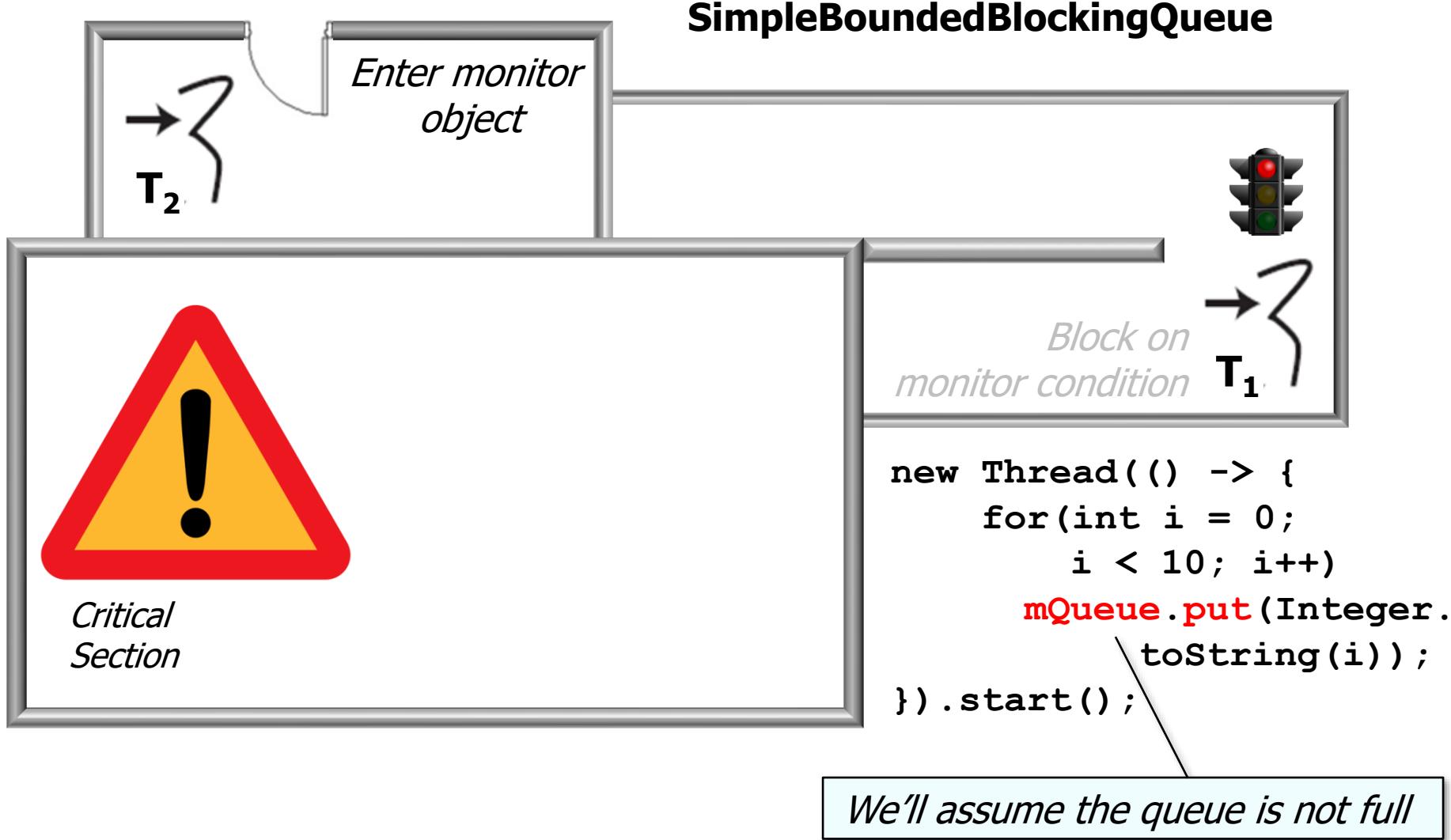
Release lock



*Block on
monitor condition*

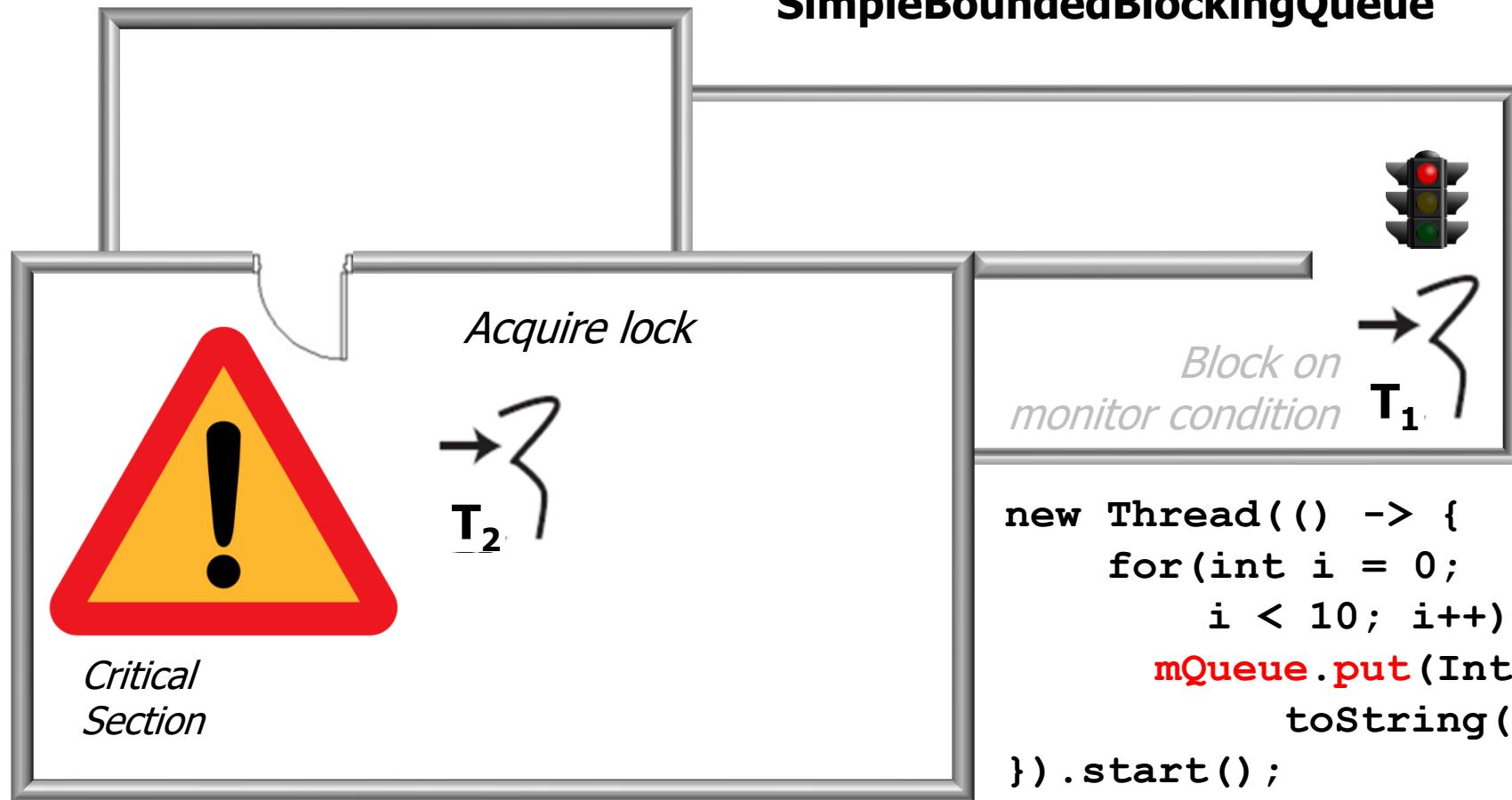
```
new Thread(() -> {  
    while(true)  
        System.out.println  
            (mQueue.take());  
    }).start();
```

Visual Analysis of SimpleBoundedBlockingQueue



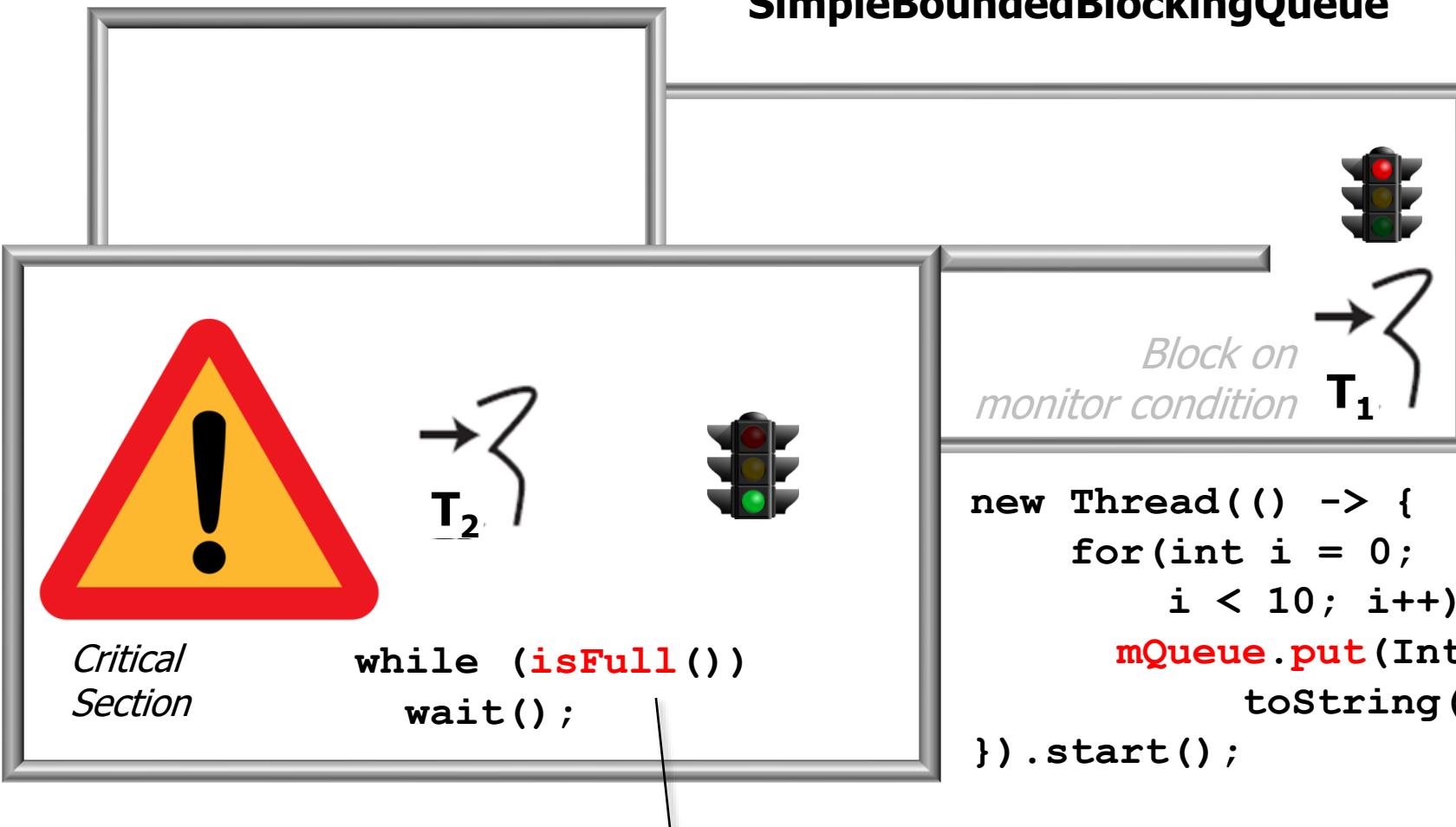
Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



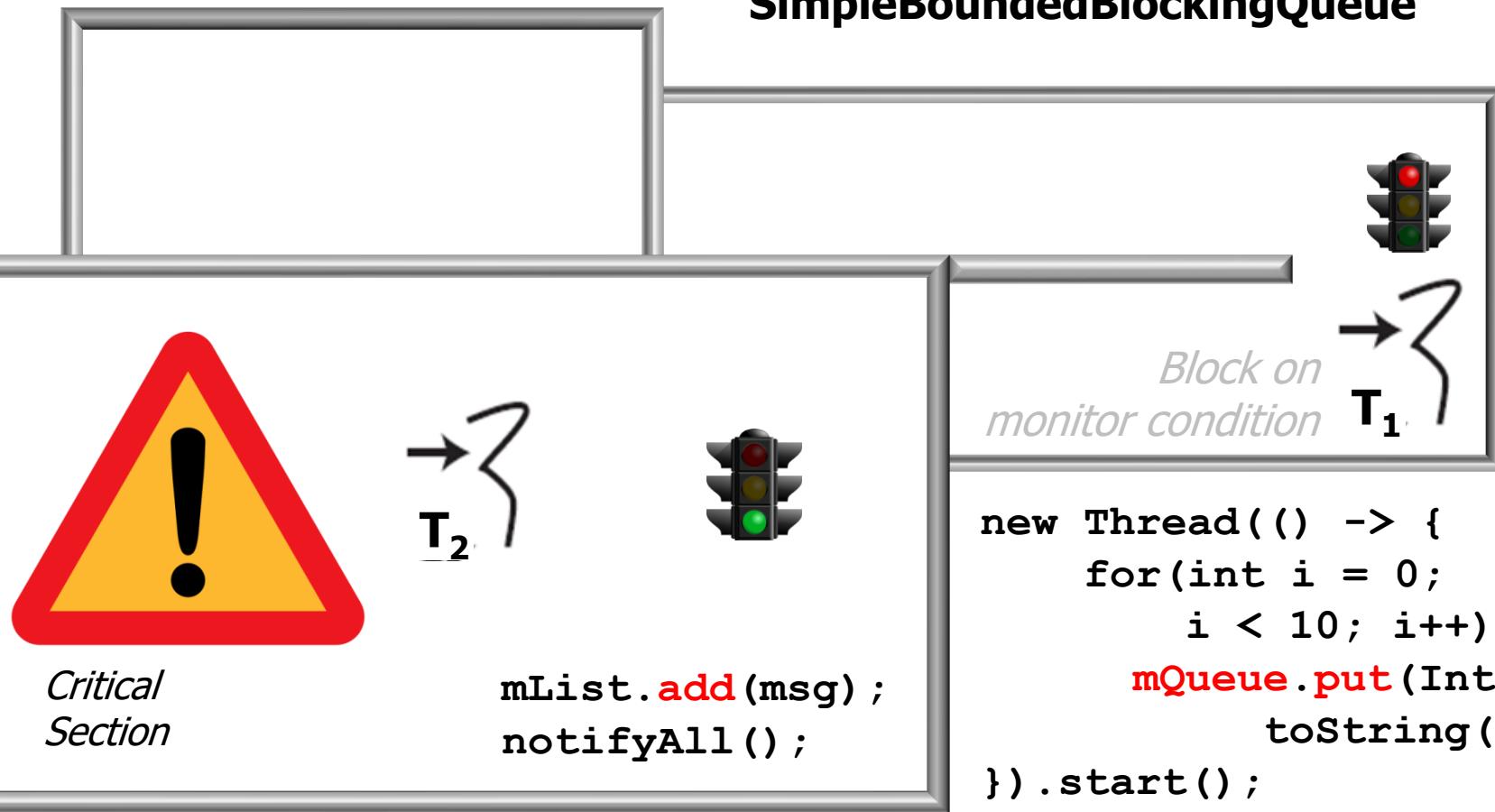
Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



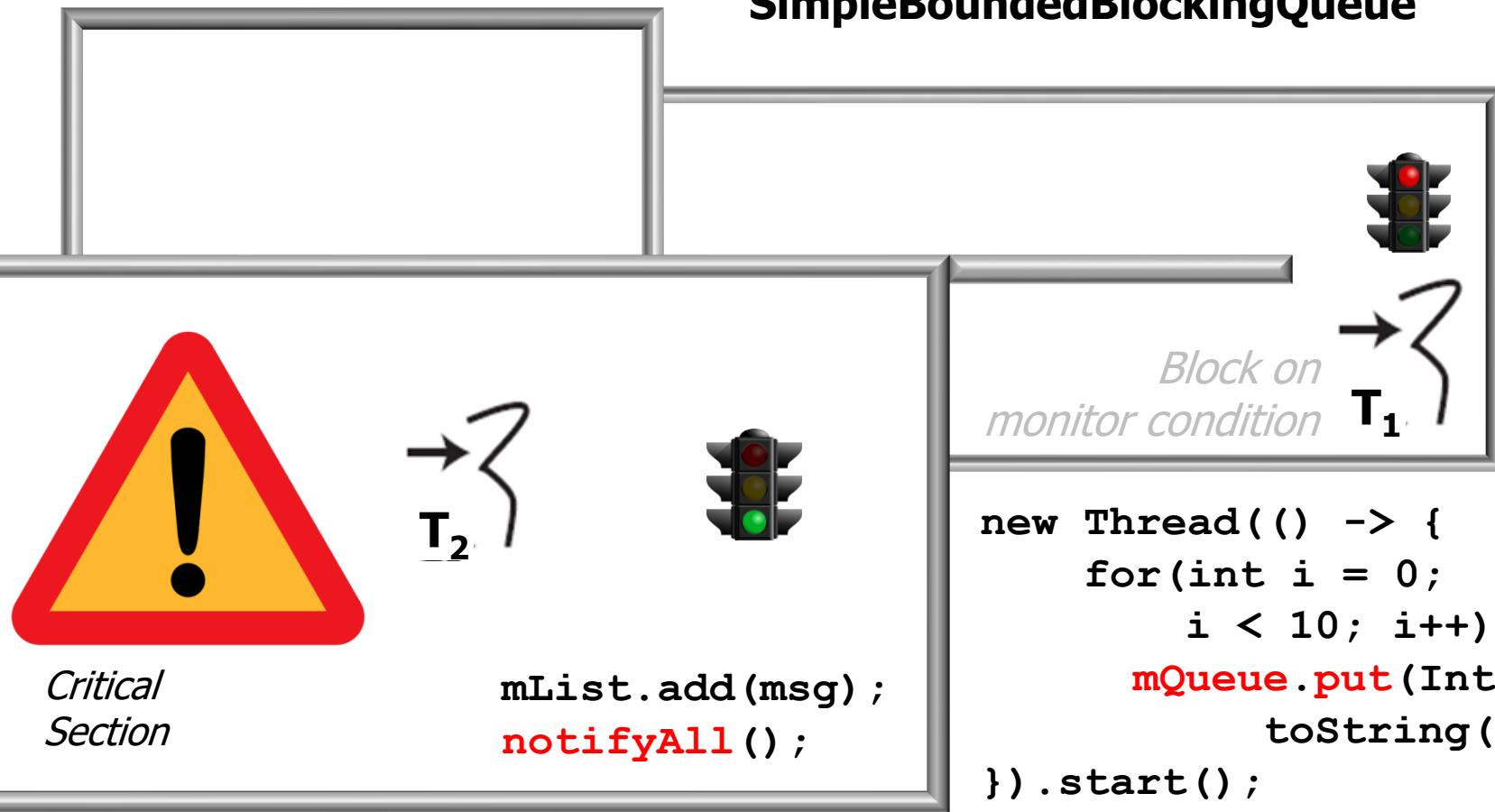
Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



Visual Analysis of SimpleBoundedBlockingQueue

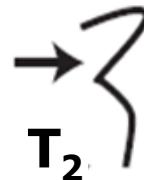
SimpleBoundedBlockingQueue

Thread T_1 wakes up,
but can't get lock

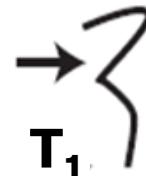
Unblock on
monitor condition



Critical
Section

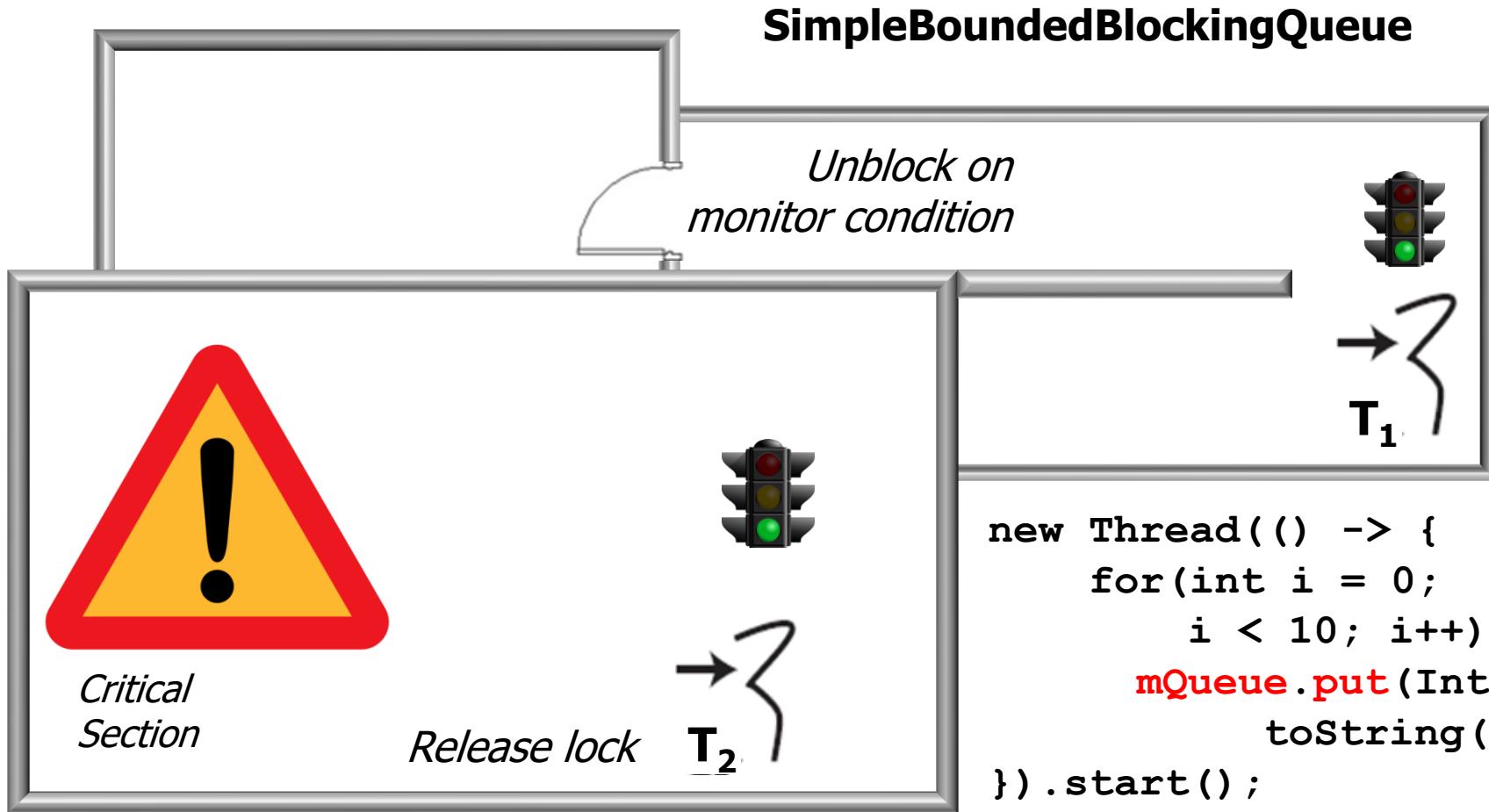


mList.add(msg);
notifyAll();

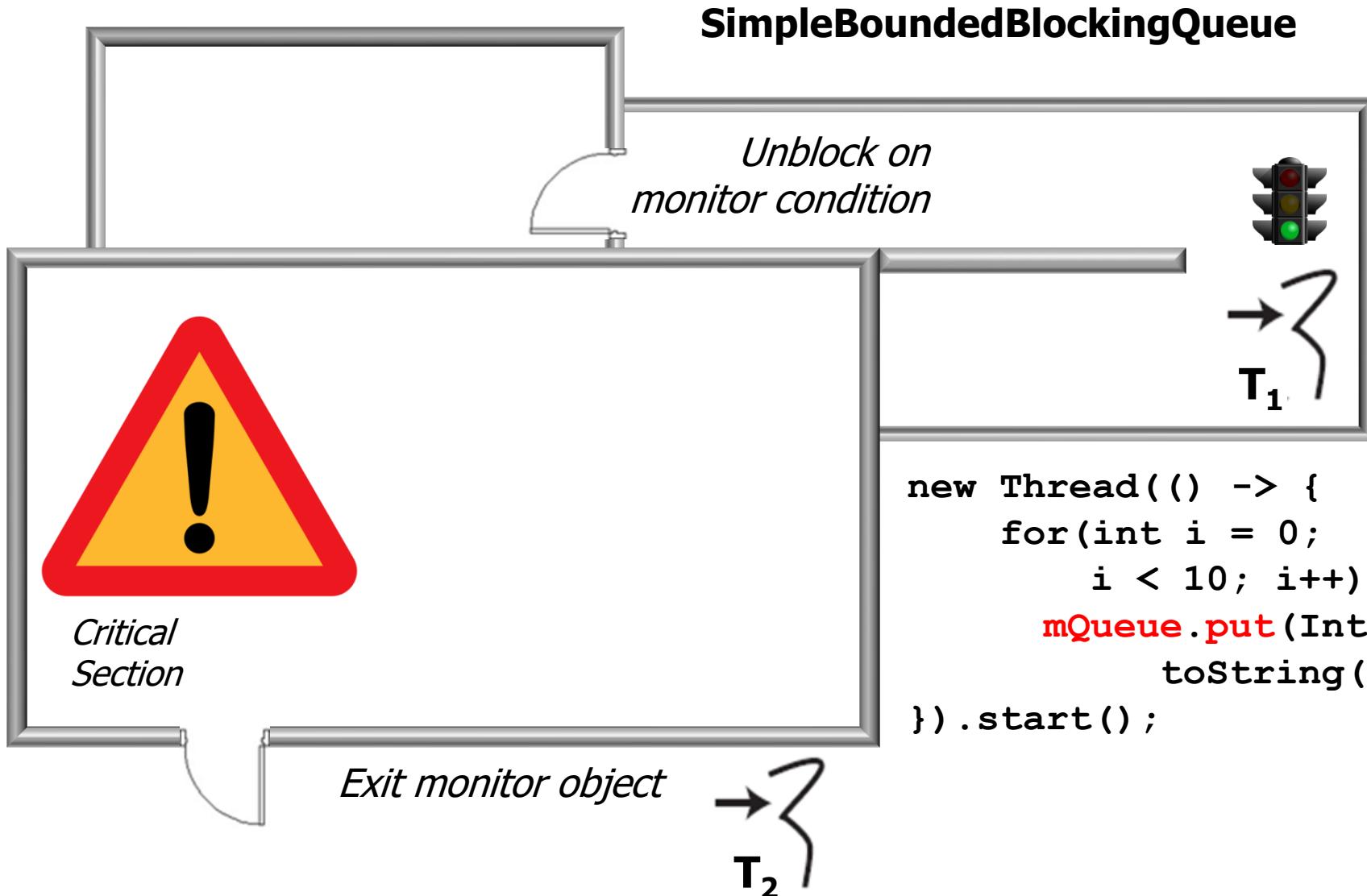


```
new Thread(() -> {  
    for(int i = 0;  
        i < 10; i++)  
        mQueue.put(Integer.  
                    toString(i));  
}).start();
```

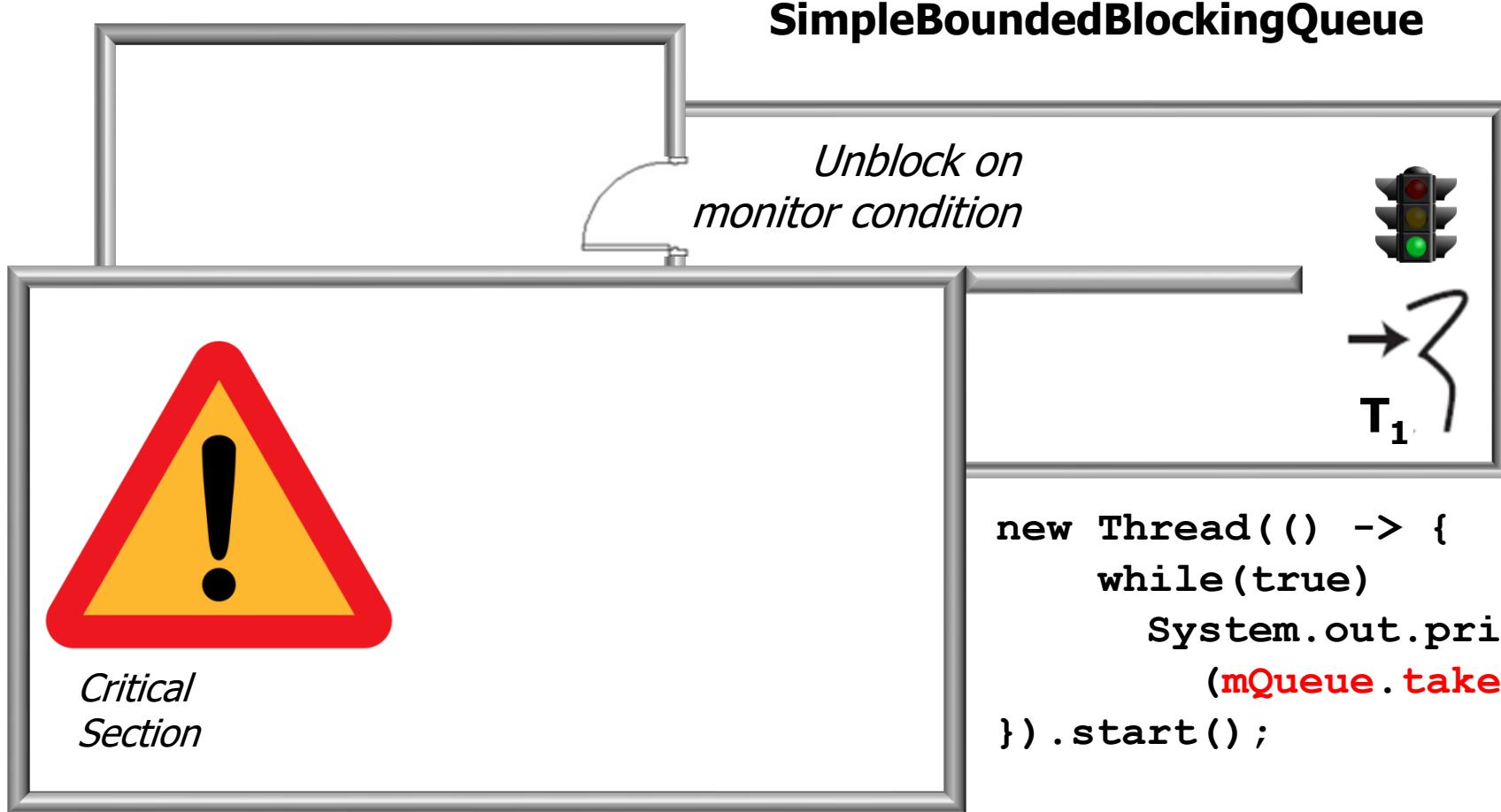
Visual Analysis of SimpleBoundedBlockingQueue



Visual Analysis of SimpleBoundedBlockingQueue

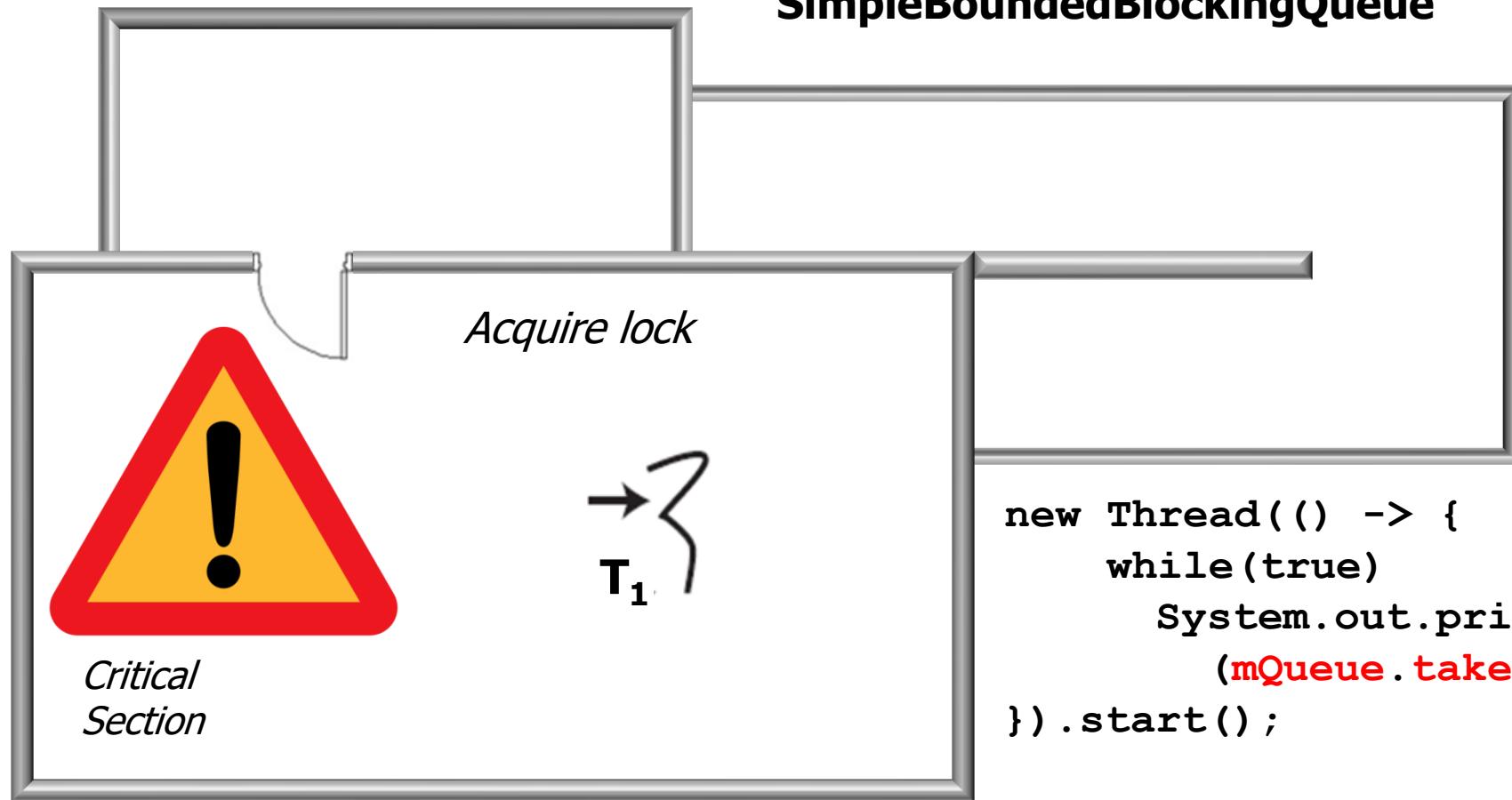


Visual Analysis of SimpleBoundedBlockingQueue



Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue

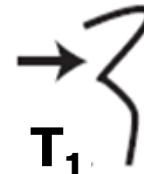


Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



Critical
Section



```
while(isEmpty())
    wait();
```

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

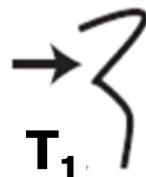
The queue is no longer empty, so continue past the guard

Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



Critical
Section



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

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

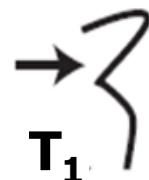
Calling notifyAll() before removing/returning the front item in the queue is ok since the monitor lock is held & only one method can be in the monitor object

Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



*Critical
Section*



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

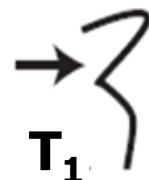
```
new Thread(() -> {  
    while(true)  
        System.out.println  
            (mQueue.take());  
}).start();
```

Visual Analysis of SimpleBoundedBlockingQueue

SimpleBoundedBlockingQueue



*Critical
Section*



Release lock



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

Visual Analysis of SimpleBoundedBlockingQueue

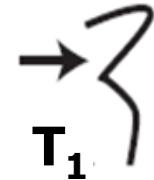
SimpleBoundedBlockingQueue



*Critical
Section*

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

Leave monitor object



End of Visualizing the Java Monitor Object Coordination Example
