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

import java.awt.*;
import java.awt.Image;
import java.util.Observer;
import java.util.Observable;

/* This is an abstract class for displaying curves
 * By defining getNewValue () in subclass, new curves can
 * be displayed
 */
abstract class IsingFever extends Canvas implements Observer {
  String title;                        // Titel (:-)   Title
  double[] y;                          // y-values of the curve
  private int n;                        // No. of values
  int width, height;                   // width and height
  int offset, top, bottom;             // offset in y, boundary lines
  int range;                           // drawing limits
  double xScale, yScale;               // scaling factor
  int index, length;                   // current ID of the curve
  double yMin, yMax, yMean;            // extreme and mean-values
  private Image img;

  private Dimension offDimension;
  private Image offImage;
  private Graphics offGraphics;

  Graphics g;
  public Color color;
  protected IsingLattice lattice;        // actual grid, here is where all the physics happen
+ s

  public Image getImage () {
    return img;
  }

  public IsingFever(String title, int n, IsingLattice lattice) {
    //System.out.println("IsingFever(int n)");
    this.lattice = lattice;
    this.title=title;
    this.n=n;
    y=new double[n];
    index=-1;
    length=0;
    color = Color.cyan;
  }

  public int getNoValue () {
    return n;
    }

  public void reset (double value) {
      index = 0;
      length = 1;
      y[index] = value;
  }
  protected void initBounds(double value, double yMin, double yMax) {
    reset (value);
    if (value < yMin || value > yMax || yMin > yMax)
      yMean=this.yMin=this.yMax=value;
```

```
    else {
      this.yMin=yMin;
      this.yMax=yMax;
      yMean=(yMin+yMax)/2;
    }
  }

  public void initGraphics() {
    width = offDimension.width / 3;
    height = offDimension.height - 5;
    bottom=(int)(0.95*height+0.5)-1;
    top=(int)(0.05*height+0.5)-1;
    range=(int)(0.9*height);
    offset=height/2;
    xScale=(double)width/n;
  }

  private void draw(Color color) {
    int i, j, k;

    if (yMax > yMin)
      yScale=range/(yMax-yMin);
        else
          yScale=1;
    if (offGraphics != null) {
      offGraphics.setColor(color);
      i=index;
      k=length;
      while (--k > 0) {
        j=(n+i-1)%n;
        offGraphics.drawLine((int)(xScale*k),
                             offset-(int)((y[i]-yMean)*yScale+0.5),
                             (int)(xScale*(k-1)),
                             offset-(int)((y[j]-yMean)*yScale+0.5));
        i=j;
      }
      offGraphics.setColor (Color.black);
    }
  }

  private void drawLines() {
    if (offGraphics != null) {
      offGraphics.setColor(Color.lightGray);
      offGraphics.drawLine(0,offset,width-1,offset);
      offGraphics.setColor(Color.white);
      offGraphics.drawLine(0,top,width-1,top);
      offGraphics.drawLine(0,bottom,width-1,bottom);
    }
  }

  /* Add values and draw new part of curve  */
  public synchronized void addValue(double value) {
    //System.out.println("addValue(double value)");
    draw(Color.black);            // erase old curve
    drawLines();
    index=(index + 1)%n;
    y[index]=value;
    yMean=(yMin+yMax)/2;
    if (length < n) ++length;
    draw(color);                  // draw new curve
  }

  private void clearParams () {
      if (offGraphics != null) {
```

```
          offGraphics.setColor (getBackground ());
          offGraphics.fillRect
              (width + 8, 0, offDimension.width - width - 8,
               offDimension.height);
      }
  }
  /* Label */
  public void drawParams() {
    Font f;
    FontMetrics fm;
    int x0, y0;
    String s1, s2;
    int w1, w2, wMax;

    if (offGraphics == null) {
        return;
    }
    f=new Font("TimesRoman",Font.PLAIN,10);
    offGraphics.setFont(f);
    fm=offGraphics.getFontMetrics();
    s1=new Double(yMax).toString();
    s2=new Double(yMin).toString();
    w1=fm.stringWidth(s1);
    w2=fm.stringWidth(s2);
    wMax=(w1 <= w2) ? w2 : w1;

    offGraphics.setColor(Color.black);
    x0=width+8+wMax-w1;
    y0=top+f.getSize()/2;
    offGraphics.drawString(s1,x0,y0);

    x0=width+8+wMax-w2;
    y0=bottom+f.getSize()/2;
    offGraphics.drawString(s2,x0,y0);

    x0=width+8;
    y0=offset+f.getSize()/2;
    offGraphics.drawString(title,x0,y0);
  }


  /**
   * Paint the fever on the canvas
   */
  public void paint  (Graphics g) {
    update (g);
  }
  /**
   * Set the preferred size of this canvas
   */
  public Dimension getPreferredSize  () {
    return new Dimension (180, 80);
  }
  /**
   * The minimum size of the IsingFever
   */
  public Dimension getMinimumSize  () {
    return getPreferredSize ();
  }
  /**
   * Print out the graph
   */
  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 ();
      initGraphics ();
      clearParams ();
      drawParams ();
    }
    offGraphics.setColor (Color.black);
    offGraphics.fillRect (0, 0, width, height);
    drawLines ();
    draw (color);
    g.drawImage (offImage, 0, 0, this);
  }

  public Insets getInsets () {
    return new Insets (5, 5, 5, 5);
  }

  public void update (Observable o, Object arg) {
    addValue (getNewValue ());
  }
  /* get new value for the lattice */
  abstract double getNewValue () ;
}
```