Listing 8.16 Methods for each step of the hard disk system.

```
public void step() {
   // finds minimum collision time from list of collision times
   minimumCollisionTime();
   // moves particles for time equal to minimum collision time
   move();
   t += timeToCollision;
   // changes velocities of two colliding particles
   contact();
   // sets collision times to bigTime for those particles set to
   // collide with two colliding particles
   setDefaultCollisionTimes();
   // finds new collision times between all particles and the two
   // colliding particles
   newCollisionTimes();
   numberOfCollisions++;
}

public void minimumCollisionTime() {
   // sets collision time very large to find minimum collision time
   timeToCollision = bigTime;
   for(int k = 0; k<N; k++) {
      if(collisionTime[k]<timeToCollision) {
         timeToCollision = collisionTime[k];
         nextCollider = k;
      }
   }
   nextPartner = partner[nextCollider];
}

public void move() {
   for(int k = 0; k<N; k++) {
      collisionTime[k] -= timeToCollision;
      x[k] = PBC.position(x[k]+vx[k]*timeToCollision, Lx);
      y[k] = PBC.position(y[k]+vy[k]*timeToCollision, Ly);
   }
}

public void contact() {
   // computes collision dynamics between nextCollider and nextPartner
   // at contact
   double dx = PBC.separation(x[nextCollider]-x[nextPartner], Lx);
   double dy = PBC.separation(y[nextCollider]-y[nextPartner], Ly);
   double dvx = vx[nextCollider]-vx[nextPartner];
   double dvy = vy[nextCollider]-vy[nextPartner];
   double factor = dx*dvx+dy*dvy;
   double delvx = -factor*dx;
   double delvy = -factor*dy;
   vx[nextCollider] += delvx;
   vy[nextCollider] += delvy;
   vx[nextPartner] -= delvx;
   vy[nextPartner] -= delvy;
   virialSum += delvx*dx+delvy*dy;
}

public void setDefaultCollisionTimes() {
   collisionTime[nextCollider] = bigTime;
```



```
   collisionTime[nextPartner] = bigTime;
   // sets collision times to bigTime for all particles set to collide
   // with the two colliding particles
   for(int k = 0; k<N; k++) {
      if(partner[k]==nextCollider) {
         collisionTime[k] = bigTime;
      } else if(partner[k]==nextPartner) {
         collisionTime[k] = bigTime;
      }
   }
}

public void newCollisionTimes() {
  // finds new collision times for all particles that were set to
  // collide with two colliding particles; also finds new collision
  // times for two colliding particles
   for(int k = 0; k<N; k++) {
      if((k!=nextCollider)&&(k!=nextPartner)) {
         checkCollision(k, nextPartner);
         checkCollision(k, nextCollider);
      }
   }
}
```

The colliding pair and the next collision time are found in `minimumCollisionTime`, and all the particles are moved forward in `move` until contact occurs. The collision dynamics of the colliding pair is computed in method `contact`, where the contribution to the virial is also found. In `setDefaultCollisionTimes` we set all the collision times to an arbitrarily large value, `bigTime`, for all pairs of particles that need to be updated. Then in `newCollisionTimes` we update the collision times for those particles in step 5.

In method `initialize` we initialize various variables and, most importantly, compute the minimum collision time for each particle using method `checkCollision`. The ith element in the array, `collisionTime`, stores the minimum collision time for particle i with all the other particles. The array element `partner[i]` stores the particle label of the collision partner corresponding to this time. The collision time for each particle is initially set to an arbitrarily large value, `bigTime`, to take into account that at any given time, some particles have no collision partners. The methods for setting the initial positions and velocities are the same as those used for simulating Lennard–Jones particles.

Listing 8.17 Method for generating the initial configuration of hard disks.

```
public void initialize(String configuration) {
   resetAverages();
   x = new double[N];
   y = new double[N];
   vx = new double[N];
   vy = new double[N];
   collisionTime = new double[N];
   partner = new int[N];
   if(configuration.equals("regular")) {
      setRegularPositions();
   } else {
      setRandomPositions();
   }
   setVelocities();
```