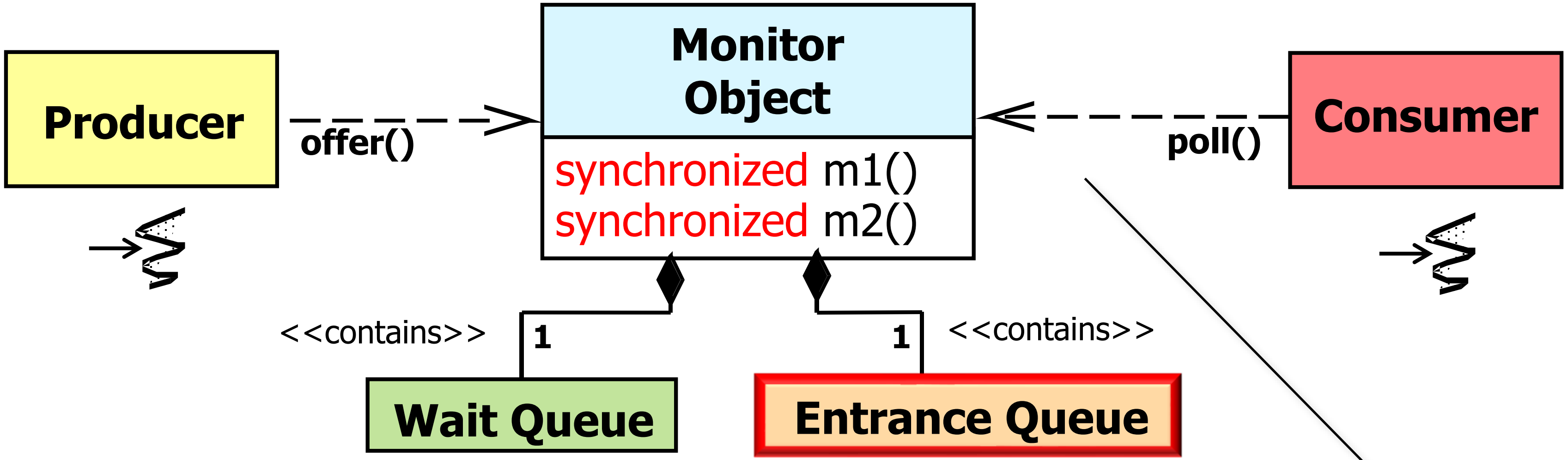
Java Monitor Objects: Synchronized Methods

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 build-in monitor objects to support *mutual exclusion*



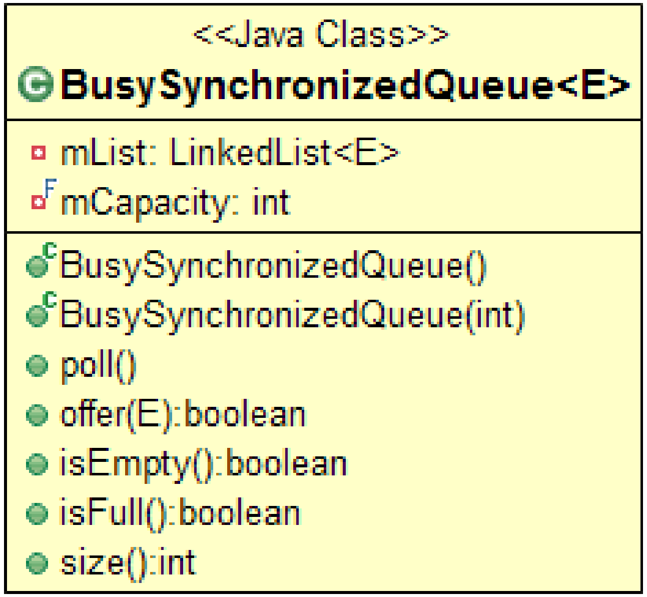
Mutual exclusion is used to protect shared state from corruption due to concurrent access by multiple threads

See en.wikipedia.org/wiki/Mutual_exclusion

Java Synchronized Methods

Java Synchronized Methods

- The BusySynchronizedQueue class showcases Java built-in synchronization mechanisms



<<Java Class>>
BusySynchronizedQueue<E>

mList: LinkedList<E>
mCapacity: int

BusySynchronizedQueue()
BusySynchronizedQueue(int)
poll()
offer(E):boolean
isEmpty():boolean
isFull():boolean
size():int

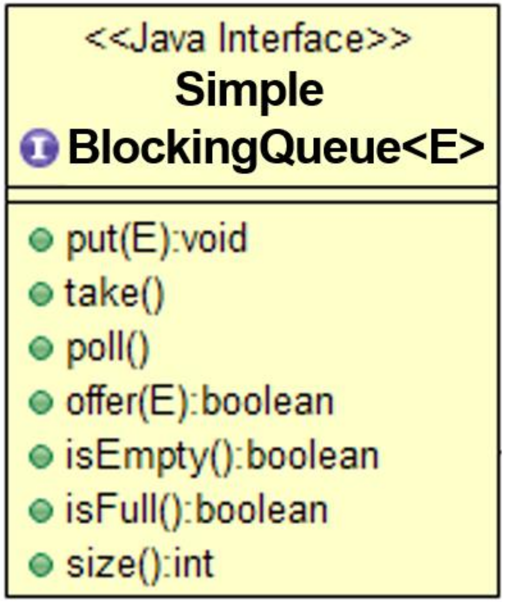
```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E>
{
  private LinkedList<E> mList;
  private int mCapacity;

  BusySynchronizedQueue(int capacity) {
    mList = new LinkedList<E>();
    mCapacity = capacity;
  }
  ...
```

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

Java Synchronized Methods

- The BusySynchronizedQueue class showcases Java built-in synchronization mechanisms

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

  BusySynchronizedQueue(int capacity){
    mList = new LinkedList<E>();
    mCapacity = capacity;
  }
  ...
```



<<Java Interface>>
Simple BlockingQueue<E>

- put(E):void
- take()
- poll()
- offer(E):boolean
- isEmpty():boolean
- isFull():boolean
- size():int

This interface is a variant of what’s available in Java’s BlockingQueue interface

Java Synchronized Methods

- The BusySynchronizedQueue class showcases Java built-in synchronization mechanisms

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

  BusySynchronizedQueue(int capacity){
    mList = new LinkedList<E>();
    mCapacity = capacity;
  }
  ...
```

The state in this class must be protected against race conditions

See en.wikipedia.org/wiki/Race_condition

Java Synchronized Methods

- The BusySynchronizedQueue class showcases Java built-in synchronization mechanisms

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

  BusySynchronizedQueue(int capacity){
    mList = new LinkedList<E>();
    mCapacity = capacity;
  }
  ...
```

The constructor initializes the state

A constructor is called once by a single thread, so synchronization isn't needed

Java Synchronized Methods

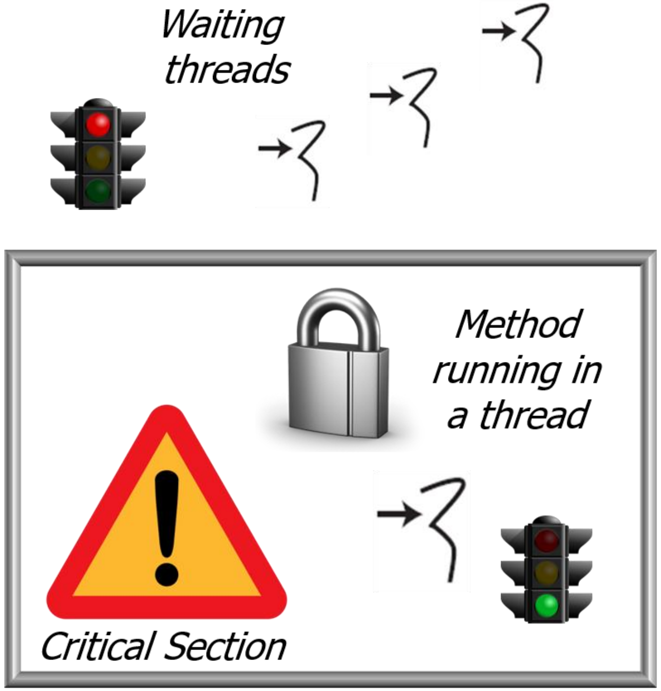
- Methods in a built-in monitor object can be marked with the synchronized keyword

```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E>
{
  ...
  public synchronized boolean
                          offer(E e)
  { ... }

  public synchronized E poll()
  { ... }

  public synchronized boolean isFull()
  { ... }

  ...
```

See docs.oracle.com/javase/tutorial/essential/concurrency/syncmeth.html

Java Synchronized Methods

- Methods in a built-in monitor object can be marked with the synchronized keyword
  - A synchronized method is serialized wrt other synchronized methods in an object



```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E>
{
  ...
  public synchronized boolean
                           offer(E e)
  { ... }

  public synchronized E poll()
  { ... }

  public synchronized boolean isFull()
  { ... }

  ...
```

Java Synchronized Methods

- Methods in a built-in monitor object can be marked with the synchronized keyword
  - A synchronized method is serialized wrt other synchronized methods in an object

```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E>
{
  ...
  public synchronized boolean
                         offer(E e)
  { ... }

  public synchronized E poll()
  { ... }

  public synchronized boolean isFull()
  { ... }

  ...
```

See earlier lesson on "*Java ReentrantLock*"

Java Synchronized Methods

- Methods in a built-in monitor object can be marked with the synchronized keyword
  - A synchronized method is serialized wrt other synchronized methods in an object
  - When used in the method declaration, the entire body of the method is serialized

```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E>
{
  ...
  public synchronized boolean
                          offer(E e)
  { if (!isFull()) {
       mList.add(e);
       return true;
    } else
       return false;
  }

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

  public synchronized boolean isFull()
  { return mList.size() == mCapacity; }

  ...
```

Java Synchronized Methods

- The synchronized keyword is not considered to be part of a method's signature

```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E>
{
  ...
  public synchronized boolean
                          offer(E e)
  { ... }

  public synchronized E poll()
  { ... }

  public synchronized boolean isFull()
  { ... }

  ...
```

Synchronization is considered an "implementation detail"

See gee.cs.oswego.edu/dl/cpj/mechanics.html#synchronization

Java Synchronized Methods

- The synchronized keyword is not considered to be part of a method's signature
  - synchronized is *not* inherited when subclasses override superclass methods

```
class SynchronizedQueue<E>
      extends BusySynchronizedQueue<E>
{
  ...
  public boolean offer(E e)
  { ... }

  public E poll()
  { ... }

  public boolean isFull()
  { ... }

  ...
```

These methods will not be synchronized unless the implementation decides to synchronize them explicitly

Java Synchronized Methods

- **Pros of synchronized methods**

See stackoverflow.com/questions/574240/is-there-an-advantage-to-use-a-synchronized-method-instead-of-a-synchronized-blo/574525#574525

Java Synchronized Methods

- **Pros of synchronized methods**
  - Synchronized methods can be identified by examining the method interfaces

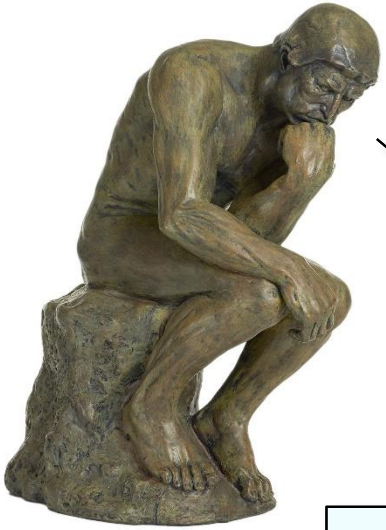
```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E>
{
  ...
  public synchronized boolean
                      offer(E e)
  { ... }

  public synchronized E poll()
  { ... }

  public synchronized boolean isFull()
  { ... }

  ...
```

See stackoverflow.com/questions/574240/is-there-an-advantage-to-use-a-synchronized-method-instead-of-a-synchronized-blo/574525#574525

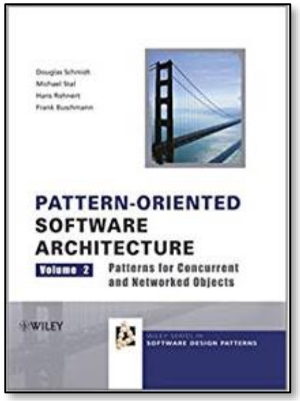
Java Synchronized Methods

- **Pros of synchronized methods**
  - Synchronized methods can be identified by examining the method interfaces
  - The "method" is the unit of synchronization

```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E>
{
  ...
  public synchronized boolean
                          offer(E e)
  { ... }

  public synchronized E poll()
  { ... }

  public synchronized boolean isFull()
  { ... }

  ...
```



It's easier to reason about method-oriented synchronization



Douglas Schmidt
Michael Stal
Hans Rohnert
Frank Buschmann

PATTERN-ORIENTED SOFTWARE ARCHITECTURE
Volume 2 Patterns for Concurrent and Networked Objects

WILEY

WILEY SERIES IN SOFTWARE DESIGN PATTERNS

See www.dre.vanderbilt.edu/~schmidt/PDF/monitor.pdf

Java Synchronized Methods

- **Pros of synchronized methods**
  - Synchronized methods can be identified by examining the method interfaces
  - The "method" is the unit of synchronization
  - The syntax is compact

```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E>
{
  ...
  public synchronized boolean
                          offer(E e)
  { if (!isFull()) {
        mList.add(e);
        return true;
    } else
        return false;
  }

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

  public synchronized boolean isFull()
  { return mList.size() == mCapacity; }
  ...
```

The code is more legible since there are no explicit synchronization statements

Java Synchronized Methods

- **Pros of synchronized methods**
  - Synchronized methods can be identified by examining the method interfaces
  - The "method" is the unit of synchronization
  - The syntax is compact
  - Support reentrant mutex semantics

```
class BusySynchronizedQueue<E>
      implements SimpleBlockingQueue<E>
{
  ...
  public synchronized boolean
                          offer(E e)
  { if (!isFull()) {
        mList.add(e);
        return true;
    } else
        return false;
  }

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

  public synchronized boolean isFull()
  { return mList.size() == mCapacity; }
  ...
```

isFull() reacquires the intrinsic lock when called from offer()

See en.wikipedia.org/wiki/Reentrant_mutex

Java Synchronized Methods

- **Cons of synchronized methods**

See stackoverflow.com/questions/574240/is-there-an-advantage-to-use-a-synchronized-method-instead-of-a-synchronized-blo/574525#574525

Java Synchronized Methods

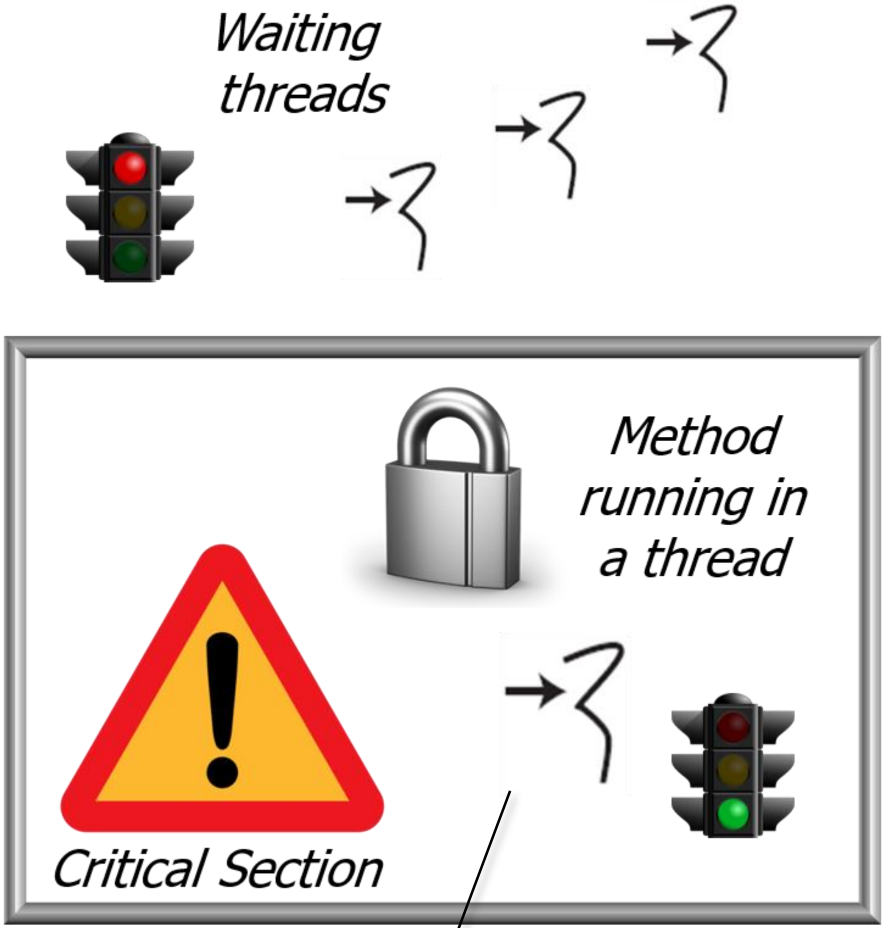
- **Cons of synchronized methods**
  - Synchronizes on the "intrinsic lock" (this), so it is possible for other objects to synchronize with it too

```
BusySynchronizedQueue<Long> q
  = new BusySynchronizedQueue<>();

// Thread T1
while (q.isEmpty())
   ...

// Thread T2
synchronized(q) {
   ...
}
```

T2 will keep Thread T1 from accessing `q`'s critical section

Java Synchronized Methods

- **Cons of synchronized methods**
  - Synchronizes on the "intrinsic lock" (this), so it is possible for other objects to synchronize with it too
  - The granularity of synchronization is "coarse-grained"



Synchronization is a per-object & per-method basis

End of Java Monitor Objects: Synchronized Methods