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](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 BoundedQueue<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](github.com/douglascraigschmidt/POSA/tree/master/ex/M3/Queues/BusySynchronizedQueue)
Java Synchronized Methods

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

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

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

    ...  
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingQueue.html](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingQueue.html)
Java Synchronized Methods

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

```java
class BusySynchronizedQueue<E> implements BoundedQueue<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)
class BusySynchronizedQueue<E> implements BoundedQueue<E> {
    private LinkedList<E> mList;
    private int mCapacity;

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

    //...
Java Synchronized Methods

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

```java
class BusySynchronizedQueue<E>
    implements BoundedQueue<E> {

    ... public synchronized boolean offer(E e) {
        ... }

    public synchronized E poll() {
        ... }

    public synchronized boolean isEmpty() {
        ... }

    ... 
```

See [docs.oracle.com/javase/tutorial/essential/concurrency/syncmeth.html](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 BoundedQueue<E> {
    ...
    public synchronized boolean offer(E e) {
        ... }
    
    public synchronized E poll() {
        ... }
    
    public synchronized boolean isEmpty() {
        ... }
    
    ...
```
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 BoundedQueue<E> {
    ...
    public synchronized boolean offer(E e) {
        ...
    }
    public synchronized E poll() {
        ...
    }
    public synchronized boolean isEmpty() {
        ...
    }
    ...
}
```

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 BoundedQueue<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 isEmpty() {
        return mList.size() == 0;
    }

    ...
```
Java Synchronized Methods

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

```java
class BusySynchronizedQueue<E>
    implements BoundedQueue<E> {
    ...
    public synchronized boolean offer(E e) {
        ... } 
    
    public synchronized E poll() {
        ... }
    
    public synchronized boolean isEmpty() {
        ... }
    ...

    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

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

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 BoundedQueue<E> {
    ...
    public synchronized boolean offer(E e) {
        ...
    }

    public synchronized E poll() {
        ...
    }

    public synchronized boolean isEmpty() {
        ...
    }
    ...
}
```

See https://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

```java
class BusySynchronizedQueue<E> implements BoundedQueue<E> {
    ...
    public synchronized boolean offer(E e) {
        ...
    }
    public synchronized E poll() {
        ...
    }
    public synchronized boolean isEmpty() {
        ...
    }
    ...
}
```

It’s easier to reason about method-oriented synchronization

See [www.dre.vanderbilt.edu/~schmidt/PDF/monitor.pdf](http://www.dre.vanderbilt.edu/~schmidt/PDF/monitor.pdf)
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 Synchronized Methods

```java
class BusySynchronizedQueue<E> implements BoundedQueue<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 isEmpty() {
        return mList.size() == 0;
    }
    ...
}
```
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

```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”.

*Waiting threads*  
Critical Section  
Method running in a thread  
Synchronization is a per-object & per-method basis
End of Java Monitor Objects: Synchronized Methods