javaf / bathroom-lock

A Bathroom Lock allows N genders (thread types) to access a common bathroom (critical section) such that different genders do not clash.

repl.it/@wolfram77/bathroom-lock#readme.md

MIT License

0 stars 0 forks

Star Unwatch

Code Issues Pull requests Actions Projects Wiki Security

master

wolfram77 better description i hope ... 3 days ago 3

View code

README.md

A Bathroom Lock allows N genders (thread types) to access a common bathroom (critical section) such that different genders do not clash.

Bathroom Lock allows threads of the same type (gender) to enter at the same time, but disallows different types of threads to occupy the critical section (bathroom) at the same time.

```
lock():
1. Acquire common lock.
2. Wait until there is no other gender.
3. Increment my gender count.
4. Release common lock.


unlock():
1. Acquire common lock.
2. Decrement my gender count.
3. If my gender cleared, signal others.
4. Release common lock.
```

```
## OUTPUT
Starting 100 unsafe males ...
Starting 100 unsafe females ...
F101: saw 5 males
F108: saw 2 males
F109: saw 2 males
F107: saw 2 males
F108: saw 2 males
F110: saw 2 males
F109: saw 2 males
F101: saw 2 males
F110: saw 2 males
F103: saw 2 males
F107: saw 2 males
F106: saw 2 males
F103: saw 2 males
F104: saw 2 males
F106: saw 2 males
F100: saw 5 males
F104: saw 2 males
F100: saw 2 males
F105: saw 2 males
F102: saw 2 males
M99: saw 2 females
F105: saw 2 males
F102: saw 2 males
F112: saw 1 males
Clashes occurred: 24

Starting 100 safe males ...
Starting 100 safe females ...
Clashes occurred: 0
```

See BathroomLock.java for code, Main.java for test, and repl.it for output.

references

- The Art of Multiprocessor Programming :: Maurice Herlihy, Nir Shavit

Languages

- Java 100.0%

javaf / bathroom-lock

Code Issues Pull requests Actions Projects Wiki Security

master

bathroom-lock / AbstractLock.java / Jump to

wolfram77 moved from shared-bathroom-problem History

1 contributor

Raw Blame

28 lines (24 sloc) 485 Bytes

```java
import java.util.concurrent.*;
import java.util.concurrent.locks.*;

abstract class AbstractLock implements Lock {
  @Override
  public void lockInterruptibly()
  throws InterruptedException {
    lock();
  }

  @Override
  public boolean tryLock() {
    lock();
    return true;
  }

  @Override
  public boolean tryLock(long arg0, TimeUnit arg1)
  throws InterruptedException {
    lock();
    return true;
  }

  @Override
  public Condition newCondition() {
    return null;
  }
```

```
28  }
```

javaf / bathroom-lock

Code Issues Pull requests Actions Projects Wiki Security

master

bathroom-lock / BathroomLock.java / Jump to

wolfram77 better description i hope History

1 contributor

Raw Blame

79 lines (69 sloc) 1.95 KB

```
1  import java.util.concurrent.locks.*;
2
3  // A Bathroom Lock allows N genders (thread types) to
4  // access a common bathroom (critical section) such that
5  // different genders do not clash.
6  //
7  // Bathroom Lock allows threads of the same type
8  // (gender) to enter at the same time, but disallows
9  // different types of threads to occupy the
10 // critical section (bathroom) at the same time.
11
12 class BathroomLock {
13   Lock lock;
14   Condition condition;
15   Lock[] genderLocks;
16   int[] counts;
17   // lock: common lock
18   // condition: no-one in bathroom
19   // genderLocks: for ith gender to use bathroom
20   // counts: no. of ith gender in bathroom
21
22   public BathroomLock(int types) {
23     lock = new ReentrantLock();
24     condition = lock.newCondition();
25     genderLocks = new GenderLock[types];
26     for (int i=0; i<types; i++)
27       genderLocks[i] = new GenderLock(i);
```

```
28      counts = new int[types];
29    }
30
31    public Lock genderLock(int i) {
32      return genderLocks[i];
33    }
34
35
36    class GenderLock extends AbstractLock {
37      int type;
38      // type: gender type
39
40      public GenderLock(int i) {
41        type = i;
42      }
43
44      // 1. Acquire common lock.
45      // 2. Wait until there is no other gender.
46      // 3. Increment my gender count.
47      // 4. Release common lock.
48      @Override
49      public void lock() {
50        lock.lock(); // 1
51        try {
52          while (!oneGender()) // 2
53            condition.await(); // 2
54          counts[type]++; // 3
55        }
56        catch (InterruptedException e) {}
57        finally { lock.unlock(); } // 4
58      }
59
60      // 1. Acquire common lock.
61      // 2. Decrement my gender count.
62      // 3. If my gender cleared, signal others.
63      // 4. Release common lock.
64      @Override
65      public void unlock() {
66        lock.lock(); // 1
67        counts[type]--; // 2
68        if (counts[type] == 0)   // 3
69          condition.signalAll(); // 3
70        lock.unlock(); // 4
71      }
72
73      private boolean oneGender() {
74        for (int i=0; i<counts.length; i++)
75          if (i!=type && counts[i]>0) return false;
```

```
76          return true;
77        }
78      }
79    }
```

javaf / bathroom-lock

Code Issues Pull requests Actions Projects Wiki Security

master

bathroom-lock / Main.java / Jump to

wolfram77 moved from shared-bathroom-problem History

1 contributor

Raw Blame

101 lines (80 sloc) 2.32 KB

```
1  import java.util.concurrent.locks.*;
2  import java.util.concurrent.atomic.*;
3
4
5  class Main {
6    static BathroomLock lock;
7    static AtomicInteger males, females, clashes;
8    static int ML = 100, FM = 100;
9    static int CS = 2;
10   // males: number of males in bathroom
11   // females: number of females in bathroom
12   // clashes: times males & females were together
13   // ML: number of males
14   // FM: number of females
15   // CS: checks per person (thread)
16
17
18   static Thread male(int id, boolean safe) {
19     return new Thread(() -> {
20       if (safe) lock.genderLock(0).lock();
21       males.incrementAndGet();
22
23       for (int i=0; i<CS; i++) {
24         sleep(1);
25         int f = females.get();
26         if (f == 0) continue;
27         clashes.incrementAndGet();
```

```
28          log("M"+id+": saw "+f+" females");
29        }
30
31        males.decrementAndGet();
32        if (safe) lock.genderLock(0).unlock();
33      });
34    }
35
36
37    static Thread female(int id, boolean safe) {
38      return new Thread(() -> {
39        if (safe) lock.genderLock(1).lock();
40        females.incrementAndGet();
41
42        for (int i=0; i<CS; i++) {
43          sleep(1);
44          int m = males.get();
45          if (m == 0) continue;
46          clashes.incrementAndGet();
47          log("F"+id+": saw "+m+" males");
48        }
49
50        females.decrementAndGet();
51        if (safe) lock.genderLock(1).unlock();
52      });
53    }
54
55
56    // Tests to see if males and females entered
57    // bathroom separately.
58    static void testThreads(boolean safe) {
59      String type = safe? "safe" : "unsafe";
60      log("Starting "+ML+" "+type+" males ...");
61      log("Starting "+FM+" "+type+" females ...");
62
63      males.set(0);
64      females.set(0);
65      clashes.set(0);
66
67      Thread[] t = new Thread[ML+FM];
68      for (int i=0; i<ML+FM; i++) {
69        t[i] = i<ML? male(i, safe) : female(i, safe);
70        t[i].start();
71      }
72
73      try {
74      for (int i=0; i<ML+FM; i++)
75        t[i].join();
```

```
76      }
77      catch(InterruptedException e) {}
78      log("Clashes occurred: "+clashes.get()+"\n");
79    }
80
81
82    public static void main(String[] args) {
83      lock = new BathroomLock(2);
84      males = new AtomicInteger(0);
85      females = new AtomicInteger(0);
86      clashes = new AtomicInteger(0);
87
88      testThreads(false);
89      testThreads(true);
90    }
91
92
93    static void sleep(long ms) {
94      try { Thread.sleep(ms); }
95      catch (InterruptedException e) {}
96    }
97
98    static void log(String x) {
99      System.out.println(x);
100   }
101 }
```