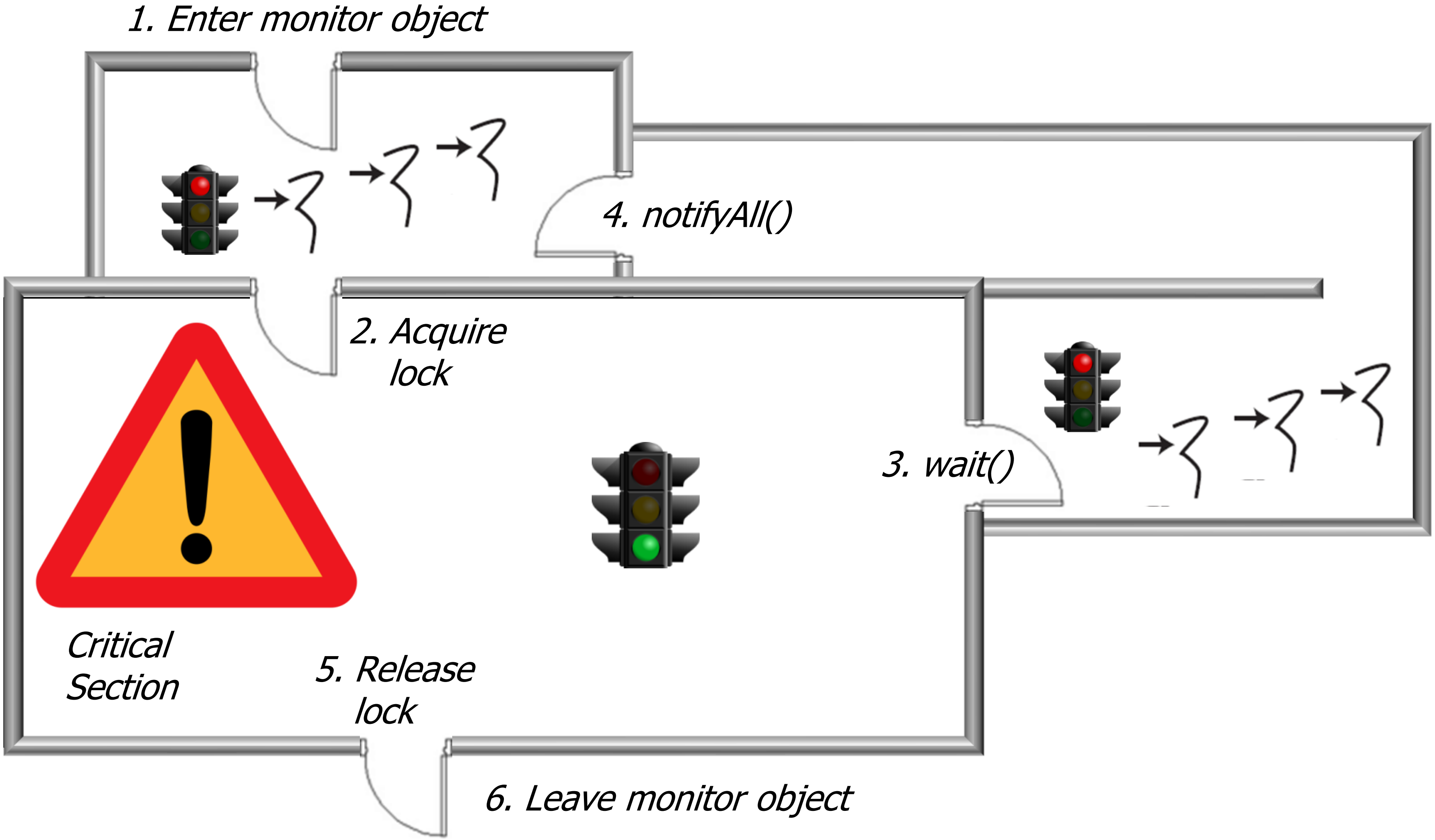
Java Monitor Objects: Coordination Example Visualization

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

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

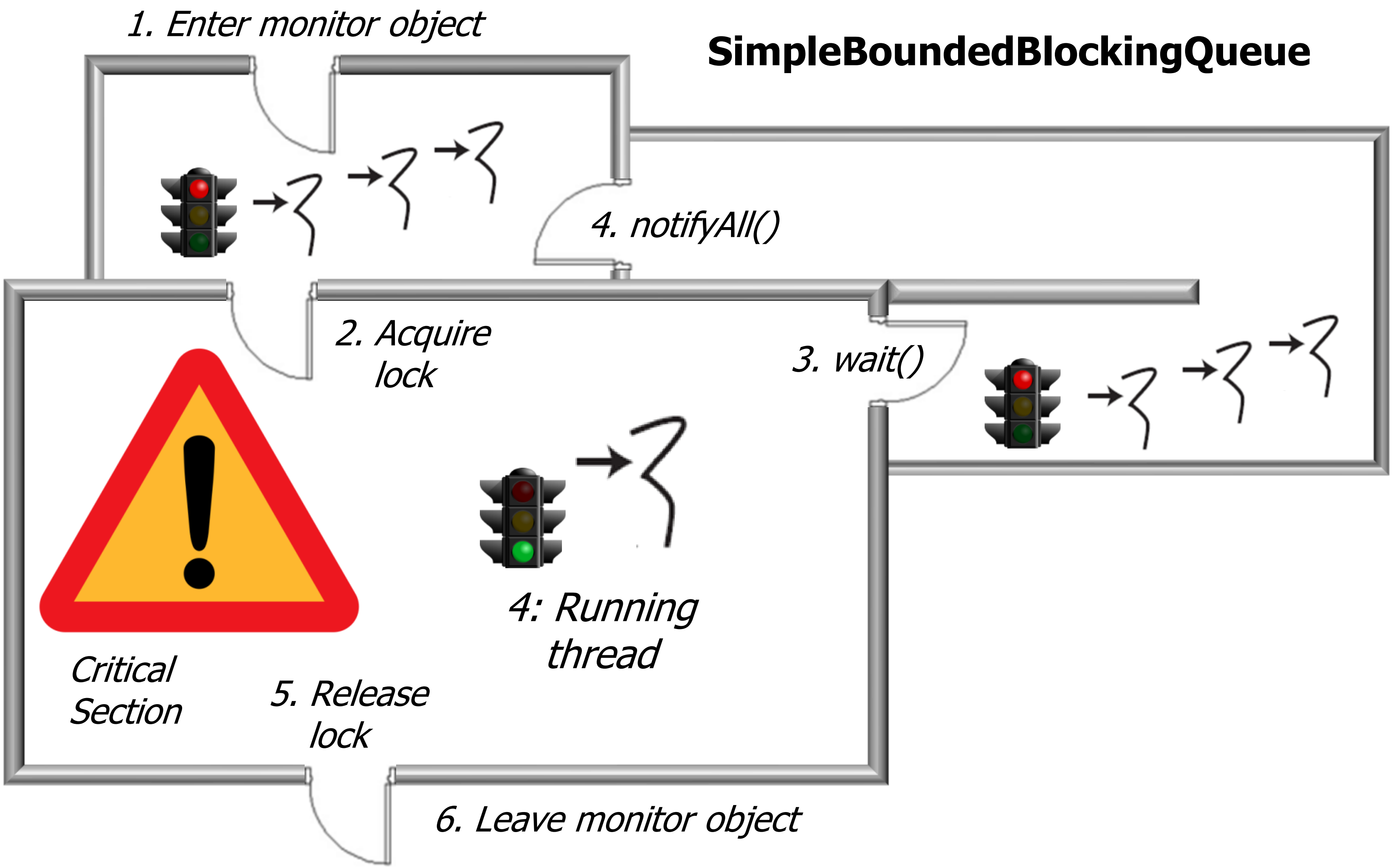
Learning Objectives in this Part of the Lesson

- 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



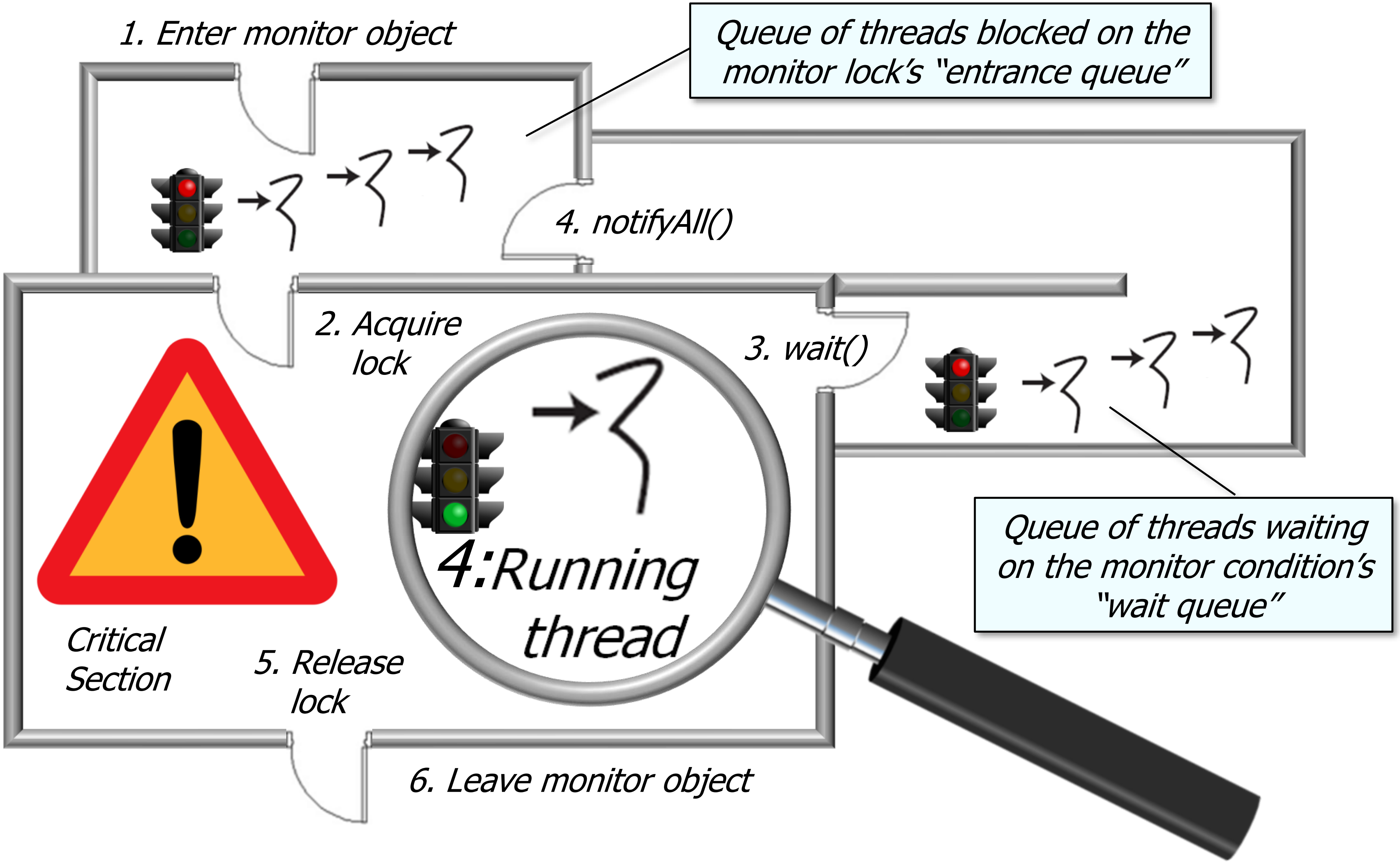
Visual Analysis of the SimpleBlockingBounded Queue Example

Visual Analysis of SimpleBoundedBlockingQueue



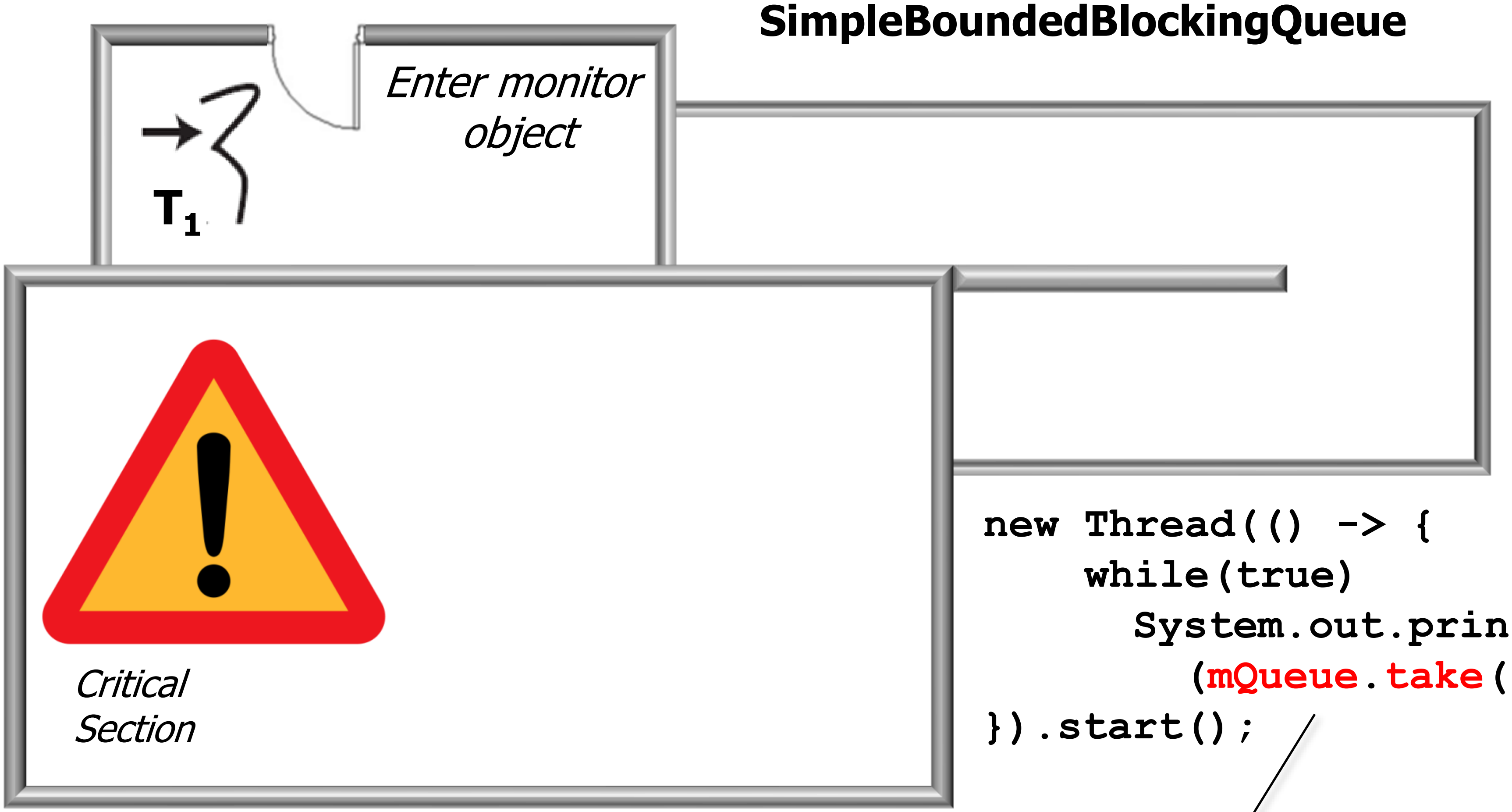
See github.com/douglascraigschmidt/POSA/tree/master/ex/M3/Queues/SimpleBoundedBlockingQueue

Visual Analysis of SimpleBoundedBlockingQueue



See en.wikipedia.org/wiki/Monitor_(synchronization)#Implicit_condition_variable_monitors

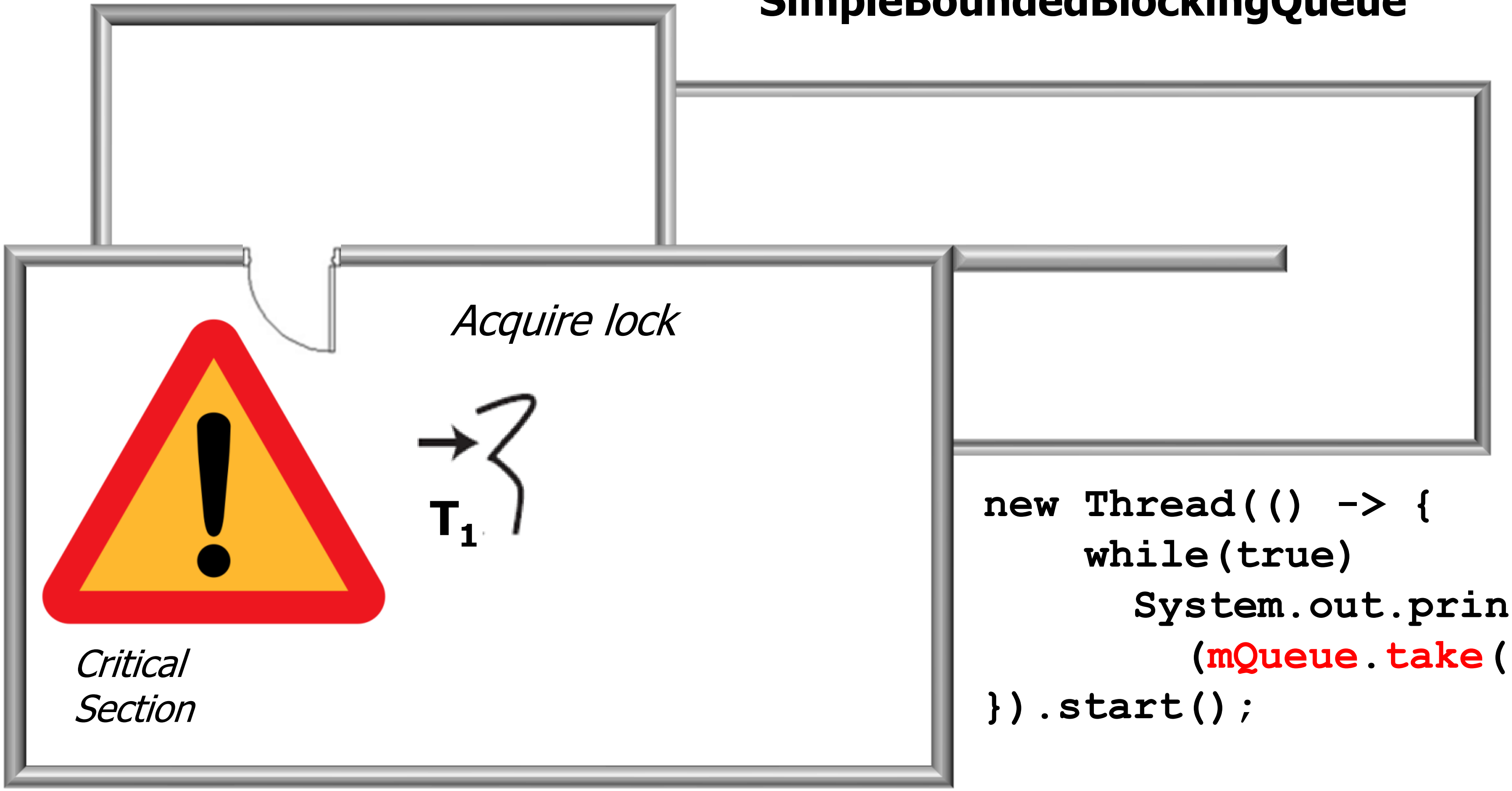
Visual Analysis of SimpleBoundedBlockingQueue



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

We'll assume the queue is initially empty

Visual Analysis of SimpleBoundedBlockingQueue


SimpleBoundedBlockingQueue

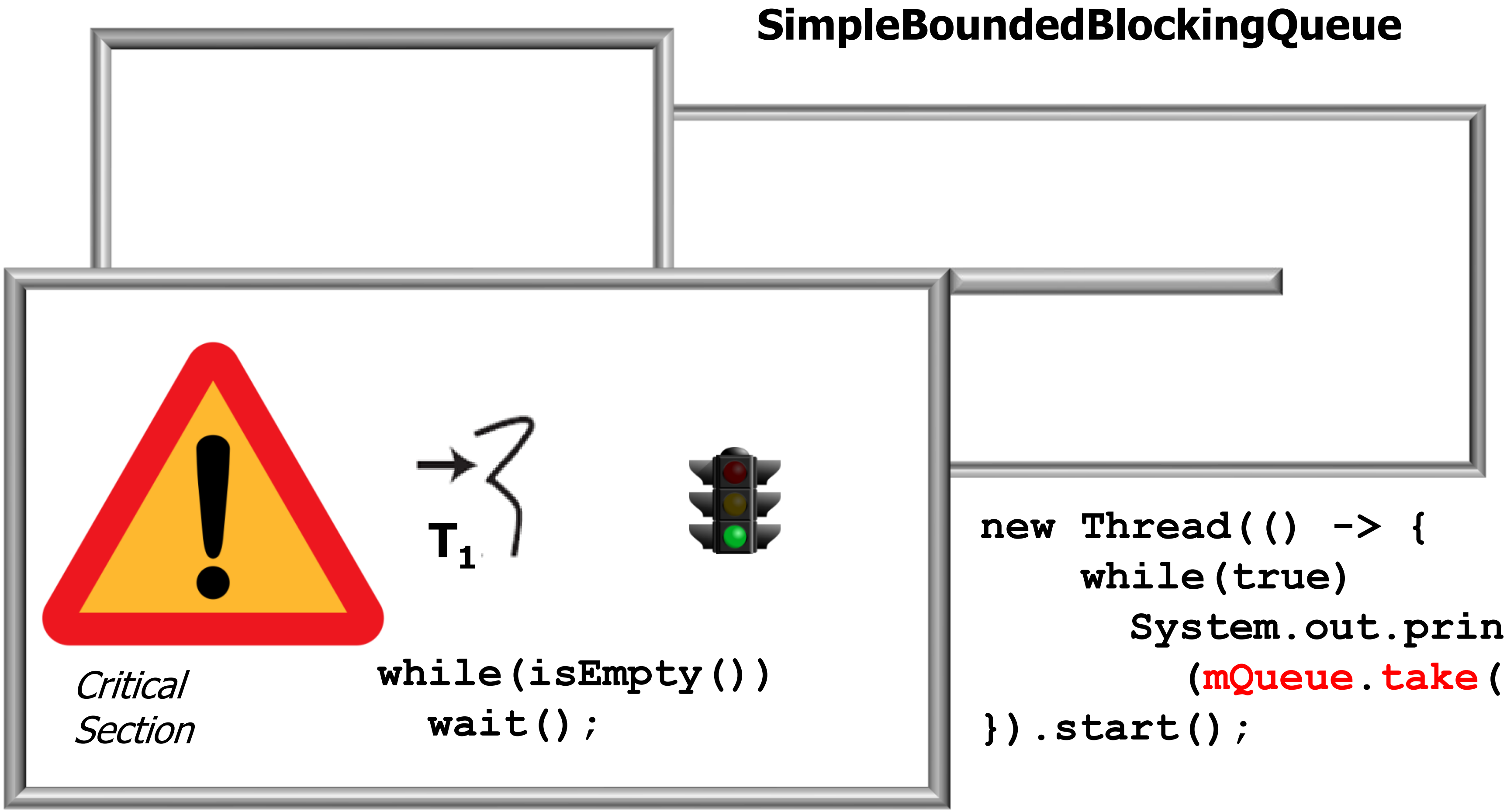
Acquire lock

T_1

Critical Section

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

Visual Analysis of SimpleBoundedBlockingQueue



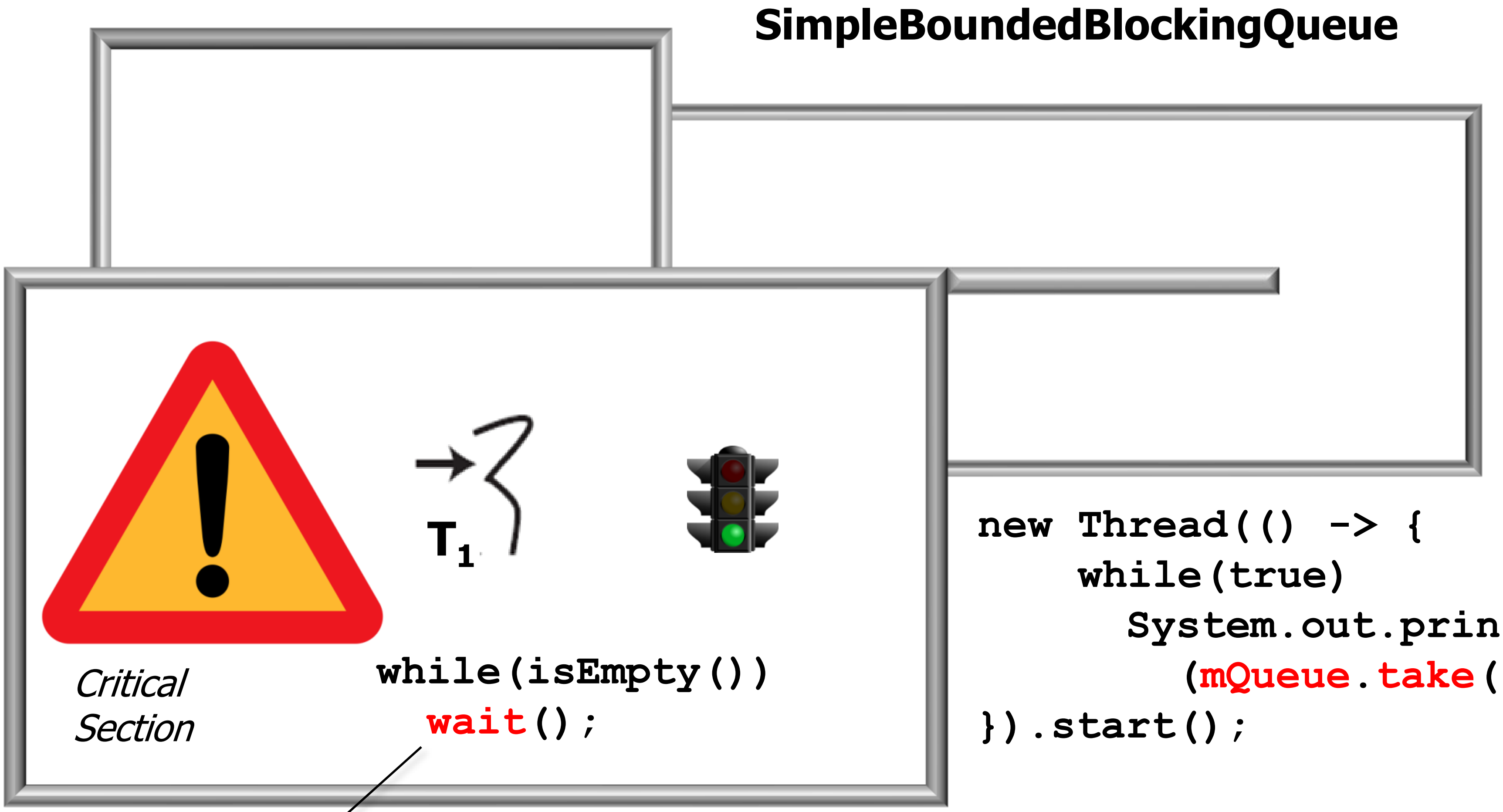
SimpleBoundedBlockingQueue

Critical Section

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

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

Visual Analysis of SimpleBoundedBlockingQueue

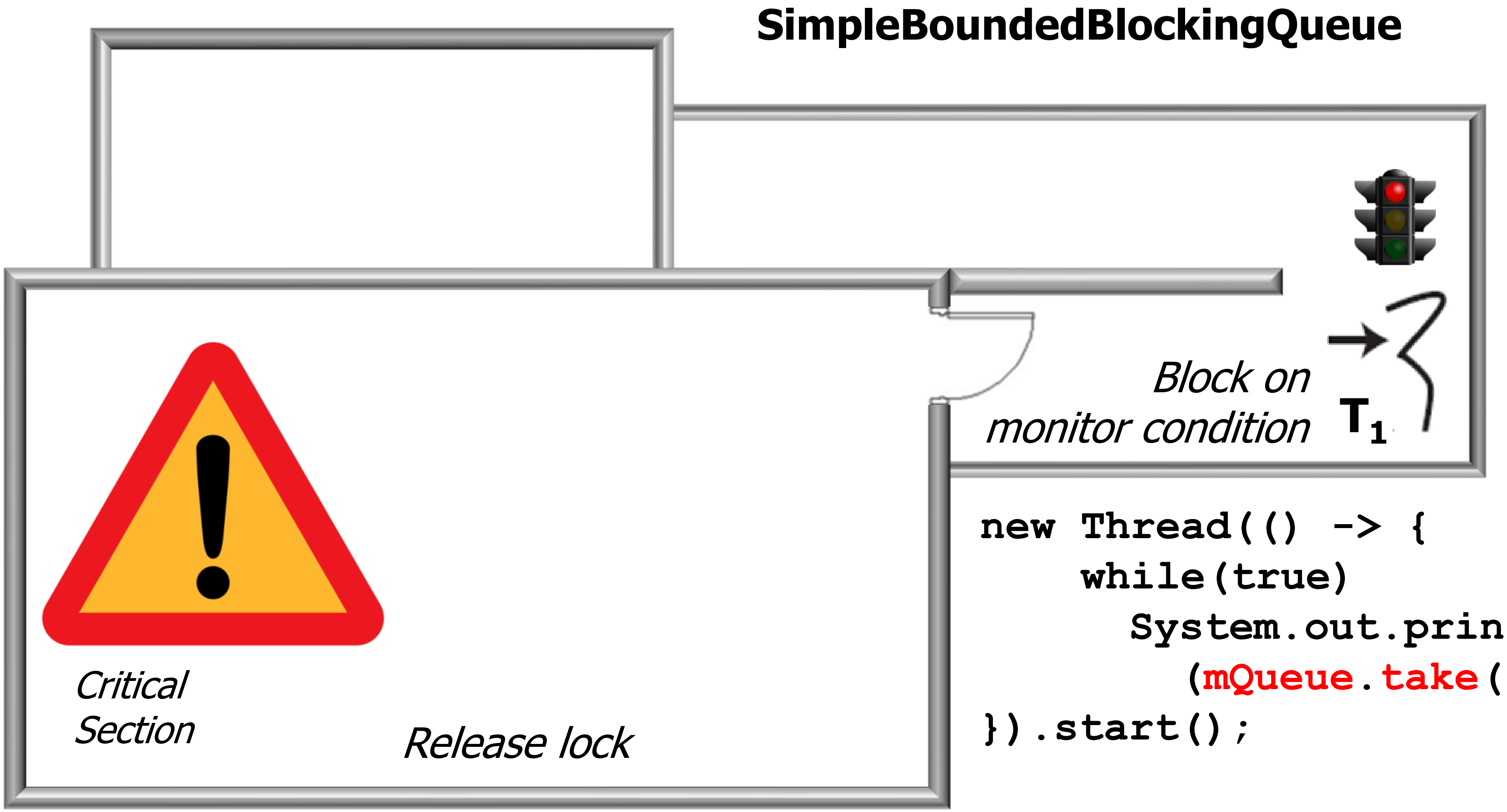


```
while(isEmpty())
  wait();
```

```
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

Block on monitor condition

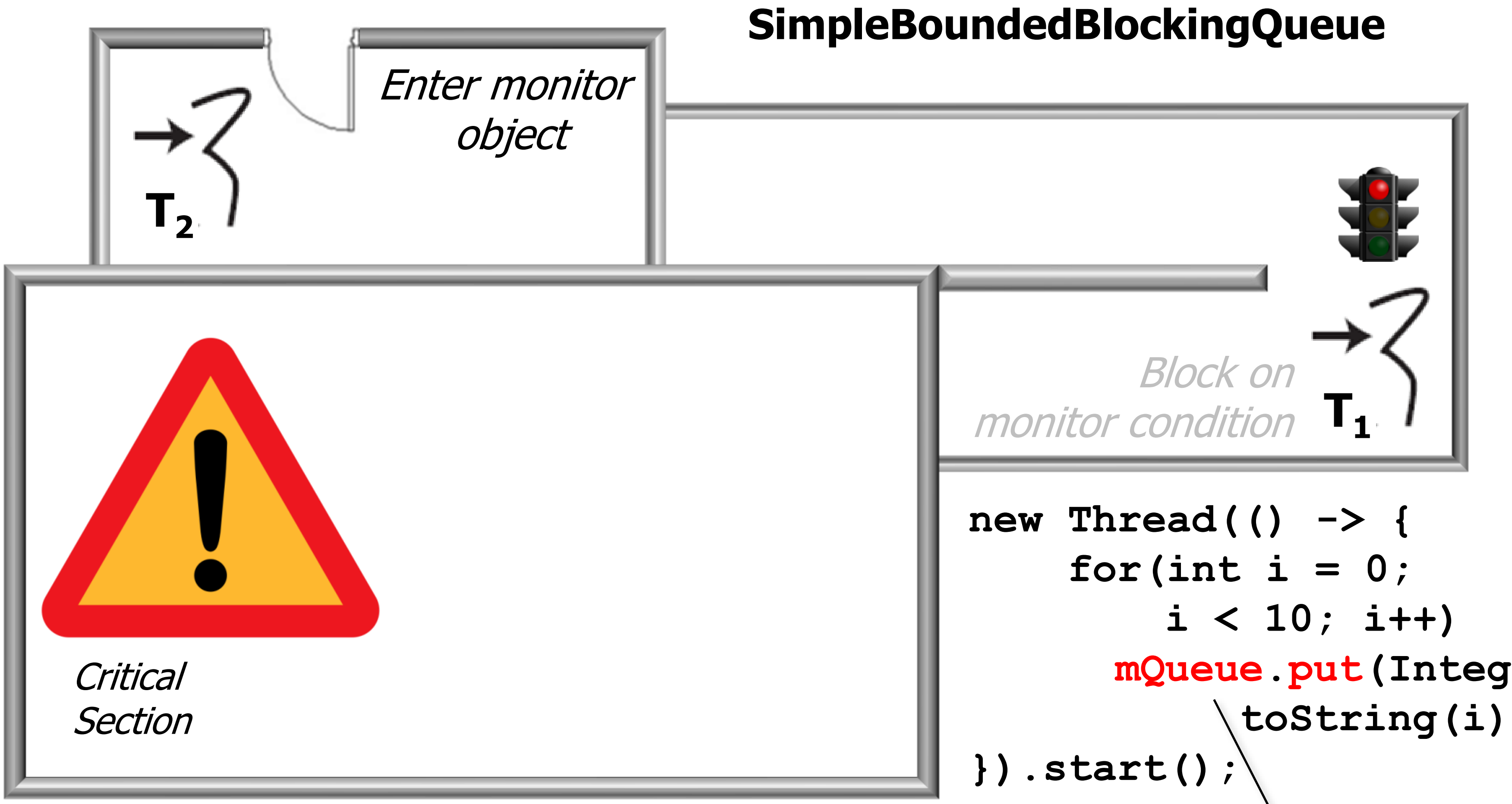
T_1

Critical Section

Release lock

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

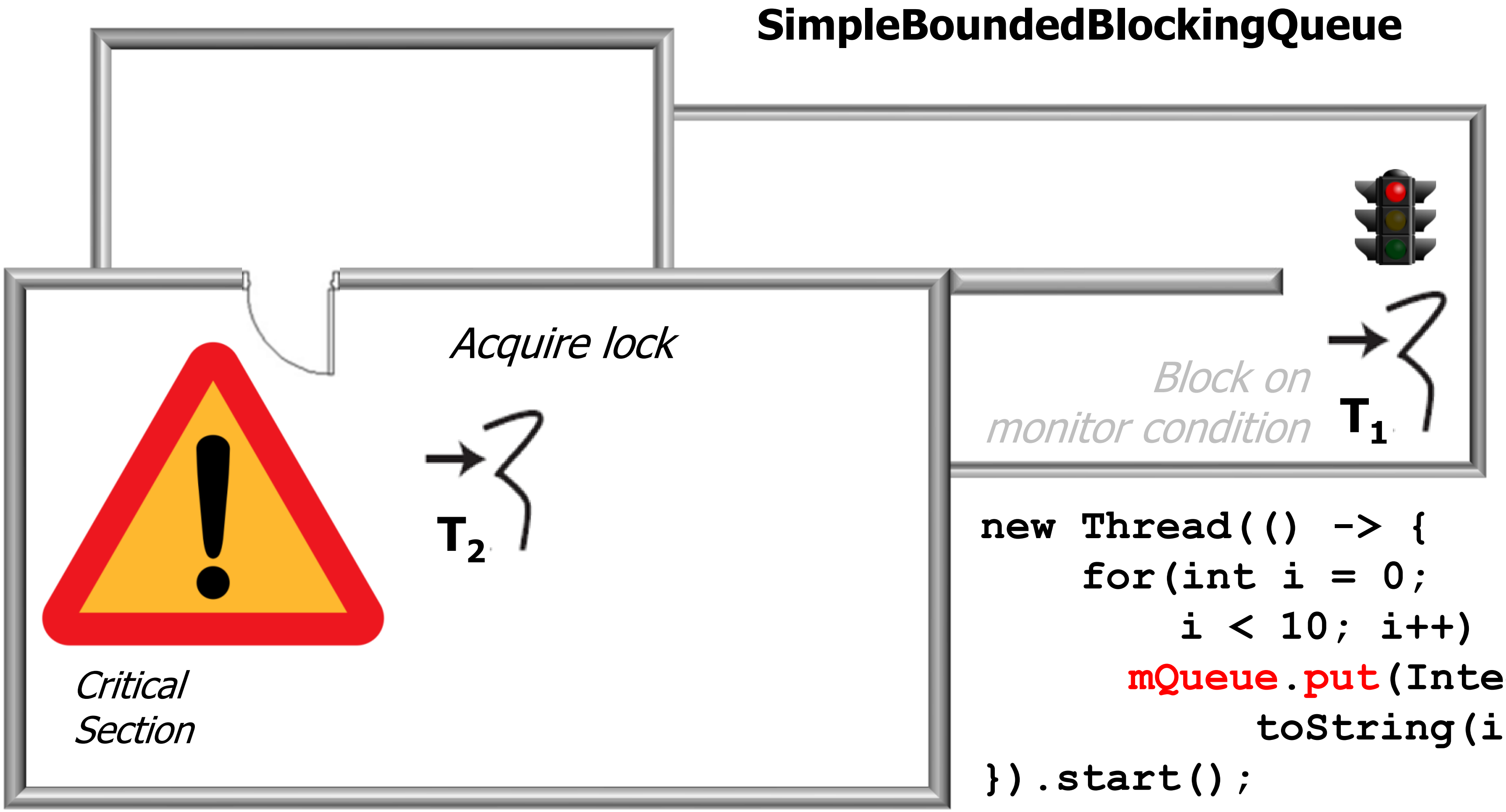
Visual Analysis of SimpleBoundedBlockingQueue



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

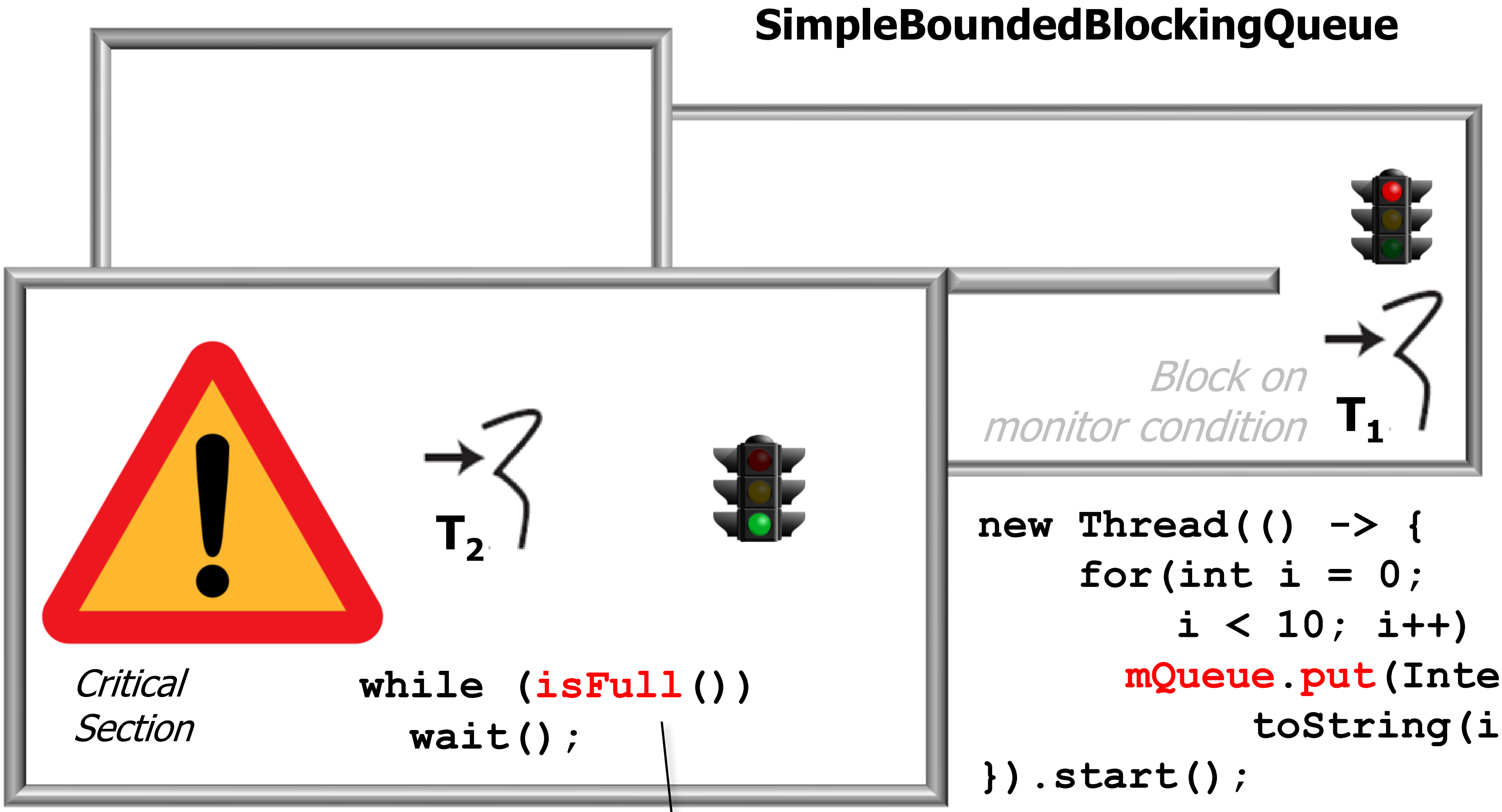
We'll assume the queue is not full

Visual Analysis of SimpleBoundedBlockingQueue



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

Visual Analysis of SimpleBoundedBlockingQueue

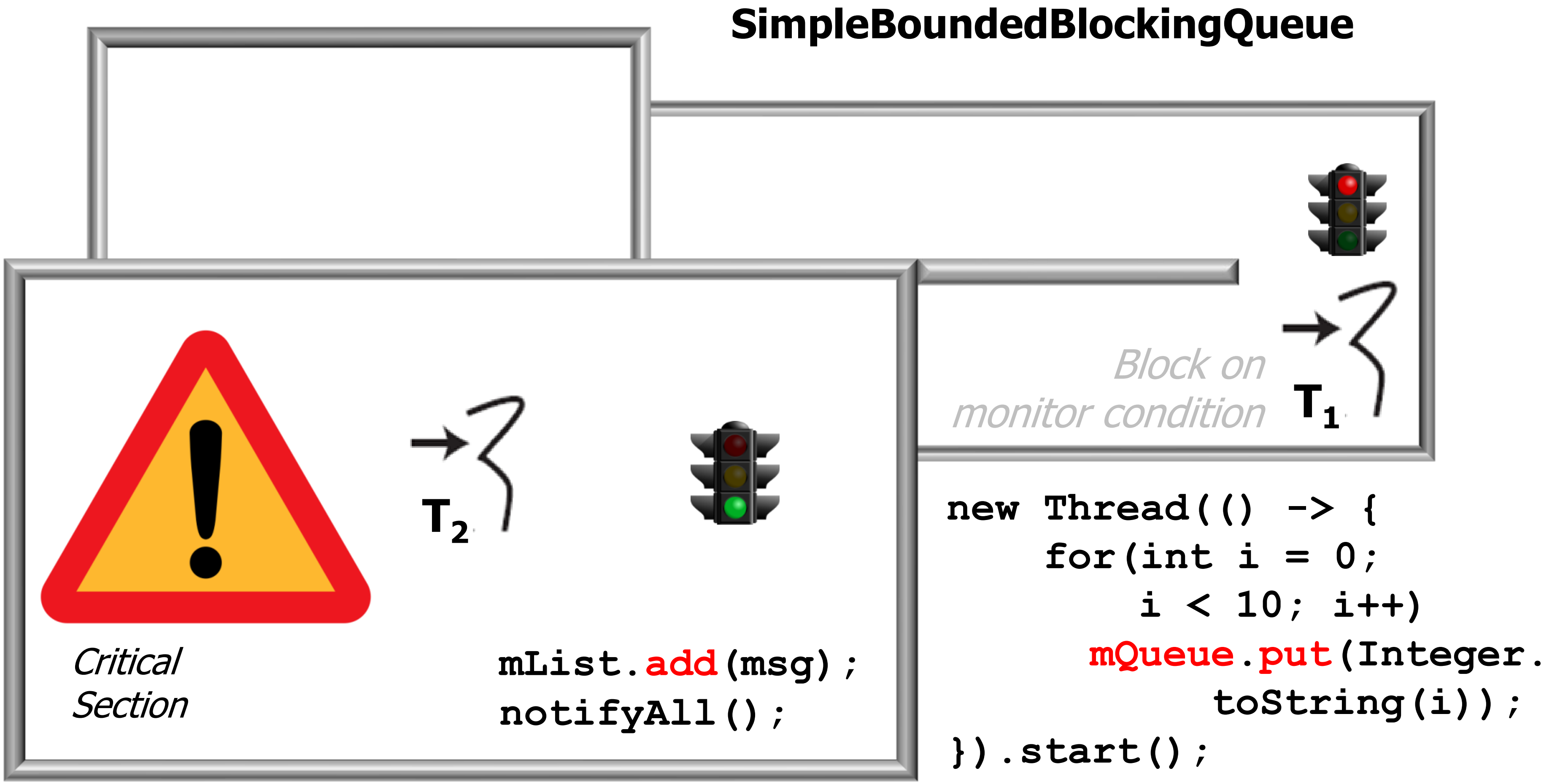


```
while (isFull())
  wait();
```

```
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



SimpleBoundedBlockingQueue

Block on monitor condition

T_1

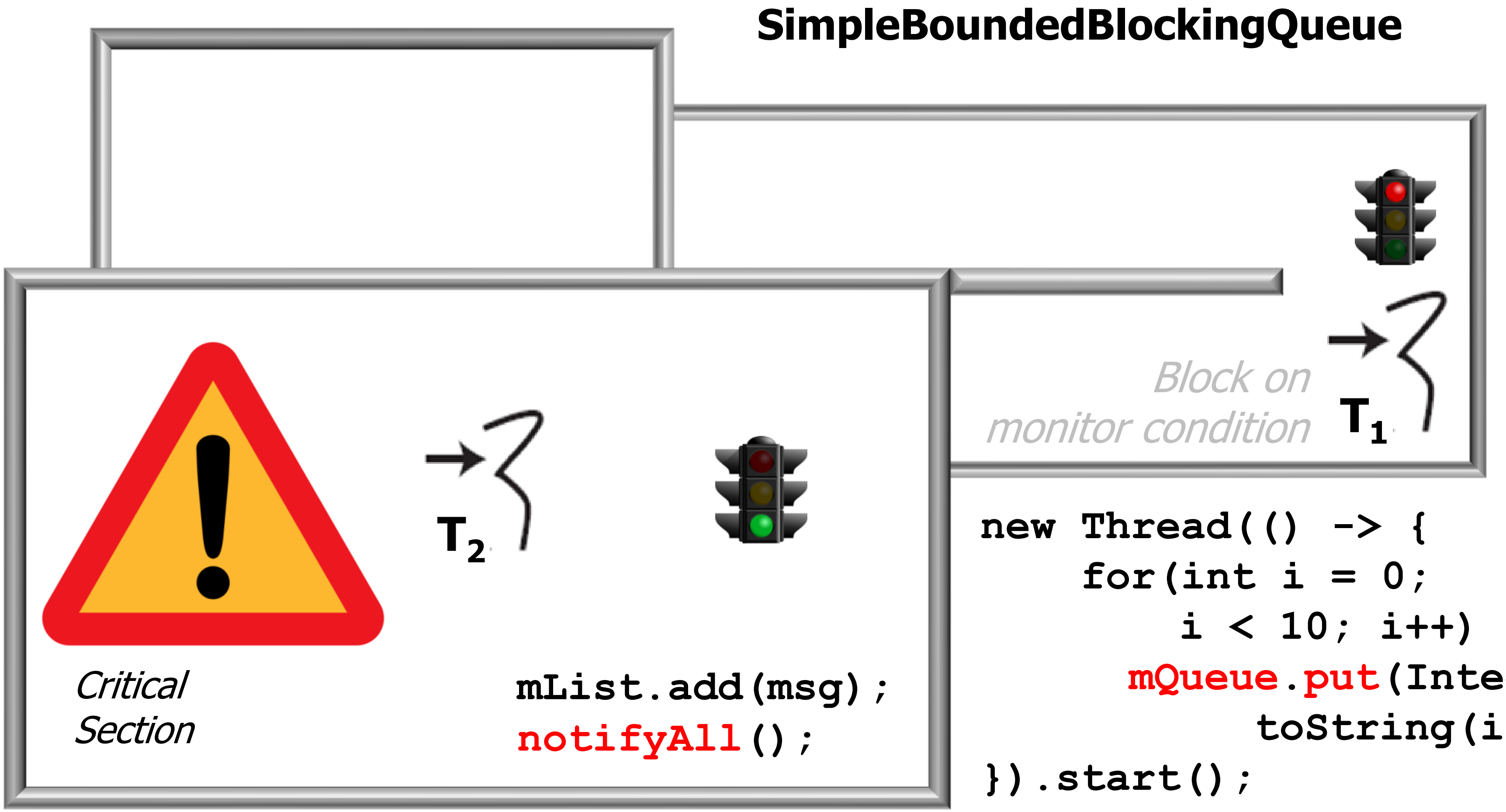
T_2

Critical Section

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

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

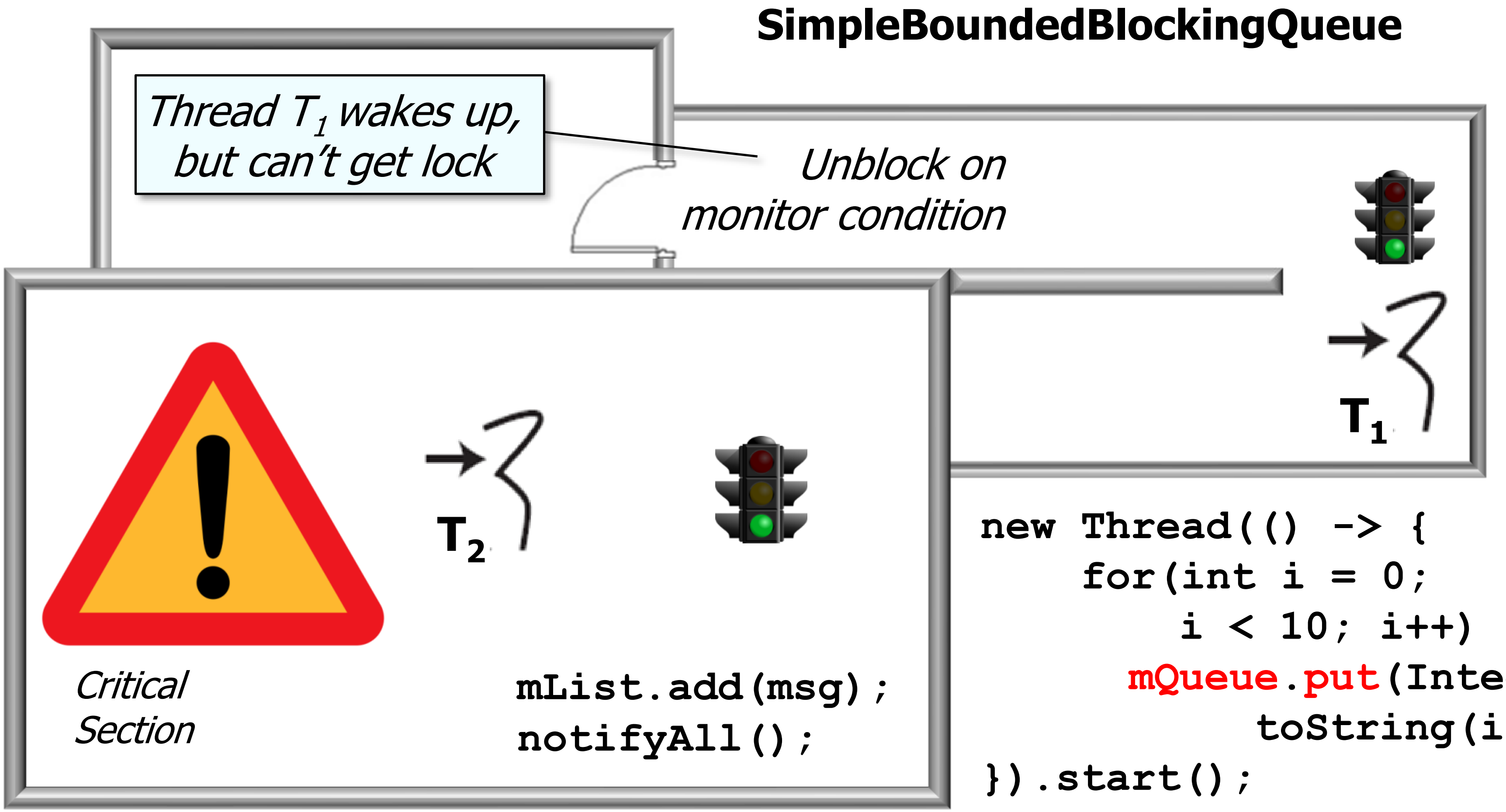
Visual Analysis of SimpleBoundedBlockingQueue



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

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

Visual Analysis of SimpleBoundedBlockingQueue



Thread T_1 wakes up, but can't get lock

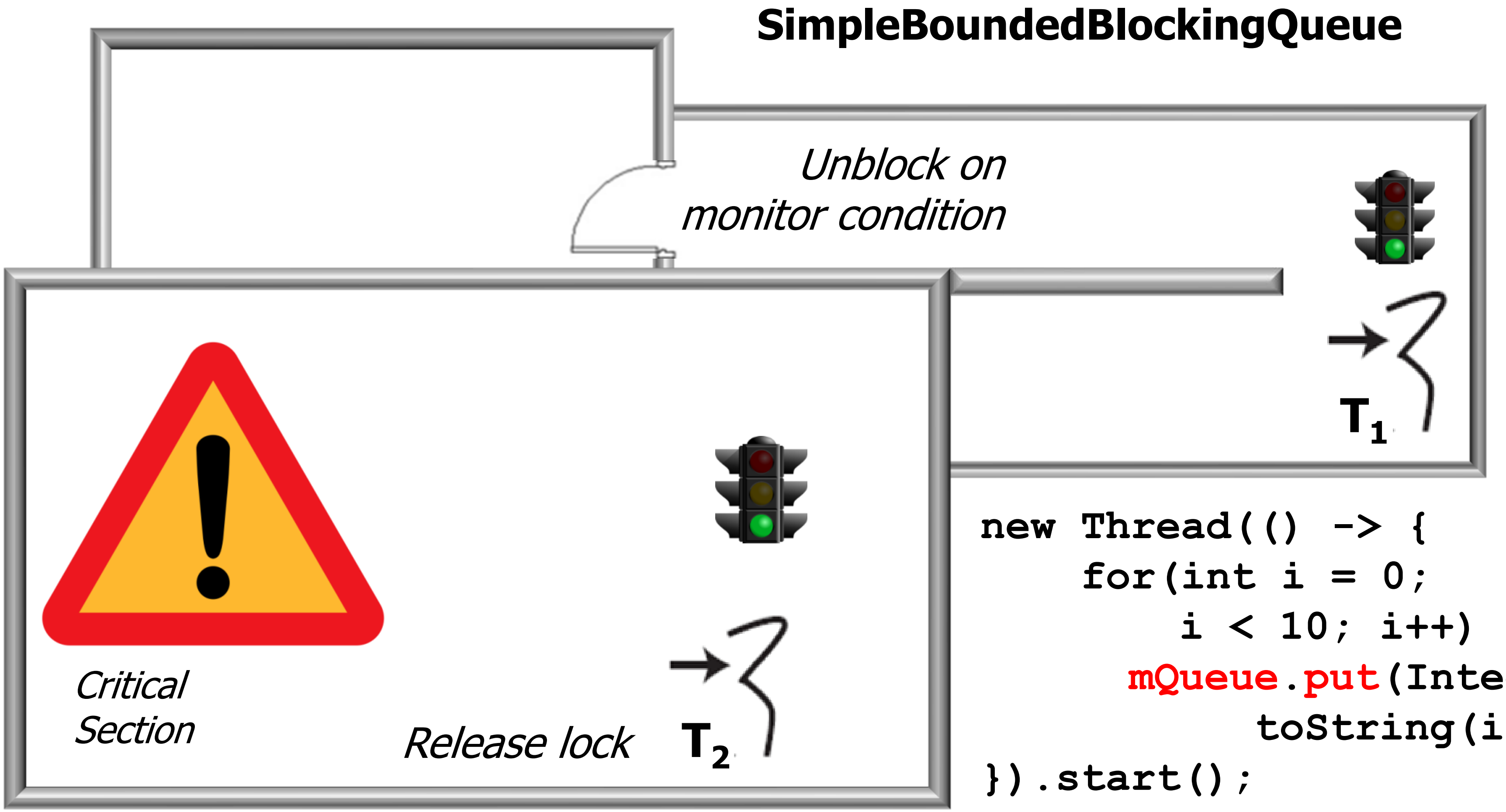
Unblock on monitor condition

Critical Section

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

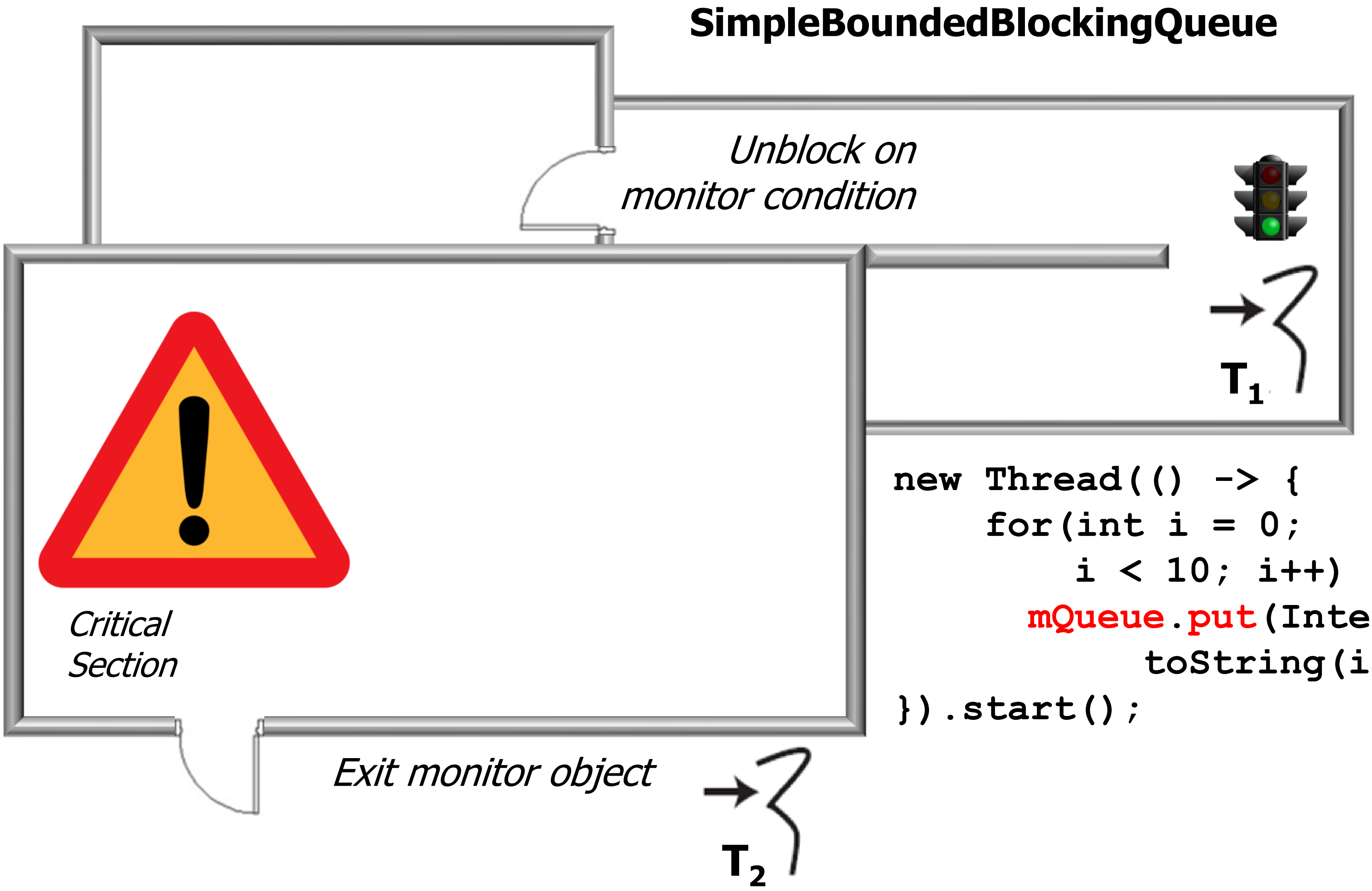
```
new Thread(() -> {
  for(int i = 0;
    i < 10; i++)
   mQueue.put(Integer.
     toString(i));
}).start();
```

Visual Analysis of SimpleBoundedBlockingQueue



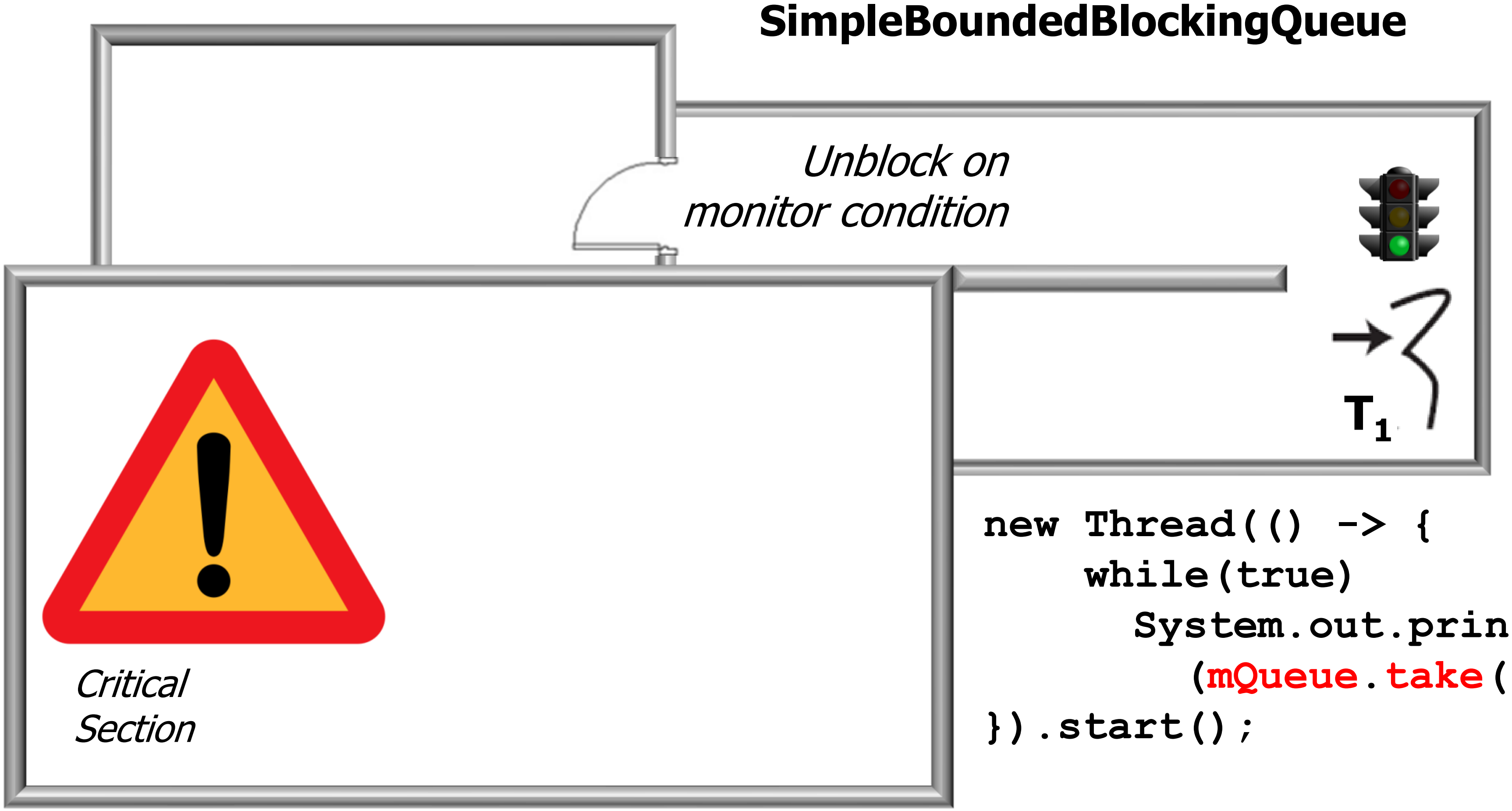
```
new Thread(() -> {
    for(int i = 0;
        i < 10; i++)
      mQueue.put(Integer.
           toString(i));
}).start();
```

Visual Analysis of SimpleBoundedBlockingQueue



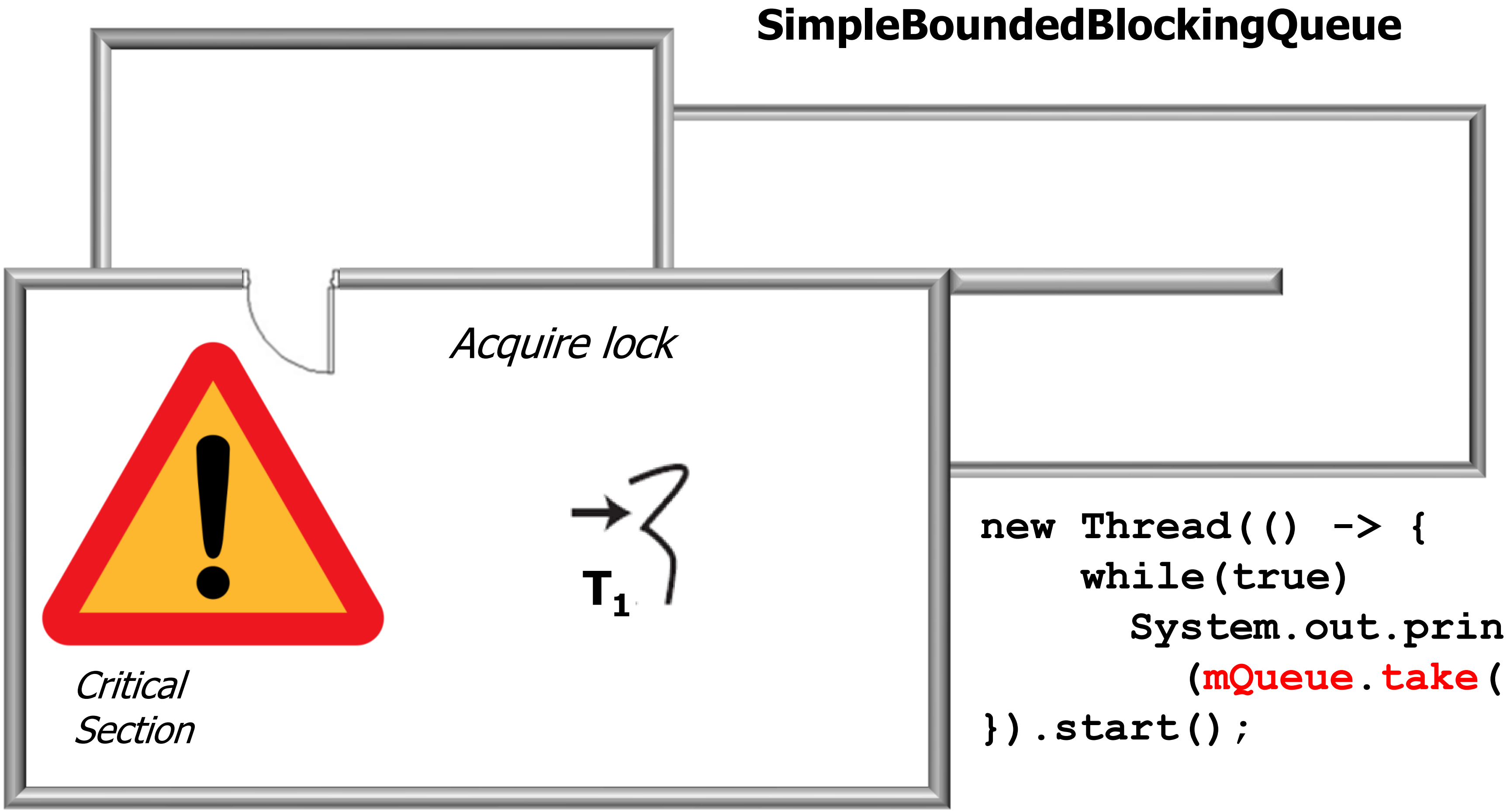
```
new Thread(() -> {
     for(int i = 0;
          i < 10; i++)
        mQueue.put(Integer.
             toString(i));
}).start();
```

Visual Analysis of SimpleBoundedBlockingQueue



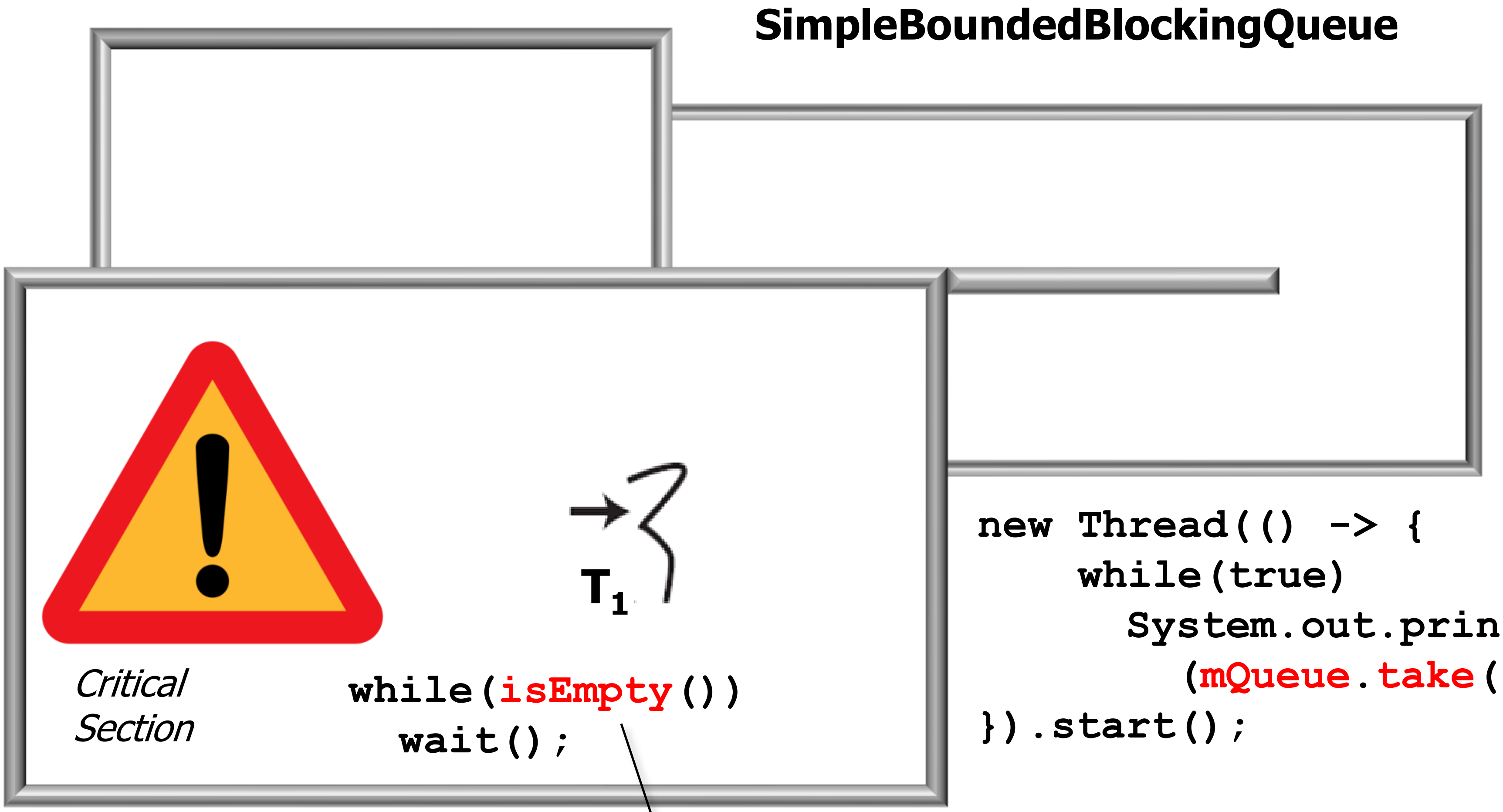
```
new Thread(() -> {
    while(true)
      System.out.println
        (mQueue.take());
}).start();
```

Visual Analysis of SimpleBoundedBlockingQueue



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

Visual Analysis of SimpleBoundedBlockingQueue

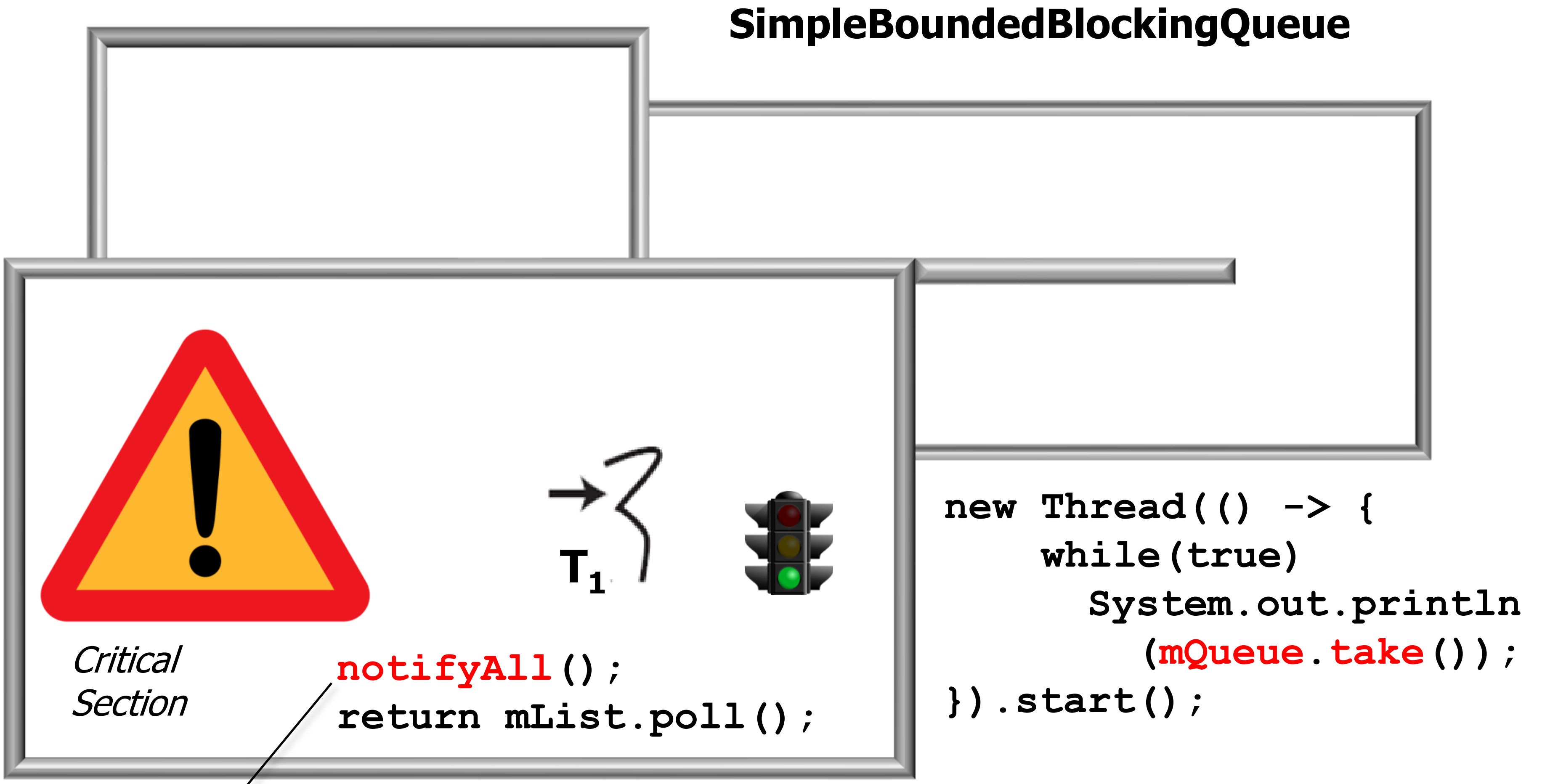


```
while(isEmpty())
  wait();
```

```
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

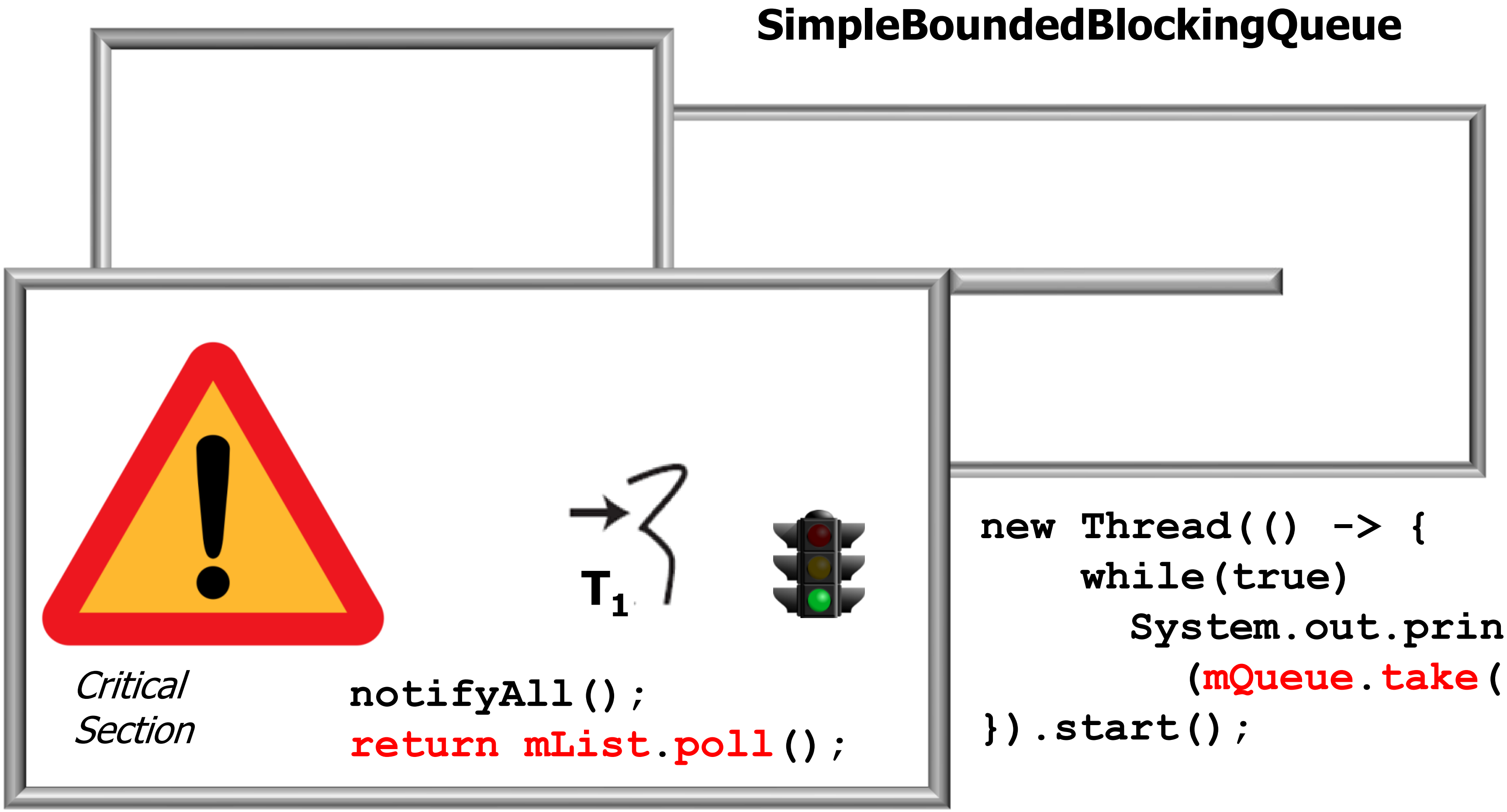


```
notifyAll();
return mList.poll();
```

```
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



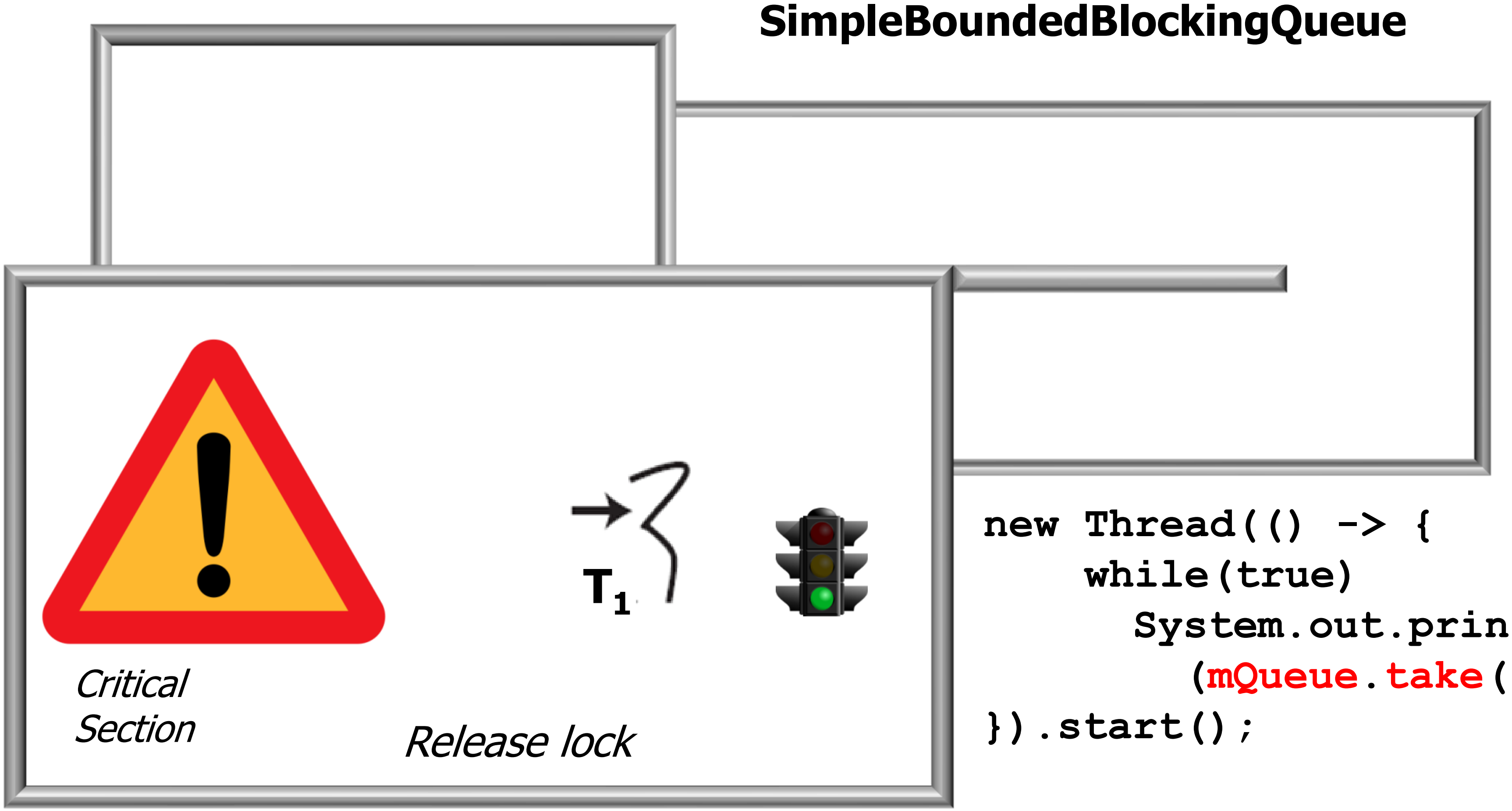
SimpleBoundedBlockingQueue

Critical Section

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

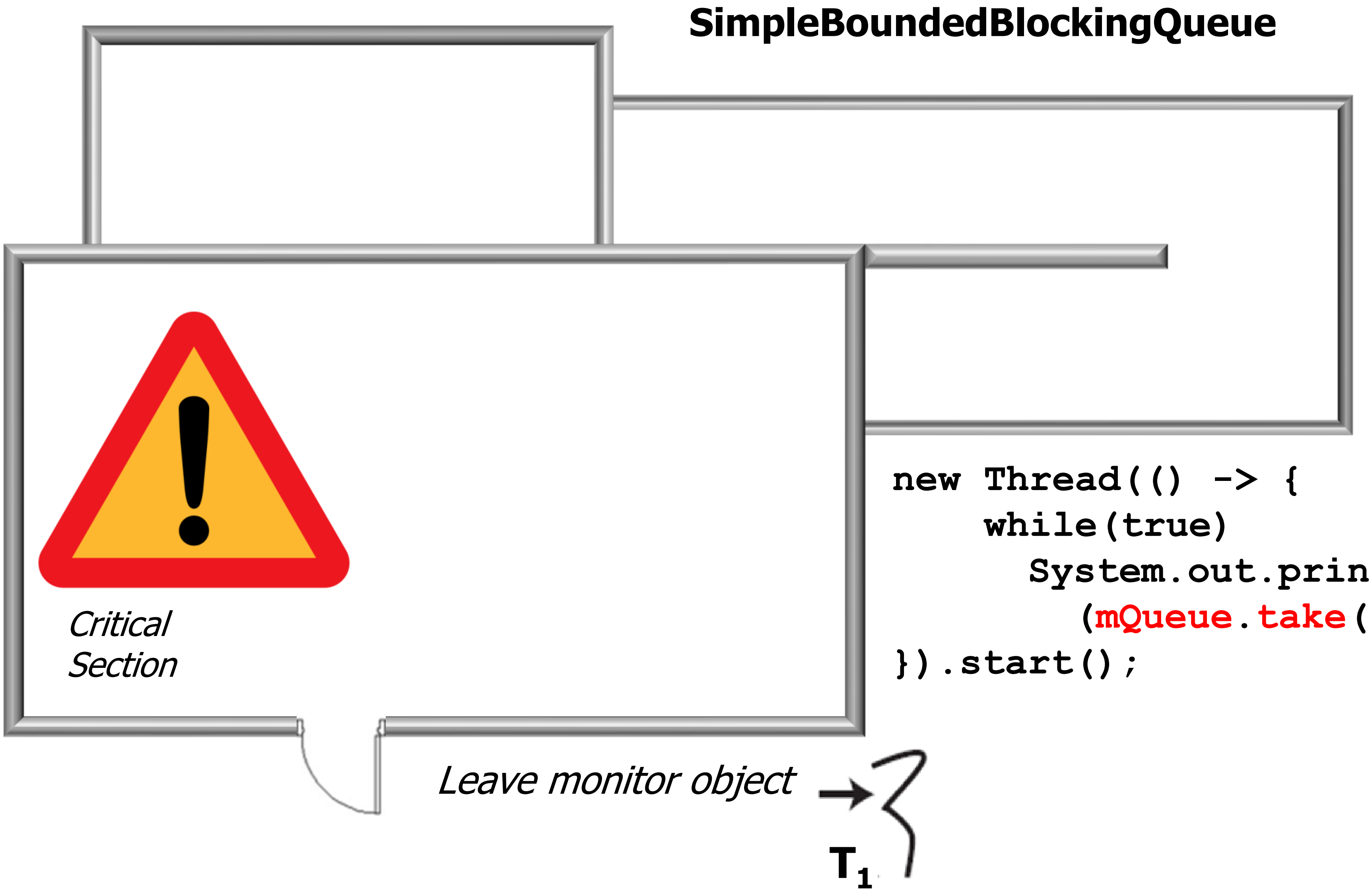
```
new Thread(() -> {
    while(true)
      System.out.println
        (mQueue.take());
}).start();
```

Visual Analysis of SimpleBoundedBlockingQueue



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

Visual Analysis of SimpleBoundedBlockingQueue



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

End of Java Monitor Object: Coordination Example Visualization