Example Application of Java StampedLock

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

- Understand the structure, functionality of the Java StampedLock class
- Know the key methods in Java StampedLock
- Recognize how to apply Java StampedLock in practice

```java
class Point {
    // ... other methods

    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock();
        try {
            while (x == 0.0 && y == 0.0) {
                long ws = sl.tryConvertToWriteLock(stamp);
                if (ws != 0L) {
                    stamp = ws;
                    x = newX; y = newY;
                    break;
                } else {
                    sl.unlockRead(stamp);
                    stamp = sl.writeLock();
                }
        }
    }
    // ... other methods
}
```
Learning Objectives in this Part of the Lesson

- Understand the structure, functionality of the Java StampedLock class
- Know the key methods in Java StampedLock
- Recognize how to apply Java StampedLock in practice

class Point {
    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock();
        try {
            while (x == 0.0 && y == 0.0) {
                long ws = sl.tryConvertToWriteLock(stamp);
                if (ws != 0L) {
                    stamp = ws;
                    x = newX; y = newY;
                    break;
                } else {
                    sl.unlockRead(stamp);
                    stamp = sl.writeLock();
                }
            }
        } finally {
            sl.unlockWrite(stamp);
        }
    }
    ...
}

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/locks/StampedLock.html

The discussion is based on examples from the Java StampedLock documentation!
Applying Java Stamped Lock in Practice
Applying Java StampedLock in Practice

- The Point class shows how to program with StampedLock

```java
class Point {

    private double x;
    private double y;

    private final StampedLock sl =
        new StampedLock();

    ...

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/locks/StampedLock.html
```
The Point class shows how to program with StampedLock

```java
class Point {
    private double x;
    private double y;

    private final StampedLock sl =
        new StampedLock();

    ...
}
```

Maintains two-dimensional points
The Point class shows how to program with StampedLock

```java
class Point {
    private double x;
    private double y;

    private final StampedLock sl =
        new StampedLock();
...
```

State that must be protected
The Point class shows how to program with StampedLock

class Point {
    private double x;
    private double y;

    private final StampedLock sl =
        new StampedLock();

    ...

    StampedLock that does
    the protecting
Applying Java StampedLock: Writing Mode
Performing an exclusive write with a StampedLock

```java
class Point {
    ...  
    void move(double deltaX, double deltaY) {
        long stamp = sl.writeLock();
        try {
            x += deltaX;
            y += deltaY;
        } finally {
            sl.unlockWrite(stamp);
        }
    }
    ...  
}
```

This method atomically moves a point to a new location.
Performing an exclusive write with a StampedLock

class Point {
    ...

    void move(double deltaX, double deltaY) {
        long stamp = sl.writeLock();
        try {
            x += deltaX;
            y += deltaY;
        } finally {
            sl.unlockWrite(stamp);
        }
    }
    ...
}
Applying Java StampedLock: Writing Mode

- Performing an exclusive write with a StampedLock

```java
class Point {
    ...

    void move(double deltaX, double deltaY) {
        long stamp = sl.writeLock();
        try {
            x += deltaX;
            y += deltaY;
        } finally {
            sl.unlockWrite(stamp);
        }
    }
    ...

    // Modify the state atomically
```


Performing an exclusive write with a StampedLock

```java
class Point {
    ...

    void move(double deltaX, double deltaY) {
        long stamp = sl.writeLock();
        try {
            x += deltaX;
            y += deltaY;
        } finally {
            sl.unlockWrite(stamp);
        }
    }
    ...
}
```

Release the write lock
Applying Java StampedLock: Optimistic & Reading Mode
Performing a optimistic read with a StampedLock

```java
class Point {
    ...
    double distanceFromOrigin() {
        long stamp = sl.tryOptimisticRead();
        double currX = x, currY = y;
        if (!sl.validate(stamp)) {
            stamp = sl.readLock();
            try {
                currX = x; currY = y;
            } finally {
                sl.unlockRead(stamp);
            }
        }
        return Math.sqrt(currX * currX + currY * currY);
    }
    ...
```
Performing an optimistic read with a StampedLock

```java
class Point {
    ...
    double distanceFromOrigin() {
        long stamp = sl.tryOptimisticRead();
        double currX = x, currY = y;
        if (!sl.validate(stamp)) {
            stamp = sl.readLock();
            try {
                currX = x; currY = y;
            }
            finally {
                sl.unlockRead(stamp);
            }
        }
        return Math.sqrt(currX * currX + currY * currY);
    }
    ...

    Attempt to get an “observation” stamp

    Applying Java StampedLock: Optimistic & Reading Mode
```
Performing an optimistic read with a StampedLock class:

```java
class Point {
    ...
    double distanceFromOrigin() {
        long stamp = sl.tryOptimisticRead();
        double currX = x, currY = y;
        if (!sl.validate(stamp)) {
            stamp = sl.readLock();
            try {
                currX = x; currY = y;
            } finally {
                sl.unlockRead(stamp);
            }
        }
        return Math.sqrt(currX * currX + currY * currY);
    }
    ...
}
```

"Optimistically" read state into local variables

Code using optimistic reading mode typically copies the values of fields & holds them in local variables for use after they are validated.
Performing a optimistic read with a StampedLock

```java
class Point {
    ...
    double distanceFromOrigin() {
        long stamp = sl.tryOptimisticRead();
        double currX = x, currY = y;
        if (!sl.validate(stamp)) {
            stamp = sl.readLock();
            try {
                currX = x; currY = y;
            } finally {
                sl.unlockRead(stamp);
            }
        }
        return Math.sqrt(currX * currX + currY * currY);
    }
    ...
}
```

Check if another thread acquired the lock for writing after earlier call to tryOptimisticRead()
Performing an optimistic read with a StampedLock

class Point {

    double distanceFromOrigin() {
        long stamp = sl.tryOptimisticRead();
        double currX = x, currY = y;
        if (!sl.validate(stamp)) {
            stamp = sl.readLock();
            try {
                currX = x; currY = y;
            } finally {
                sl.unlockRead(stamp);
            }
        }
        return Math.sqrt(currX * currX + currY * currY);
    }

    ...

}
Performing a optimistic read with a StampedLock

```java
class Point {
    
    double distanceFromOrigin() {
        long stamp = sl.tryOptimisticRead();
        double currX = x, currY = y;
        if (!sl.validate(stamp)) {
            stamp = sl.readLock();
            try {
                currX = x; currY = y;
            } finally {
                sl.unlockRead(stamp);
            }
        }
        return Math.sqrt(currX * currX + currY * currY);
    }
    
    ...
```
• Performing a optimistic read with a StampedLock

class Point {
    ...
    double distanceFromOrigin() {
        long stamp = sl.tryOptimisticRead();
        double currX = x, currY = y;
        if (!sl.validate(stamp)) {
            stamp = sl.readLock();
            try {
                currX = x; currY = y;
            } finally {
                sl.unlockRead(stamp); }
        }
        return Math.sqrt(currX * currX + currY * currY);
    }
    ...
    Release read lock
}
Performing a optimistic read with a StampedLock

```java
class Point {
    ... 
    double distanceFromOrigin() {
        long stamp = sl.tryOptimisticRead();
        double currX = x, currY = y;
        if (!sl.validate(stamp)) {
            stamp = sl.readLock();
            try {
                currX = x; currY = y;
            } finally {
                sl.unlockRead(stamp);
            }
        }
        return Math.sqrt(currX * currX + currY * currY);
    }
    ...
```
Performing an optimistic read with a StampedLock

```java
class Point {
    ... 
    double distanceFromOrigin() {
        long stamp = sl.tryOptimisticRead();
        double currX = x, currY = y;
        if (!sl.validate(stamp)) {
            stamp = sl.readLock();
            try {
                currX = x; currY = y;
            } finally {
                sl.unlockRead(stamp);
            }
        }
        return Math.sqrt(currX * currX + currY * currY);
    }
    ...
```

Do computation with the copied values
Applying Java Stamped Lock: Conditional Write
Performing a conditional write with a StampedLock class

```java
void moveIfAtOrigin(double newX, double newY) {
    long stamp = sl.readLock();
    try {
        while (x == 0.0 && y == 0.0) {
            long ws = sl.tryConvertToWriteLock(stamp);
            if (ws != 0L) {
                stamp = ws;
                x = newX; y = newY;
                break;
            } else {
                sl.unlockRead(stamp);
                stamp = sl.writeLock();
            }
        }
    }
    ...
}
```

Move a point only if it’s current at the origin

Applying Java StampedLock: Conditional Write
Performing a conditional write with a StampedLock

class Point {
    ...
    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock(); /* Acquire a read lock */
        try {
            while (x == 0.0 && y == 0.0) {
                long ws = sl.tryConvertToWriteLock(stamp);
                if (ws != 0L) {
                    stamp = ws;
                    x = newX; y = newY;
                    break;
                } else {
                    sl.unlockRead(stamp);
                    stamp = sl.writeLock();
                }
            }
        } catch (InterruptedException e) {
            // Handle exception
        }
    }
    ...
}
Performing a conditional write with a StampedLock

```java
class Point {
    ...
    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock();
        try {
            while (x == 0.0 && y == 0.0) {
                long ws = sl.tryConvertToWriteLock(stamp);
                if (ws != 0L) {
                    stamp = ws;
                    x = newX; y = newY;
                    break;
                } else {
                    sl.unlockRead(stamp);
                    stamp = sl.writeLock();
                }
            }
        } finally {
            ...
        }
    }
    ...
}
```

Check whether x & y are at the origin

Applying Java StampedLock: Conditional Write

This loop only executes at most twice!
Performing a conditional write with a StampedLock class

```java
class Point {
    ...
    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock();
        try {
            while (x == 0.0 && y == 0.0) {
                long ws = sl.tryConvertToWriteLock(stamp);
                if (ws != 0L) {
                    stamp = ws;
                    x = newX; y = newY;
                    break;
                } else {
                    sl.unlockRead(stamp);
                    stamp = sl.writeLock();
                }
            }
        } catch (
```}

Try to upgrade to a write lock w/out blocking

tryConvertToWriteLock() atomically releases the read lock & acquires the write lock if there are no other readers
Performing a conditional write with a StampedLock

class Point {
    ...
    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock();
        try
            while (x == 0.0 && y == 0.0) {
                long ws = sl.tryConvertToWriteLock(stamp);
                if (ws != 0L) {
                    Upgrade succeeded w/out blocking!
                    stamp = ws;
                    x = newX; y = newY;
                    break;
                } else {
                    sl.unlockRead(stamp);
                    stamp = sl.writeLock();
                }
            }
    }
    ...

Applying Java StampedLock: Conditional Write
Applying Java StampedLock: Conditional Write

• Performing a conditional write with a StampedLock

```java
class Point {
    ...
    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock();
        try {
            while (x == 0.0 && y == 0.0) {
                long ws = sl.tryConvertToWriteLock(stamp);
                if (ws != 0L) {
                    stamp = ws;
                    x = newX; y = newY;
                    break;
                } else {
                    sl.unlockRead(stamp);
                    stamp = sl.writeLock();
                }
            }
        } finally {
            ...
        }
    } ...
```
Performing a conditional write with a StampedLock

class Point {
    ...
    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock();
        try
            while (x == 0.0 && y == 0.0) {
                long ws = sl.tryConvertToWriteLock(stamp);
                if (ws != 0L) {
                    stamp = ws;
                    x = newX; y = newY;
                    break;  // Exit the loop
                } else {
                    sl.unlockRead(stamp);
                    stamp = sl.writeLock();
                }
            }
        ...
    }
    ...
}

Applying Java StampedLock: Conditional Write
Performing a conditional write with a StampedLock

class Point {
    ...
    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock();
        try
            while (x == 0.0 && y == 0.0) {
                long ws = sl.tryConvertToWriteLock(stamp);
                if (ws != 0L) {
                    stamp = ws;
                    x = newX; y = newY;
                    break;
                } else {
                    sl.unlockRead(stamp);
                    stamp = sl.writeLock();
                }
            }
    }...

The x & y field values may change between unlockRead() & writeLock()!
Performing a conditional write with a StampedLock

```java
class Point {
    ...
    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock();
        try {
            while (x == 0.0 && y == 0.0) {
                long ws = sl.tryConvertToWriteLock(stamp);
                if (ws != 0L) {
                    stamp = ws;
                    x = newX; y = newY;
                    break;
                } else {
                    sl.unlockRead(stamp);
                    stamp = sl.writeLock();
                }
            }
        } finally {
            ...
        }
    }
}
```

**Must retest loop condition since x & y field values may change between unlockRead() & writeLock()!**
Performing a conditional write with a StampedLock

class Point {
    ...
    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock();
        try {
            while (x == 0.0 && y == 0.0) {
                long ws = sl.tryConvertToWriteLock(stamp);
                if (ws != 0L) {
                    stamp = ws;
                    x = newX; y = newY;
                    break;
                } else {
                    sl.unlockRead(stamp);
                    stamp = sl.writeLock();
                }
            }
        } finally {
            ...
        }
    }
    ...
}

This conversion will always succeed since stamp is now a write lock
Applying Java StampedLock: Conditional Write

• Performing a conditional write with a StampedLock

```java
class Point {

    ... ...

    void moveIfAtOrigin(double newX, double newY) {
        long stamp = sl.readLock();
        try {
            while (x == 0.0 && y == 0.0) {
                ... ...
                stamp = ws;
                ... ...
                stamp = sl.writeLock();
            }
        } finally {
            sl.unlock(stamp);
        }
    }

    ... ...

    Release the appropriate lock
```
End of Example Application of Java StampedLock