```
/*
 * Thermometer.java
 * ----------------
 * Separated and modified by David Wong
 */
import java.awt.*;
import java.awt.event.MouseMotionListener;
import java.awt.event.MouseEvent;
import java.util.Observer;
import java.util.Observable;

/* This class is a GUI for displaying temperature */
class Thermometer extends Canvas implements MouseMotionListener, Observer {
  private Temperature tempValueHolder; // variable that maintain the value
  // of temperature
  private double maxTemp;
  private double minTemp;
  private double critTemp;
  static private final double verticalSpacingRatio = 0.02;
  private int controlRangeMinX;
  private int controlRangeMaxX;
  private int controlRangeMinY;
  private int controlRangeMaxY;

  Thermometer (Temperature tempValueHolder,
               double minTemp, double maxTemp, double critTemp) {
    this.tempValueHolder = tempValueHolder;
    this.maxTemp = maxTemp;
    this.minTemp = minTemp;
    this.critTemp = critTemp;
    controlRangeMinX = 0;
    controlRangeMaxX = 0;
    controlRangeMinY = 0;
    controlRangeMaxY = 0;
    addMouseMotionListener (this);
    tempValueHolder.addObserver (this);
  }


  // set the coord. for the control ranges
  // of the  thermometer
  private void getControlRange (Point thermoLocation, double tubeHeight,
                                double tubeRadius) {
    controlRangeMinX = (int) (thermoLocation.x - tubeRadius);
    controlRangeMaxX = (int) (thermoLocation.x + tubeRadius);
    controlRangeMinY = (int) (thermoLocation.y - tubeHeight);
    controlRangeMaxY = thermoLocation.y;
  }


  public void update (Graphics g) {
        paint (g); // remove flashing
  }

  public void paint (Graphics g) {
    double x,y,r1,r2,h;
    double factor=Math.PI/180;
    int x0,y0,x1,y1,h0,alpha,gamma;
    Dimension mySize = getSize ();
    int wh = mySize.height;

    r1=0.14*mySize.width;
    r2=r1/2;
```

```
    //  h=0.8*wh;
    h = mySize.height * ( 1 - 2 * verticalSpacingRatio ) - r2 -
      r1 - Math.sqrt (r1*r1 - r2 * r2);

    Point thermoLocation = computePosition (mySize, h, r2);
    x = thermoLocation.x;
    y = thermoLocation.y;

    getControlRange (thermoLocation, h, r2);

    alpha=90+(int)(Math.asin(r2/r1)/factor);
    gamma=360-2*(alpha-90);
    x0=(int)(x-r1);
    y0=(int)(y-(r1-r1*Math.cos(factor*(alpha-90))));
    g.setColor(Color.red);
    g.fillArc(x0,y0,(int)(2*r1),(int)(2*r1),alpha,gamma);
    g.setColor(Color.black);
    g.drawArc(x0,y0,(int)(2*r1),(int)(2*r1),alpha,gamma);
    x0=(int)(x-r2);
    y0=(int)(y-h*tempValueHolder.getTemperature ()/maxTemp);
    h0=(int)(h*tempValueHolder.getTemperature ()/maxTemp+
             r1*Math.cos(factor*(alpha-90)));
    g.setColor(Color.red);
    g.fillRect(x0,y0,(int)(2*r2),h0);
    y0=(int)(y-h);
    h0=(int)(h-h*tempValueHolder.getTemperature ()/maxTemp);
    g.setColor(Color.lightGray);
    g.fillRect(x0,y0,(int)(2*r2),h0);
    g.setColor(Color.black);
    g.drawLine(x0,y0,x0,(int)(y0+h));
    g.drawLine(x0+(int)(2*r2),y0,x0+(int)(2*r2),(int)(y0+h));
    y0=(int)(y-h-r2);
    g.setColor(Color.lightGray);
    g.fillArc(x0,y0,(int)(2*r2),(int)(2*r2),0,180);
    g.setColor(Color.black);
    g.drawArc(x0,y0,(int)(2*r2),(int)(2*r2),0,180);
    drawTempCrit (g);
  }
  int TempCritxy[] = new int [4];
  private void drawTempCrit(Graphics g) {
    Font f;
    FontMetrics fm;
    int x, y;
    String s1="T", s2="Crit";

    f=new Font("TimesRoman",Font.PLAIN,12);
    g.setFont(f);
    fm=g.getFontMetrics();
    x= controlRangeMaxX + 6;
    y= controlRangeMaxY - (int)(critTemp/(maxTemp-minTemp)*
                               (controlRangeMaxY - controlRangeMinY));

    TempCritxy[0]= controlRangeMaxX;
    TempCritxy[1]=y-f.getSize()/2;
    g.setColor(new Color(127,0,127));
    g.drawLine(controlRangeMaxX+1,y,controlRangeMaxX+4,y);
    y+=f.getSize()/2;
    g.drawString(s1,x,y);
    x+=fm.stringWidth(s1);
    f=new Font("TimesRoman",Font.PLAIN,12);
    g.setFont(f);
    fm=g.getFontMetrics();
    y+=f.getSize()/2;
    g.drawString(s2,x,y);
```

```
    x+=fm.stringWidth(s2);
    TempCritxy[2]=x;
    TempCritxy[3]=y;
  }
  public Dimension getMinimumSize () {
    return getPreferredSize ();
  }

  public Dimension getPreferredSize () {
    return new Dimension (90, 100);
  }


  boolean isInControlRange (int x, int y) {
    return (controlRangeMinX <= x && x <= controlRangeMaxX
            && controlRangeMinY <= y && y <= controlRangeMaxY);
  }
  public void mouseDragged (MouseEvent e) {
    if (isInControlRange (e.getX (), e.getY())) {
      tempValueHolder.setTemperature
        ((double)(controlRangeMaxY -e.getY ())/
         (double)(controlRangeMaxY - controlRangeMinY)*
         (IsingLattice.TempMax-IsingLattice.TempMin));
      return ;
    } else if (TempCritxy[0]<=e.getX () && e.getX ()<=TempCritxy[2] &&
                  TempCritxy[1]<=e.getY () && e.getY ()<=TempCritxy[3]) {
      tempValueHolder.setTemperature (IsingLattice.TempCrit);
      return ;
    }
  }
  // compute the location of the thermometer
  private Point computePosition (Dimension boxDim, double tubeHeight,
                                 double tubeRadius) {
    double verticalSpacing = verticalSpacingRatio * boxDim.height;
    return new Point
      (boxDim.width/2, (int) (verticalSpacing + tubeRadius + tubeHeight));
  }

  public void mouseMoved (MouseEvent e) {
  }

  public void update (Observable o, Object arg) {
    repaint ();
  }
}
```