```
/*
 * SpinsCanvas.java
 * ---------------------------
 * Separated and modified from Ising.java by David Wong
 */

import java.awt.*;
import java.awt.event.*;

/**
 * Canvas for spins simulation
 */
class SpinsCanvas extends Canvas implements MouseMotionListener, MouseListener {
  private Graphics offGraphics;
  private Image offImage;
  private Dimension offDimension;
  private int n;
  private IsingLattice lattice;
  private double dxy;
  private boolean drawGrid;
  private boolean spinsDrawn;  // for optimization purpose
  private short[] spin;
  private short plotValue;
  private int translatedX;
  private int translatedY;

/**
 * Constructor of SpinsCanvas
 */
    SpinsCanvas  (IsingLattice lattice, int n, short plotValue) {
        this.lattice = lattice;
        this.n = n;
        drawGrid = false;
        spinsDrawn = false;
        this.plotValue = plotValue;
        spin = new short [lattice.getNInd ()];
        addMouseMotionListener (this);
        addMouseListener (this);
    }

    public void paint (Graphics g) {
        update (g);
    }

    public Dimension getPreferredSize () {
        return new Dimension (180, 180);
    }
    public Dimension getMinimumSize () {
        return getPreferredSize ();
    }

    public void update  (Graphics g) {
        Dimension d = getSize ();
        if ((offGraphics == null)
            || (d.width != offDimension.width)
            || (d.height != offDimension.height)) {
            offDimension = d;
            offImage = createImage (d.width, d.height);
            offGraphics = offImage.getGraphics ();
            double dx = d.width / n;
            double dy = d.height / n;
            dxy = (dx <= dy) ? dx : dy;
            translatedX = (int) (d.width - n*dxy)/2;
            translatedY = (int) (d.height - n*dxy)/2;
```

```
            offGraphics.translate (translatedX, translatedY);
        }
        spinsDrawn = false;
        offGraphics.setColor (getBackground ());
        offGraphics.fillRect (0, 0, d.width, d.height);
        int wh = (int) (n*dxy);
        drawSpins (wh);
        g.drawImage (offImage, 0, 0, this);
    }


    private void drawSpins(int wh) {
        int i, j, t, pos=0;

        offGraphics.setColor(Color.blue);
        t=lattice.t;

        if (drawGrid) {
            if (!spinsDrawn)
                for (j=0; j<n; ++j)
                    for (i=0; i<n; ++i)
                        if (lattice.sigma[t][pos++] > 0)
                            offGraphics.fillRect((int)(i*dxy)+1,(int)(j*dxy)+1,
                                                 (int)dxy-1,(int)dxy-1);
                        else
                            offGraphics.clearRect((int)(i*dxy)+1,(int)(j*dxy)+1,
                                                  (int)dxy-1,(int)dxy-1);
            else
                for (j=0; j<n; ++j)
                    for (i=0; i<n; ++i) {
                        if (lattice.sigma[t][pos] != spin[pos])
                            if (lattice.sigma[t][pos] > 0)
                                offGraphics.fillRect((int)(i*dxy)+1,(int)(j*dxy)+1,
                                                     (int)dxy-1,(int)dxy-1);
                            else
                                offGraphics.clearRect((int)(i*dxy)+1,(int)(j*dxy)+1,
                                                      (int)dxy-1,(int)dxy-1);
                        ++pos;
                    }
            paintGrid ();
        } else {
            offGraphics.clearRect(0,0,wh+1,wh+1);
            for (j=0; j<n; ++j)
                for (i=0; i<n; ++i) {
                    if (lattice.sigma[t][pos++] > 0)
                        offGraphics.fillRect((int)(i*dxy),(int)(j*dxy),
                                             (int)dxy,(int)dxy);
                }
        }
        spin=lattice.sigma[t];
        spinsDrawn=true;
    }


    void setDrawGrid (boolean b) {
        drawGrid = b;

    }

    public Insets getInsets () {
        return new Insets (10, 10, 10, 10);
    }

    private void paintGrid() {
```

```
        int i;
        int size = (int)(n * dxy);
        offGraphics.setColor(Color.black);
        for (i=0; i<=n; i++) {
            offGraphics.drawLine(0,(int)(i*dxy),size - 1, (int)(i*dxy));
            offGraphics.drawLine((int)(i*dxy),0,(int)(i*dxy),size - 1);
        }
    }

    /* Two interfaces for Mouse Motion Listener */
    public void mouseDragged (MouseEvent e) {
        if (0<=e.getX () && e.getX ()<n*dxy &&
            0<=e.getY () && e.getY ()<n*dxy) {
            plot((int)(e.getX ()/dxy),
                 (int)(e.getY ()/dxy),plotValue);
            //redraw(false,true,false,false);
            repaint ();
        }
    }

    public void mouseMoved (MouseEvent e) {
    }

    public void plot(int x, int y, short value) {
      if (x >= 0 && x < n && y >=0 && y < n) {
        spin[x+n*y]=lattice.sigma[lattice.t][x+n*y]=value;
        offGraphics.setColor(Color.blue);
        if (drawGrid)
          if (value > 0)
            offGraphics.fillRect((int)(x*dxy)+1,(int)(y*dxy)+1,
                                 (int)dxy-1,(int)dxy-1);
          else
            offGraphics.clearRect((int)(x*dxy)+1,(int)(y*dxy)+1,
                                  (int)dxy-1,(int)dxy-1);
        else
          if (value > 0)
            offGraphics.fillRect((int)(x*dxy),(int)(y*dxy),
                                 (int)dxy,(int)dxy);
          else
            offGraphics.clearRect((int)(x*dxy),(int)(y*dxy),
                                  (int)dxy,(int)dxy);
      }
    }

  /* Five interfaces for Mouse Listener */
  public void mouseClicked (MouseEvent e) {
  }

  public void mouseEntered (MouseEvent e) {
  }

  public void mouseExited (MouseEvent e) {
  }

  public void mousePressed (MouseEvent e) {
    if (translatedX <=e.getX () && e.getX ()< (n*dxy + translatedX) &&
        translatedY <=e.getY () && e.getY ()< (n*dxy + translatedY)) {
      plotValue=(short)-lattice.getValue((int)((e.getX () - translatedX)/dxy) ,
                                         (int)((e.getY () - translatedY)/ dxy));
    }
  }

  public void mouseReleased (MouseEvent e) {
  }
```

```
}
```