

LAB EXERCISE X6 – SCRIBBLE

Task: http://www.tfh-berlin.de/~ischmied/Inf1/Exercises/X6_Scribble.html

Prep Questions

I made the report after I programmed Scribble. And I answered these questions while making the report, so they are more after-prep-questions.

? Are you going to call drawLine in the paint method or elsewhere?

! The method drawLine() needs to be implemented with an event handler. But the method paint() does not work together with handlers. So I put drawLine() into the mouseDragged, where it can get the coordinates of the mouse as parameters.

? Will your drawings survive repaint calls?

! I hope they will not. I call repaint() to erase my drawings...

? How will your drawings react to window resizing? Will they grow and shrink accordingly?

! The drawings will disappear immediately on every change on window size, because our drawings are not stored permanently somewhere.

? Do you expect the opening of the combo box to cause any problems?

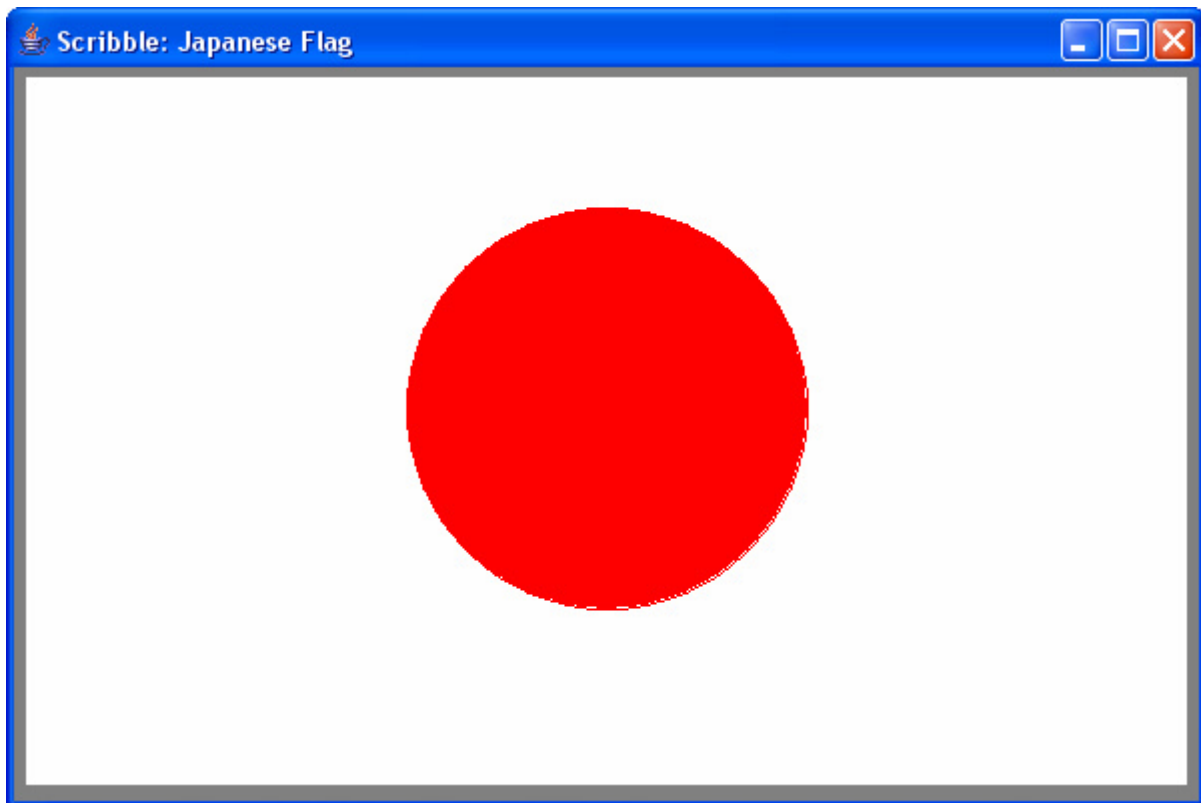
! Yes, that can cause problems. If the opened combo box is a layer over the drawings it will erase them, because the drawings are not permanently stored. That is a problem of our drawings: if it is invisible (by something over it or minimizing or...) it will be erased.

? How will you implement the clear button handler, i.e. what does the actionPerformed method do?

! This is a very short answer: just the method repaint() will do this for us.

Voluntary pre-lab: Japanese Flag

The pre-lab was relatively easy. I was able to finish it directly in lab.



My class extended JFrame and the window/GUI is implemented in the constructor. In the main method I instantiated my own class, so the constructor was called. The drawing itself was implemented in the method paint(). The complete source code:

```
package scribble;

import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JFrame;

public class JapanFlag extends JFrame {

    public JapanFlag() {
        this.setTitle("Scribble: Japanese Flag");
        this.setSize(600, 400);
        this.setBackground(Color.gray);
        this.setVisible(true);
    }

    public static void main (String [] args) {
        new JapanFlag();
    }

    public void paint(Graphics g) {
        g.setColor(Color.white);
        g.fillRect(10, 35, 580, 353);
        g.setColor(Color.red);
        g.drawOval(200, 100, 200, 200);
        g.fillOval(200, 100, 200, 200);
    }
}
```

Scribble Program Setup

As described in our task I defined two classes: ScribbleFrame and ScribblePanel. On the one ScribbleFrame provides the GUI and contains the main method, one the other hand ScribblePanel provides the MouseListeners.

```
package scribble;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;

import javax.swing.*;

public class ScribbleFrame extends JFrame
{
    ScribblePanel panel;

    public ScribbleFrame(ScribblePanel sp) {
        panel = sp;
        init();
        this.setTitle("Scribble 1.0");
        this.setSize(300, 400);
        this.setBackground(Color.gray);
        this.setVisible(true);
    }

    public static void main (String [] args)
    {
        ScribblePanel sPanel = new
        ScribblePanel();
        new ScribbleFrame(sPanel);
    }

    private void init() {
        setLayout(new BorderLayout());
        ...
    }
}
```

```
package scribble;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class ScribblePanel extends JPanel
implements MouseMotionListener,
MouseListener {

    private int x1, y1, x2, y2;
    public String currentColor = "schwarz";

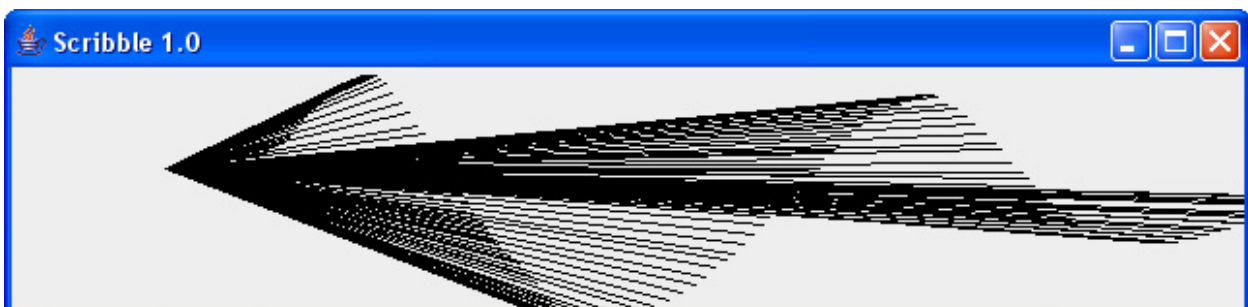
    public ScribblePanel() {
        addMouseListener(this);
        addMouseMotionListener(this);
    }

    public void mouseDragged (MouseEvent e) {
        Graphics g = getGraphics();
        ...
    }
}
```

A lot of imports, as you can see...

Drawing

I introduced four attributes of type integer to work with the mouse coordinates. In mousePressed() I stored the starting coordinates. In mouseDragged() I got the current position of the mouse and could draw a line with these four coordinates. Then I updated the first two coordinates. Without updating the coordinates something like that would be normal (the line starting point is always the same):



The other handler methods can stay empty, I do not need to implement them.

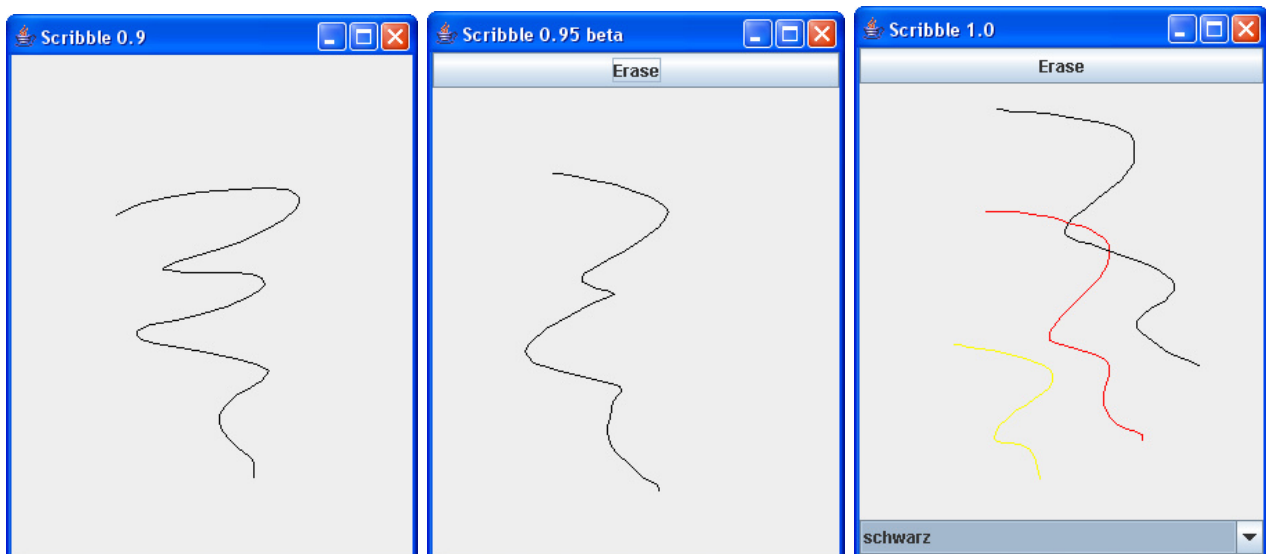
Erase button

I created a new JButton called "erase" and registered an anonymous listener to it. The method actionPerformed() contains only repaint() to erase all drawings. The anonymous listener is recognizable by the characteristic }}); at the end:

```
JButton erase = new JButton ("Erase");
erase.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent a) {
        repaint();
    }
});
```

Color Choice

Now its time to show my Scribble. With these three screenshots you can see the development through the different tasks:



I added the color JComboBox to the SOUTH of the BorderLayout. The different colors are handled as of type String. In mousePressed() I checked the currentColor and changed it when necessary by calling the method setColor().

The first extra work did I already, but this was not really extra work. I think it would be also easy to implement the ColorChooser, but because of all our exams in this weeks I did not want to use my time for it.

Source Code: I saved my whole project in Scribble.zip, there you can find the source code of all classes.

As every time some people helped me: Moritz Liepe gave me some tips for the Japanese flag and Henrike Lode explained problems concerning Scribble in Tutorium.

Time spent on lab and report: about 5 hours