Java 实现画笔小程序实验报告

软件学院软件工程专业 2019 级 2 班 韦诗睿 1913184

一、实验内容

（一）主要内容：使用 Java GUI 的相关知识做一个画笔小程序。

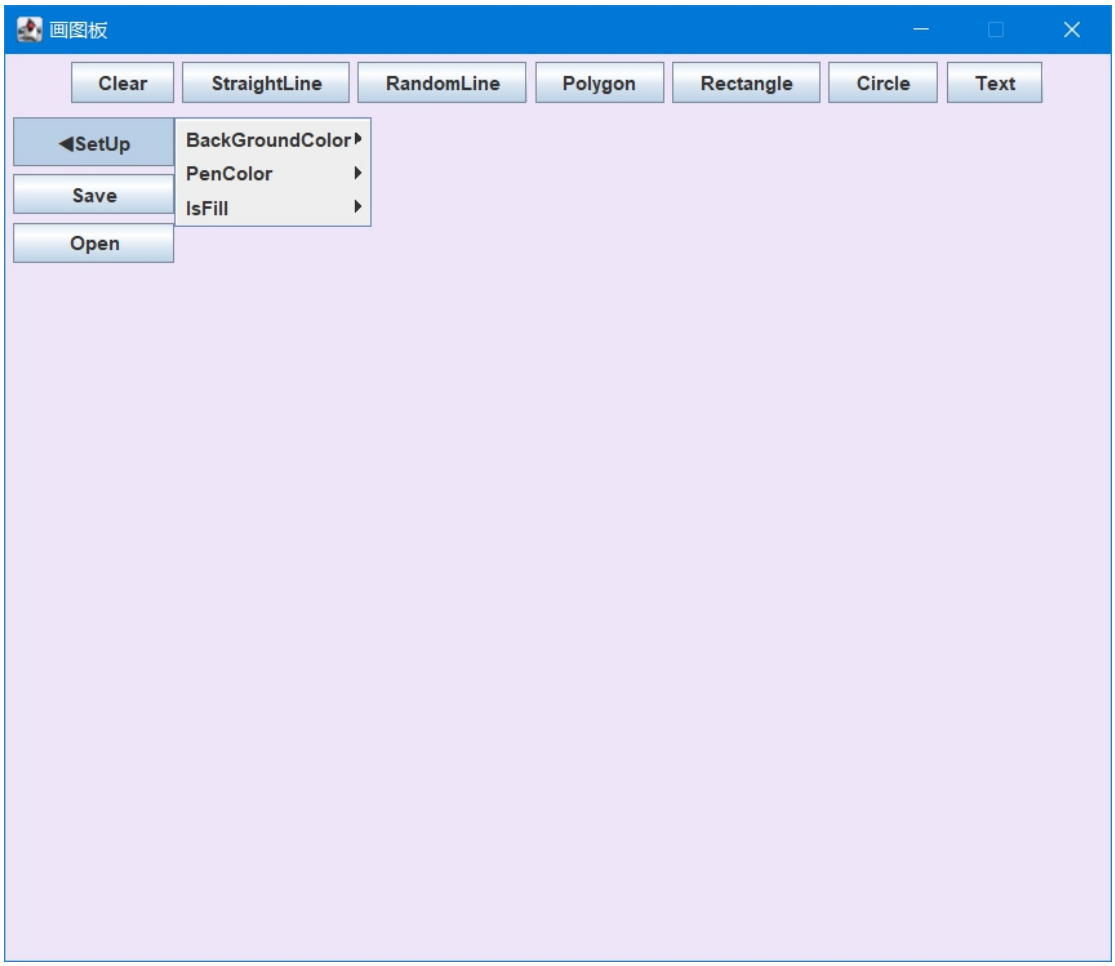
（二）基本功能：

1. 提供直线、矩形、椭圆等形状，及文字的绘制。
2. 背景色和前端颜色可选。
3. 图形的绘制可以选择填充模式或者非填充模式。
4. 图形可以保存，并且可以打开已保存的图形文件（采用对象输入和输出流）。

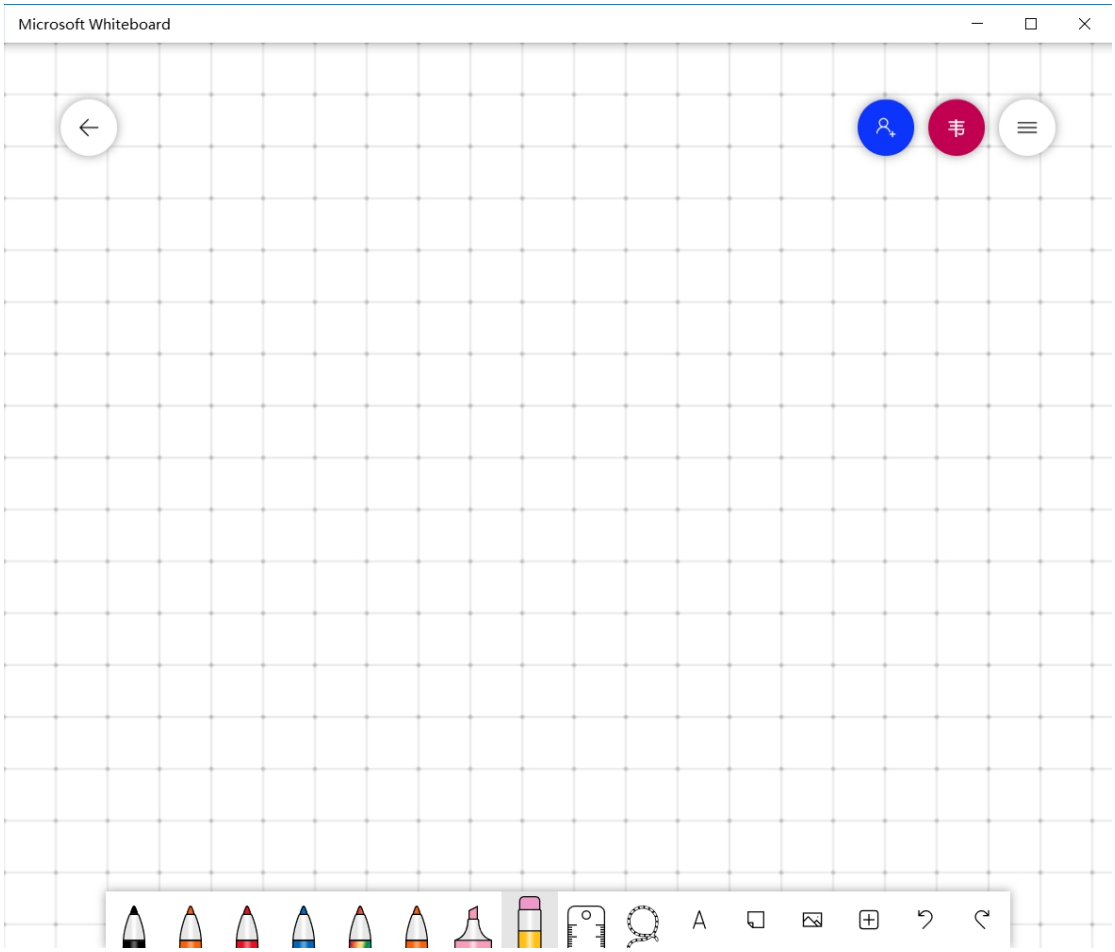
二、设计分析

（一） 页面设计：

1. 页面展示：



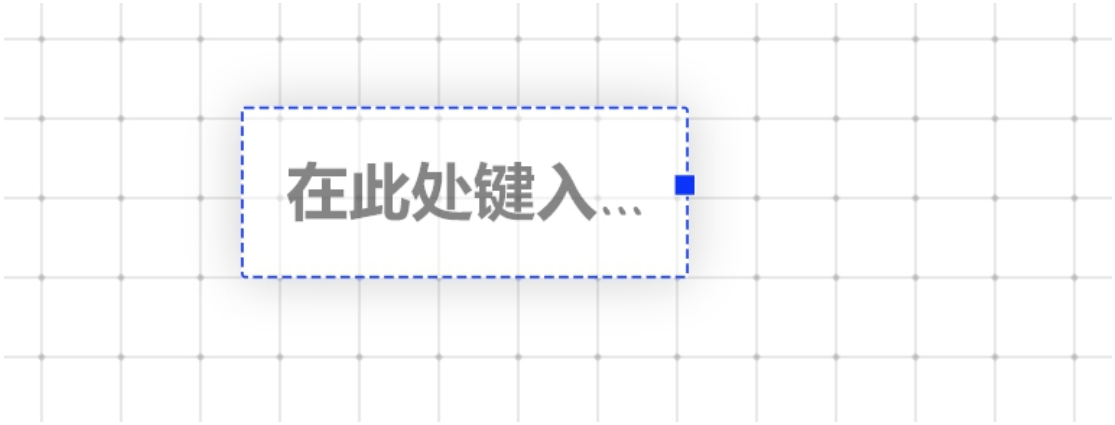
2. 设计思路：参考 Microsoft Whiteboard 和 OneNote for Win10 的页面进行设计。



（1）工具栏面板：

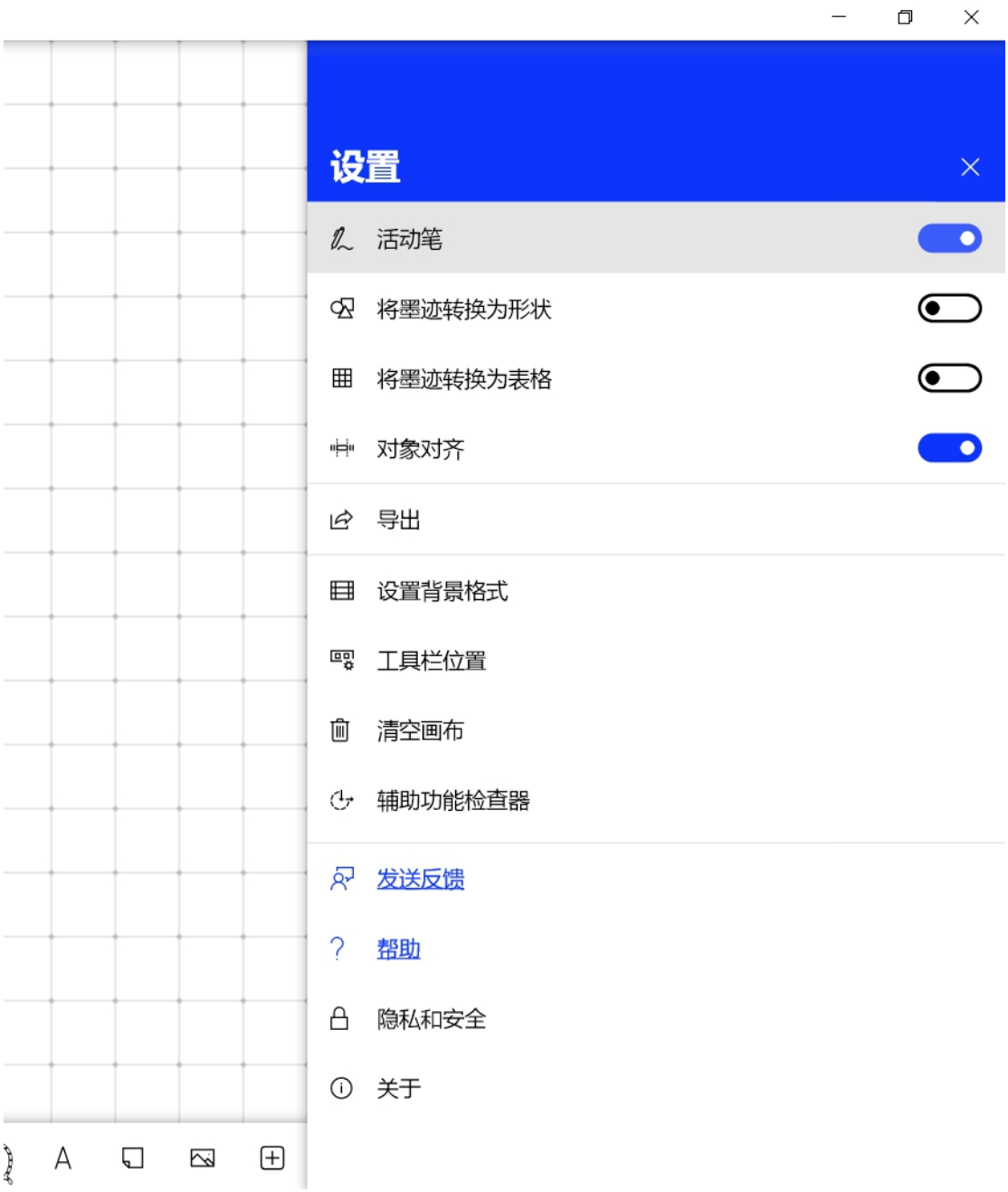
考虑到画板的简易性且无需用多只画笔，将画笔设计转变为绘制不同形状的工具，根据需求设计有直线（StraightLine）、曲线（RandomLine）、多边形（Polygon）、矩形（Rectangle）、圆形（Circle）、文本（Text）、清空（Clear）七种工具。

（2）文字效果：参考 Microsoft Whiteboard，在某处点击后即可键入。



（3）设置面板：

参考 Microsoft Whiteboard，将页面颜色、画笔颜色、是否填充设置和打开文件、保存文件放置在一起作为设置选项。



（二）代码设计：

代码主体部分分为 3 类，分别为存储图形的 Shape 类，实现事件监听的 ButtonListener 类和 ColorButton 类，主程序面板设计 drawline 类。

1. 主程序面板设计：

（1）框架：JFrame 作为画图窗口

（2）面板：3 个 JPanel，分为画图面板、图形选项面板和设置面板。在图形选项面板和设置面板设置一系列功能按钮，通过绑定事件监听器对画图面板进行一系列操作。

1）设置面板：采用弹出式菜单设计，点击后出现三个菜单项，每个分别设置各自的子菜单。

2. 事件监听：

对一系列功能的命令进行响应。

（1）命令按钮（ButtonListener）响应：对清空、打开文件、保存文件、是否填充的按钮设置进行响应。

（2）颜色按钮（ColorButton）响应：设置页面背景颜色。

（3）鼠标响应：

1）鼠标拖动：绘制曲线

2）鼠标点击：绘制多边形，在最后一个点双击即可进行封闭图形连线；设置文本输入位置

3）鼠标按压和松开：绘制曲线、直线、矩形、圆形。

（4）键盘响应：输入 Enter 键即可绘制输入的文字

3. 存储图形：实现重绘功能。

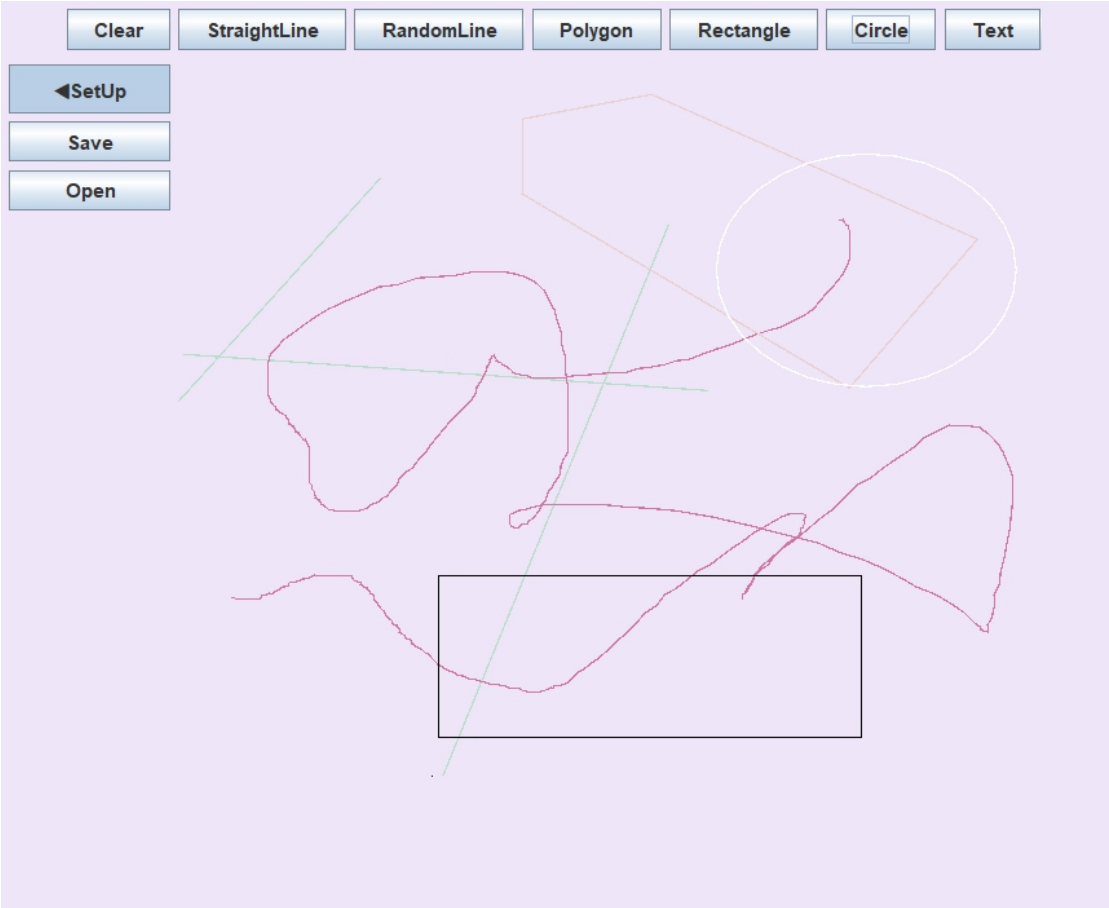
在对面板进行更改时，为了重绘画上的图形，需要对面板上绘制的形状进行保存。设置形状类 Shape 对绘制图形的种类、绘制位置、绘制的颜色、是否填充的设置进行保存，并采用 ArrayList 的数据结构存储。

在保存文件时，也是采取保存形状 ArrayList 的方式而不是保存为 jpg 等图片的形式。在打开文件时即调用重绘函数对保存的 ArrayList 里的图案进行绘制，达到打开文件的效果。

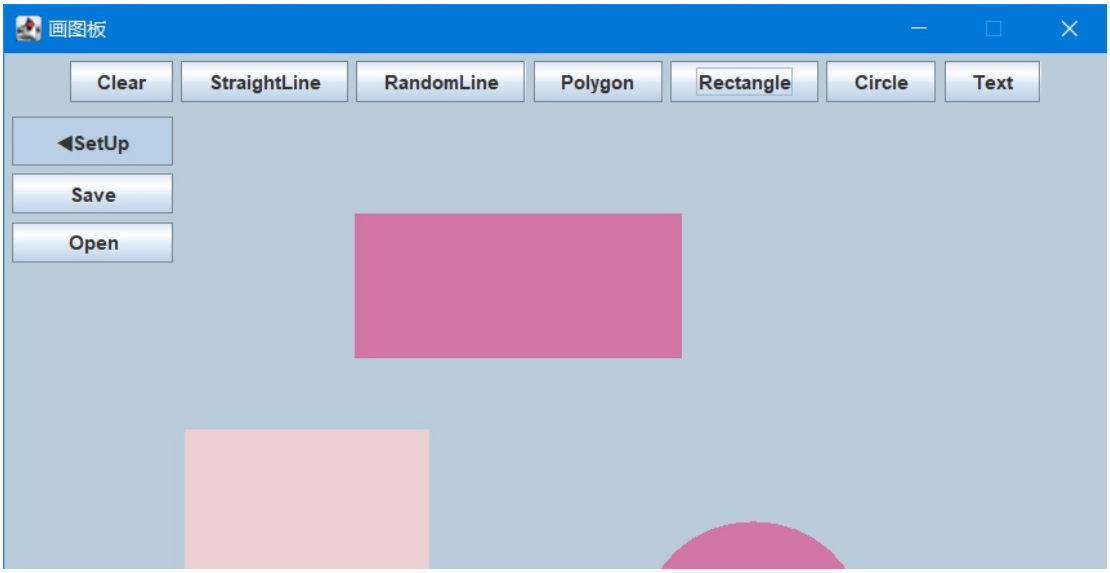
三、效果展示：

1. 形状绘制：

（1）包含直曲线、图形（未填充）绘制，可以对画笔进行不同颜色的设置



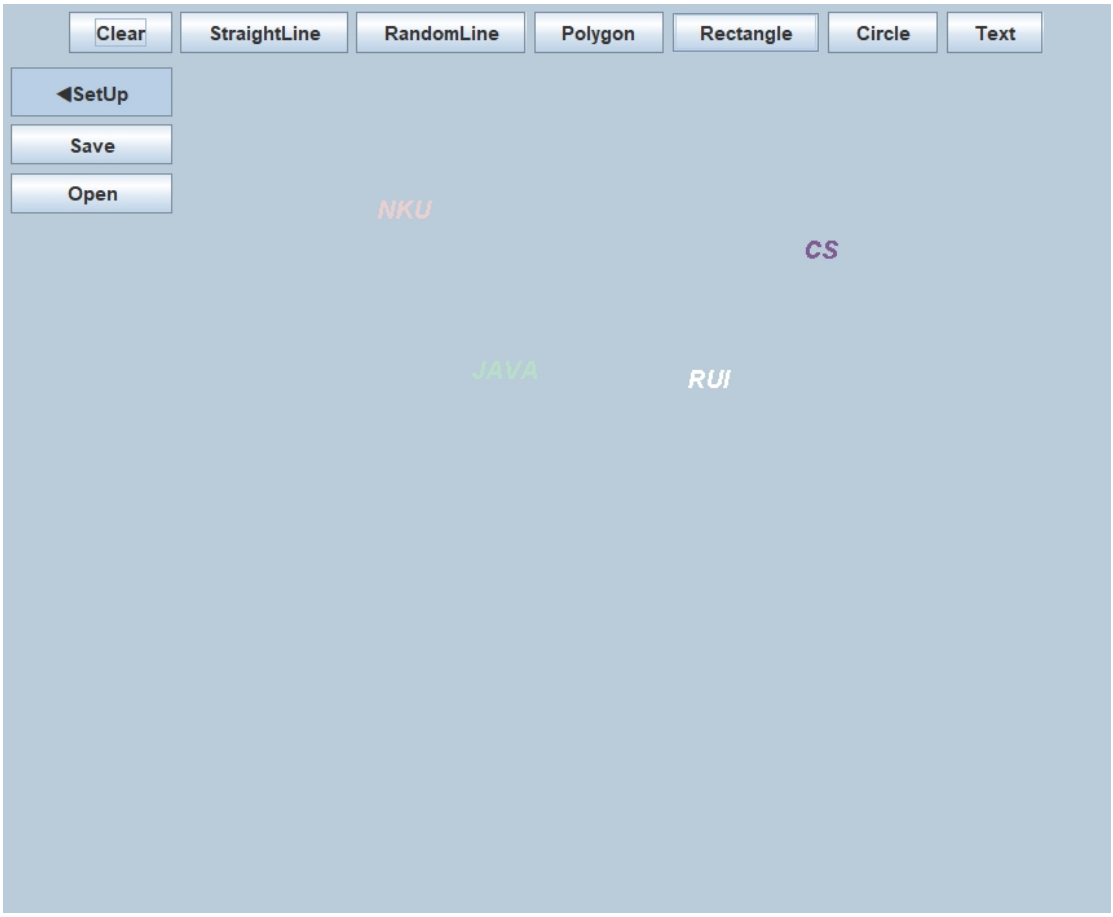
（2）绘制图形填充：图形填充颜色和边框颜色保持相同。



（3）文字效果：

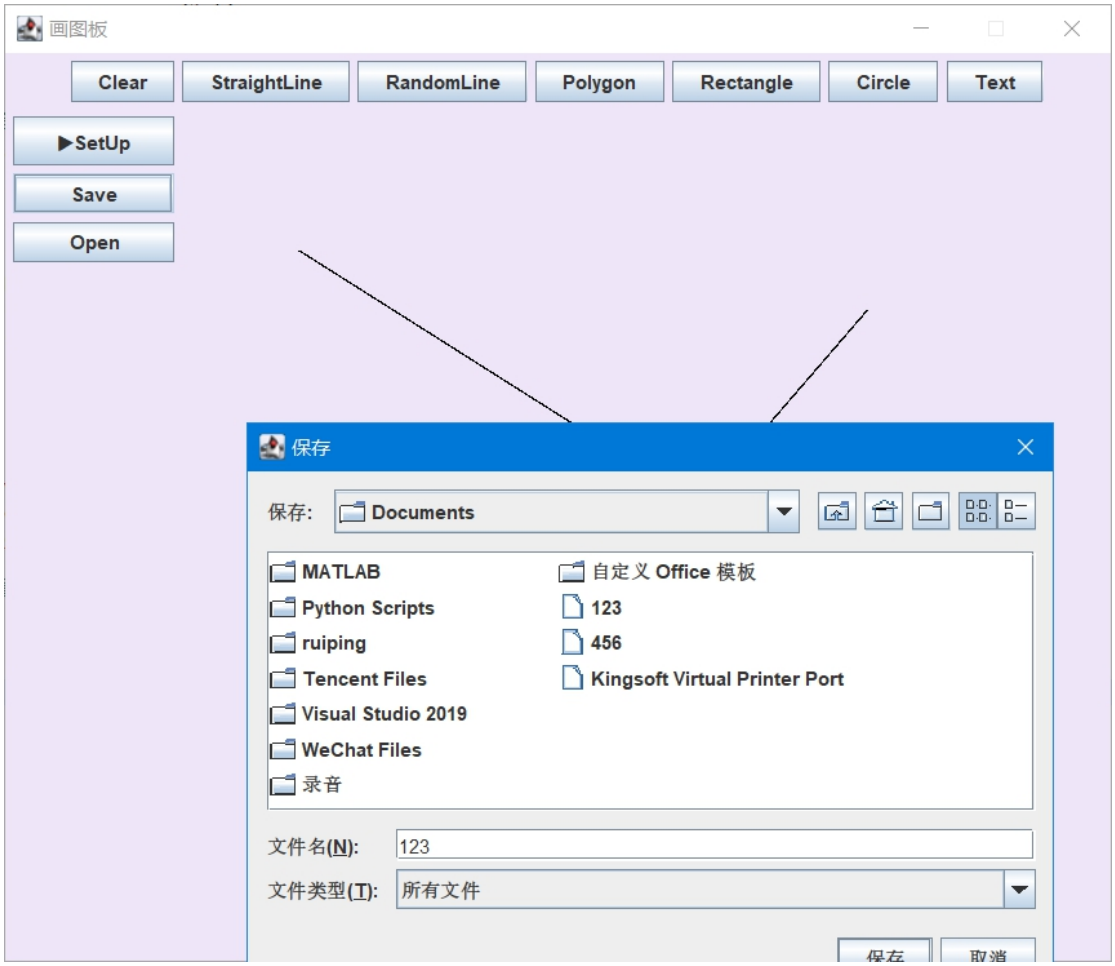
输入效果：

绘制效果：

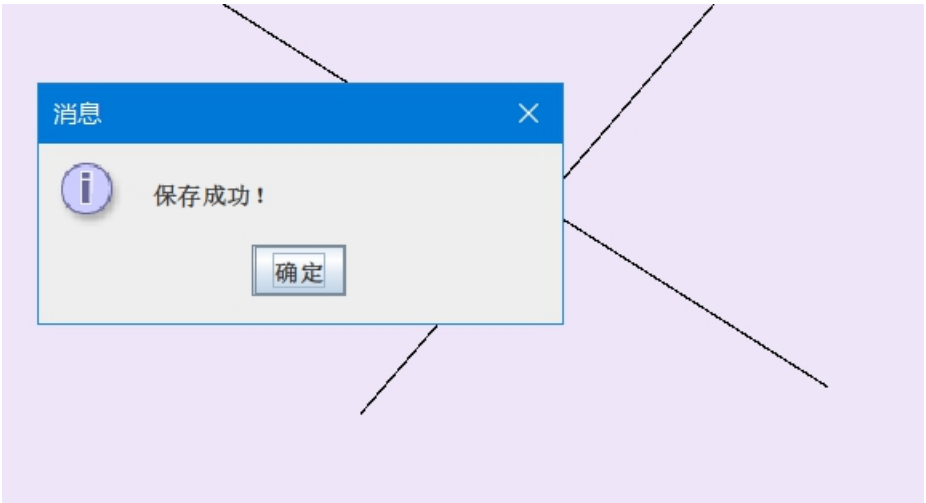


2.文件操作：

（1）保存文件：

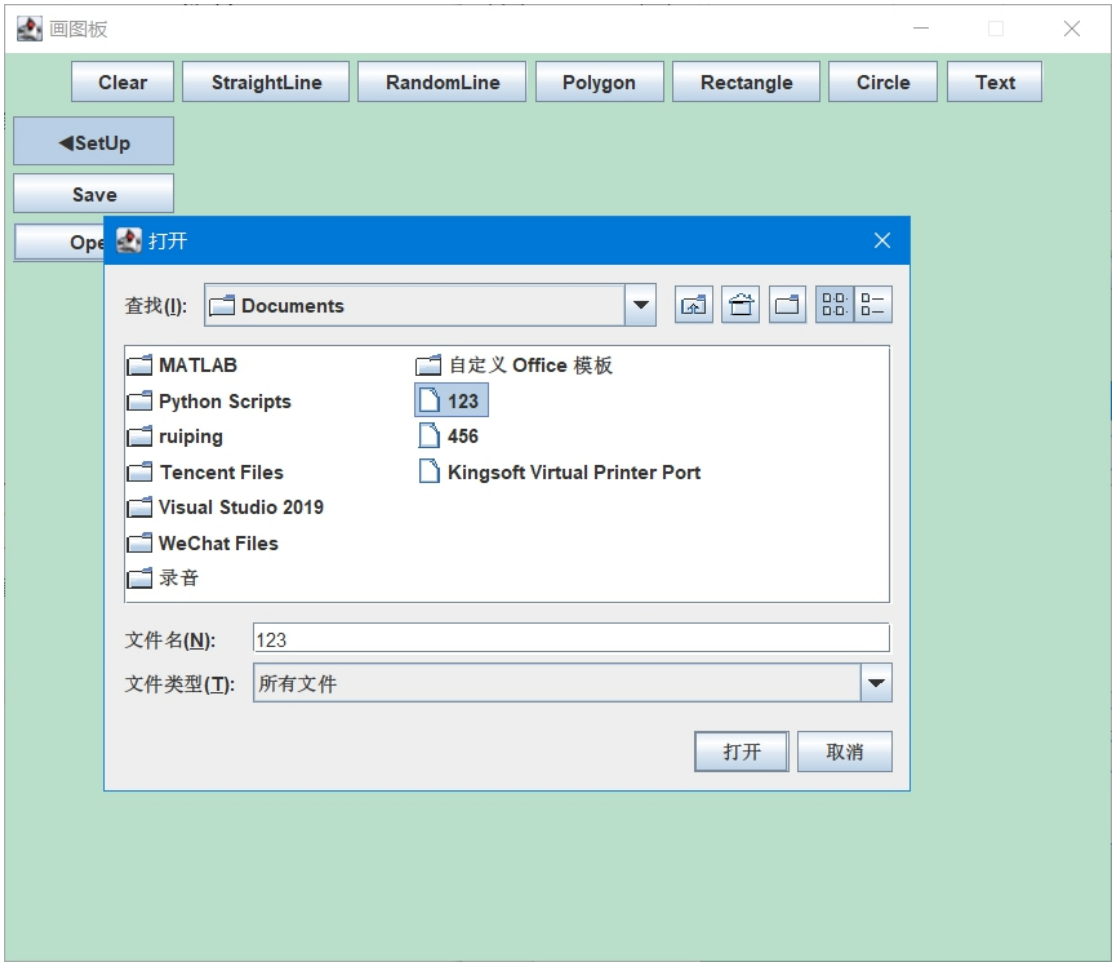


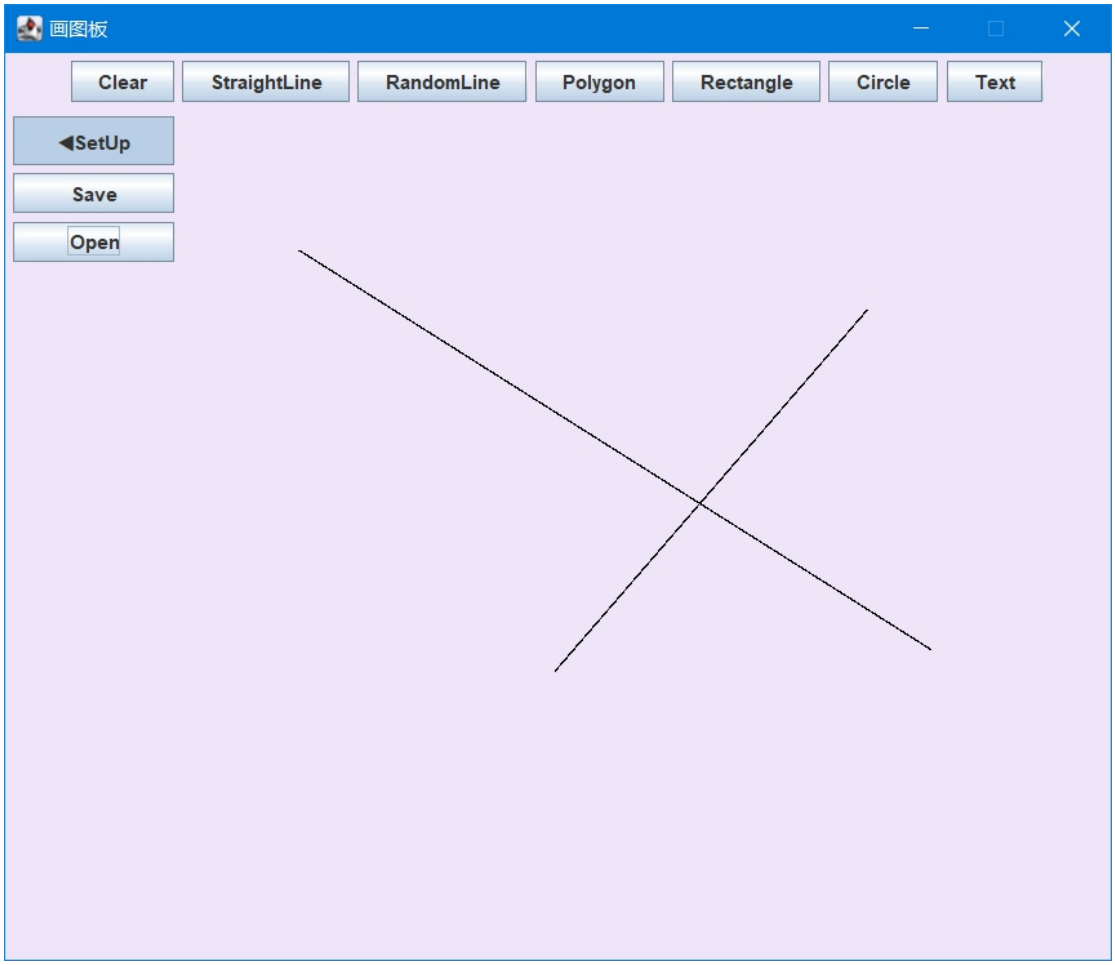
保存成功消息提示：



（3）打开文件：

保存时连同背景颜色一起保存。

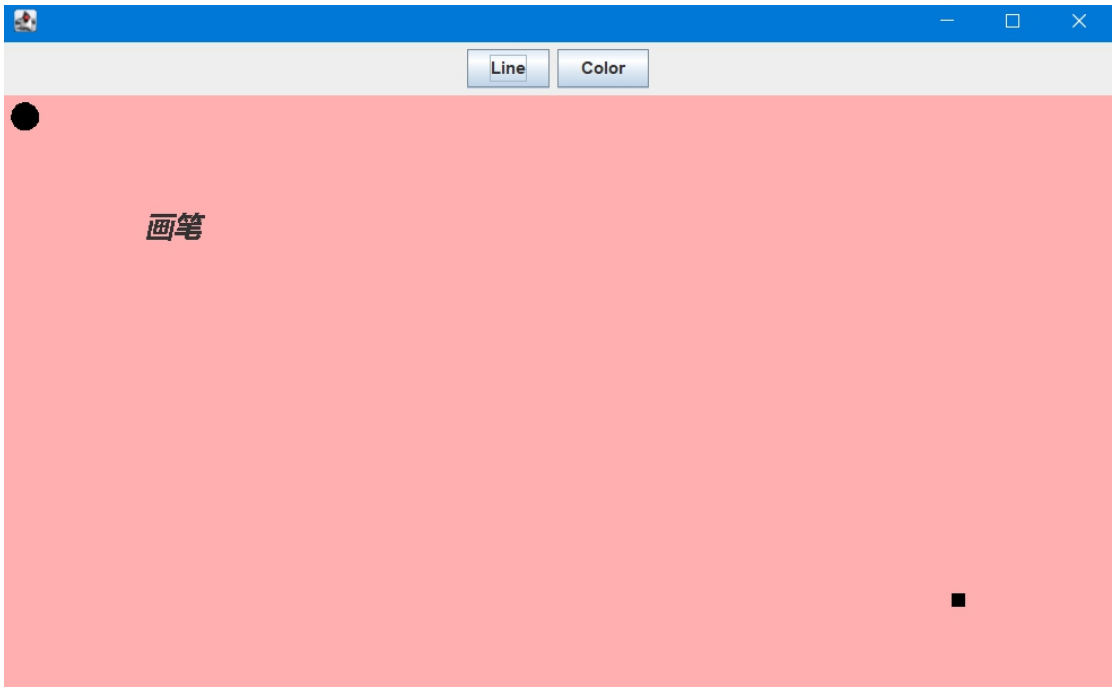




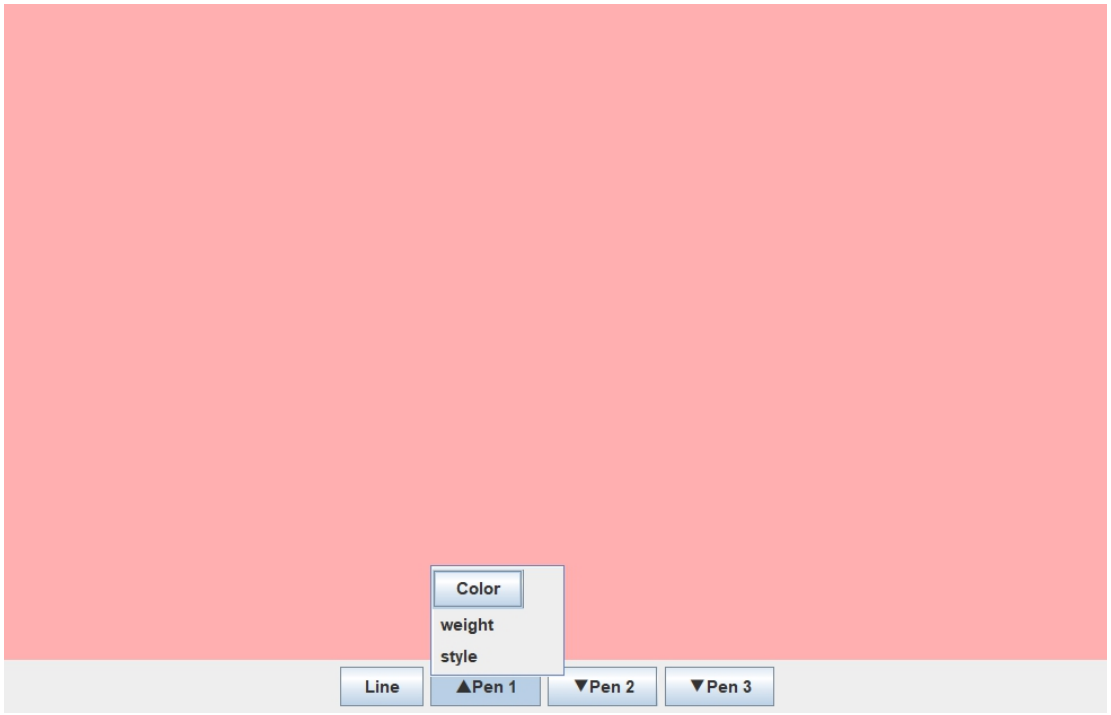
四、心得体会

一开始虽然在上机课上跟着一步一步敲了代码，但回头自己动手的时候才发现整体框架并没有弄清楚，在如何处理不同面板上的按钮之间的关系和动作响应上疑惑了很久。其次是调整布局令人精疲力尽，有点像当初调整 CSS、Html 的感觉。由于对布局的不熟悉也导致 Debug 了很长时间，重绘的时候线段的位置老是不对，以为是布局的问题，先后尝试了 flowlayout、null、borderlayout 都没有效果，结果发现是保存形状的位置坐标写错了，发现的时候令人无奈。还有对于布局设置的 setsize 和 setpreferredsize 上没有事先弄清楚，导致画图时一直画不出来，一行行的检查后才发现在 flowlayout 下使用了 setsize。诸如此类问题让我在写本次大作业的时候耗时很多但做出来的效果与自己预想的相差甚远……但回过头来究其原因还是因为自己对 GUI 的基础知识掌握并不熟也没有进行其他的动手实践，在后期逐渐体会到面板、框架和事件响应之间的关系后速度便快了很多。期待之后能学到页面效果更好的 GUI 框架。

附上最开始的样子的记录（试图模仿 Microsoft Whiteboard）



开始调按钮：



五、源代码：

GitHub:*https://github.com/RuiNov1st/2021NKU_java_course*

1. 形状存储类：

```
1.    //利用形状类进行画图
2.    class Shape implements Serializable{
3.        int x1,x2,y1,y2;//坐标
4.        String name;//类型
5.        String s;//文字
6.        Color c;//记录颜色
7.        boolean isfill;//记录是否填充
8.        Color bgc;//记录背景颜色
9.        public Shape(int x1,int y1,int x2,int y2,String name,Color c,boolean isfill,C
    olor bgc) {
10.         this.x1 = x1;
11.         this.x2 = x2;
12.         this.y1 = y1;
13.         this.y2 = y2;
14.         this.name = name;
15.         this.c = c;
16.         this.isfill = isfill;
17.         this.bgc = bgc;
18.       }
19.       public Shape(int x1,int y1,String name,String s,Color c,Color bgc) {
20.         this.x1 = x1;
21.         this.y1 = y1;
22.         this.name = name;
23.         this.s = s;
24.         this.c = c;
25.         this.bgc = bgc;
26.       }
27.       //在形状类中进行重绘操作
28.       public void repaint(Graphics g) {
29.         g.setColor(c);
30.         Font font = new Font("华文行楷",Font.BOLD + Font.ITALIC,15);
31.     g.setFont(font);
32.         switch(name) {
33.         //直线
34.         case "StraightLine":
35.           g.drawLine(x1, y1, x2, y2);
36.           break;
37.         //曲线
38.         case "RandomLine":
39.           g.drawLine(x1, y1, x2, y2);
```

```
40.          break;
41.         //矩形
42.         case "Rectangle":
43.          if(isfill) {
44.           g.fillRect(Math.min(x1,x2), Math.min(y1, y2), Math.abs(x1-x2), Math.ab
    s(y1-y2));
45.          }else {
46.          g.drawRect(Math.min(x1,x2), Math.min(y1, y2), Math.abs(x1-x2), Math.a
    bs(y1-y2));
47.          }
48.          break;
49.         //圆形
50.         case "Circle":
51.          if(isfill) {
52.           g.fillOval(x1, y1, x2, y2);
53.          }else {
54.          g.drawOval(x1, y1, x2, y2);
55.          }
56.          break;
57.         //多边形
58.         case "Polygon":
59.          g.drawLine(x1, y1, x2, y2);
60.          break;
61.         //文本
62.         case "Text":
63.          g.drawString(s,x1,y1);
64.          break;
65.         }
66.
67.       }
68.   }
```

2. 事件监听：

```
1.    import javax.imageio.ImageIO;
2.    import javax.swing.*;
3.
4.    import java.awt.*;
5.    import java.awt.event.*;
6.    import java.awt.geom.Ellipse2D;
7.    import java.awt.geom.Line2D;
8.    import java.awt.image.BufferedImage;
9.    import java.io.File;
10.   import java.io.FileInputStream;
11.   import java.io.FileOutputStream;
12.   import java.io.ObjectInputStream;
```

```
13.  import java.io.ObjectOutputStream;
14.  import java.io.OutputStream;
15.  import java.io.Serializable;
16.  import java.util.ArrayList;
17.  import java.util.HashMap;
18.
19.  class ButtonListener implements ActionListener,MouseListener,MouseMotion
     Listener {
20.   String s = "";
21.   boolean flag = false;
22.   private int index=0;
23.   //private Shape ShArr[];
24.   private ArrayList<Shape>ShArr;
25.   //public ArrayList<Shape>ShArr;
26.   private JPanel jp;
27.   private java.awt.Graphics gr;
28.   private String commend="";
29.   boolean flag2 = false;
30.   int x0=0,y0=0,x1=0,y1=0,x2=0,y2=0,x3=0,x4=0,y3=0,y4=0,start_x=0,start_y
     =0;
31.   public void set_jp(JPanel jp) {
32.    this.jp=jp;
33.    }
34.   public void set_gr(java.awt.Graphics G) {
35.    this.gr = G;
36.   }
37.   public void set_ShArr(ArrayList<Shape>ShArr) {
38.    this.ShArr = ShArr;
39.   }
40.   public int get_len() {
41.    return index;
42.   }
43.
44.   public void openpic() {
45.   //弹出选择对话框，选择需要读入的文件
46.    JFileChooser chooser = new JFileChooser();
47.    chooser.showOpenDialog(null);
48.    File file =chooser.getSelectedFile();
49.   //如果为选中文件
50.   if(file==null){
51.    JOptionPane.showMessageDialog(null, "没有选择文件");
52.   }
53.   else {
54.    try {
```

```
55.      ShArr.clear();
56.      //选中了相应的文件，则柑橘选中的文件创建对象输入流
57.      FileInputStream fis = new FileInputStream(file);
58.      ObjectInputStream ois = new ObjectInputStream(fis);
59.      //将读出来的对象转换成父类对象的容器进行接收
60.      ArrayList<Shape> l =(ArrayList<Shape>)ois.readObject();
61.      //遍历容器里面的具体对象，将取出来的对象保存到容器里面
62.      for (int i = 0; i <l.size(); i++) {
63.       Shape shape=(Shape)l.get(i);
64.       ShArr.add(shape);
65.      }
66.      ois.close();
67.      jp.paint(gr);
68.     }
69.     catch (Exception e1) {
70.     e1.printStackTrace();
71.    }
72.     }
73.   }
74.
75.
76.    public void savepic() {
77.     //选择要保存的位置以及文件名字和信息
78.     JFileChooser chooser = new JFileChooser();
79.     chooser.showSaveDialog(null);
80.     File file =chooser.getSelectedFile();
81.
82.     if(file==null){
83.      JOptionPane.showMessageDialog(null, "没有选择文件");
84.     }else {
85.      try {
86.      //根据要保存的文件创建对象输出流
87.      FileOutputStream fis = new FileOutputStream(file);
88.      ObjectOutputStream oos = new ObjectOutputStream(fis);
89.      //将容器里面所绘制的图形利用对象流全部写入选中的文件中
90.      oos.writeObject(ShArr);
91.      JOptionPane.showMessageDialog(null, "保存成功！ ");
92.      oos.close();
93.     } catch (Exception e1) {
94.      e1.printStackTrace();
95.     }
96.    }
97.   }
98.
```

```
99.  public void actionPerformed(ActionEvent e)
100. {
101.  // TODO Auto-generated method stub
102.  x1 = 200; y1 = 200;
103.  if (e.getActionCommand()=="") {
104.   JButton j = (JButton)e.getSource();
105.   gr.setColor(j.getBackground());
106.  }
107.  else {
108.   commend = e.getActionCommand();
109.   if("Clear".equals(commend)) {
110.    index = 0;
111.    ShArr.clear();
112.    jp.repaint();
113.    x4 = 0;
114.    y4 = 0;
115.   }else {
116.    if("Save".equals(commend)) {
117.     savepic();
118.    }else {
119.     if("Open".equals(commend)) {
120.      openpic();
121.     }else {
122.      if("yes".equals(commend)){
123.       flag2 = true;
124.      }else {
125.       if("no".equals(commend)) {
126.        flag2 = false;
127.       }
128.      }
129.     }
130.    }
131.   }
132.  }
133. }
134. public void mouseDragged(MouseEvent e) {
135.  //System.out.println("Drag");
136.  if("RandomLine".equals(commend)) {
137.   x1 = x0; y1 = y0; x0 = e.getX(); y0 = e.getY();
138.   gr.drawLine(x1,y1,x0,y0);
139.   //ShArr[index++] = new Shape(x1,y1,x0,y0,commend);
140.   ShArr.add(new Shape(x1,y1,x0,y0,commend,gr.getColor(),flag2,jp.getBackg
    round()));
```

```
141.    }
142.   }
143.
144.   public void mouseClicked(MouseEvent e) {
145.    // TODO Auto-generated method stub
146.    if("Polygon".equals(commend)) {
147.
148.     if(x4==0&&y4==0){
149.      x4 = e.getX(); y4 = e.getY();
150.      start_x = x4; start_y = y4;
151.      //gr.drawLine(x3, y3, x4, y4);
152.     }
153.     else {
154.      x3 = x4; y3 = y4; x4 = e.getX(); y4 = e.getY();
155.      gr.drawLine(x3, y3, x4, y4);
156.      //ShArr[index++] = new Shape(x3,y3,x4,y4,commend);
157.      ShArr.add(new Shape(x3,y3,x4,y4,commend,gr.getColor(),flag2,jp.getBack
    ground()));
158.     }
159.     if(e.getClickCount()==2){
160.     x4 =0; y4=0;
161.     gr.drawLine(start_x, start_y, e.getX(), e.getY());
162.     //ShArr[index++] = new Shape(start_x,start_y,e.getX(),e.getY(),commend);
163.     ShArr.add(new Shape(start_x,start_y,e.getX(),e.getY(),commend,gr.getColor
    (),flag2,jp.getBackground()));
164.     }
165.    }else {
166.     if("Text".equals(commend)) {
167.     //获取坐标
168.      x4 = e.getX();
169.      y4 = e.getY();
170.     //jp.setLayout(null);
171.      JTextField jt = new JTextField();
172.      jp.add(jt);
173.      jt.setBounds(x4,y4,100,20);
174.     //阻塞式监听
175.      jt.addKeyListener(new KeyAdapter() {
176.       @Override
177.       public void keyTyped(KeyEvent ek) {
178.        if((char)ek.getKeyChar()==KeyEvent.VK_ENTER) {
179.         s = jt.getText();//获取文本
180.         Font font = new Font("华文行楷",Font.BOLD + Font.ITALIC,15);
181.         gr.setFont(font);
182.         gr.drawString(s,e.getX(),e.getY());
```

```
183.         //jp.remove(jt);//删去
184.         jt.setVisible(false);
185.         //ShArr[index++] = new Shape(x4,y4,commend,s);
186.         ShArr.add(new Shape(e.getX(),e.getY(),commend,s,gr.getColor(),jp.getBa
    ckground()));
187.         //jp.revalidate();
188.         //jp.repaint();
189.         //gr.drawString("hahaha", x4, y4);
190.         //System.out.print("Stirng set");
191.       }
192.      }
193.    });
194.   }
195.  }
196.
197. }
198.
199. @Override // 按下
200. public void mousePressed(MouseEvent e) {
201. // System.out.println("按下");
202.
203.  if("RandomLine".equals(commend)){
204.   x0 = e.getX();
205.   y0 = e.getY();
206.  }
207.  // TODO Auto-generated method stub
208.  x1 = e.getX();
209.  y1 = e.getY();
210.
211. }
212. @Override // 松开
213. public void mouseReleased(MouseEvent e) {
214.  // TODO Auto-generated method stub
215.  x2 = e.getX();
216.  y2 = e.getY();
217. // System.out.println("松开");
218.  if("StraightLine".equals(commend)){
219.   gr.drawLine(x1, y1, x2, y2);
220.   ShArr.add(new Shape(x1, y1, x2, y2,commend,gr.getColor(),flag2,jp.getBac
    kground()));
221.   //System.out.print(ShArr.size());
222.   //ShArr[index++] = new Shape(x1,y1,x2,y2,commend);
223.  }
224.  else if("Rectangle".equals(commend)) {
```

```
225.     if(flag2) {
226.      gr.fillRect(Math.min(x1, x2), Math.min(y1, y2), Math.abs(x1-x2), Math.abs
    (y1-y2));
227.     }else {
228.     gr.drawRect(Math.min(x1, x2), Math.min(y1, y2), Math.abs(x1-x2), Math.ab
    s(y1-y2));
229.     }
230.     ShArr.add(new Shape(x1,y1,x2,y2,commend,gr.getColor(),flag2,jp.getBackg
    round()));
231.     //System.out.print(ShArr.size());
232.     //ShArr[index++] = new Shape(x1,y1,x2,y2,commend);
233.    }
234.    else if("Circle".equals(commend)) {
235.     if(flag2) {
236.      gr.fillOval(Math.min(x1, x2), Math.min(y1, y2), Math.abs(x1-x2), Math.abs
    (y1-y2));
237.     }else {
238.     gr.drawOval(Math.min(x1, x2), Math.min(y1, y2), Math.abs(x1-x2), Math.a
    bs(y1-y2));
239.     }
240.     ShArr.add(new Shape(x1,y1,x2,y2,commend,gr.getColor(),flag2,jp.getBackg
    round()));
241.     //System.out.print(ShArr.size());
242.    }
243.   }
244.   @Override
245.   public void mouseEntered(MouseEvent e) {
246.    // TODO Auto-generated method stub
247.
248.   }
249.
250.   @Override
251.   public void mouseExited(MouseEvent e) {
252.    // TODO Auto-generated method stub
253.
254.   }
255.   public void mouseMoved(MouseEvent e) {
256.
257.   }
258.  }
259.
260. //颜色按钮，不当作匿名类
261. class ColorButton implements ActionListener{
262.   Color c;
```

```
263.  JButton jb;
264.  JPanel jp;
265.  JPanel setpanel;
266.  JPanel drawpanel;
267.  ColorButton(JPanel jp,JPanel setpanel,JPanel drawpanel){
268.   this.jp = jp;
269.   this.setpanel = setpanel;
270.   this.drawpanel = drawpanel;
271.  }
272.  public void actionPerformed(ActionEvent e)
273.  {
274.    jb = (JButton)e.getSource();
275.    c = jb.getBackground();
276.    jp.setBackground(c);
277.    setpanel.setBackground(c);
278.    drawpanel.setBackground(c);
279.  }
280. }
```

3. 面板设置：

```
1.    //用弹出式菜单设置背景颜色
2.    class MenuButton extends JToggleButton{
3.     private JPopupMenu menu;
4.     public MenuButton(final String label){
5.      super(label);
6.      this.setText("▶ "+label);
7.      this.setHorizontalTextPosition(SwingConstants.RIGHT );
8.      addActionListener(new ActionListener(){
9.       @Override
10.      public void actionPerformed(ActionEvent arg0) {
11.       if(isSelected()){
12.        setText("◀ "+label);
13.        menu.show(MenuButton.this,MenuButton.this.getWidth(), MenuButton.thi
    s.getHeight()-30);
14.       }else{
15.        setText("▶ "+label);
16.        menu.setVisible(false);
17.       }
18.      }
19.     });
20.    }
21.    public void addMenu(JPopupMenu menu){
22.     this.menu=menu;
```

```
23.     }
24.
25.    }
26.
27.    public class drawline extends JPanel{
28.     int len = 0;
29.     private ArrayList<Shape>ShArr = new ArrayList<Shape>();
30.     public static void main(String[] args) {
31.      drawline p = new drawline();
32.     }
33.
34.     JFrame jf = new JFrame();
35.
36.     ButtonListener bl = new ButtonListener();
37.
38.     JPanel drawpanel = new JPanel();
39.     JPanel setpanel = new JPanel();
40.     ColorButton cb = new ColorButton(this,drawpanel,setpanel);
41.     public drawline() {
42.      //实现一个窗体
43.      jf.setTitle("画图板");
44.      jf.setLocation(450, 100);
45.      jf.setSize(700, 600);
46.      //this.setSize(new Dimension(600, 500));
47.      //当采用布局的时候就需要用 setpreferredsize
48.      this.setPreferredSize(new Dimension(600, 500));
49.      this.setBackground(new Color(238,229,248));
50.      drawpanel.setPreferredSize(new Dimension(600,35));
51.      drawpanel.setBackground(new Color(238,229,248));
52.      setpanel.setPreferredSize(new Dimension(110,50));
53.      setpanel.setBackground(new Color(238,229,248));
54.      //setpanel.setBackground(Color.black);
55.      //setpanel.setLayout(new BorderLayout());
56.      //声明两个数组，包含各种指令
57.      String[] command = { "Clear", "StraightLine", "RandomLine", "Polygon", "
    Rectangle","Circle","Text","Save","Open"};
58.      Color[] color = {Color.white,Color.black,new Color(193,203,215),new Color
    (238,229,248),new Color(234,208,209),new Color(210,118,163),new Color(129,
    92,148),new Color(186,204,217),new Color(185,222,201) };
59.      //设置布局为流式布局
60.      //jf.setLayout(new FlowLayout());
61.      jf.setLayout(new BorderLayout());
62.      jf.setResizable(false);
63.
```

```
64.     for (int i = 0; i < command.length-2; i++) {
65.      JButton jb = new JButton(command[i]);
66.      jb.addActionListener(bl);
67.      drawpanel.add(jb);
68.     }
69.     jf.add(drawpanel,"North");
70.
71.     //菜单项
72.     JMenu coloritem = new JMenu("BackGroundColor");
73.     //coloritem.setPreferredSize(new Dimension(35,10));
74.     JMenu coloritem2 = new JMenu("PenColor");
75.     JMenu coloritem3 = new JMenu("IsFill");
76.     //coloritem3.setLayout(new BorderLayout());
77.     for (int i = color.length - 1; i >= 0; i--) {
78.      JButton jb = new JButton();
79.      jb.setBackground(color[i]);//设置背景颜色
80.      Dimension dm = new Dimension(20, 20);//设置大小
81.      jb.setPreferredSize(dm);
82.      jb.addActionListener(cb);
83.      coloritem.add(jb);
84.     }
85.     JButton yesb = new JButton("yes");
86.     JButton nob = new JButton("no");
87.     Dimension dm1 = new Dimension(55, 20);//设置大小
88.     yesb.setPreferredSize(dm1);
89.     yesb.addActionListener(bl);
90.     nob.setPreferredSize(dm1);
91.     nob.addActionListener(bl);
92.     coloritem3.add(yesb);
93.     coloritem3.add(nob);
94.     //弹出式菜单设计
95.     JPopupMenu penmenu=new JPopupMenu();
96.     //penmenu.setPreferredSize(new Dimension(30,20));
97.     penmenu.add(coloritem);
98.     penmenu.add(coloritem2);
99.     penmenu.add(coloritem3);
100.
101.    //创建按钮
102.    MenuButton penbutton1=new MenuButton("SetUp");
103.
104.    //把菜单放到按钮上
105.    penbutton1.addMenu(penmenu);
106.    penbutton1.setPreferredSize(new Dimension(100,30));
107.    //把按钮放到控件上
```

```
108.    setpanel.add(penbutton1,"North");
109.    //保存和打开按钮
110.    for(int i = command.length-2;i<command.length;i++) {
111.     JButton jb = new JButton(command[i]);
112.     jb.addActionListener(bl);
113.     jb.setPreferredSize(new Dimension(100,25));
114.     setpanel.add(jb,"North");
115.    }
116.    jf.add(setpanel,"West");
117.
118.    for (int i = color.length - 1; i >= 0; i--) {
119.     JButton jb = new JButton();
120.     jb.setBackground(color[i]);//设置背景颜色
121.     Dimension dm = new Dimension(25, 20);//设置大小
122.     jb.setPreferredSize(dm);
123.     jb.addActionListener(bl);
124.     coloritem2.add(jb);
125.    }
126.
127.    //将 JPanel 对象添加进入 jf 窗体对象中，让他生效
128.    jf.add(this,"Center");
129.    jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
130.    jf.setVisible(true); //设置可见
131.
132.
133.    Graphics gf = this.getGraphics(); //获取画笔，我们的 this 代表当前类的对
       象，正好是一个 JPanel 的对象
134.    this.addMouseListener(bl); //添加鼠标监听器，用于画图
135.    this.addMouseMotionListener(bl); //添加鼠标模式监听器，用于绘画曲线
136.
137.    bl.set_jp(this); //设置另外一个类的 JPanel 容器
138.    bl.set_gr(gf); //设置另外一个类的画笔
139.    bl.set_ShArr(ShArr); //设置另外一个类的数组
140.   }
141.   public void paint(Graphics g) {
142.    super.paint(g); //调用父类的 paint 方法，用来画出窗体
143.    //len = bl.get_len(); //获取数组的长度
144.    int size = ShArr.size();
145.    //重绘我们的图案
146.    if(size!=0) {
147.     this.setBackground(ShArr.get(0).bgc);
148.     setpanel.setBackground(ShArr.get(0).bgc);
149.     drawpanel.setBackground(ShArr.get(0).bgc);
150.    }
```

```
151.    for(int i=0;i<size;i++) {
152.     if(ShArr.get(i)!=null) {
153.      //重写绘画图像，但是我只重绘了图形，忘记加颜色了；
154.       ShArr.get(i).repaint(g);
155.     }
156.     else {
157.      break;
158.     }
159.    }
160.   }
161.
162.  }
163.
```