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 how the synchronized methods/statements provided by Java built-in monitor objects 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](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>
    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](https://github.com/douglascraigschmidt/POSA/tree/master/ex/M3/Queues/BusySynchronizedQueue)
The BusySynchronizedQueue class showcases Java built-in synchronization mechanisms. 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

```java
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](en.wikipedia.org/wiki/Race_condition)
Java Synchronized Methods

- The `BusySynchronizedQueue` class showcases Java built-in synchronization mechanisms.

```java
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

```java
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](http://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.

```java
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.

```java
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

```java
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

```java
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](http://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

```java
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
Pros of synchronized methods

Synchronized methods can be identified by examining the method interfaces.

```java
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](stackoverflow.com/questions/574240/is-there-an-advantage-to-use-a-synchronized-method-instead-of-a-synchronized-blo/574525#574525)
Pros of synchronized methods

- Synchronized methods can be identified by examining the method interfaces
- The “method” is the unit of synchronization

Java Synchronized Methods

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

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

```java
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*
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;
    }
    ...

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-block/574525#574525](https://stackoverflow.com/questions/574240/is-there-an-advantage-to-use-a-synchronized-method-instead-of-a-synchronized-block/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

```java
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”
End of Java Monitor Object
Synchronized Methods