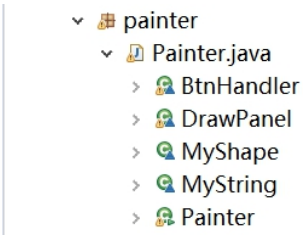
GUI 实验报告

马斓轩 1813076

一、程序框架

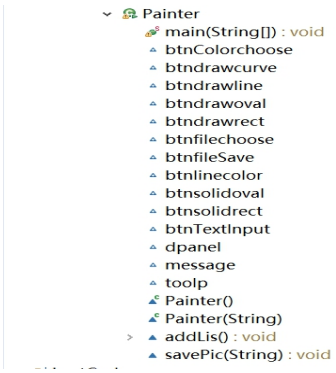


✧ 布局设计——Painter

整体布局使用 GridLayout，包括工具栏（1 列），右侧文字区，中间绘图区，初始背景为浅灰色。

工具栏包含 11 项：打开文件（File），选择背景颜色（Back），选择线条颜色（Front），输入文字（Text），保存文件（Save），画直线（Line），画空心矩形（Rect），画空心椭圆（Oval），画实心矩形（SolidRect），以及画实心椭圆（SolidOval）。

使用 setBackground 为 DrawPanel 区设置背景颜色，使用 setSize 设置窗口大小。在 addLis 方法中，所有的 JButton 都加入了事件响应 BtnHandler 中。savePic 方法则用来保存图片。




```
this.getContentPane().add("West",toolp);
dpanel=new DrawPanel();
dpanel.setBackground(Color.lightGray);
this.getContentPane().add("Center",dpanel);
message=new JTextArea(3,20);
this.getContentPane().add("East",message);
this.setSize(1200,800);
this.setDefaultCloseOperation(EXIT_ON_CLOSE);
addLis();
this.setVisible(true);
```

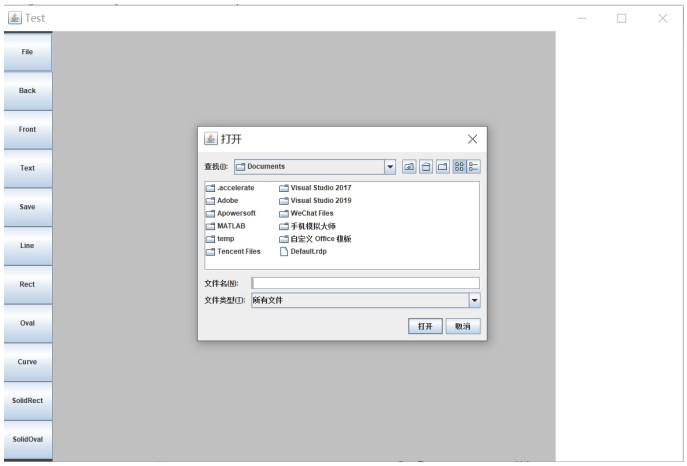
✧ 事件监听

工具按钮——BtnHandler

获取 ActionEvent 的类，若为 JButton，判断对应的字符，分别做出响应。

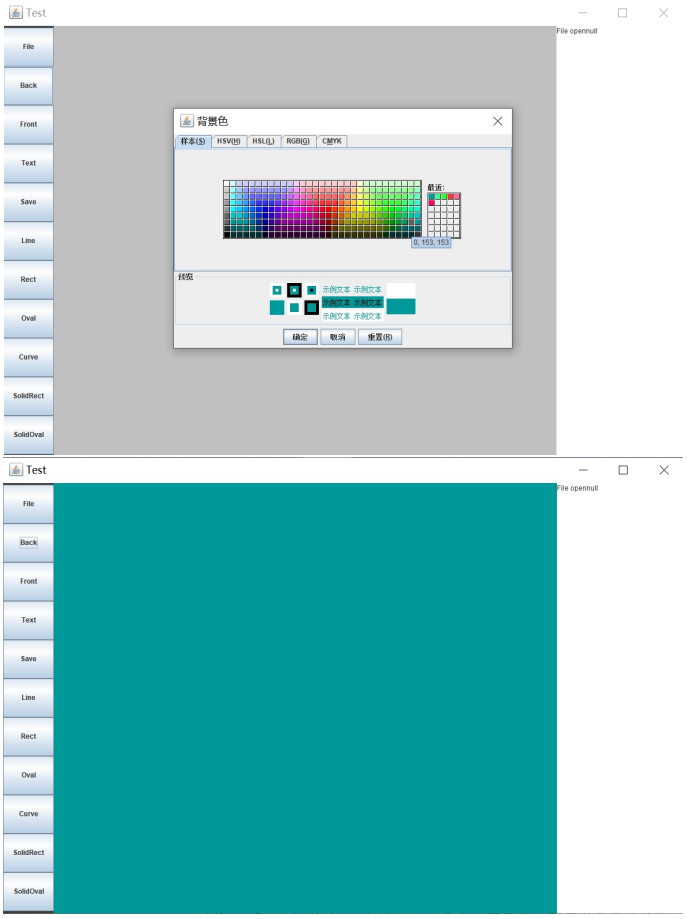
“File”:使用 JFileChooser 打开对话框，选择要打开的文件。

```
if(sur.getText().equals("File")) {
    JFileChooser jf=new JFileChooser();
    jf.showOpenDialog(parent);
    parent.message.append("File open"+jf.getName());
}
```



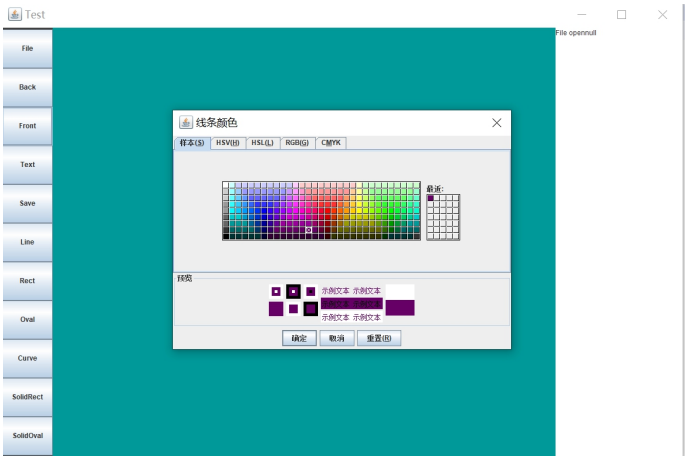
“Back”:使用 setBackground 为 Jpanel 区选择背景颜色。

```
if(sur.getText().equals("Back")) {
    JColorChooser jf=new JColorChooser();
    Color c=jf.showDialog(parent, "背景色", null);
    parent.dpanel.setBackground(c);
}
```



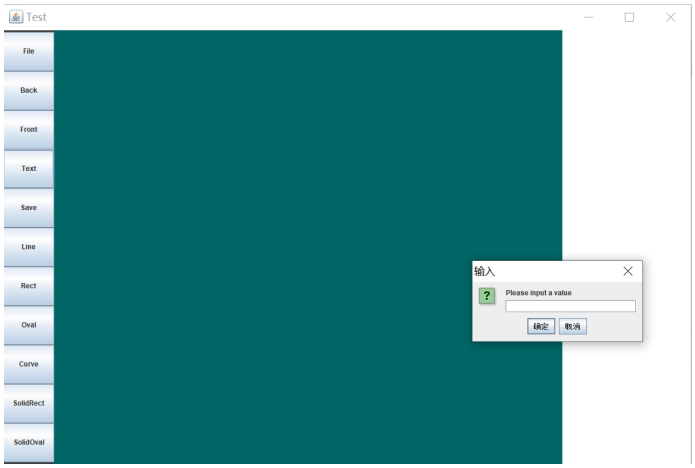
“Front”:直接对 Jpanel 区的 currColor 参数进行赋值以改变线条颜色。

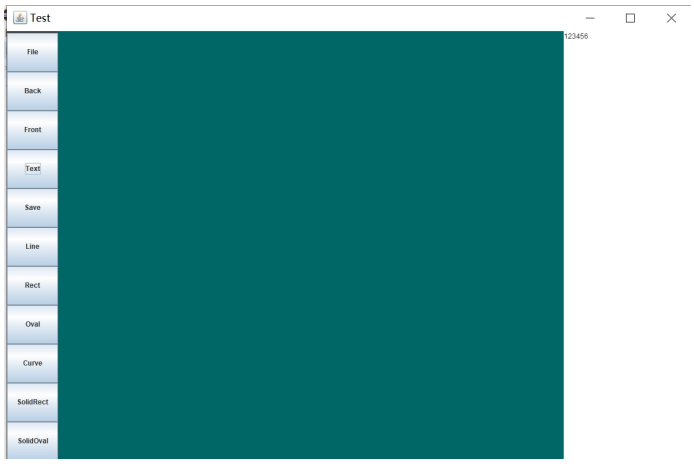
```
if(sur.getText().equals("Front")) {
    JColorChooser jf=new JColorChooser();
    Color c=jf.showDialog(parent, "线条颜色", null);
    parent.dpanel.currColor=c;
}
```



“Text”:弹出输入文字窗口。

```
btnTextInput.addMouseListener(new MouseAdapter() {
    @Override
    public
    void mouseClicked(MouseEvent me) {
        if(me.getClickCount()==1) {
            //message.append("single click\n");
            String inputValue = JOptionPane.showInputDialog("Please input a value");
            message.append(inputValue);
        }
    }
});
```

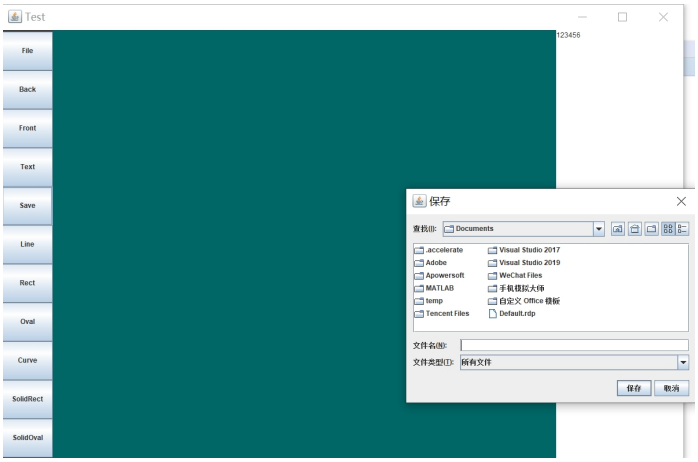




“Save”： 使用 JFileChooser 打开对话框，确定保存的路径，传入 savePic 方法中。savePic 方法位于 Painter 类中，主要使用了类 Robot 抓取当前图片，再使用 imageIO 将图片输出。

```
if(sur.getText().equals("Save")) {

    JFileChooser jf = new JFileChooser();
    int cnt = jf.showDialog(null,"保存");
    if(cnt == 0 ) {
       File file = jf.getSelectedFile();
       parent.savePic(file.getAbsolutePath()+".jpg");
    }
}
```



“Line”:将类 Drawpanel 中的成员“type”赋值为“Line”。

“Rect”：将类 Drawpanel 中的成员“type”赋值为“Rect”。

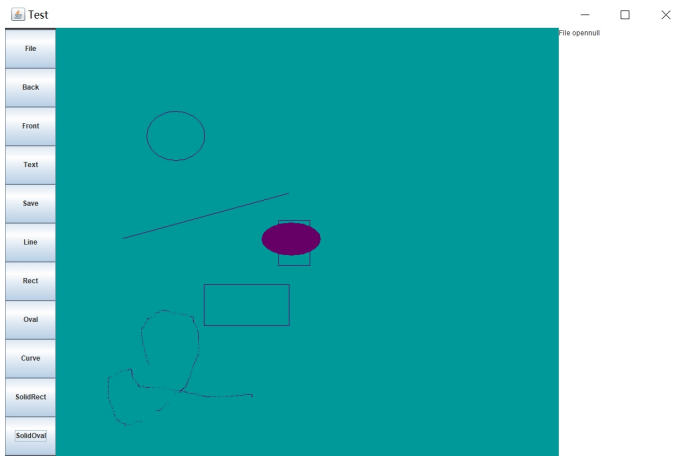
“Oval”：将类 Drawpanel 中的成员“type”赋值为“Oval”。

“Curve”：将类 Drawpanel 中的成员“type”赋值为“Curve”。

“SolidRect”：将类 Drawpanel 中的成员“type”赋值为“SolidRect”。

“SolidOval”：将类 Drawpanel 中的成员“type”赋值为“SolidOval”。

```
if(sur.getText().equals("Line")) {
    parent.dpanel.type="Line";
}
if(sur.getText().equals("Rect")) {
    parent.dpanel.type="Rect";
}
if(sur.getText().equals("Oval")) {
    parent.dpanel.type="Oval";
}
if(sur.getText().equals("Curve")) {
    parent.dpanel.type="Curve";
}
if(sur.getText().equals("SolidRect")) {
    parent.dpanel.type="SolidRect";
}
if(sur.getText().equals("SolidOval")) {
    parent.dpanel.type="SolidOval";
}
```



鼠标事件——DrawPanel

鼠标事件主要发生在 Jpanel，作为 DrawMouseL 和 DrawMouseML 的内部类。

“mouseDragged”:鼠标拖拽前首先判断当前的“type”，若不为曲线，拖拽发生后将 tempshape 转换成当前形状并放入动态数组中；若为曲线，则实时描绘连续曲线。

```
@Override
public void mouseDragged(MouseEvent e) {
    if(tempshape!=null) {
        if(type.equals("Oval")||type.equals("SolidOval")) {
            Ellipse2D el = (Ellipse2D)(tempshape.s);
            el.setFrame(el.getX(), el.getY(), e.getX()-el.getX(),e.getY()-el.getY());
        }
        else if(type.equals("Rect")||type.equals("SolidRect")) {
            Rectangle2D rl=(Rectangle2D)(tempshape.s);
            rl.setFrame(rl.getX(), rl.getY(), e.getX()-rl.getX(), e.getY()-rl.getY());
        }
        else if(type.equals("Line")){
            Line2D l=(Line2D)(tempshape.s);
            l.setLine(l.getX1(), l.getY1(), e.getX(), e.getY());
        }

        shapelist.add(tempshape);
    }
    if(type.equals("Curve")) {
        tempshape=new MyShape();
        Line2D l=new Line2D.Double();
        l.setLine(e.getX(), e.getY(), e.getX(), e.getY());
        tempshape.c=currColor;
        tempshape.s=l;
        repaint();
    }
}
```

“mousePressed”:鼠标按压时，为不同的形状创建不同的类型，其中实心和空心通过 Myshape 中的布尔型变量“f”来确定。在 paintComponent 中，若 f 为 true 则说明选择的是实心图形,此时调用 graphic2D 中的 fill 方法，否则调用 draw 方法。

```
@Override
public void mousePressed(MouseEvent e){
    x=e.getX();
    y=e.getY();
    if(tempshape!=null) {
        shapelist.add(tempshape);
    }
    tempshape=new MyShape();
    if(type.equals("Oval")||type.equals("SolidOval")) {
        Ellipse2D el = new Ellipse2D.Double();
        el.setFrame(el.getX(), el.getY(), e.getX()-el.getX(),e.getY()-el.getY());
        tempshape.c=currColor;
        tempshape.s=el;
        if(type.equals("Oval")) {
            tempshape.f=false;
        }
        else {
            tempshape.f=true;
        }
    }
```

```
else if(type.equals("Rect")||type.equals("SolidRect")) {
    Rectangle2D rl = new Rectangle2D.Double();
    rl.setFrame(rl.getX(), rl.getY(), e.getX()-rl.getX(), e.getY()-rl.getY());
    tempshape.c=currColor;
    tempshape.s=rl;
    if(type.equals("Rect")) {
        tempshape.f=false;
    }
    else {
        tempshape.f=true;
    }
}
else if(type.equals("Line")) {
    Line2D line = new Line2D.Double();
    line.setLine(line.getX1(), line.getY1(), e.getX(), e.getY());
    tempshape.c=currColor;
    tempshape.s=line;
    tempshape.f=false;
}
else {
    Line2D l=new Line2D.Double();
    l.setLine(e.getX(), e.getY(), e.getX(), e.getY());
    tempshape.c=currColor;
    tempshape.s=l;
    tempshape.f=false;
}
```

eReleased”:鼠标释放与鼠标按压相对应，确定当前的图形。

```
@Override
public void mouseReleased(MouseEvent e){
    int x1=e.getX();
    int y1=e.getY();
    if(type.equals("Line")) {
        Line2D l=(Line2D)(tempshape.s);
        l.setLine(x, y, x1, y1);
    }
    else if(type.equals("Rect")||type.equals("SolidRect")) {
        Rectangle2D rl=(Rectangle2D)(tempshape.s);
        rl.setFrame(x, y, x1-x, y1-y);
    }
    else if(type.equals("Oval")||type.equals("SolidOval")) {
        Ellipse2D el = (Ellipse2D)(tempshape.s);
        el.setFrame(x, y, x1-x, y1-y);
    }
    else {
        Line2D l=(Line2D)(tempshape.s);
        l.setLine(l.getX1(), l.getY1(), e.getX(), e.getY());
    }
    repaint();

}
```

“mouseClicked”:鼠标点击与文字输入相关。点击鼠标右键，调用 showInputDialog 会弹出文字输入框。

```
@Override
public void mouseClicked(MouseEvent e){
  //  System.out.println(e.getButton());
    if(e.getButton()==MouseEvent.BUTTON3) {
        String inputValue = JOptionPane.showInputDialog("Please input a value");
        Font f=new Font("宋体",10,10);
        MyString ms=new MyString();
        ms.s=inputValue;
        ms.c=currColor;
        ms.x=e.getX();
        ms.y=e.getY();
        stringlist.add(ms);
        repaint();
        }
}
```

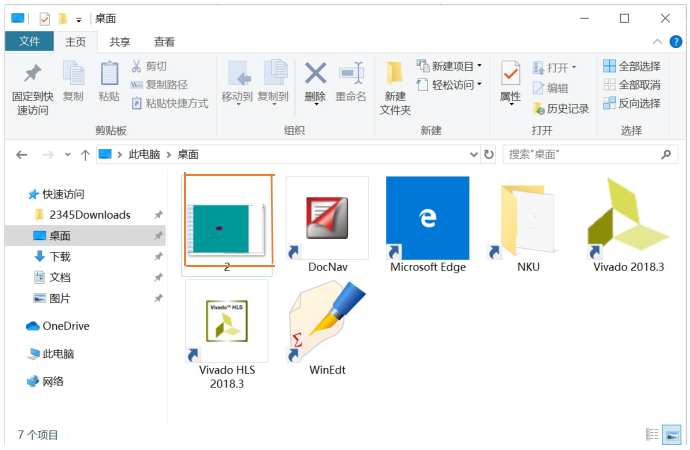
二、问题与解决

✧ 输出功能

对象序列化作为输出的首选方法，在保存过程中容易出现乱码。其次尝试使用 BufferedImage，在保存过程也频频出现错误，并不能保存绘画图像，而是保存空白 panel。

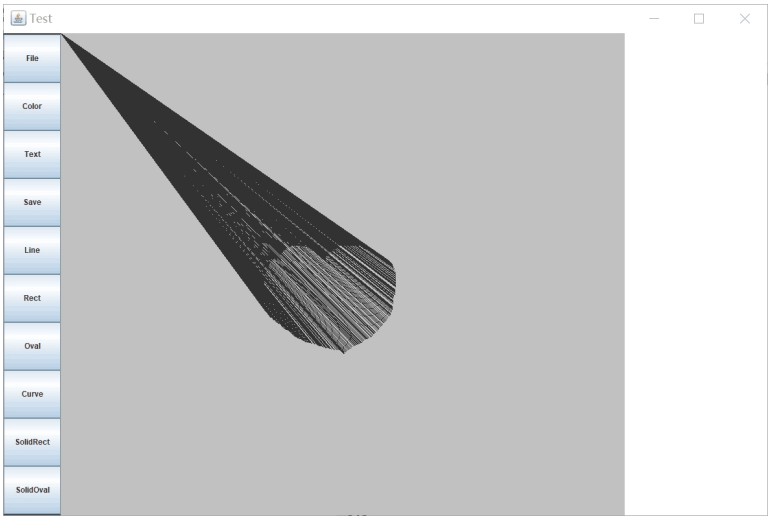
选择类 Robot，可以实现自由绘图并以 jpg 格式保存。抓取某一个矩形区域，还可以抓取某一个控件，返回 BufferedImage 对象，最后以 imageIO 输出保存。

```
void savePic(String path){
    BufferedImage myImage = null;
    try {
        myImage = new Robot().createScreenCapture(
                new Rectangle(this.getX(), this.getY(), this.getWidth(), this.getHeight()));
        ImageIO.write(myImage, "jpg", new File(path));
        } catch (AWTException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```



✧ 鼠标拖拽

对于规则图形，如直线、矩形等，tempshape 中的 s 不为空，不需要拖拽绘图，只需要确定起止点就可以了。需要拖拽绘图的只有自由曲线部分。在刚开始时将所有的形状都做了拖拽绘图，出现了从左上角到鼠标位置的线条。将该部分删掉就可以了。（下图为直线拖拽效果）



✧ 颜色设置

起初并没有设置两个按钮，希望仅使用一个按钮 Color，在按钮响应逻辑上做出前景色与背景色二者相关的编写，并不能很好地区分开。后改用两个按钮对其进行区分响应。

✧ 填充图形

首次尝试在 mouseReleased 中创建 Graphics2D 对象，分别使用 fillRect 和 fillOval 进行填充图形的绘制，出现了不能共存的情况。问题在于并没有将当前形状存入动态数组，因此在 paintComponet 中没有绘制，只是以临时值的状态出现。

对 Myshape 结构进行修改，增加的一个布尔型参数 f，用来判断是否为实心图形。实心矩形和空心矩形、实心椭圆和空心椭圆在本质上是相同的，分别使用 Rectangle2D 和 Eliipse2D 创建对象，区别在于使用 Graphics2D 绘图时使用的是 fill 方法还是 draw 方法。因此，在鼠标事件中不做区分，只在 paintComponent 中对其参数 f 进行判断。

```
class MyShape{
    Shape s;
    Color c;
    boolean f;
}
```

```
if(tempshape!=null) {
    g2d.setColor(tempshape.c);
    if(tempshape.f) {
        g2d.fill(tempshape.s);
    }
    else {
        g2d.draw(tempshape.s);
    }

}
```

源代码：

```
import java.awt.AWTException;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GridLayout;
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.Shape;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionAdapter;
import java.awt.event.MouseMotionListener;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Line2D;
import java.awt.geom.Rectangle2D;
```

```
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JColorChooser;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;


class MyShape{
    Shape s;
    Color c;
    boolean f;
}
class MyString{
    Font f;
    String s;
    Color c;
    int x;
    int y;
}

class DrawPanel extends JPanel{
    ArrayList<MyShape> shapelist=new ArrayList();
    ArrayList<MyString> stringlist=new ArrayList();
    String type="";
    int x,y;
    MyShape tempshape;
    Color currColor;
    DrawPanel(){
        DrawMouseL ml=new DrawMouseL();
        this.addMouseListener((MouseListener) ml);
        DrawMouseML mml=new DrawMouseML();
        this.addMouseMotionListener((MouseMotionListener) mml);

    }
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
```

```
        Graphics2D g2d=(Graphics2D) g;
        for(MyShape ms:shapelist) {
            g2d.setColor(ms.c);
            g2d.draw(ms.s);
        }
        if(tempshape!=null) {
            g2d.setColor(tempshape.c);
            if(tempshape.f) {
                g2d.fill(tempshape.s);
            }
            else {
                g2d.draw(tempshape.s);
            }

        }
        for(MyString ms:stringlist) {
            g2d.setColor(ms.c);
            g2d.setFont(ms.f);
            g2d.drawString(ms.s,ms.x,ms.y);
        }
    }
    class DrawMouseML extends MouseMotionAdapter{
        @Override
        public void mouseDragged(MouseEvent e) {
            if(tempshape!=null) {
                if(type.equals("Oval")||type.equals("SolidOval")) {
                    Ellipse2D el = (Ellipse2D)(tempshape.s);
                    el.setFrame(el.getX(), el.getY(),
e.getX()-el.getX(),e.getY()-el.getY());
                }
                else if(type.equals("Rect")||type.equals("SolidRect")) {
                    Rectangle2D rl=(Rectangle2D)(tempshape.s);
                    rl.setFrame(rl.getX(), rl.getY(), e.getX()-rl.getX(),
e.getY()-rl.getY());
                }
                else if(type.equals("Line")){
                    Line2D l=(Line2D)(tempshape.s);
                    l.setLine(l.getX1(), l.getY1(), e.getX(), e.getY());
                }

            shapelist.add(tempshape);
            }
            if(type.equals("Curve")) {
                tempshape=new MyShape();
```

```
                Line2D l=new Line2D.Double();
                l.setLine(e.getX(), e.getY(), e.getX(), e.getY());
                tempshape.c=currColor;
                tempshape.s=l;
                repaint();
            }
    }
}
    class DrawMouseL extends MouseAdapter{
        @Override
        public void mousePressed(MouseEvent e){
            x=e.getX();
            y=e.getY();
            if(tempshape!=null) {
            shapelist.add(tempshape);
            }
            tempshape=new MyShape();
            if(type.equals("Oval")||type.equals("SolidOval")) {
            Ellipse2D el = new Ellipse2D.Double();
            el.setFrame(el.getX(), el.getY(),
e.getX()-el.getX(),e.getY()-el.getY());
            tempshape.c=currColor;
                tempshape.s=el;
                if(type.equals("Oval")) {
                    tempshape.f=false;
                }
                else {
                    tempshape.f=true;
                }
            }
            else if(type.equals("Rect")||type.equals("SolidRect")) {
            Rectangle2D rl = new Rectangle2D.Double();
            rl.setFrame(rl.getX(), rl.getY(), e.getX()-rl.getX(),
e.getY()-rl.getY());
            tempshape.c=currColor;
                tempshape.s=rl;
                if(type.equals("Rect")) {
                    tempshape.f=false;
                }
                else {
                    tempshape.f=true;
                }
            }
            else if(type.equals("Line")) {
```

```
            Line2D line = new Line2D.Double();
            line.setLine(line.getX1(), line.getY1(), e.getX(), e.getY());
            tempshape.c=currColor;
                tempshape.s=line;
                tempshape.f=false;
            }
            else {
            Line2D l=new Line2D.Double();
                l.setLine(e.getX(), e.getY(), e.getX(), e.getY());
                tempshape.c=currColor;
                tempshape.s=l;
                tempshape.f=false;
            }
        }
        @Override
        public void mouseReleased(MouseEvent e){
            int x1=e.getX();
            int y1=e.getY();
            if(type.equals("Line")) {
            Line2D l=(Line2D)(tempshape.s);
                l.setLine(x, y, x1, y1);
            }
            else if(type.equals("Rect")||type.equals("SolidRect")) {
            Rectangle2D rl=(Rectangle2D)(tempshape.s);
                rl.setFrame(x, y, x1-x, y1-y);
            }
            else if(type.equals("Oval")||type.equals("SolidOval")) {
                Ellipse2D el = (Ellipse2D)(tempshape.s);
                el.setFrame(x, y, x1-x, y1-y);
            }
            else {
            Line2D l=(Line2D)(tempshape.s);
                l.setLine(l.getX1(), l.getY1(), e.getX(), e.getY());
            }
            repaint();

        }
        @Override
        public void mouseClicked(MouseEvent e){
          //  System.out.println(e.getButton());
            if(e.getButton()==MouseEvent.BUTTON3) {
                String inputValue = JOptionPane.showInputDialog("Please input
a value");
                Font f=new Font("宋体",10,10);
```

```
                MyString ms=new MyString();
                ms.s=inputValue;
                ms.c=currColor;
                ms.x=e.getX();
                ms.y=e.getY();
                stringlist.add(ms);
                repaint();
                }
        }
    }
}

class BtnHandler implements ActionListener{

    Painter parent;

    public BtnHandler(Painter parent) {
        super();
        this.parent = parent;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        if(e.getSource().getClass()==JButton.class) {
            JButton sur=(JButton)e.getSource();
            if(sur.getText().equals("File")) {
                JFileChooser jf=new JFileChooser();
                jf.showOpenDialog(parent);
                parent.message.append("File open"+jf.getName());
            }
            if(sur.getText().equals("Back")) {
                JColorChooser jf=new JColorChooser();
                Color c=jf.showDialog(parent, "背景色", null);
                parent.dpanel.setBackground(c);
            }
            if(sur.getText().equals("Front")) {
                JColorChooser jf=new JColorChooser();
                Color c=jf.showDialog(parent, "线条颜色", null);
                parent.dpanel.currColor=c;
            }
            if(sur.getText().equals("Save")) {

                 JFileChooser jf = new JFileChooser();
```

```
                int cnt = jf.showDialog(null,"保存");
                if(cnt == 0 ) {
                  File file = jf.getSelectedFile();
                  parent.savePic(file.getAbsolutePath()+".jpg");
                }
            }
            if(sur.getText().equals("Line")) {
                parent.dpanel.type="Line";
            }
            if(sur.getText().equals("Rect")) {
                parent.dpanel.type="Rect";
            }
            if(sur.getText().equals("Oval")) {
                parent.dpanel.type="Oval";
            }
            if(sur.getText().equals("Curve")) {
                parent.dpanel.type="Curve";
            }
            if(sur.getText().equals("SolidRect")) {
                parent.dpanel.type="SolidRect";
            }
            if(sur.getText().equals("SolidOval")) {
                parent.dpanel.type="SolidOval";
            }
        };
    }

}

public class Painter extends JFrame{

    JPanel toolp;
    JButton btnfilechoose;
    JButton btnColorchoose;
    JButton btnlinecolor;
    JButton btnTextInput;
    JButton btnfileSave;
    JButton btndrawline;
    JButton btndrawrect;
    JButton btndrawoval;
    JButton btndrawcurve;
    JButton btnsolidrect;
    JButton btnsolidoval;
    DrawPanel dpanel;
```

```
JTextArea message;
Painter(){
    super();
    toolp=new JPanel(new GridLayout(11,1));
    toolp.setBackground(Color.darkGray);
    btnfilechoose=new JButton("File");
    btnColorchoose=new JButton("Back");
    btnlinecolor=new JButton("Front");
    btnTextInput=new JButton("Text");
    btnfileSave=new JButton("Save");
    btndrawline=new JButton("Line");
    btndrawrect=new JButton("Rect");
    btndrawoval=new JButton("Oval");
    btndrawcurve=new JButton("Curve");
    btnsolidrect=new JButton("SolidRect");
    btnsolidoval=new JButton("SolidOval");
    toolp.add(btnfilechoose);
    toolp.add(btnColorchoose);
    toolp.add(btnlinecolor);
    toolp.add(btnTextInput);
    toolp.add(btnfileSave);
    toolp.add(btndrawline);
    toolp.add(btndrawrect);
    toolp.add(btndrawoval);
    toolp.add(btndrawcurve);
    toolp.add(btnsolidrect);
    toolp.add(btnsolidoval);
    this.getContentPane().add("West",toolp);
    dpanel=new DrawPanel();
    dpanel.setBackground(Color.lightGray);
    this.getContentPane().add("Center",dpanel);
    message=new JTextArea(3,20);
    this.getContentPane().add("East",message);
    this.setSize(1200,800);
    this.setDefaultCloseOperation(EXIT_ON_CLOSE);
    addLis();
    this.setVisible(true);

}
Painter(String title){
    this();
    this.setTitle(title);

}
```

```
    void addLis(){
        BtnHandler bh=new BtnHandler(this);
        btnfilechoose.addActionListener(bh);
        btnColorchoose.addActionListener(bh);
        btnlinecolor.addActionListener(bh);
        btnfileSave.addActionListener(bh);
        btndrawline.addActionListener(bh);
        btndrawrect.addActionListener(bh);
        btndrawoval.addActionListener(bh);
        btndrawcurve.addActionListener(bh);
        btnsolidrect.addActionListener(bh);
        btnsolidoval.addActionListener(bh);
        btnTextInput.addMouseListener(new MouseAdapter() {
            @Override
            public
            void mouseClicked(MouseEvent me) {
                if(me.getClickCount()==1) {
                    //message.append("single click\n");
                    String inputValue = JOptionPane.showInputDialog("Please
input a value");
                    message.append(inputValue);
                }
            }
        });
    }
    void savePic(String path){
         BufferedImage myImage = null;
         try {
             myImage = new Robot().createScreenCapture(
                     new Rectangle(this.getX(), this.getY(), this.getWidth(),
this.getHeight()));
             ImageIO.write(myImage, "jpg", new File(path));
             } catch (AWTException e) {
                 e.printStackTrace();
             } catch (IOException e) {
                 e.printStackTrace();
             }
         }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Painter ge=new Painter("Test");
```

```
    }

}
```