THE BAKERY ALGORITHM

Multiple customers are planning to eat at a bakery. For each item, the pick a "latest" ticket and wait for their turn to arrive. When their turn comes, they place the order, clear their ticket, goto eating. When they are hungry again, the process repeats.

Each customer picks a maximum ticket, but since this maximum can be the same for two people more preference is given to customer with smaller id. Also, if picking maximum is not made atomic, it is possible for two customers to end up ordering at the same time, so a "choosing" status is present for each customer.

CODE: https://repl.it/@wolfram77/bakery-algorithm#Main.java

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


class Main {
static int[] ticket;
static boolean[] choosing;
static int N = 10;


static void customer(int i) {
new Thread(() -> {
try {
while(true) {
lock(i);
log(i+": placing order");
unlock(i);
log(i+": eating");
Thread.sleep((long)(Math.random()*1000));
log(i+": done");
}
}
catch(InterruptedException e) {}
}).start();
}


static void lock(int i) {
try {
```

```
log(i+": choosing ticket");
choosing[i] = true;
ticket[i] = max(ticket, 0) + 1;
choosing[i] = false;
log(i+": got ticket="+ticket[i]);
log(i+": waiting for turn");
for(int p=0; p<N; p++) {
while(choosing[p]) Thread.sleep(10);
while(
(ticket[i] > 0) &&
(ticket[p] < ticket[i] ||
(ticket[p] == ticket[i] && p < i)))
Thread.sleep(10);
}
}
catch(InterruptedException e) {}
}


static void unlock(int i) {
ticket[i] = 0;
}


public static void main(String[] args) {
ticket = new int[N];
choosing = new boolean[N];
for(int i=0; i<N; i++)
customer(i);
}
static int max(int[] x, int vd) {
int a = vd;
for(int i=0; i<x.length; i++)
a = Math.max(a, x[i]);
return a;
}
static void log(String x) {
System.out.println(x);
}
}
```

OUTPUT: https://bakery-algorithm.wolfram77.repl.run

```
6: choosing ticket
3: choosing ticket
4: choosing ticket
0: choosing ticket
2: choosing ticket
9: choosing ticket
7: choosing ticket
8: choosing ticket
1: choosing ticket
5: choosing ticket
4: got ticket=2
0: got ticket=3
2: got ticket=4
2: waiting for turn
8: got ticket=6
8: waiting for turn
0: waiting for turn
7: got ticket=5
4: waiting for turn
6: got ticket=10
6: waiting for turn
5: got ticket=8
1: got ticket=7
1: waiting for turn
7: waiting for turn
5: waiting for turn
3: got ticket=1
3: waiting for turn
3: placing order
3: eating
9: got ticket=9
9: waiting for turn
3: done
3: choosing ticket
3: got ticket=11
3: waiting for turn
4: placing order
4: eating
4: done
4: choosing ticket
4: got ticket=12
4: waiting for turn
0: placing order
0: eating
...
```