

# Java Monitor Objects: Motivating Example



**Douglas C. Schmidt**

**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**

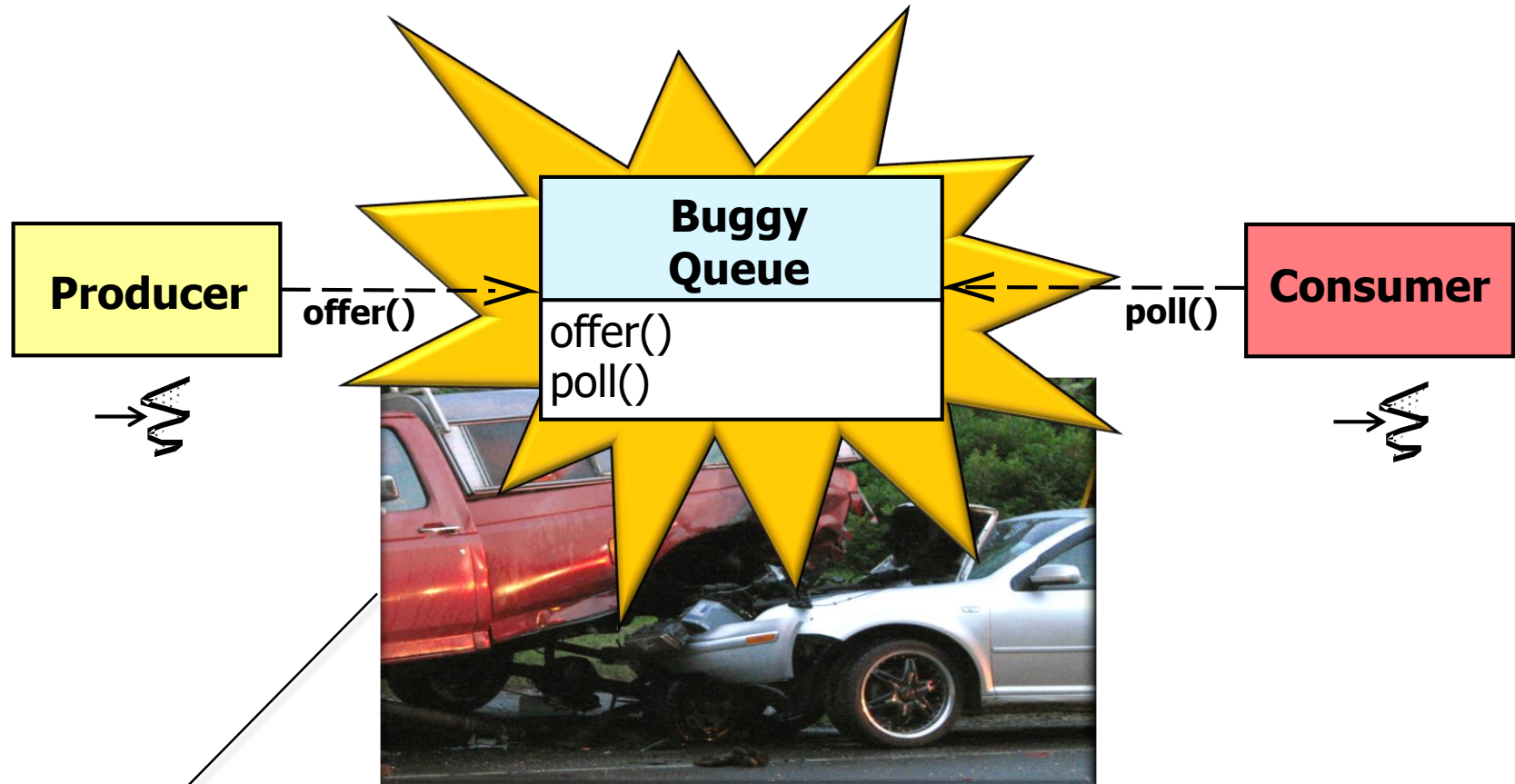
**[www.dre.vanderbilt.edu/~schmidt](http://www.dre.vanderbilt.edu/~schmidt)**

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



# Learning Objectives in this Part of the Lesson

- Understand synchronization problems in a buggy concurrent Java program



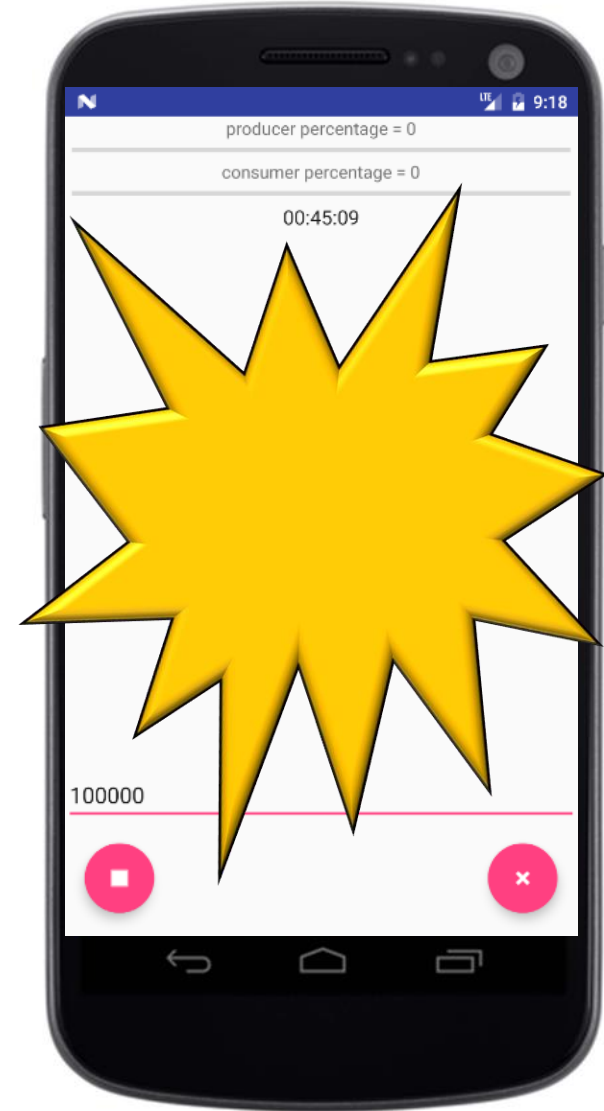
*Concurrent calls to `offer()` & `poll()` corrupt internal state in the `BuggyQueue` fields*

---

# A Buggy Producer/ Consumer App

# A Buggy Producer/Consumer App

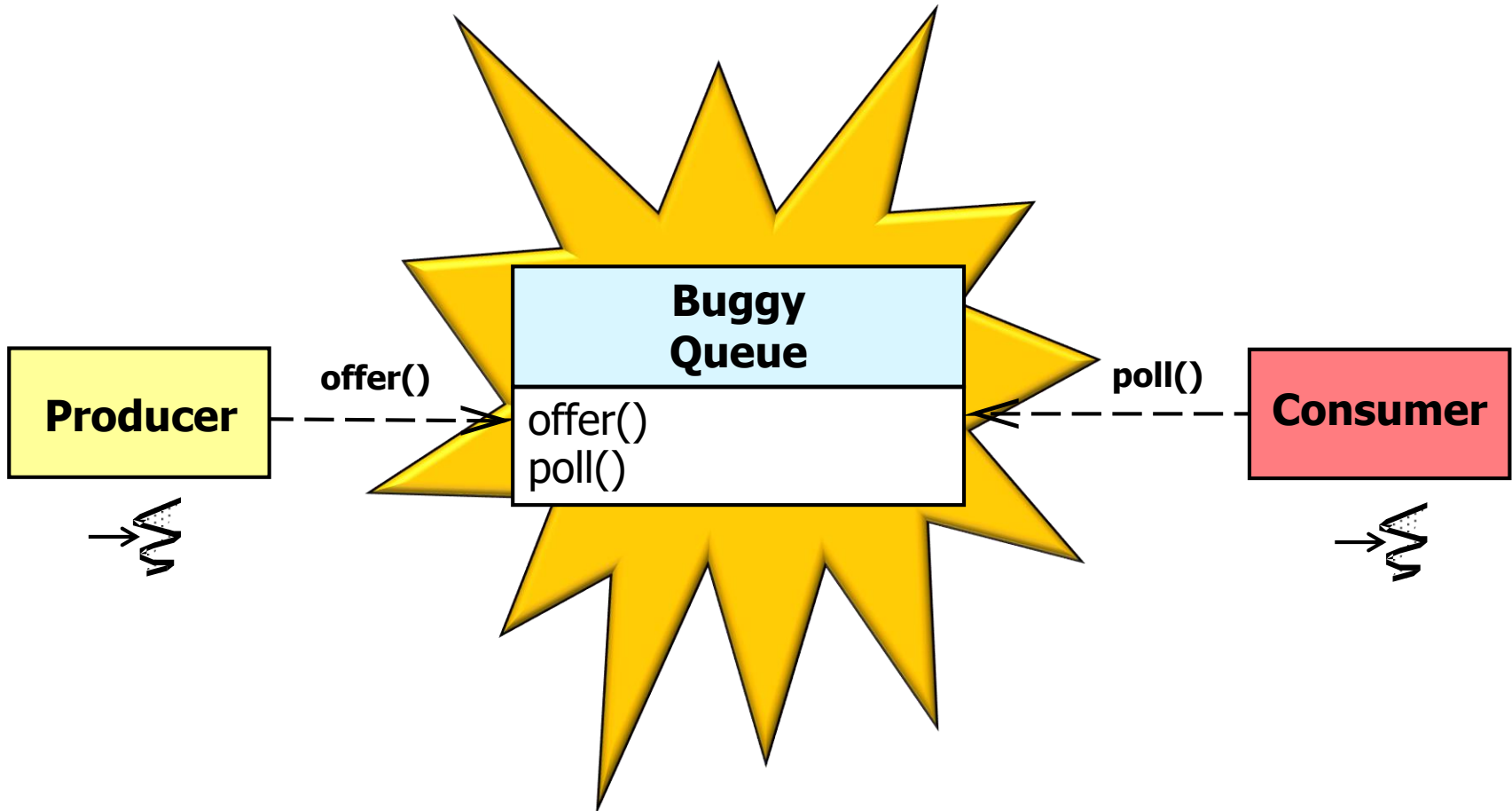
- A concurrent producer/consumer app that attempts to pass messages via an "BuggyQueue" class



See [github.com/douglasraigschmidt/POSA/tree/master/ex/M3/Queues/BuggyQueue](https://github.com/douglasraigschmidt/POSA/tree/master/ex/M3/Queues/BuggyQueue)

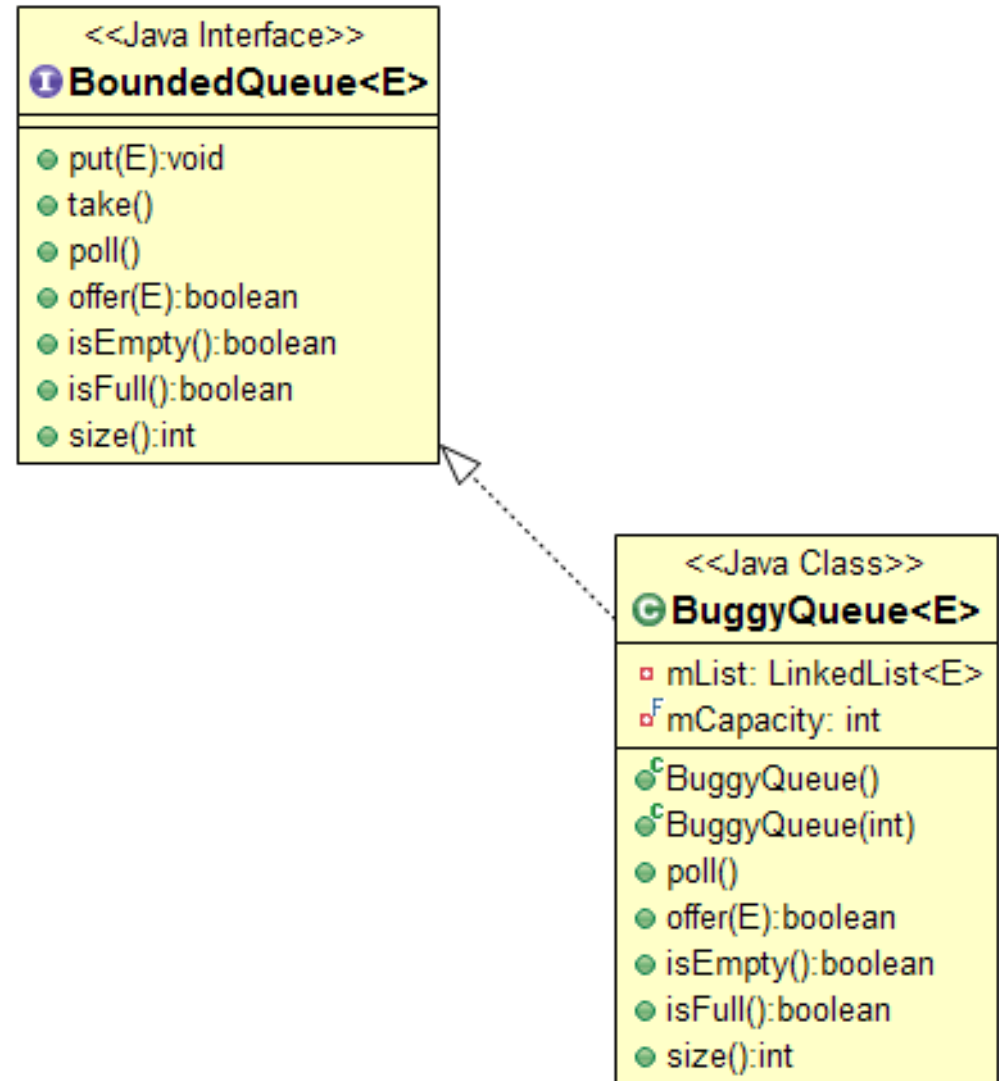
# A Buggy Producer/Consumer App

- The BuggyQueue class is modeled on the Java ArrayBoundedQueue class



# A Buggy Producer/Consumer App

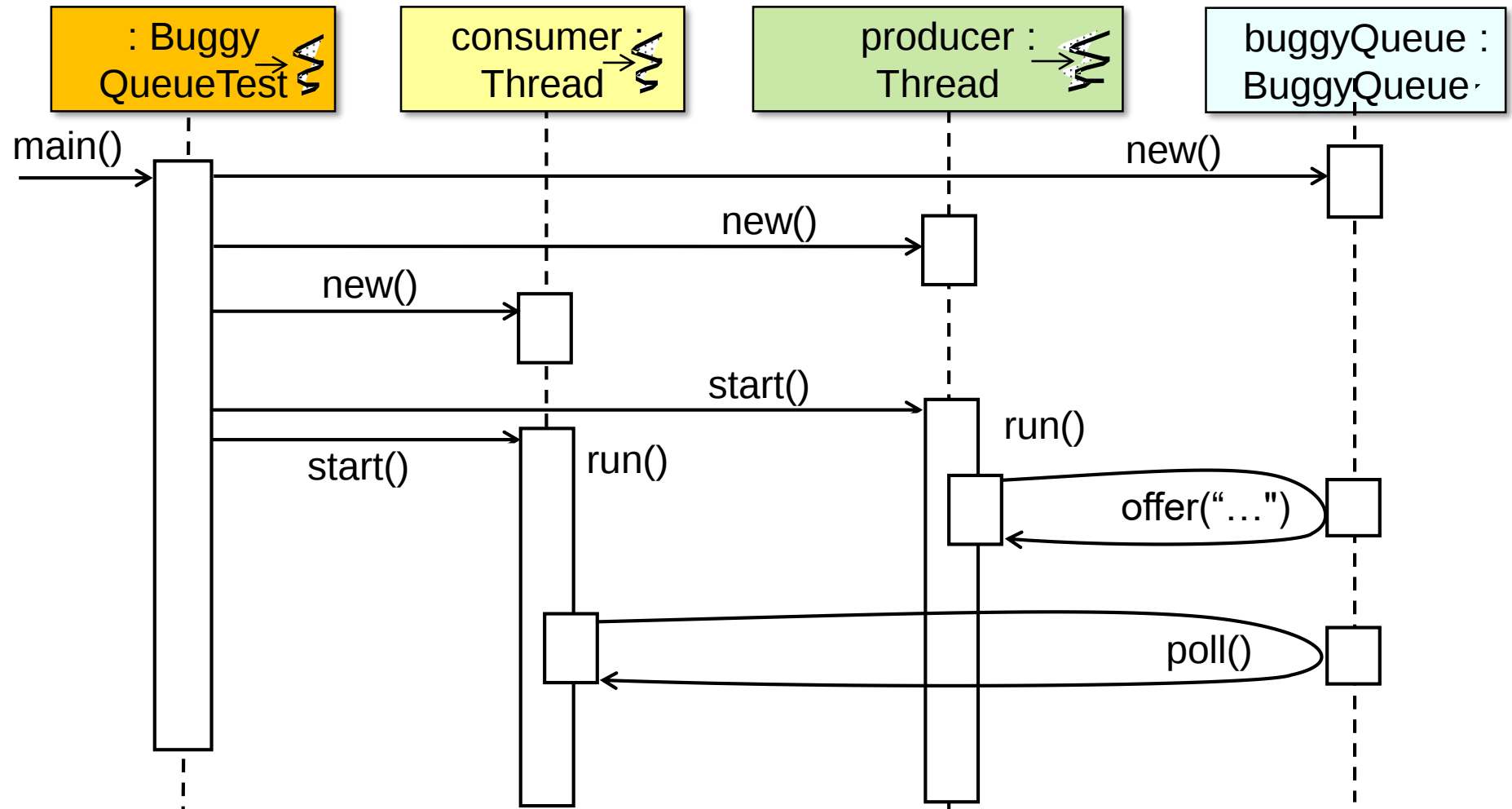
- UML class diagram showing the design of the BuggyQueue



See [github.com/douglasraigschmidt/POSA/tree/master/ex/M3/Queues/BuggyQueue/app/src/main/java/edu/vandy/buggyqueue/model](https://github.com/douglasraigschmidt/POSA/tree/master/ex/M3/Queues/BuggyQueue/app/src/main/java/edu/vandy/buggyqueue/model)

# A Buggy Producer/Consumer App

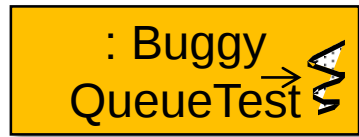
- UML sequence diagram of the BuggyQueue producer/consumer unit test



See [github.com/douglasraigschmidt/POSA/tree/master/ex/M3/Queues/BuggyQueue/app/src/test/java/edu/vandy/buggyqueue](https://github.com/douglasraigschmidt/POSA/tree/master/ex/M3/Queues/BuggyQueue/app/src/test/java/edu/vandy/buggyqueue)

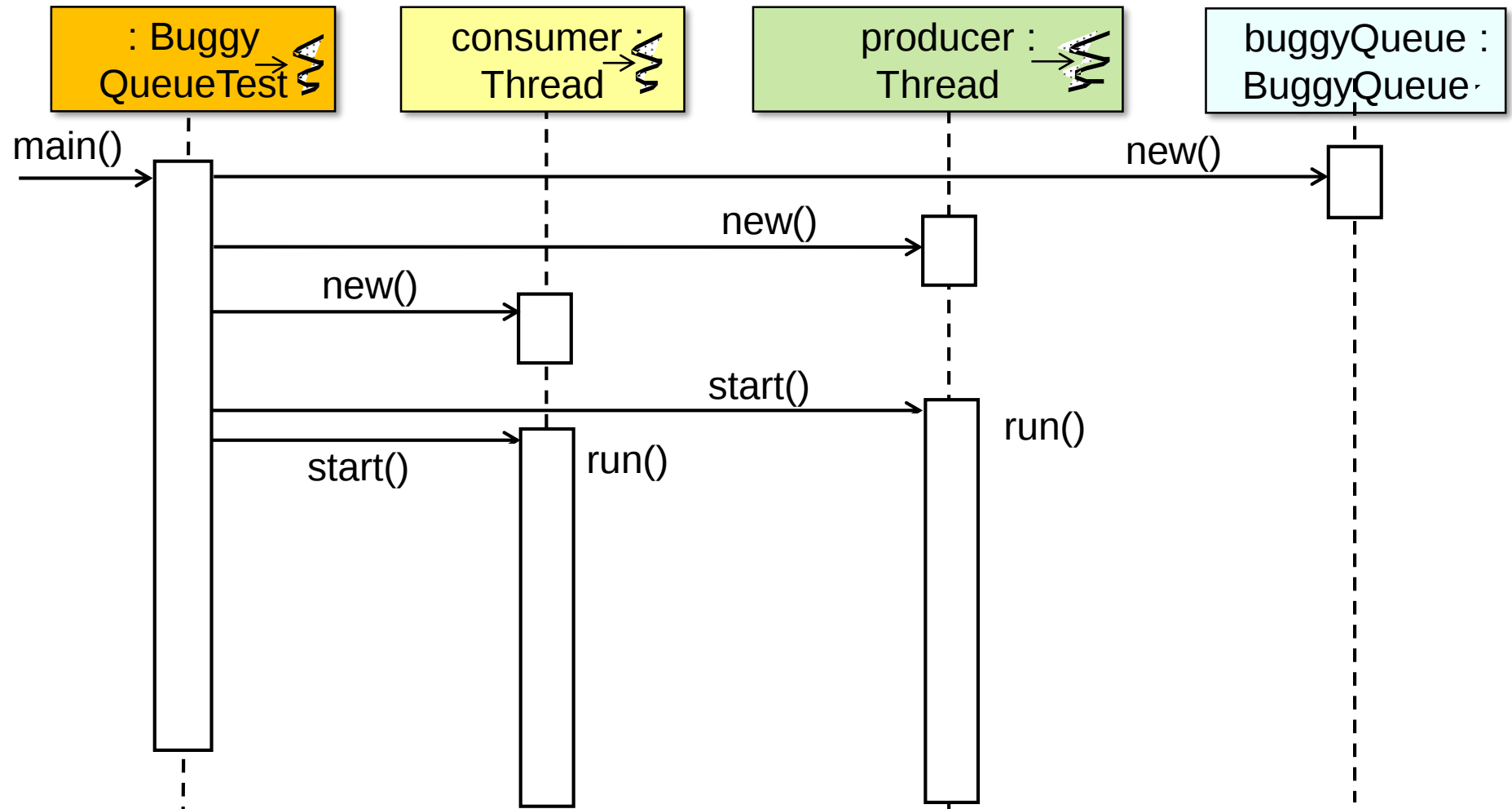
# A Buggy Producer/Consumer App

- UML sequence diagram of the BuggyQueue producer/consumer unit test



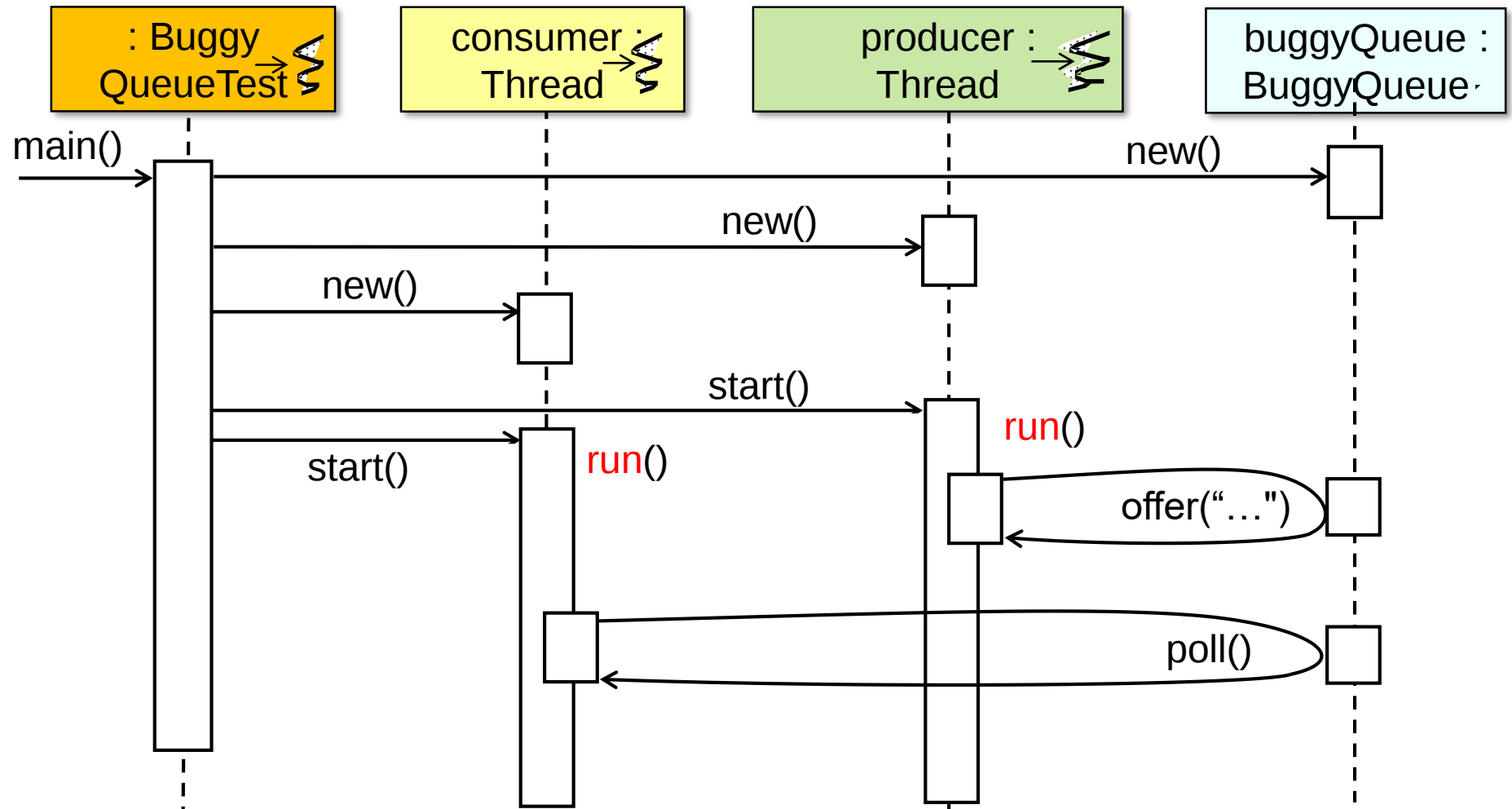
# A Buggy Producer/Consumer App

- UML sequence diagram of the BuggyQueue producer/consumer unit test



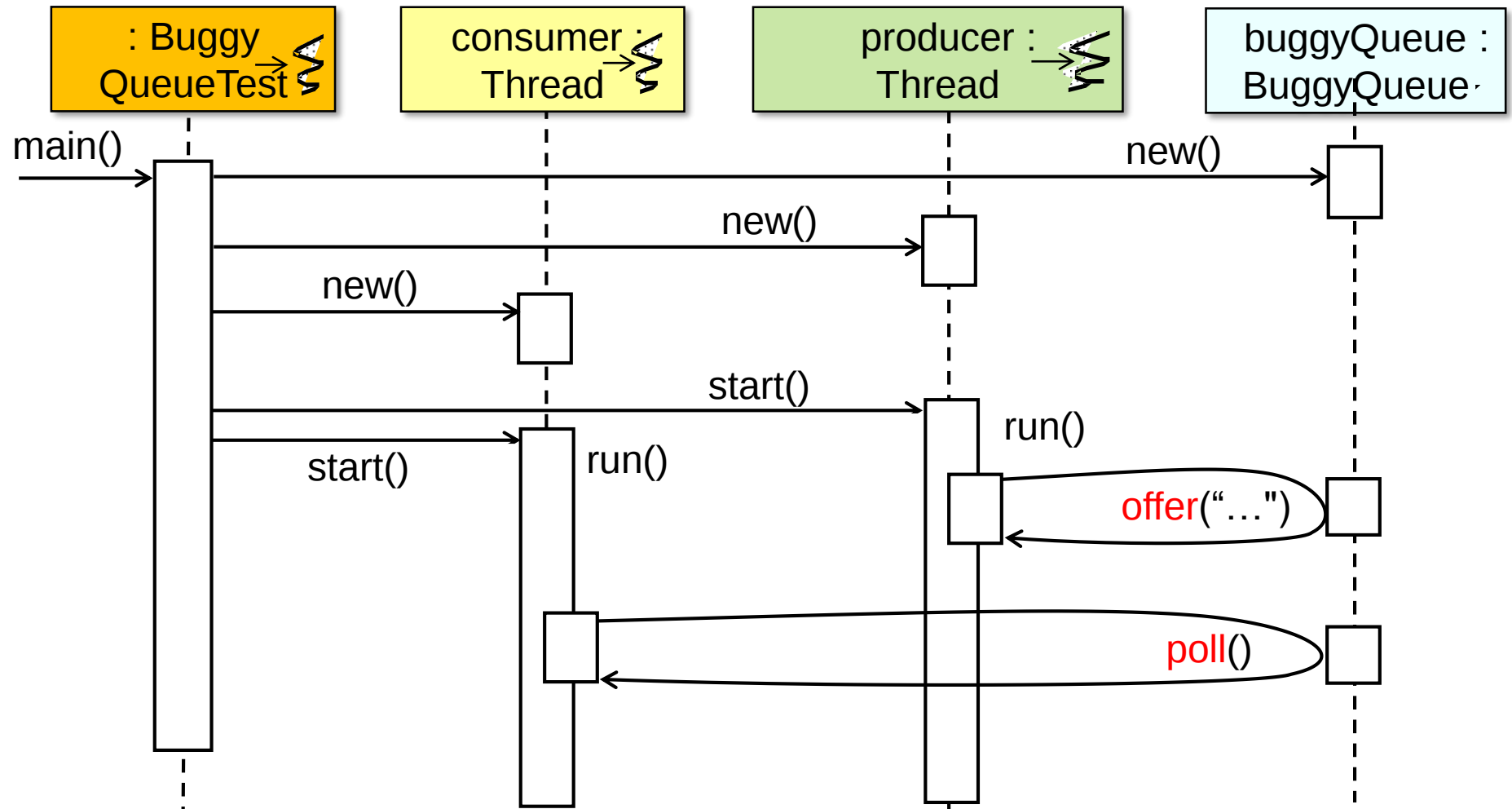
# A Buggy Producer/Consumer App

- UML sequence diagram of the BuggyQueue producer/consumer unit test



# A Buggy Producer/Consumer App

- UML sequence diagram of the BuggyQueue producer/consumer unit test



Since the `offer()` & `poll()` methods aren't synchronized chaos & insanity will result when this app & unit test is run!!

---

# The BuggyQueue Implementation

# The BuggyQueue Implementation

- The BuggyQueue class is a simply wrapper around Java's LinkedList class

```
static class BuggyQueue<E> implements BoundedQueue<E> {  
    private LinkedList<E> mList = new LinkedList<>(); ...
```

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

```
    public E poll() {  
        if (!isEmpty())  
            return mList.remove(0);  
        else  
            return null;  
    }
```

...

|                                                                                                                |  |
|----------------------------------------------------------------------------------------------------------------|--|
| <<Java Class>>                                                                                                 |  |
|  <b>BuggyQueue&lt;E&gt;</b> |  |
|  mList: LinkedList<E>       |  |
|  mCapacity: int             |  |
|  BuggyQueue()               |  |
|  BuggyQueue(int)            |  |
|  poll()                     |  |
|  offer(E): boolean          |  |
|  isEmpty(): boolean       |  |
|  isFull(): boolean        |  |
|  size(): int              |  |

See [github.com/douglasraigschmidt/POSA/tree/master/ex/M3/Queues/BuggyQueue/app/src/main/java/edu/vandy/buggyqueue/model](https://github.com/douglasraigschmidt/POSA/tree/master/ex/M3/Queues/BuggyQueue/app/src/main/java/edu/vandy/buggyqueue/model)

# The BuggyQueue Implementation

- The BuggyQueue class is a simply wrapper around Java's LinkedList class

```
static class BuggyQueue<E> implements BoundedQueue<E> {  
    private LinkedList<E> mList = new LinkedList<>(); ...
```

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

```
    public E poll() {  
        if (!isEmpty())  
            return mList.remove(0);  
        else  
            return null;  
    }  
    ...
```

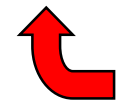
| <<Java Interface>> |                   |
|--------------------|-------------------|
| 1                  | BoundedQueue<E>   |
| ●                  | put(E):void       |
| ●                  | take()            |
| ●                  | poll()            |
| ●                  | offer(E):boolean  |
| ●                  | isEmpty():boolean |
| ●                  | isFull():boolean  |
| ●                  | size():int        |

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)

# The BuggyQueue Implementation

- The BuggyQueue class is a simply wrapper around Java's LinkedList class

```
static class BuggyQueue<E> implements BoundedQueue<E> {  
    private LinkedList<E> mList = new LinkedList<>(); ...
```

 **Linked list  
implementation**

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

```
    public E poll() {  
        if (!isEmpty())  
            return mList.remove(0);  
        else  
            return null;  
    }  
    ...
```

See [docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html)

# The BuggyQueue Implementation

---

- The BuggyQueue class is a simply wrapper around Java's LinkedList class

```
static class BuggyQueue<E> implements BoundedQueue<E> {  
    private LinkedList<E> mList = new LinkedList<>(); ...
```

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



**Non-synchronized public  
methods**

```
    public E poll() {  
        if (!isEmpty())  
            return mList.remove(0);  
        else  
            return null;  
    }  
    ...
```

# The BuggyQueue Implementation

- The BuggyQueue class is a simply wrapper around Java's LinkedList class

```
static class BuggyQueue<E> implements BoundedQueue<E> {  
    private LinkedList<E> mList = new LinkedList<>(); ...
```

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

**Add & remove elements  
into/from the queue**



```
public E poll() {  
    if (!isEmpty())  
        return mList.remove(0);  
    else  
        return null;  
}  
...
```



See [en.wikipedia.org/wiki/Robot\\_B-9](https://en.wikipedia.org/wiki/Robot_B-9)

---

# End of Java Monitor Objects: Motivating Example