

STIWK3014 REAL TIME PROGRAMMING

Tutorial / Exercise 9: Safe Lock and ReentrantLock Method

```
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import java.util.Random;

public class Safelock {
    static class Friend {
        private final String name;
        private final Lock lock = new ReentrantLock();

        public Friend(String name) {
            this.name = name;
        }

        public String getName() {
            return this.name;
        }

        public boolean impendingBow(Friend bower) {
            Boolean myLock = false;
            Boolean yourLock = false;
            try {
                myLock = lock.tryLock();
                yourLock = bower.lock.tryLock();
            } finally {
                if (! (myLock && yourLock)) {
                    if (myLock) {
                        lock.unlock();
                    }
                    if (yourLock) {
                        bower.lock.unlock();
                    }
                }
            }
            return myLock && yourLock;
        }

        public void bow(Friend bower) {
            if (impendingBow(bower)) {
                try {
                    System.out.format("%s: %s has"
                        + " bowed to me!\n",
                        this.name, bower.getName());
                    bower.bowBack(this);
                } finally {
                    lock.unlock();
                    bower.lock.unlock();
                }
            } else {
                System.out.format("%s: %s started"
                    + " to bow to me, but saw that"
                    + " I was already bowing to"
                    + " him.\n",
                    this.name, bower.getName());
            }
        }

        public void bowBack(Friend bower) {
            System.out.format("%s: %s has"
                + " bowed back to me!\n",
                this.name, bower.getName());
        }
    }

    static class BowLoop implements Runnable {
        private Friend bower;
        private Friend bowee;

        public BowLoop(Friend bower, Friend bowee) {
            this.bower = bower;
            this.bowee = bowee;
        }
    }
}
```

```

public void run() {
    Random random = new Random();
    for (;;) {
        try {
            Thread.sleep(random.nextInt(10));
        } catch (InterruptedException e) {}
        bowee.bow(bower);
    }
}

```

```

public static void main(String[] args) {
    final Friend alphonse =
        new Friend("Alphonse");
    final Friend gaston =
        new Friend("Gaston");
    new Thread(new BowLoop(alphonse, gaston)).start();
    new Thread(new BowLoop(gaston, alphonse)).start();
}
}

```

Coding

```
m pom.xml (Tutorial9) x SafeLock.java x
1 import java.util.concurrent.locks.Lock;
2 import java.util.concurrent.locks.ReentrantLock;
3 import java.util.Random;
4
5 public class SafeLock {
6     static class Friend { 11 usages
7         private final String name; 5 usages
8         private final Lock lock = new ReentrantLock(); 6 usages
9
10        public Friend(String name) { 2 usages
11            this.name = name;
12        }
13
14        public String getName() { 3 usages
15            return this.name;
16        }
17
18        public boolean impedingBow(Friend bower) { 1 usage
19            Boolean myLock = false;
20            Boolean yourLock = false;
21            try {
22                myLock = lock.tryLock();
23                yourLock = bower.lock.tryLock();
24            } finally {
25                if (!myLock && yourLock) {
26                    if (myLock) {
27                        lock.unlock();
28                    }
29                    if (yourLock) {
30                        bower.lock.unlock();
31                    }
32                }
33            }
34
35            return myLock && yourLock;
36        }
37
38        public void bow(Friend bower) { 1 usage
39            if (impedingBow(bower)) {
40                try {
41                    System.out.format("%s: %s has bowed to me!\n", this.name, bower.getName());
42                    bower.bowBack(this);
43                } finally {
44                    lock.unlock();
45                    bower.lock.unlock();
46                }
47            } else {
48                System.out.format("%s: %s started to bow to me, but saw that I was already bowing to him.\n",
49                    this.name, bower.getName());
50            }
51        }
52
53        public void bowBack(Friend bower) { 1 usage
54            System.out.format("%s: %s has bowed back to me!\n", this.name, bower.getName());
55        }
56    }
57
58    static class BowLoop implements Runnable { 2 usages
59        private Friend bower; 2 usages
60        private Friend bowee; 2 usages
61
62        public BowLoop(Friend bower, Friend bowee) { 2 usages
63            this.bower = bower;
64            this.bowee = bowee;
65        }
66
67        public void run() {
68            Random random = new Random();
69            for (;;) {
70                try {
71                    Thread.sleep(random.nextInt(bound: 10));
72                    bowee.bow(bower);
73                } catch (InterruptedException e) {
74                    Thread.currentThread().interrupt(); // Best practice
75                }
76            }
77        }
78
79        public static void main(String[] args) {
80            final Friend alphonse = new Friend(name: "Alphonse");
81            final Friend gaston = new Friend(name: "Gaston");
82            new Thread(new BowLoop(alphonse, gaston)).start();
83            new Thread(new BowLoop(gaston, alphonse)).start();
84        }
85    }
```

Output

The image shows a screenshot of an IDE (Integrated Development Environment) with a dark theme. At the top, there's a 'Project' view showing a file named 'Tutorial9'. Below it, a 'Run' button is visible, and a 'SafeLock' window is open. The main area is a terminal window showing the output of a Java program. The output consists of a series of messages: 'Gaston: Alphonse has bowed to me!', 'Alphonse: Gaston has bowed back to me!', and so on, alternating between the two characters. The messages end with 'Gaston: Alphonse has bowed back to me!'. The IDE interface includes various icons on the left for navigation and a status bar at the bottom showing the current file path and some statistics.

Submission:

Platform: 1. Online Learning - Sample Coding & Output in PdF form
2. GitHub – Upload the coding file to your GitHub account.

Date: 21 May 2025 (Wednesday, before 12.30 noon)