```
      heatCapacity = heatCapacity/(temperature*temperature);
      return(heatCapacity/N);
   }

   public double susceptibility() {
      double magnetizationSquaredAverage =
            magnetizationSquaredAccumulator/mcs;
      double magnetizationAverage = magnetizationAccumulator/mcs;
      return(magnetizationSquaredAverage -
            Math.pow(magnetizationAverage, 2))/(temperature*N);
   }

   public void resetData() {
      mcs = 0;
      energyAccumulator = 0;
      energySquaredAccumulator = 0;
      magnetizationAccumulator = 0;
      magnetizationSquaredAccumulator = 0;
      acceptedMoves = 0;
   }

   public void doOneMCStep() {
      for(int k = 0;k<N;++k) {
         int i = (int) (Math.random()*L);
         int j = (int) (Math.random()*L);
         int dE = 2*lattice.getValue(i, j)*
               (lattice.getValue((i+1)%L, j)
               +lattice.getValue((i-1+L)%L, j)
               +lattice.getValue(i, (j+1)%L)
               +lattice.getValue(i, (j-1+L)%L));
         if((dE<=0)||(w[dE]>Math.random())) {
            int newSpin = -lattice.getValue(i, j);
            lattice.setValue(i, j, newSpin);
            acceptedMoves++;
            energy += dE;
            magnetization += 2*newSpin;
         }
      }
      energyAccumulator += energy;
      energySquaredAccumulator += energy*energy;
      magnetizationAccumulator += magnetization;
      magnetizationSquaredAccumulator += magnetization*magnetization;
      mcs++;
   }
}
```

One of the most time consuming parts of the Metropolis algorithm is the calculation of the exponential function $e^{-\beta\Delta E}$. Because there are only a small number of possible values of $\beta\Delta E$ for the Ising model (see Figure 15.11), we store the small number of different probabilities for the spin flips in the array `w`. The values of this array are computed in method `initialize`.

To implement the Metropolis algorithm, we determine the change in the energy ΔE and then accept the trial flip if $\Delta E \leq 0$. If this condition is not satisfied, we generate a random number in the unit interval and compare it to $e^{-\beta\Delta E}$. We can use a single if statement for these two conditions, because in Java (and C/C++) the second condition of an `||` (or) statement is evaluated only if the first is false. This feature is very useful because we do not want to generate random numbers when they are not needed, as is the case for $\Delta E \leq 0$. (The same feature holds for the compound `&&` (and) operator for which the second condition is only evaluated if the first is true.)

A typical laboratory system has at least 10^{18} spins. In contrast, the number of spins that can be simulated typically ranges from 10^3 to 10^9. As we have discussed in other contexts, the use of periodic boundary conditions minimizes finite size effects. However, more sophisticated boundary conditions are sometimes convenient. For example, we can give the surface spins extra neighbors, whose direction is related to the mean magnetization of the microstate (see Saslow).

In class `Ising` data for the values of the physical observables are accumulated after each Monte Carlo step per spin. The optimum time for sampling various physical quantities is explored in Problem 15.13. Note that if a flip is rejected, the old configuration is retained. Thermal equilibrium is not described properly unless the old configuration is again included in computing the averages.

Achieving thermal equilibrium can account for a substantial fraction of the total run time for very large systems. The most practical choice of initial conditions in these cases is a configuration from a previous run that is at a temperature close to the desired temperature. The code for reading and saving configurations can be found in Appendix 8A.

Problem 15.12 Equilibration of the two-dimensional Ising model

(a) Write a target class that uses class `Ising` and plots the magnetization and energy as a function of the number of Monte Carlo steps. Your program should also display the mean magnetization, the energy, the specific heat, the susceptibility, and the acceptance probability when the simulation is stopped. Averages such as the mean energy and the susceptibility should be normalized by the number of spins so that it is easy to compare systems with different values of N. Choose the linear dimension $L = 32$ and the heat bath temperature $T = 2$. Estimate the time needed to equilibrate the system given that all the spins are initially up.

(b) Visually determine if the spin configurations are "ordered" or "disordered" at $T = 2$ after equilibrium has been established.

(c) Repeat part (a) with the initial direction of each spin chosen at random. Make sure you explicitly compute the initial energy and magnetization in `initialize`. Does the equilibration time increase or decrease?

(d) Repeat parts (a)–(c) for $T = 2.5$. ■

Problem 15.13 Comparison with exact results

In general, a Monte Carlo simulation yields exact answers only after an infinite number of configurations have been sampled. How then can we be sure that our program works correctly, and our results are statistically meaningful? One way is to reproduce exact results in known limits. In the following, we test class `Ising` by considering a small system for which the mean energy and magnetization can be calculated analytically.

(a) Calculate analytically the T-dependence of E, M, C, and χ for the Ising model on the square lattice with $L = 2$. (A summary of the calculation is given in Appendix 15C.) For simplicity, we have omitted the brackets denoting the thermal averages.