ClockAnimation

```
// требуется класс StillClock !!!
import by.riit.grafics.StillClock;
import java.awt.event.*;
import javax.swing.*;

public class ClockAnimation extends JFrame {
    private StillClock clock = new StillClock();

    public ClockAnimation() {
        add(clock);

        // Создаем timer с запаздыванием 1000 ms
        Timer timer = new Timer(1000, new TimerListener());
        timer.start();
    }

    private class TimerListener implements ActionListener {
        /**
         * ОБрабатываем событие
         */
        public void actionPerformed(ActionEvent e) {
            // Устанавливаем новое время и перерисовываем
            // часы для отражения текущего времени
            clock.setCurrentTime();
            clock.repaint();
        }
    }

    /**
     * Main method
     */
    public static void main(String[] args) {
        JFrame frame = new ClockAnimation();
        frame.setTitle("ClockAnimation");
        frame.setSize(200, 200);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

StillClock

```
import java.awt.*;
import javax.swing.*;
import java.util.*;

public class StillClock extends JPanel {
    private int hour;
    private int minute;
    private int second;

    /**
     * Construct a default clock with the current time
     */
    public StillClock() {
        setCurrentTime();
    }

    /**
     * Construct a clock with specified hour, minute, and second
     */
    public StillClock(int hour, int minute, int second) {
        this.hour = hour;
        this.minute = minute;
        this.second = second;
    }

    /**
     * Return hour
     */
    public int getHour() {
        return hour;
    }

    /**
     * Set a new hour
     */
    public void setHour(int hour) {
        this.hour = hour;
        repaint();
    }

    /**
     * Return minute
     */
    public int getMinute() {
        return minute;
    }

    /**
     * Set a new minute
     */
    public void setMinute(int minute) {
        this.minute = minute;
        repaint();
    }

    /**
     * Return second
     */
    public int getSecond() {
        return second;
    }

    /**
     * Set a new second
```

```
     */
    public void setSecond(int second) {
        this.second = second;
        repaint();
    }

    /**
     * Draw the clock
     */
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Initialize clock parameters
        int clockRadius =
                (int) (Math.min(getWidth(), getHeight()) * 0.8 * 0.5);
        int xCenter = getWidth() / 2;
        int yCenter = getHeight() / 2;

        // Draw circle
        g.setColor(Color.black);
        g.drawOval(xCenter - clockRadius, yCenter - clockRadius,
                2 * clockRadius, 2 * clockRadius);
        g.drawString("12", xCenter - 5, yCenter - clockRadius + 12);
        g.drawString("9", xCenter - clockRadius + 3, yCenter + 5);
        g.drawString("3", xCenter + clockRadius - 10, yCenter + 3);
        g.drawString("6", xCenter - 3, yCenter + clockRadius - 3);

        // Draw second hand
        int sLength = (int) (clockRadius * 0.8);
        int xSecond = (int) (xCenter + sLength *
                Math.sin(second * (2 * Math.PI / 60)));
        int ySecond = (int) (yCenter - sLength *
                Math.cos(second * (2 * Math.PI / 60)));
        g.setColor(Color.red);
        g.drawLine(xCenter, yCenter, xSecond, ySecond);

        // Draw minute hand
        int mLength = (int) (clockRadius * 0.65);
        int xMinute = (int) (xCenter + mLength *
                Math.sin(minute * (2 * Math.PI / 60)));
        int yMinute = (int) (yCenter - mLength *
                Math.cos(minute * (2 * Math.PI / 60)));
        g.setColor(Color.blue);
        g.drawLine(xCenter, yCenter, xMinute, yMinute);

        // Draw hour hand
        int hLength = (int) (clockRadius * 0.5);
        int xHour = (int) (xCenter + hLength *
                Math.sin((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
        int yHour = (int) (yCenter - hLength *
                Math.cos((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
        g.setColor(Color.green);
        g.drawLine(xCenter, yCenter, xHour, yHour);
    }

    public void setCurrentTime() {
        // Construct a calendar for the current date and time
        Calendar calendar = new GregorianCalendar();

        // Set current hour, minute and second
        this.hour = calendar.get(Calendar.HOUR_OF_DAY);
        this.minute = calendar.get(Calendar.MINUTE);
        this.second = calendar.get(Calendar.SECOND);
    }

    public Dimension getPreferredSize() {
```

```
        return new Dimension(200, 200);
    }
}
```