```
/*
 * IsingCanvas.java
 * ---------------------------
 * Separated from Ising.java by David Wong
 */

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

final class IsingCanvas extends Panel {
    private Ising is;
    int n;
    private short plotValue;
    Rectangle Rect;
    Image img, HamiltonImg, MagnetizationImg;
    Graphics imgGra;
    private IsingFever Hamilton, Magnetization;
    private boolean GridForbidden=false;
    //    short[] Spin;
    private IsingLattice lattice;
    private Temperature tempValueHolder;
    private SpinsCanvas simCanvas;
    private TimeLabelCanvas timeLabel;

    public IsingCanvas(Ising is, int[] Panxy, short plotValue, int gridExtent,
                       IsingLattice lattice, Temperature tempValueHolder) {
        this.lattice = lattice;
        this.tempValueHolder = tempValueHolder;
        this.is=is;
        n=gridExtent;

        Hamilton=new HamiltonIsingFever
          ("Energy per spin",(int)(0.8*Panxy[2]), lattice);
        Magnetization=new MagnetizationIsingFever
          ("Magnetization",(int)(0.8*Panxy[2]), lattice);
        //      Spin=new short[lattice.nind];
        this.plotValue = plotValue;

        layoutComponents ();
        setUpObservers ();
    }

    /* This method layout the components on the panel by using GridBagLayout
     * layout manager
     */
    private void layoutComponents () {
        GridBagLayout gridBag = new GridBagLayout ();
        GridBagConstraints gc = new GridBagConstraints ();
        setLayout (gridBag);

        Thermometer meter = new Thermometer (tempValueHolder, lattice.TempMin,
                                lattice.TempMax, lattice.TempCrit);
        gc.fill = GridBagConstraints.VERTICAL;

        gc.anchor = GridBagConstraints.WEST;
        add (meter, gridBag, gc, 0, 0, 1, 4);
        gc.weightx = 1;
        gc.anchor = GridBagConstraints.NORTH;
        gc.fill = GridBagConstraints.HORIZONTAL;
        gc.insets = new Insets (-3, 0, 0, 0);
        Font f=new Font("TimesRoman",Font.PLAIN,15);
        add (timeLabel = new TimeLabelCanvas (lattice, f),
             gridBag, gc, 1, 0, 3, 1);
```

```
        gc.fill = GridBagConstraints.NONE;
        gc.anchor = GridBagConstraints.CENTER;

        simCanvas = new SpinsCanvas (lattice, n, plotValue);
        add (simCanvas, gridBag, gc, 1, 1, 3, 3);

        gc.insets = new Insets (0, 0, 0, 0);
        gc.weighty = 1;
        add (Hamilton, gridBag, gc, 4, 0, GridBagConstraints.REMAINDER, 2);

        add (Magnetization, gridBag, gc, 4, 2, GridBagConstraints.REMAINDER, 2);

    }

    private void setUpObservers () {
        //      lattice.addObserver (simCanvas);
        lattice.addObserver (Hamilton);
        lattice.addObserver (Magnetization);
    }

    private  void init() {
        Hamilton.reset (lattice.Hamilton (lattice.t));
        Magnetization.reset (lattice.Magnetization (lattice.t));
        //      Hamilton.init(lattice.Hamilton(lattice.t),
        //                    lattice.HamiltonMin(),lattice.HamiltonMax());
        //Magnetization.init(lattice.Magnetization(lattice.t),-1,1);
    }

    public void reset() {
        init();
    }

    public void setGridFlag(boolean GridFlag) {
        if (!GridForbidden) {
            simCanvas.setDrawGrid (GridFlag);
        }
    }

    private void drawTitle(Graphics g, String s, int x, int y, int fs) {
        int red, green, blue;
        Font f;
        FontMetrics fm;
        int i;
        String subs;

        red=(int)(Math.random()*150);
        green=(int)(Math.random()*100);
        blue=(int)(Math.random()*50);
        f=new Font("TimesRoman",Font.ITALIC,fs);
        g.setFont(f);
        fm=g.getFontMetrics();
        y+=f.getSize();

        for (i=0; i<s.length(); i++) {
            subs=s.substring(i,i+1);
            g.setColor(new Color((red+x/2)%256,(green+x)%256,(blue+x/4)%256));
            g.drawString(subs,x,y);
            x+=fm.stringWidth(subs);
        }
    }

    public void paint(Graphics g) {
        /*      Rect=getBounds();
        dx=(Rect.width-Panxy[0]-Panxy[2])/n;
```

```
        dy=(Rect.height-Panxy[1]-Panxy[3])/n;
        dxy=(dx <= dy) ? dx : dy;
        wh=(int)(n*dxy);
        if (dxy<=2) {
            GridForbidden=true;
        }

        SpinsDrawn=false;
        */
    }

    public void update(Graphics g) {     // DO AWAY WITH FLICKERING !!!!
        paintComponents (g);
    }

    public Dimension getMinimumSize () {
        return getPreferredSize ();
    }
    public Dimension getPreferredSize () {
        return new Dimension (1000, 350);
    }
    /**
     * add a component to the gridbag lagout
     */
    private void add (Component c, GridBagLayout gbl, GridBagConstraints gbc,
                      int x, int y, int w, int h) {
        gbc.gridx = x;
        gbc.gridy = y;
        gbc.gridwidth = w;
        gbc.gridheight = h;
        gbl.setConstraints (c, gbc);
        add (c);
    }

    public boolean gridIsForbidden () {
        return GridForbidden;
    }
}
```