THE SLEEPING BARBERS PROBLEM

There is a barber sleeping in his shop. When a customer comes, he checks if the barber is sleeping and wakes him up. If there are no other customers in the waiting room, the barber cuts his hair. Else the customer takes a set in the waiting room. But if there are no seats, he leaves. Once a barber finishes cutting hair, he checks if there are any customers in the waiting room. If not, he goes back to sleep again. The idea is the barber works only when a customer arrives, and sleeps otherwise.

CODE: https://repl.it/@wolfram77/sleeping-barber-problem#Main.java

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


class Main {
static Semaphore barber;
static Semaphore customer;
static Semaphore accessSeats;
static int seats = 4;
static int N = 20;


static void barber() {
new Thread(() -> {
try {
while(true) {
log("barber: sleeping");
customer.acquire();
log("barber: got customer");
accessSeats.acquire();
seats++;
barber.release();
accessSeats.release();
log("barber: cutting hair");
Thread.sleep(1000);
log("barber: cutting done");
}
}
catch(InterruptedException e) {}
}).start();
}
static void customer(int i) {
new Thread(() -> {
try {
log("customer "+i+": checking seats");
accessSeats.acquire();
if(seats<=0) {
log("customer "+i+": no seats!");
accessSeats.release();
return;
}
seats--;
customer.release();
accessSeats.release();
log("customer "+i+": sat, seats="+seats);
barber.acquire();
log("customer "+i+": having hair cut");
}
catch(InterruptedException e) {}
}).start();
}


public static void main(String[] args) {
try {
barber = new Semaphore(0);
customer = new Semaphore(0);
accessSeats = new Semaphore(1);
barber();
for(int i=0; i<N; i++) {
Thread.sleep((long)(Math.random()*1000));
customer(i);
}
}
catch(InterruptedException e) {}
}
static void log(String x) {
System.out.println(x);
}
}
```

OUTPUT: https://sleeping-barber-problem.wolfram77.repl.run

```
barber: sleeping
customer 0: checking seats
barber: got customer
barber: cutting hair
customer 0: sat, seats=3
customer 0: having hair cut
customer 1: checking seats
customer 1: sat, seats=3
customer 2: checking seats
customer 2: sat, seats=2
customer 3: checking seats
customer 3: sat, seats=1
customer 4: checking seats
customer 4: sat, seats=0
barber: cutting done
barber: sleeping
barber: got customer
barber: cutting hair
customer 1: having hair cut
customer 5: checking seats
customer 5: sat, seats=0
customer 6: checking seats
customer 6: no seats!
barber: cutting done
barber: sleeping
barber: got customer
barber: cutting hair
customer 2: having hair cut
customer 7: checking seats
customer 7: sat, seats=0
customer 8: checking seats
customer 8: no seats!
barber: cutting done
barber: sleeping
barber: got customer
barber: cutting hair
customer 3: having hair cut
customer 9: checking seats
customer 9: sat, seats=0
customer 10: checking seats
customer 10: no seats!
barber: cutting done
barber: sleeping
...
```