```
// Clock Bean Class
// Clock.java

package HTPJB.Clock;

// Imports
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import java.io.Serializable;
import java.util.Calendar;

import javax.swing.JFrame;

public class Clock extends Canvas implements Serializable, Runnable {
  // state & properties
  private int             hour;
  private int             min;
  private int             sec;

  private transient Image     offImage;
  private transient Graphics  offGrfx;
  private transient Thread    clockThread;
  private boolean           raised;
  private boolean           digital;

  // main() for testing
  public static void main(String args[]) {
            JFrame frame = new JFrame();
            frame.setSize(300, 300);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            Clock clock = new Clock(true, false);
            frame.add(clock);
            frame.setVisible(true);
  }

  // Constructors
  public Clock() {
    this(false, false);
    this.setSize(300, 300);
  }

  public Clock(boolean r, boolean d) {
    // Allow the superclass constructor to do its thing
    super();

    // Set properties
    raised = r;
    digital = d;
    // set Background + default size + create/start the clock thread
    this.setBackground(Color.LIGHT_GRAY);
```

```
    this.setForeground(Color.BLUE);
    clockThread = new Thread(this);
    clockThread.start();

  }

  // Accessor methods
  public boolean isRaised() {
    return raised;
  }

  public void setRaised(boolean r) {
    raised = r;
    repaint();
  }

  public boolean isDigital() {
    return digital;
  }

  public void setDigital(boolean d) {
    digital = d;
    repaint();
  }

  // Other public methods
  public void run() {
    while(Thread.currentThread() == clockThread) {
          try {
                      Thread.sleep(1000);
                } catch (InterruptedException e) {
                      e.printStackTrace();
                }

          repaint();
    }

  }

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

  public synchronized void paint(Graphics g) {
    Dimension d = getSize();

    // Create the offscreen graphics context
//      if(offImage == null) {
            // Create the off screen image if it is the first time
//          offImage = createImage(d.width, d.height);
//          offGrfx = offImage.getGraphics();
```

```
//     }
   // To make the clock resizable, create offimage and offGrfx on every update
     offImage = createImage(d.width, d.height);
            offGrfx = offImage.getGraphics();

   // Paint the background with 3D effects
   offGrfx.setColor(getBackground());
   if(raised)
          offGrfx.fill3DRect(10, 10, (int)(d.getWidth() - 20), (int)(d.getHeight() - 20), raised);

   // Paint the clock
   if (digital)
     drawDigitalClock(offGrfx);
   else
     drawAnalogClock(offGrfx);

   // Paint the image onto the screen
   g.drawImage(offImage, 0, 0, null);
 }

 // Private support methods
 private void drawAnalogClock(Graphics g) {

            // Draw the clock shape
                  int centerX = getWidth() / 2;
                  int centerY = getHeight() / 2;
                  int radius = (int)(Math.min(getHeight(), getWidth()) * 0.4);
                  g.setColor(Color.BLACK);
                  g.fillOval(centerX - (int)(2.1 * radius / 2), centerY - (int)(2.1 * radius / 2),
                                    (int)(2.1 * radius), (int)(2.1 * radius));
                  g.setColor(Color.WHITE);
                  g.fillOval(centerX - radius, centerY - radius, 2 * radius, 2 * radius);

            // Draw the hour marks and numbers
                        // find the imaginary circle where the number marks are located
                  g.setColor(getForeground());
                  int fontSize = radius / 5; // try different ratio to find the best relative font size
                  g.setFont(new Font("TimesRoman", Font.PLAIN, fontSize));
                  int radiusOfNumberCircle = (int)(radius - fontSize); // circle where the centers of hour numbers
are located
                        // Draw the hour numbers
                  for(int i = 1; i < 13; i++) {
                        // use font metrics to find the location to draw the hour number string
                        FontMetrics fm = g.getFontMetrics();
                        int height = fm.getAscent();
                        int width = fm.stringWidth(i + "");
                        int numX = (int)(centerX + radiusOfNumberCircle * Math.sin(i * Math.PI / 6) - width /
2.0);
                        int numY = (int)(centerY - radiusOfNumberCircle * Math.cos(i * Math.PI / 6) + height /
2.0 - fm.getLeading());
                        g.drawString(i + "", numX, numY);
```

```
                }

                        // draw hour marks
                g.setColor(getForeground());
                for(int i = 1; i < 61; i++) {
                        int markX1 = (int)(centerX + radius * Math.sin(i * Math.PI / 30)); // (markX1, markY1) is
located on the circle
                        int markY1 = (int)(centerY - radius * Math.cos(i * Math.PI / 30));
                        int markRadius = radius - ((i % 5 == 0) ? radius / 9 : radius / 25);
                        int markX2 = (int)(centerX + markRadius * Math.sin(i * Math.PI / 30));
                        int markY2 = (int)(centerY - markRadius * Math.cos(i * Math.PI / 30));
                        g.drawLine(markX1, markY1, markX2, markY2);
                }

            // Draw the hour hand
            Calendar now = Calendar.getInstance();
            hour = now.get(Calendar.HOUR);
            min = now.get(Calendar.MINUTE);
            sec = now.get(Calendar.SECOND);

            double hours = hour + min / 60.0;
            g.setColor(Color.BLACK);
                g.drawLine(centerX, centerY, (int)(centerX + 0.4 * radius * Math.sin(hours * Math.PI / 6)),
                                (int)(centerY - 0.4 * radius * Math.cos(hours * Math.PI / 6)));

            // Draw the minute hand
                double minutes = min + sec / 60.0;
                g.setColor(Color.BLACK);
                g.drawLine(centerX, centerY, (int)(centerX + 0.65 * radius * Math.sin(min * Math.PI / 30)),
                                (int)(centerY - 0.65 * radius * Math.cos(minutes * Math.PI / 30)));

            // Draw the second hand and center
                g.setColor(Color.RED);
                g.drawLine(centerX, centerY, (int)(centerX + 0.8 * radius * Math.sin(sec * Math.PI / 30)),
                                (int)(centerY - 0.8 * radius * Math.cos(sec * Math.PI / 30)));
           }

  private void drawDigitalClock(Graphics g) {
    Dimension d = getSize();

    // Get the time as a string
    Calendar     now = Calendar.getInstance();
    hour = now.get(Calendar.HOUR);
    min = now.get(Calendar.MINUTE);
    sec = now.get(Calendar.SECOND);
    String dayPeriod = (now.get(Calendar.AM_PM) == Calendar.AM? "AM":"PM");

    // Draw the time
          // Get time string
    StringBuilder timeString = new StringBuilder("");
    timeString.append((hour < 10? "0":"") + hour + ":");
```

```
    timeString.append((min < 10? "0":"") + min + ":");
    timeString.append((sec < 10? "0":"") + sec);
    timeString.append(" " + dayPeriod);

    String time = timeString.toString();
    int fontSize;
    if(d.getHeight() * 1.0 / d.getWidth() > 0.25)
            fontSize  = (int)(d.getWidth() / 6);
    else
            fontSize = (int)(d.getHeight() / 1.5);
    g.setFont(new Font("TimesRoman", Font.PLAIN, fontSize));
    FontMetrics fm = g.getFontMetrics();
            int height = fm.getAscent();
            int width = fm.stringWidth(time);
            int centerX = getWidth() / 2;
            int centerY = getHeight() / 2;
            int stringX = (int) (centerX - width / 2.0);
            int stringY = (int) (centerY + height / 2.0 - fm.getLeading());
            g.setColor(getForeground());
    g.drawString(time, stringX, stringY);
  }
}
```