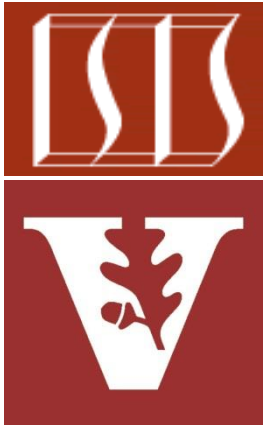


# Java “Happens-Before” Relationships: Examples



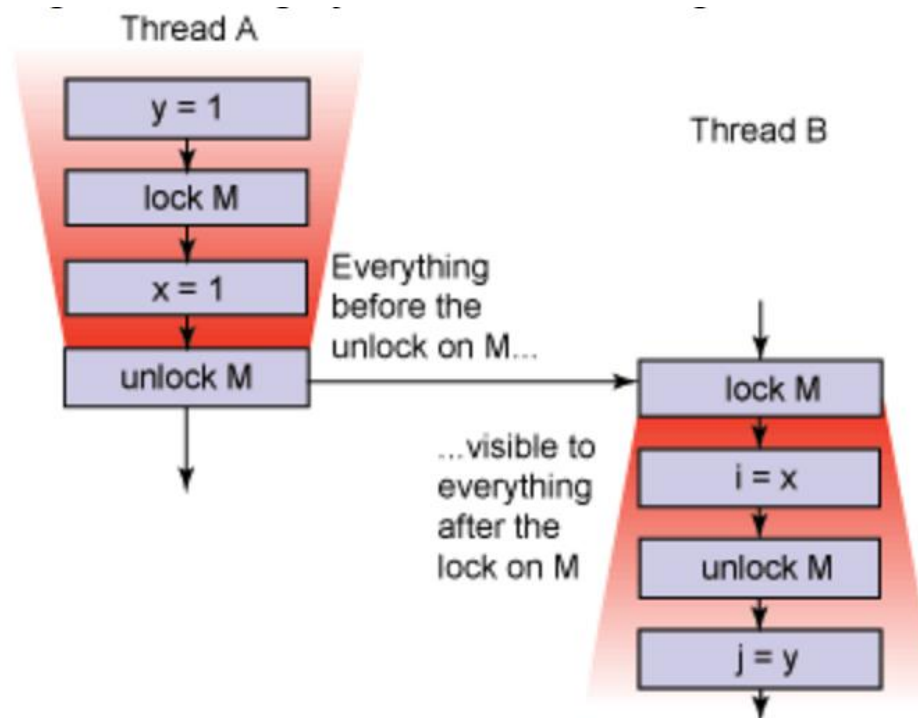
**Douglas C. Schmidt**  
**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**  
**[www.dre.vanderbilt.edu/~schmidt](http://www.dre.vanderbilt.edu/~schmidt)**

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



# Learning Objectives in this Part of the Lesson

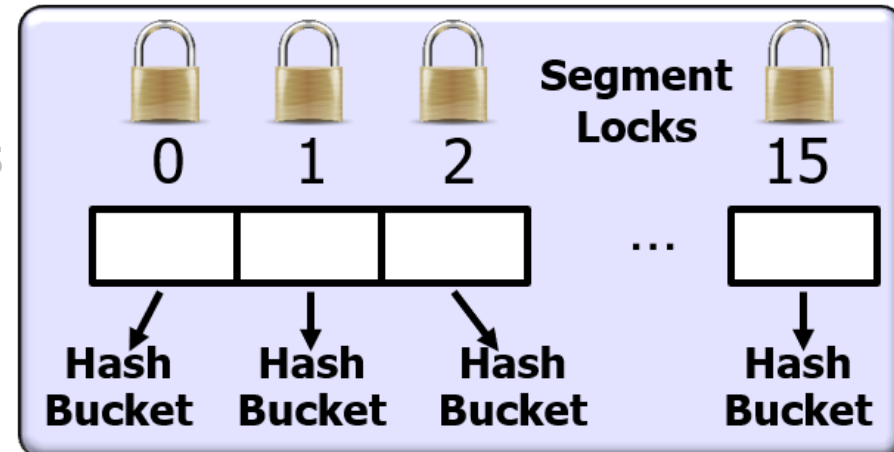
- Understand what “happens-before” relationships mean in Java
- Recognize how Java Thread methods support “happens-before” relationships



| <<Java Class>>          |   |
|-------------------------|---|
| Thread                  |   |
| yield():void            | S |
| currentThread():Thread  | S |
| sleep(long):void        | S |
| sleep(long,int):void    | S |
| Thread()                | C |
| Thread(Runnable)        | C |
| Thread(String)          | C |
| start():void            |   |
| run():void              |   |
| exit():void             |   |
| interrupt():void        |   |
| interrupted():boolean   | S |
| isInterrupted():boolean |   |
| isAlive():boolean       | F |
| setPriority(int):void   | F |
| getPriority():int       | F |
| join(long):void         | F |
| join(long,int):void     | F |
| join():void             | F |
| setDaemon(boolean):void | F |
| isDaemon():boolean      | F |

# Learning Objectives in this Part of the Lesson

- Understand what “happens-before” relationships mean in Java
- Recognize how Java Thread methods support “happens-before” relationships
- Know how Java collections support “happens-before” relationships



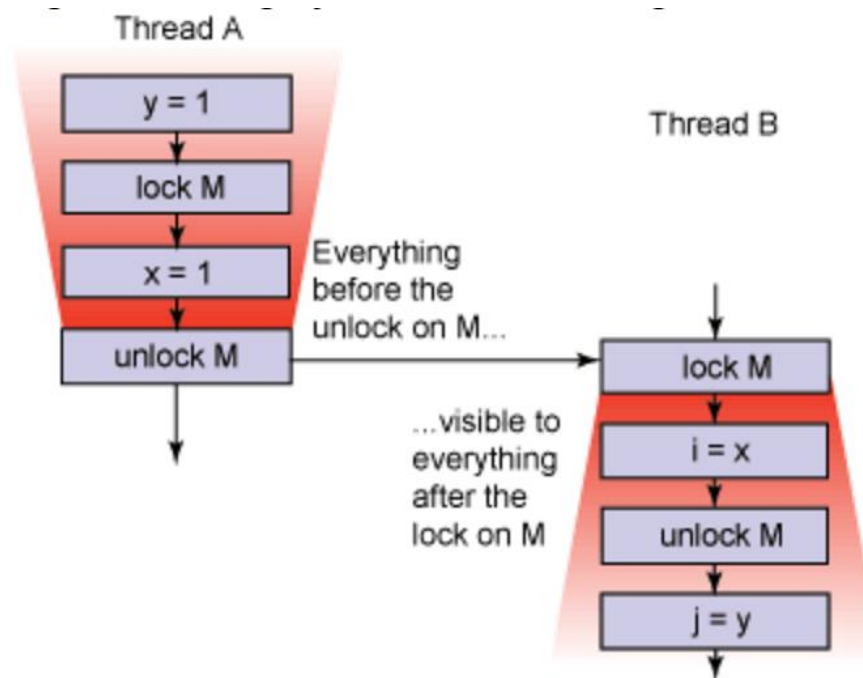
**ConcurrentHashMap**

---

# Java Thread “Happens-Before” Relationships

# Java Thread "Happens-Before" Relationships

- Methods in the Java Thread class establish "happen-before" relationships



<<Java Class>>

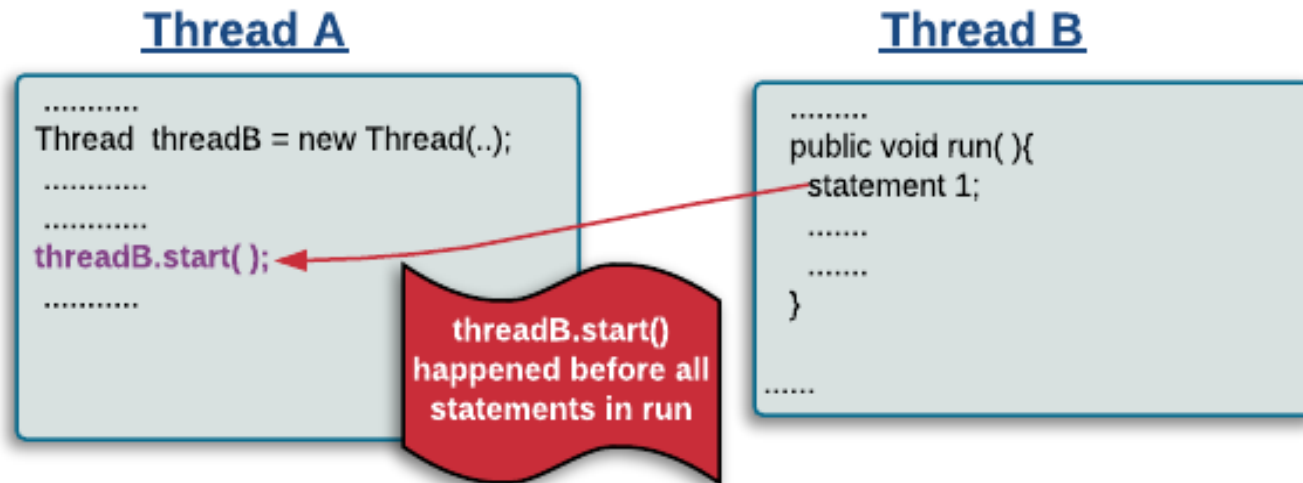
**Thread**

- <sup>S</sup>yield():void
- <sup>S</sup>currentThread():Thread
- <sup>S</sup>sleep(long):void
- <sup>S</sup>sleep(long,int):void
- <sup>C</sup>Thread()
- <sup>C</sup>Thread(Runnable)
- <sup>C</sup>Thread(String)
- start():void
- run():void
- <sup>S</sup>exit():void
- interrupt():void
- <sup>S</sup>interrupted():boolean
- isInterrupted():boolean
- <sup>F</sup>isAlive():boolean
- <sup>F</sup>setPriority(int):void
- <sup>F</sup>getPriority():int
- <sup>F</sup>join(long):void
- <sup>F</sup>join(long,int):void
- <sup>F</sup>join():void
- <sup>F</sup>setDaemon(boolean):void
- <sup>F</sup>isDaemon():boolean

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

# Java Thread "Happens-Before" Relationships

- Methods in the Java Thread class establish "happens-before" relationships
  - Starting a thread "happens-before" the run() hook method of the thread is called



| <<Java Class>> |                         |
|----------------|-------------------------|
| G Thread       |                         |
| S              | yield():void            |
| S              | currentThread():Thread  |
| S              | sleep(long):void        |
| S              | sleep(long,int):void    |
| C              | Thread()                |
| C              | Thread(Runnable)        |
| C              | Thread(String)          |
|                | start():void            |
|                | run():void              |
| ■              | exit():void             |
|                | interrupt():void        |
| S              | interrupted():boolean   |
|                | isInterrupted():boolean |
| F              | isAlive():boolean       |
| F              | setPriority(int):void   |
| F              | getPriority():int       |
| F              | join(long):void         |
| F              | join(long,int):void     |
| F              | join():void             |
| F              | setDaemon(boolean):void |
| F              | isDaemon():boolean      |

# Java Thread "Happens-Before" Relationships

- Methods in the Java Thread class establish "happens-before" relationships
- Starting a thread "happens-before" the run() hook method of the thread is called

```
Thread t1 =  
    new Thread(() ->  
        System.out.println  
            ("hello world"));  
  
t1.start();  
  
...
```

| <<Java Class>> |                         |
|----------------|-------------------------|
| G Thread       |                         |
| S              | yield():void            |
| S              | currentThread():Thread  |
| S              | sleep(long):void        |
| S              | sleep(long,int):void    |
| C              | Thread()                |
| C              | Thread(Runnable)        |
| C              | Thread(String)          |
|                | start():void            |
|                | run():void              |
| ■              | exit():void             |
|                | interrupt():void        |
| S              | interrupted():boolean   |
|                | isInterrupted():boolean |
| F              | isAlive():boolean       |
| F              | setPriority(int):void   |
| F              | getPriority():int       |
| F              | join(long):void         |
| F              | join(long,int):void     |
| F              | join():void             |
| F              | setDaemon(boolean):void |
| F              | isDaemon():boolean      |

# Java Thread "Happens-Before" Relationships

- Methods in the Java Thread class establish "happens-before" relationships
- Starting a thread "happens-before" the run() hook method of the thread is called

```
Thread t1 =  
    new Thread(() ->  
        System.out.println  
            ("hello world"));
```

```
t1.start();
```

```
...
```

*Create & start a new thread*

<<Java Class>>

**Thread**

- yield():void
- currentThread():Thread
- sleep(long):void
- sleep(long,int):void
- Thread()
- Thread(Runnable)
- Thread(String)
- start():void
- run():void
- exit():void
- interrupt():void
- interrupted():boolean
- isInterrupted():boolean
- isAlive():boolean
- setPriority(int):void
- getPriority():int
- join(long):void
- join(long,int):void
- join():void
- setDaemon(boolean):void
- isDaemon():boolean

# Java Thread "Happens-Before" Relationships

- Methods in the Java Thread class establish "happens-before" relationships
- Starting a thread "happens-before" the run() hook method of the thread is called

```
Thread t1 =  
    new Thread( () ->  
                System.out.println  
                ("hello world") );
```

```
t1.start();
```

```
...
```

*This lambda expression plays the role of the run() hook method!*

<<Java Class>>

**Thread**

- <sup>S</sup>yield():void
- <sup>S</sup>currentThread():Thread
- <sup>S</sup>sleep(long):void
- <sup>S</sup>sleep(long,int):void
- <sup>C</sup>Thread()
- <sup>C</sup>Thread(Runnable)
- <sup>C</sup>Thread(String)
- start():void
- run():void
- exit():void
- interrupt():void
- <sup>S</sup>interrupted():boolean
- isInterrupted():boolean
- <sup>F</sup>isAlive():boolean
- <sup>F</sup>setPriority(int):void
- <sup>F</sup>getPriority():int
- <sup>F</sup>join(long):void
- <sup>F</sup>join(long,int):void
- <sup>F</sup>join():void
- <sup>F</sup>setDaemon(boolean):void
- <sup>F</sup>isDaemon():boolean

# Java Thread "Happens-Before" Relationships

- Methods in the Java Thread class establish "happens-before" relationships
- Starting a thread "happens-before" the run() hook method of the thread is called

```
Thread t1 =  
    new Thread( () ->  
                System.out.println  
                ("hello world") );
```

```
t1.start();
```

```
...
```

*The state of thread t1 is consistent & visible before run() begins to execute*

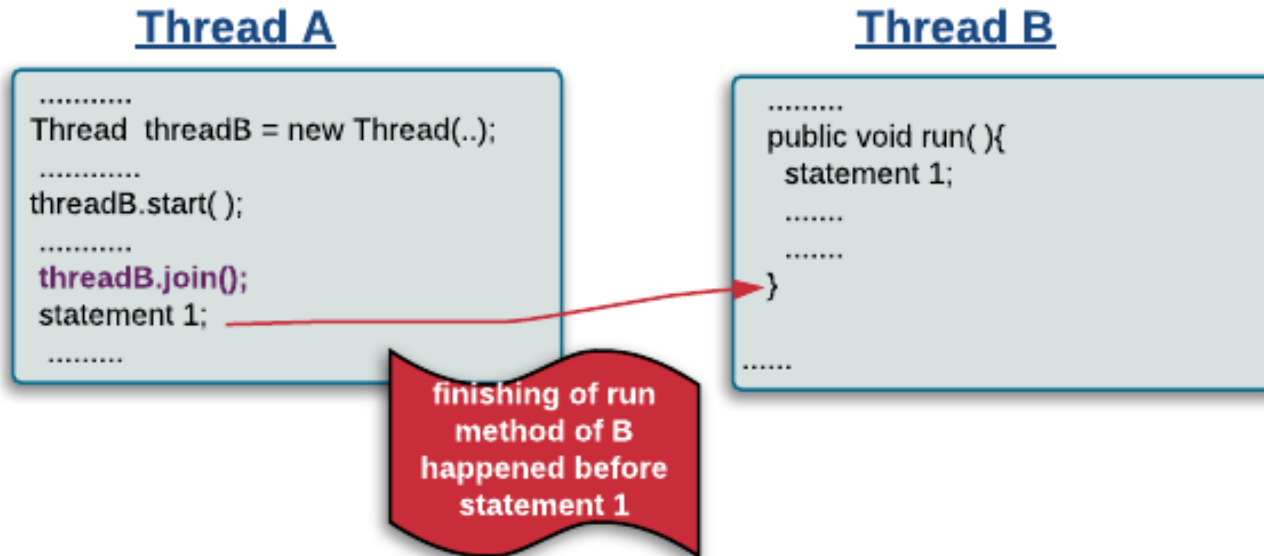
<<Java Class>>

**Thread**

- <sup>S</sup>yield():void
- <sup>S</sup>currentThread():Thread
- <sup>S</sup>sleep(long):void
- <sup>S</sup>sleep(long,int):void
- <sup>C</sup>Thread()
- <sup>C</sup>Thread(Runnable)
- <sup>C</sup>Thread(String)
- start():void
- run():void
- <sup>S</sup>exit():void
- interrupt():void
- <sup>S</sup>interrupted():boolean
- isInterrupted():boolean
- <sup>F</sup>isAlive():boolean
- <sup>F</sup>setPriority(int):void
- <sup>F</sup>getPriority():int
- <sup>F</sup>join(long):void
- <sup>F</sup>join(long,int):void
- <sup>F</sup>join():void
- <sup>F</sup>setDaemon(boolean):void
- <sup>F</sup>isDaemon():boolean

# Java Thread "Happens-Before" Relationships

- Methods in the Java Thread class establish "happens-before" relationships
  - Starting a thread "happens-before" the run() hook method of the thread is called
  - The termination of a thread "happens-before" a join() with the terminated thread



| <<Java Class>> |                         |
|----------------|-------------------------|
| G Thread       |                         |
| S              | yield():void            |
| S              | currentThread():Thread  |
| S              | sleep(long):void        |
| S              | sleep(long,int):void    |
| C              | Thread()                |
| C              | Thread(Runnable)        |
| C              | Thread(String)          |
|                | start():void            |
|                | run():void              |
| ■              | exit():void             |
|                | interrupt():void        |
| S              | interrupted():boolean   |
|                | isInterrupted():boolean |
| F              | isAlive():boolean       |
| F              | setPriority(int):void   |
| F              | getPriority():int       |
| F              | join(long):void         |
| F              | join(long,int):void     |
| F              | join():void             |
| F              | setDaemon(boolean):void |
| F              | isDaemon():boolean      |

# Java Thread "Happens-Before" Relationships

- Methods in the Java Thread class establish "happens-before" relationships
  - Starting a thread "happens-before" the run() hook method of the thread is called
- The termination of a thread "happens-before" a join() with the terminated thread

```
Thread t1 =  
    new Thread(() ->  
        System.out.println  
            ("hello world"));  
  
t1.start();  
  
...  
  
t1.join();
```

| <<Java Class>><br>G Thread |                         |
|----------------------------|-------------------------|
| S                          | yield():void            |
| S                          | currentThread():Thread  |
| S                          | sleep(long):void        |
| S                          | sleep(long,int):void    |
| C                          | Thread()                |
| C                          | Thread(Runnable)        |
| C                          | Thread(String)          |
|                            | start():void            |
|                            | run():void              |
| ■                          | exit():void             |
|                            | interrupt():void        |
| S                          | interrupted():boolean   |
|                            | isInterrupted():boolean |
| F                          | isAlive():boolean       |
| F                          | setPriority(int):void   |
| F                          | getPriority():int       |
| F                          | join(long):void         |
| F                          | join(long,int):void     |
| F                          | join():void             |
| F                          | setDaemon(boolean):void |
| F                          | isDaemon():boolean      |

# Java Thread "Happens-Before" Relationships

- Methods in the Java Thread class establish "happens-before" relationships
  - Starting a thread "happens-before" the run() hook method of the thread is called
- The termination of a thread "happens-before" a join() with the terminated thread

```
Thread t1 =  
    new Thread( () ->  
        System.out.println  
            ("hello world")) ;  
  
t1.start() ;  
  
...  
  
t1.join() ;
```



*Thread t1 terminates after its lambda expression run() processing completes*

<<Java Class>>

**Thread**

- <sup>S</sup>yield():void
- <sup>S</sup>currentThread():Thread
- <sup>S</sup>sleep(long):void
- <sup>S</sup>sleep(long,int):void
- <sup>C</sup>Thread()
- <sup>C</sup>Thread(Runnable)
- <sup>C</sup>Thread(String)
- start():void
- run():void
- <sup>■</sup>exit():void
- interrupt():void
- <sup>S</sup>interrupted():boolean
- isInterrupted():boolean
- <sup>F</sup>isAlive():boolean
- <sup>F</sup>setPriority(int):void
- <sup>F</sup>getPriority():int
- <sup>F</sup>join(long):void
- <sup>F</sup>join(long,int):void
- <sup>F</sup>join():void
- <sup>F</sup>setDaemon(boolean):void
- <sup>F</sup>isDaemon():boolean

# Java Thread "Happens-Before" Relationships

- Methods in the Java Thread class establish "happens-before" relationships
  - Starting a thread "happens-before" the run() hook method of the thread is called
- The termination of a thread "happens-before" a join() with the terminated thread

```
Thread t1 =  
    new Thread(() ->  
        System.out.println  
            ("hello world"));  
  
t1.start();  
  
...  
  
t1.join();
```



*The thread waiting on join() only resumes  
it's processing after the thread t1 terminates*

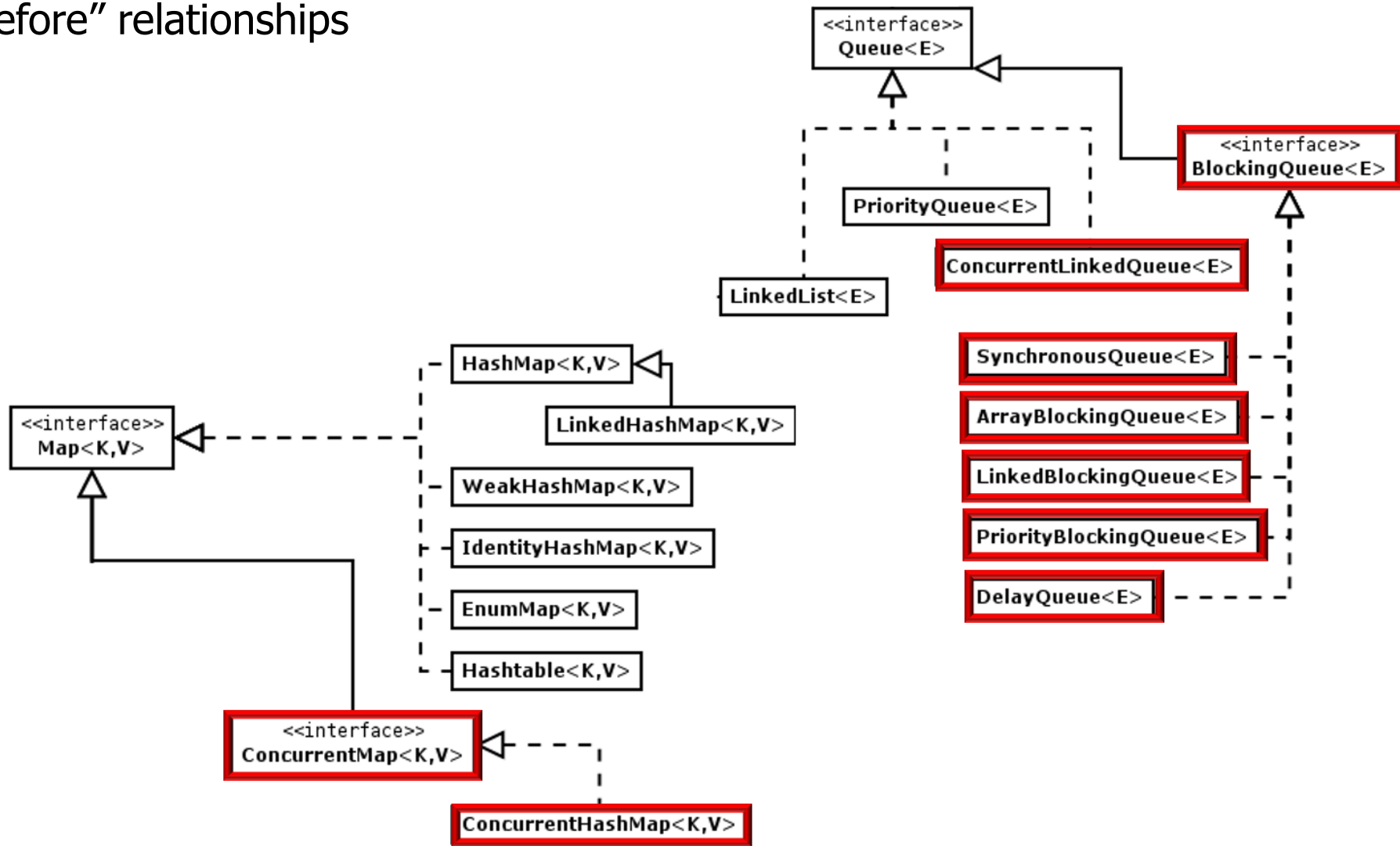
| <<Java Class>> |                         |
|----------------|-------------------------|
| G Thread       |                         |
| S              | yield():void            |
| S              | currentThread():Thread  |
| S              | sleep(long):void        |
| S              | sleep(long,int):void    |
| C              | Thread()                |
| C              | Thread(Runnable)        |
| C              | Thread(String)          |
|                | start():void            |
|                | run():void              |
| ■              | exit():void             |
|                | interrupt():void        |
| S              | interrupted():boolean   |
|                | isInterrupted():boolean |
| F              | isAlive():boolean       |
| F              | setPriority(int):void   |
| F              | getPriority():int       |
| F              | join(long):void         |
| F              | join(long,int):void     |
| F              | join():void             |
| F              | setDaemon(boolean):void |
| F              | isDaemon():boolean      |

---

# Java Collections “Happens-Before” Relationships

# Java Collections "Happens-Before" Relationships

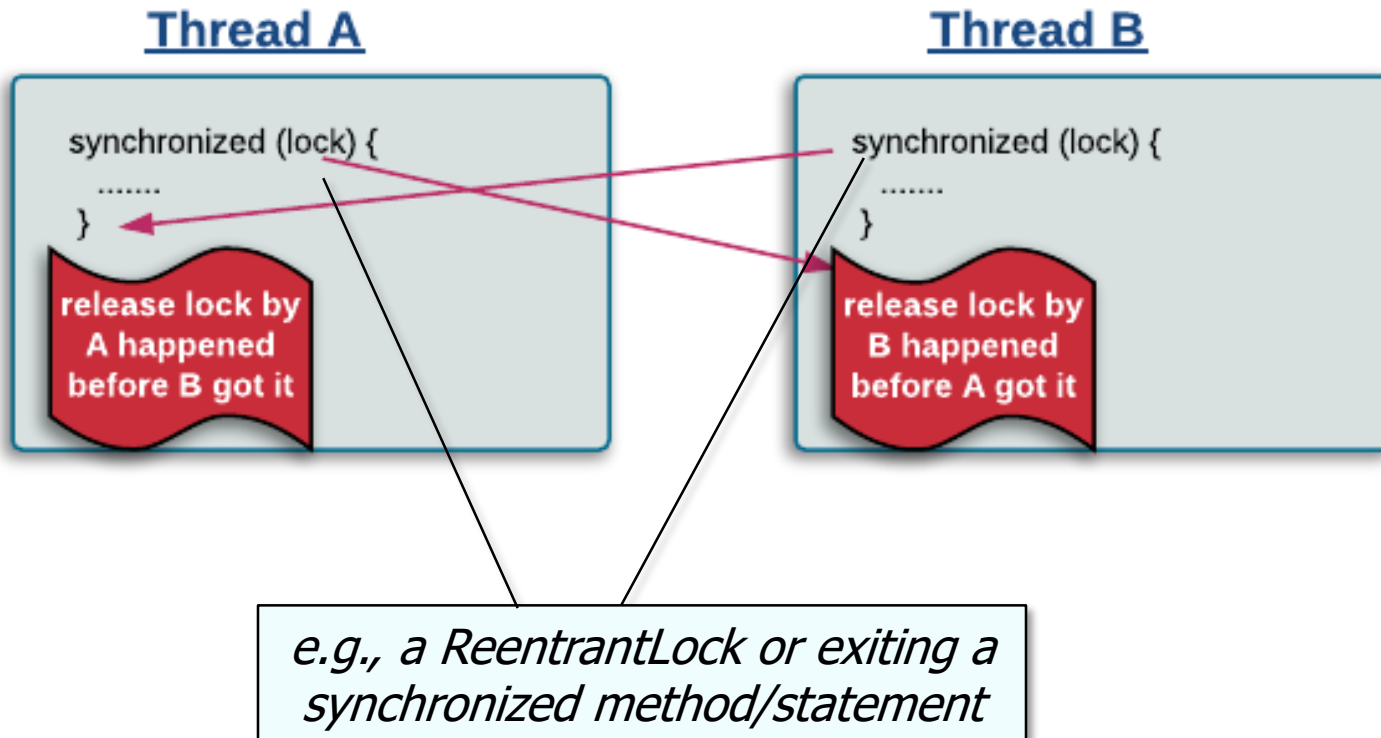
- Methods in java.util.concurrent package classes also establish "happen-before" relationships



See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/package-summary.html#MemoryVisibility](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/package-summary.html#MemoryVisibility)

# Java Collections “Happens-Before” Relationships

- Methods in `java.util.concurrent` package classes also establish “happen-before” relationships
- The release of a monitor lock “happens-before” every subsequent acquire on the same lock



See [www.logicbig.com/tutorials/core-java-tutorial/java-multi-threading/happens-before.html](http://www.logicbig.com/tutorials/core-java-tutorial/java-multi-threading/happens-before.html)

# Java Collections “Happens-Before” Relationships

- Methods in `java.util.concurrent` package classes also establish “happens-before” relationships
- The release of a monitor lock “happens-before” every subsequent acquire on the same lock

```
class ArrayBlockingQueue<E>
... { ...
public void put(E e) ... {
...
final ReentrantLock lock =
    this.lock;
lock.lockInterruptibly();
try { ...
} finally {
    lock.unlock();
}
}
```

```
class ArrayBlockingQueue<E>
... { ...
public E take() ... {
    final ReentrantLock lock
        = this.lock;
    lock.lockInterruptibly();
    try { ...
    } finally {
        lock.unlock();
    }
}
```

# Java Collections "Happens-Before" Relationships

- Methods in `java.util.concurrent` package classes also establish "happens-before" relationships
- The release of a monitor lock "happens-before" every subsequent acquire on the same lock

```
class ArrayBlockingQueue<E>
... { ...
public void put(E e) ... {
    ...
    final ReentrantLock lock =
        this.lock;
    lock.lockInterruptibly();
    try { ...
    } finally {
        lock.unlock();
    }
}
```

```
class ArrayBlockingQueue<E>
... { ...
public E take() ... {
    final ReentrantLock lock
        = this.lock;
    lock.lockInterruptibly();
    try { ...
    } finally {
        lock.unlock();
    }
}
```

*Consider the put() & take() methods in ArrayBlockingQueue*

See earlier lessons on "Java ReentrantLock" & "Java ConditionObject"

# Java Collections "Happens-Before" Relationships

- Methods in `java.util.concurrent` package classes also establish "happens-before" relationships
- The release of a monitor lock "happens-before" every subsequent acquire on the same lock

```
class ArrayBlockingQueue<E>
... { ...
public void put(E e) ... {
...
final ReentrantLock lock =
    this.lock;
lock.lockInterruptibly();
try { ...
} finally {
    lock.unlock();
}
}
```

```
class ArrayBlockingQueue<E>
... { ...
public E take() ... {
    final ReentrantLock lock
        = this.lock;
    lock.lockInterruptibly();
    try { ...
    } finally {
        lock.unlock();
    }
}
```

*Actions prior to "releasing" the ReentrantLock must happen-before actions subsequent to a successful "acquiring" of this lock*

See earlier lessons on "Java ReentrantLock" & "Java ConditionObject"

# Java Collections “Happens-Before” Relationships

- Methods in `java.util.concurrent` package classes also establish “happen-before” relationships
  - The release of a monitor lock “happens-before” every subsequent acquire on the same lock
- Actions in a thread prior to placing an object into any concurrent collection “happen-before” actions subsequent to the access or removal of that element from the collection in another thread

```
ConcurrentMap concurrentMap = new ConcurrentHashMap();  
  
// Thread t1  
concurrentMap.put("key", "value");  
  
// Thread t2  
Object value = concurrentMap.get("key");
```

# Java Collections “Happens-Before” Relationships

- Methods in `java.util.concurrent` package classes also establish “happen-before” relationships
  - The release of a monitor lock “happens-before” every subsequent acquire on the same lock
- Actions in a thread prior to placing an object into any concurrent collection “happen-before” actions subsequent to the access or removal of that element from the collection in another thread

```
ConcurrentMap concurrentMap = new ConcurrentHashMap();  
  
// Thread t1  
concurrentMap.put("key", "value");  
  
// Thread t2  
Object value = concurrentMap.get("key");
```

*Consider a ConcurrentHashMap that supports concurrent retrievals & high expected concurrency for updates*

# Java Collections “Happens-Before” Relationships

- Methods in `java.util.concurrent` package classes also establish “happen-before” relationships
  - The release of a monitor lock “happens-before” every subsequent acquire on the same lock
- Actions in a thread prior to placing an object into any concurrent collection “happen-before” actions subsequent to the access or removal of that element from the collection in another thread

```
ConcurrentMap concurrentMap = new ConcurrentHashMap();
```

```
// Thread t1
```

```
concurrentMap.put("key", "value");
```

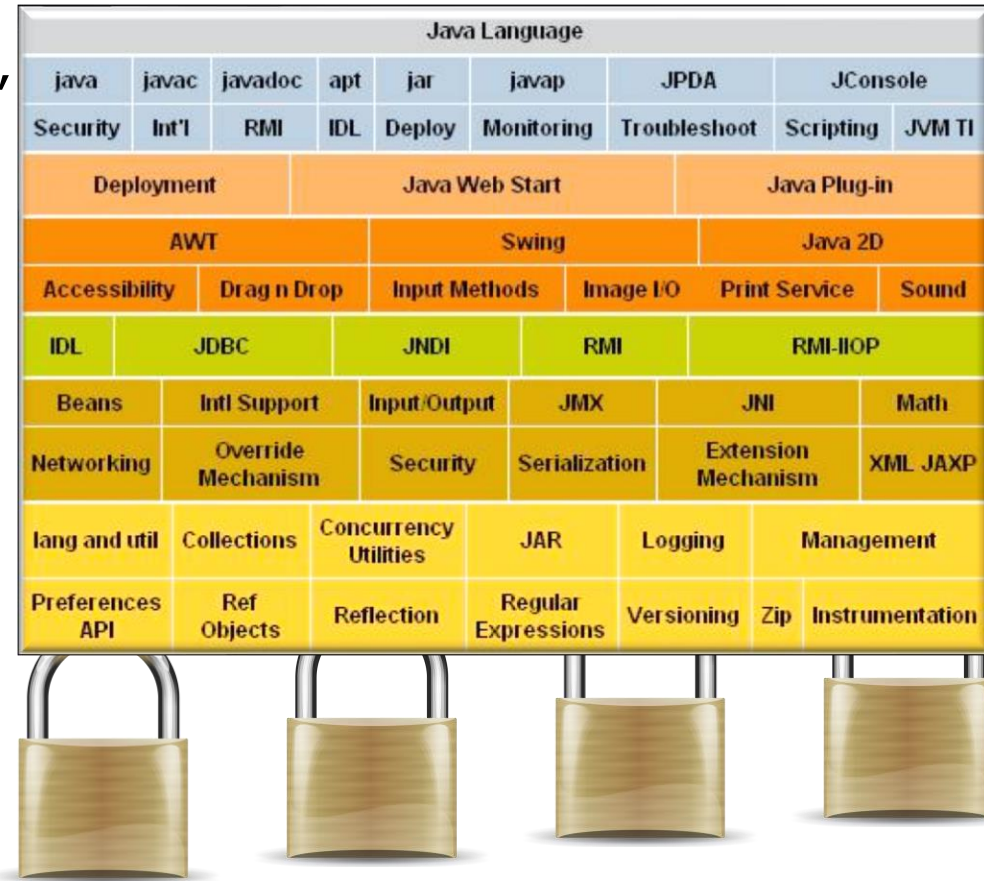
```
// Thread t2
```

```
Object value = concurrentMap.get("key");
```

*Placing a “key/value” element into a `ConcurrentHashMap` must happen-before accessing or removing this element from the map*

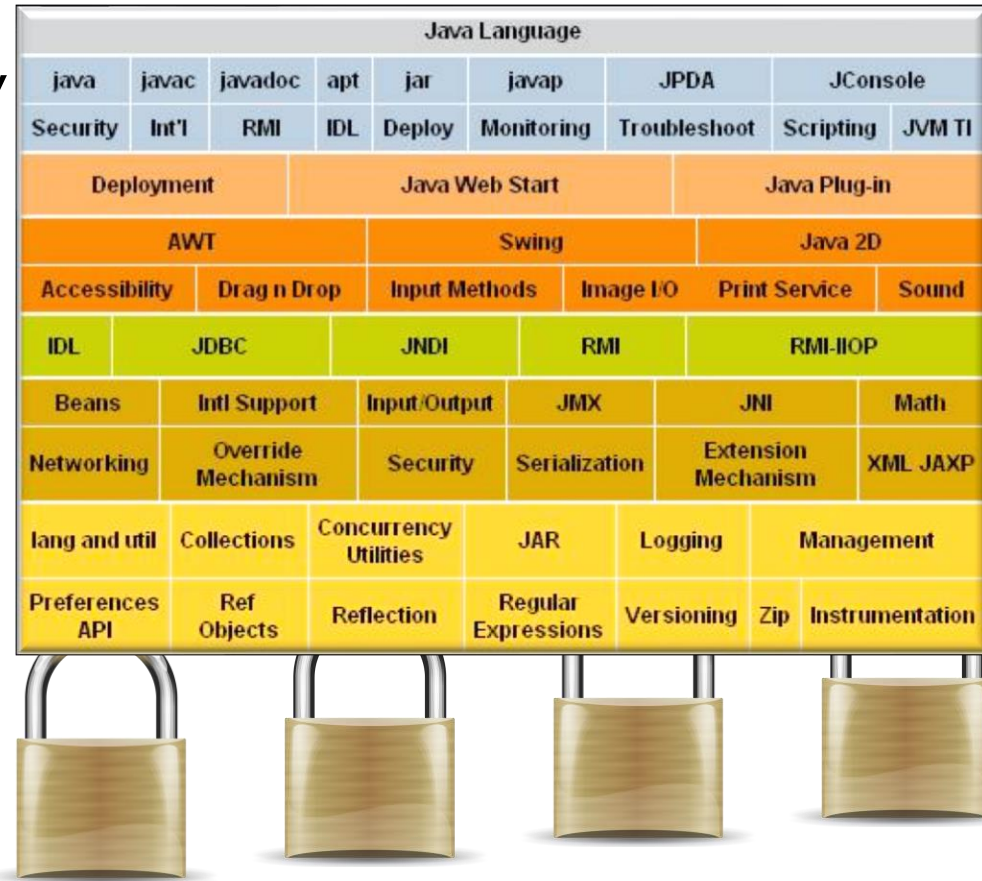
# Java Collections “Happens-Before” Relationships

- Java’s class libraries are responsible for ensuring these “happens-before” relationships are preserved



# Java Collections “Happens-Before” Relationships

- Java’s class libraries are responsible for ensuring these “happens-before” relationships are preserved



You don't need to understand all the nitty-gritty details of Java's memory model – you just need to understand how to use synchronizers properly!

---

# End of “Happens-Before” Relationships: Examples