```java
import java.awt.Color;
import java.awt.Graphics;
public class Timestamp
{
    private int x;
    private int y;
    private Color color;
    private String value;

    public Timestamp(int x, int y, Color color, String value)
    {
        this.x = x;
        this.y = y;
        this.color = color;
        this.value = value;
    }

    public void draw(Graphics g)
    {
        g.setColor(color);
        g.drawString(value, x, y);
    }
}
```

```java
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Point;
import java.util.ArrayList;
public class Chronometer
{
    // le temps écoulé en dixièmes de secondes
    private int time;

    private ArrayList<Timestamp> alTimestamps = new ArrayList<>();

    public Chronometer()
    {
        time = 0;
    }

    public void reset()
    {
        time = 0;
        alTimestamps.clear();
    }

    public void nextTick()
    {
        time++;
    }

    public int getTime()
    {
        return time;
    }

    public int getTenthsOfSeconds()
    {
        return time % 10;
    }

    public int getSeconds()
    {
        return (time / 10) % 60;
    }

    public int getMinutes()
    {
        return (time / 600) % 60;
    }

    public int getHours()
    {
        return (time / 36000);
        // rajouter %24 si on sépare les heures d'une journée et les jours
    }

    public String toString()
    {
        return getHours() + " hours, " + getMinutes() + " minutes, " + getSeconds() + " seconds, " + getTenthsOfSeconds() + "/10 seconds";
    }

    public Color generateRandomColor()
    {
        int r = (int)(Math.random()*256);
        int g = (int)(Math.random()*256);
        int b = (int)(Math.random()*256);
        int a = (int)(Math.random()*256);

        return new Color(r,g,b,a);
    }

    public void markTime(Point position)
    {
        Color color = generateRandomColor();
        String value = toString();
        Timestamp t = new Timestamp(position.x, position.y, color, value);
        alTimestamps.add(t);
    }

    public void draw(Graphics g)
    {
        for (int i=0; i<alTimestamps.size();i++)
            alTimestamps.get(i).draw(g);
    }
}
```

```java
import java.awt.Color;
import java.awt.Graphics;
public class DrawPanel extends javax.swing.JPanel
{
    private Chronometer chronometer = null;

    public void setChronometer(Chronometer chronometer)
    {
        this.chronometer = chronometer;
    }

    public void paintComponent(Graphics g)
    {
        g.setColor(Color.white);
        g.fillRect(0, 0, getWidth(), getHeight());

        if (chronometer != null)
            chronometer.draw(g);
    }

    public DrawPanel()
    {
        initComponents();
    }
// Skipped: ... initComponents { ... }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    // End of variables declaration//GEN-END:variables
}
```

```java
import javax.swing.Timer;
public class MainFrame extends javax.swing.JFrame
{
    private Chronometer chrono = null;
    private Timer timer = null;

    public MainFrame()
    {
        initComponents();
        chrono = new Chronometer();
        drawPanel.setChronometer(chrono);

        timer = new Timer(100, stepButton.getActionListeners()[0]);

        stepButton.setVisible(false);
        stopButton.setEnabled(false);
        resetButton.setEnabled(false);
        updateView();
    }

    public void updateView()
    {
        chronoLabel.setText(String.valueOf(chrono.getTime()));
        betterChronoLabel.setText(chrono.toString());

        repaint();
    }
// Skipped: ... initComponents { ... }
    private void startButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_startButtonActionPerformed
        timer.start();
        startButton.setEnabled(false);
        stopButton.setEnabled(true);
        resetButton.setEnabled(true);
    }//GEN-LAST:event_startButtonActionPerformed

    private void stopButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_stopButtonActionPerformed
        timer.stop();
        startButton.setEnabled(true);
        stopButton.setEnabled(false);
    }//GEN-LAST:event_stopButtonActionPerformed

    private void resetButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_resetButtonActionPerformed
        timer.stop();
        chrono.reset();
        startButton.setEnabled(true);
        stopButton.setEnabled(false);
        resetButton.setEnabled(false);
        updateView();
    }//GEN-LAST:event_resetButtonActionPerformed

    private void stepButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_stepButtonActionPerformed
        chrono.nextTick();
        updateView();
    }//GEN-LAST:event_stepButtonActionPerformed

    private void drawPanelMousePressed(java.awt.event.MouseEvent evt)//GEN-FIRST:event_drawPanelMousePressed
    {//GEN-HEADEREND:event_drawPanelMousePressed
        if (timer.isRunning())
        {
            chrono.markTime(evt.getPoint());
            repaint();
        }
    }//GEN-LAST:event_drawPanelMousePressed
// Skipped: ... Look & Feel
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel betterChronoLabel;
    private javax.swing.JLabel chronoLabel;
    private DrawPanel drawPanel;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JButton resetButton;
    private javax.swing.JButton startButton;
    private javax.swing.JButton stepButton;
    private javax.swing.JButton stopButton;
    // End of variables declaration//GEN-END:variables
}
```