```
public class RateLimiter {
    static final int LIMIT = 100; // example value
    public int quota = LIMIT;
    private Lock lock = new ReentrantLock();
    private Condition needQuota = lock.newCondition();
    public void increaseQuota() { // called once per minute
        synchronized(lock) {      // grab the lock
            if (quota < LIMIT) {  // if some of the quote has been used up:
                quota = LIMIT;    // increase quota to LIMIT
                needQuota.signal(); // wake up a sleeper
            }
        }                         // unlock
    }
    private void throttle(int weight) {
        synchronized(lock) {      // grab the lock
            while (quota < weight) { // while not enough quota:
                needQuota.await(); // sleep until increased
            }
            quota -= weight;      // claim my job's part of the quota
            if (quota > 0) {      // if still quota left over:
                needQuota.signal(); // wake up another sleeper
            }
        }                         // unlock
    }
    public void run(Runnable job, int weight) {
        throttle(weight);         // sleep if under quota
        job.run();                // run my job
    }
}
```

FIGURE 8.15 A proposed RateLimiter class implementation.