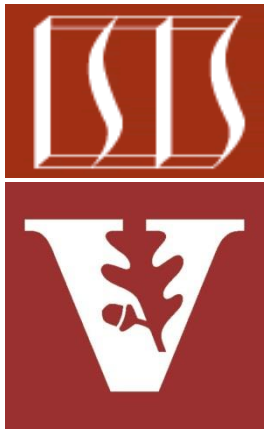
Evaluating Java Monitor Object Synchronized Methods



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

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

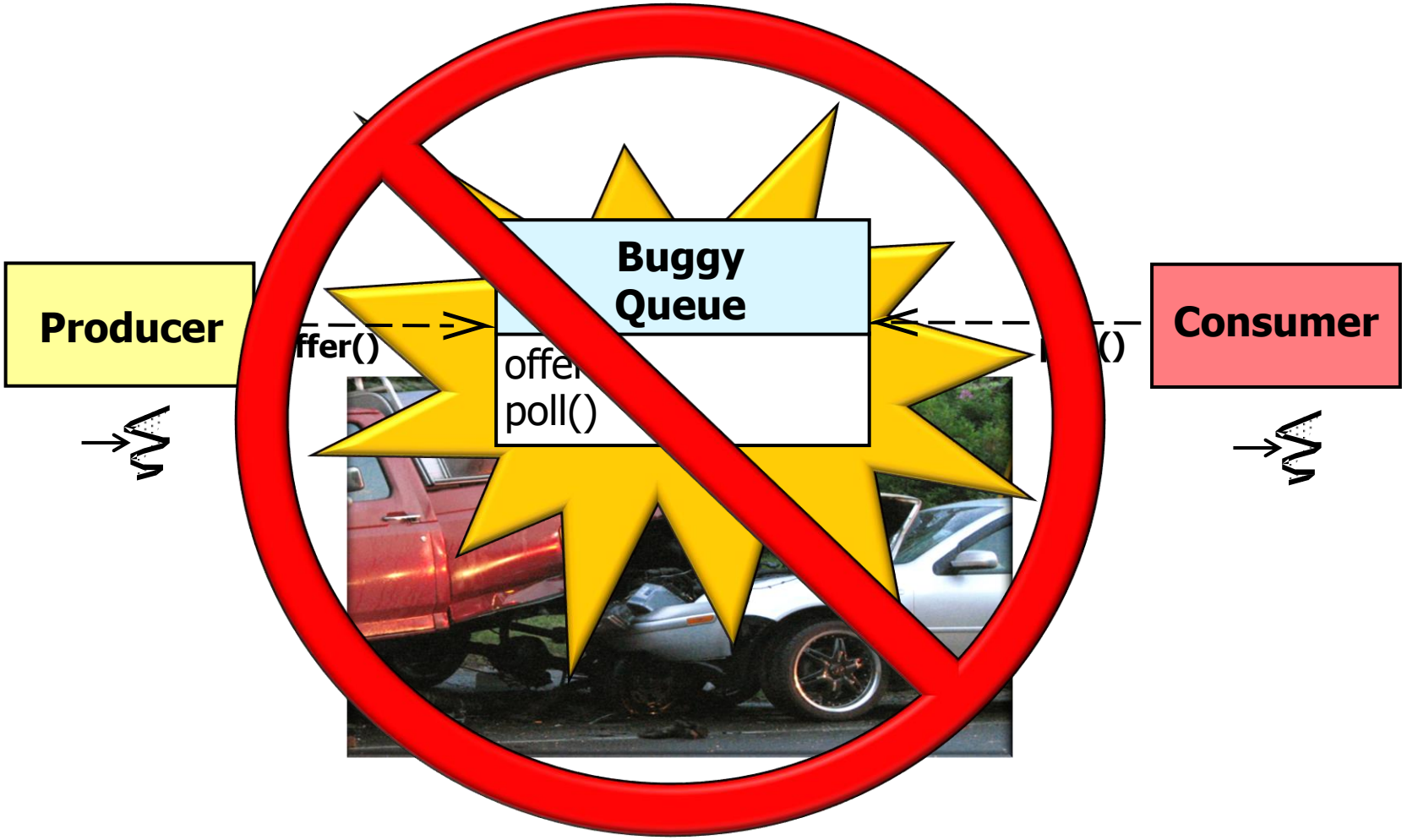
Learning Objectives in this Part of the Lesson

- Recognize the synchronized methods/ statements provided by Java built-in monitor objects to support *mutual exclusion*
- Understand how to fix race conditions in the buggy concurrent Java app by using synchronized methods
- Evaluate the pros & cons of applying Java synchronized methods to the BusySynchronizedQueue class & case study app

Evaluating the Busy SynchronizedQueue

Evaluating the BusySynchronizedQueue

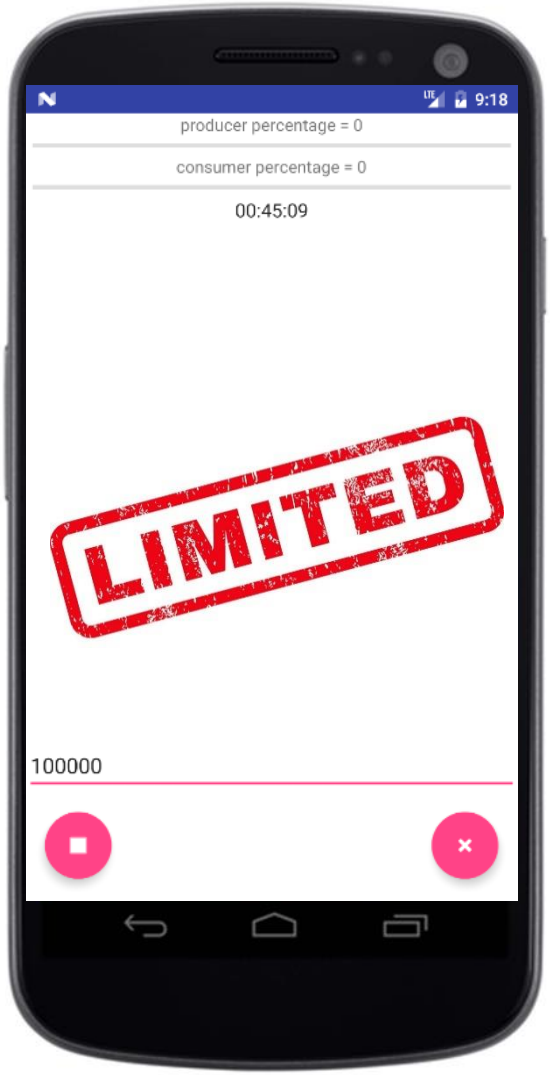
- Applying Java synchronized methods to BusySynchronizedQueue certainly fixed the race condition problems in BuggyQueue



See earlier lessons on "*Java Monitor Objects: Motivating Example*"

Evaluating the BusySynchronizedQueue

- However, Java synchronized methods can be limited when used in isolation



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

Evaluating the BusySynchronizedQueue

- However, Java synchronized methods can be limited when used in isolation

```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E> {
  private List<E> mList;
  private int mCapacity;

  public BusySynchronizedQueue(int capacity) {
    mCapacity = capacity; mList = new LinkedList<>();
  }

  public synchronized boolean offer(E e) {
      if (!isFull())
      { mList.add(e); return true; }
      else
         return false;
    }

  public E synchronized poll() { return mList.poll(); }
  ...
```

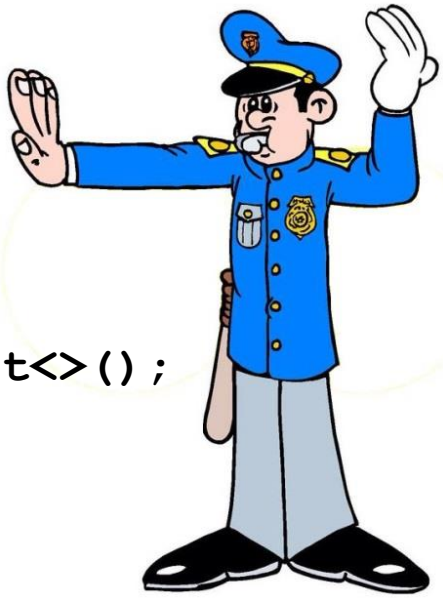
Concurrent calls to these methods will "busy wait"..

See en.wikipedia.org/wiki/Busy_waiting

Evaluating the BusySynchronizedQueue

- However, Java synchronized methods can be limited when used in isolation

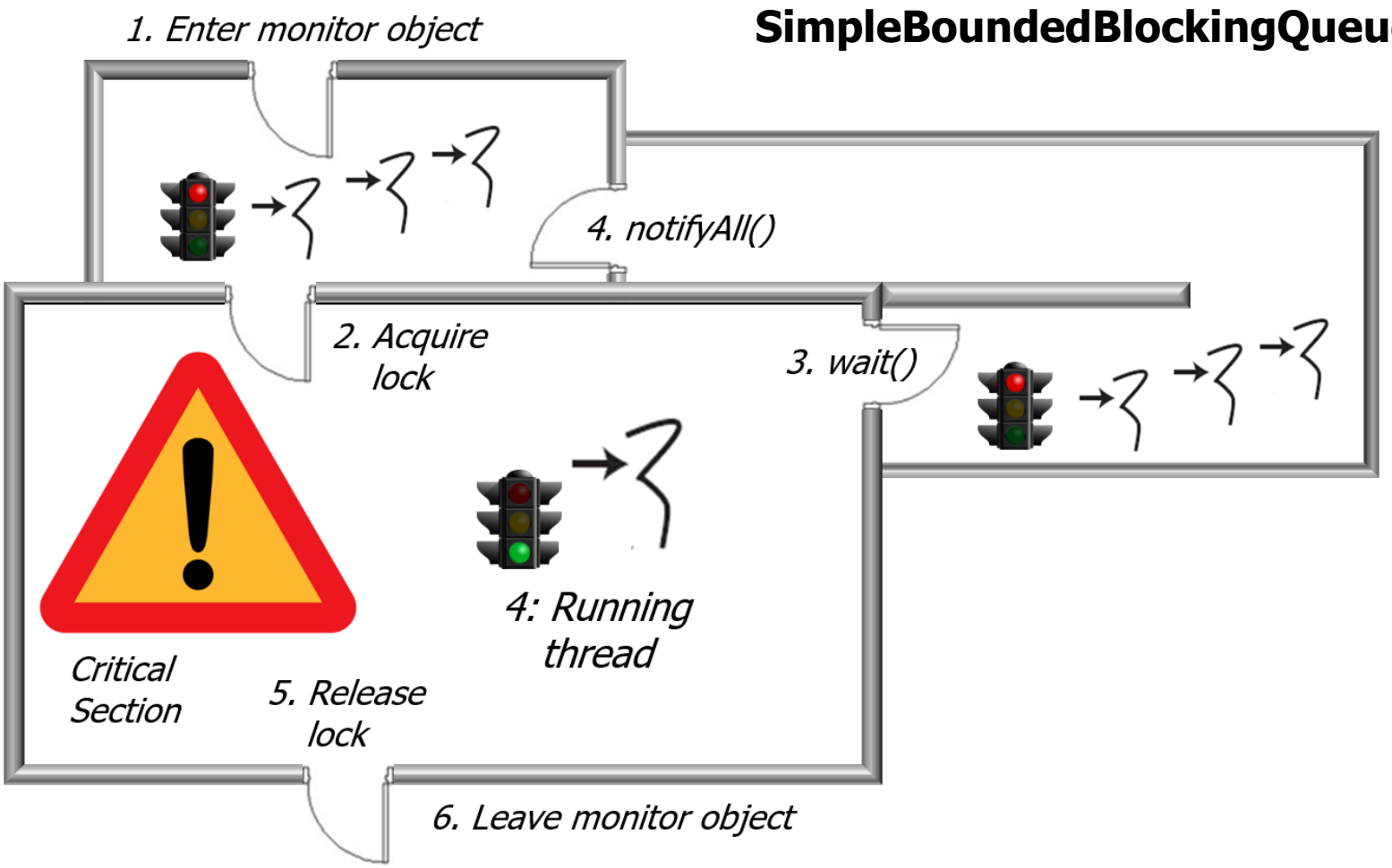
```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E> {
  private List<E> mList;
  private int mCapacity;

  public BusySynchronizedQueue(int capacity) {
    mCapacity = capacity; mList = new LinkedList<>();
  }

  public synchronized boolean offer(E e) {
      if (!isFull())
      { mList.add(e); return true; }
      else
         return false;
    }

  public E synchronized poll() { return mList.poll(); }
  ...
```



Need to coordinate offer() & poll() so they won't busy wait when there's nothing to do

Evaluating the BusySynchronizedQueue

- To avoid busy waiting, therefore, Java monitor objects provide "wait" & "notify" mechanisms



See upcoming lesson on "*Java Monitor Objects: Coordination Methods*"

End of Evaluating Java Monitor Object Synchronized Methods