```
public int perimeterListX[], perimeterListY[];
public int numberOfPerimeterSites;
public boolean ok = true;
public LatticeFrame lattice;

public Invasion(LatticeFrame latticeFrame) {
   lattice = latticeFrame;
   lattice.setIndexedColor(0, Color.blue);
   lattice.setIndexedColor(1, Color.black);
}

public void initialize() {
   Lx = 2*Ly;
   site = new double[Lx][Ly];
   perimeterListX = new int[Lx*Ly];
   perimeterListY = new int[Lx*Ly];
   for(int y = 0;y<Ly;y++) {
      site[0][y] = 1; // occupy first column
      lattice.setValue(0, y, 1);
   }
   for(int y = 0;y<Ly;y++) {
      for(int x = 1;x<Lx;x++) {
         site[x][y] = Math.random();
         lattice.setValue(x, y, 0);
      }
   }
   numberOfPerimeterSites = 0;
   for(int y = 0;y<Ly;y++) { // second column is perimeter sites
      site[1][y] += 2;       // perimeter sites have site > 2
      numberOfPerimeterSites++;
      // inserts site in perimeter list in order
      insert(1, y);
   }
   ok = true;
}

public void insert(int x, int y) {
   int insertionLocation = binarySearch(x, y);
   for(int i = numberOfPerimeterSites-1;i>insertionLocation;i--) {
      perimeterListX[i] = perimeterListX[i-1];
      perimeterListY[i] = perimeterListY[i-1];
   }
   perimeterListX[insertionLocation] = x;
   perimeterListY[insertionLocation] = y;
}

public int binarySearch(int x, int y) {
   int firstLocation = 0;
   int lastLocation = numberOfPerimeterSites-2;
   if(lastLocation<0) {
      lastLocation = 0;
   }
   int middleLocation = (firstLocation+lastLocation)/2;
   // determine which half of list new number is in
   while(lastLocation-firstLocation>1) {
      int middleX = perimeterListX[middleLocation];
```

```
      int middleY = perimeterListY[middleLocation];
      if(site[x][y]>site[middleX][middleY]) {
         lastLocation = middleLocation;
      } else {
         firstLocation = middleLocation;
      }
      middleLocation = (firstLocation+lastLocation)/2;
   }
   return lastLocation;
}

// goes in order looking for location to insert
public int linearSearch(int x, int y) {
   if(numberOfPerimeterSites==1) {
      return 0;
   } else {
      for(int i = 0;i<numberOfPerimeterSites-1;i++) {
         if(site[x][y]>site[perimeterListX[i]][perimeterListY[i]]) {
            return i;
         }
      }
   }
   return numberOfPerimeterSites-1;
}

public void step() {
   if(ok) {
      int nx[] = {1, -1, 0, 0};
      int ny[] = {0, 0, 1, -1};
      int x = perimeterListX[numberOfPerimeterSites-1];
      int y = perimeterListY[numberOfPerimeterSites-1];
      if(x>Lx-3) {
         // if cluster gets near the end, stop simulation
         ok = false;
      }
      numberOfPerimeterSites--;
      site[x][y] -= 1;
      lattice.setValue(x, y, 1);
      for(int i = 0;i<4;i++) { // finds new perimeter sites
         int perimeterX = x+nx[i];
         int perimeterY = (y+ny[i])%Ly;
         if(perimeterY==-1) {
            perimeterY = Ly-1;
         }
         if(site[perimeterX][perimeterY]<1) { // new perimeter site
            site[perimeterX][perimeterY] += 2;
            numberOfPerimeterSites++;
            insert(perimeterX, perimeterY);
         }
      }
   }
}

public void computeDistribution(PlotFrame data) {
   int numberOfBins = 20;
   int numberOccupied = 0;
```