题目 基础图形绘制软件

姓名 韩传兵 学号 161220044 邮箱 161220044@smail.nju.edu.cn 联系方式 18326199705

(南京大学 计算机科学与技术系，南京 210093)

1 引言

AWT 的全称是抽象窗口工具集。它是 Sun 最早提供的 GUI 库，提供了一些基本的功能，但功能比较有限，后来又提供了 Swing 库。Swing 库替代了绝大部分的 AWT 组件，但需要使用 AWT 的事件处理机制。通过使用 AWT 和 Swing 提供的图形界面组件库，Java 的图形界面编程可以变得比较简单，程序只需依次创建所需的图形组件，并以合适的方式将这些组件组织在一起，就可以开发出不错的用户界面。

（1）AWT 的容器：

任何窗口都可被分解为一个空的容器，容器里盛满了大的基本组件，通过设置这些组件的大小、位置等属性，就可以对该空的容器和基本的组件组成一个整体的窗口。

AWT 提供了两种主要的容器类型：

Window：可独立存在的顶级窗口

Panel：可作为容器容纳其他组件，但不能独立存在，必须被添加到其它的容器中

（2）AWT 的布局管理器：

Java 提供了布局管理器来设计组件在容器中的布局，而不是直接设置组件位置和大小。

AWT 中主要有四种布局管理器：

FlowLayout：组件向某个方向排列，遇到边界就折回，从头开始排列

BorderLayout：将容器分为东北南北中五个区域，普通组件被放置在这五个区域的任意一个中

GirdLayout：将容器分割成纵横线分割的网格，每个网格所占的区域大小相同

CardLayout：让多个组件共享同一个显示空间

（3）菜单栏的实现：

在 Java 中，一个完整的菜单栏是由三种菜单栏类所构建的（MenuBar、Menu 和 MenuItem）。要创建完整的菜单栏，必须分别创建 Menubar、Menu 与 MenuItem 对象，然后先用 add()把 Menu 对象加到 MenuBar 对象中，再把 MenuItem 对象加到 Menu 对象中。

（4）事件处理：

Frame 和组件本身没有事件处理的能力，必须由特定对象（事件监听器）来处理。

实现事件处理机制的步骤：

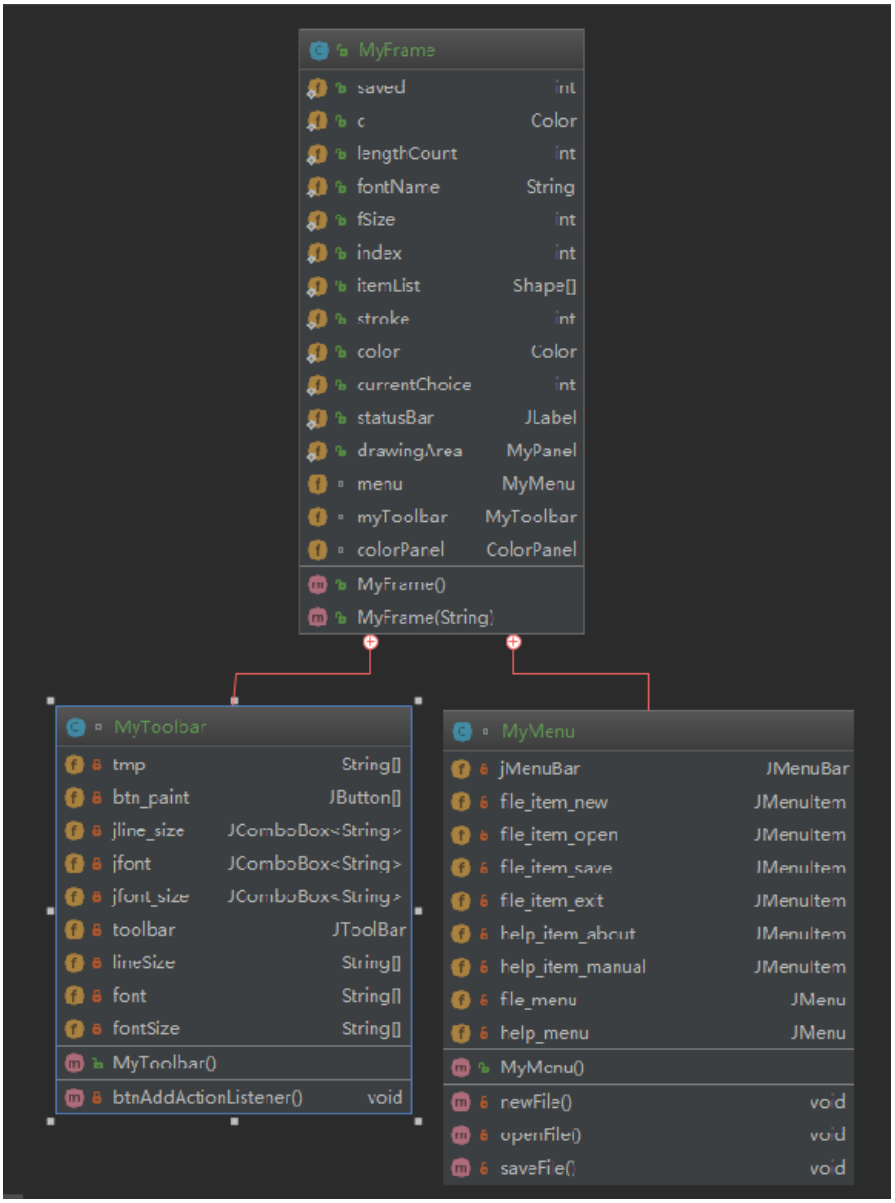
1. 实现事件监听类，必须实现 xxxListener 接口
2. 创建普通组件（事件源），创建事件监听对象
3. 调用 addxxxListener()方法，将事件监听器注册给普通组件，当事件源上发生指定的事件时，AWT 会触发事件监听器，由事件监听器调用相应的方法(事件处理器)来处理事件，事件源上发生的事件会作为参数传入事件处理器。

Swing 是一个为 Java 设计的 GUI 工具包，是 Java 基础类的一部分。包括了图形用户界面（GUI）器件如：文本框，按钮，分隔窗格和表。此外，Swing 提供许多比 AWT 更好的屏幕显示元素。它们用纯 Java 写成，所以同 Java 本身一样可以跨平台运行，这一点不像 AWT。它们是 JFC 的一部分。它们支持可更换的面板和主题，然而不是真的使用原生平台提供的设备，而是仅仅在表面上模仿它们。这意味着可以在任意平台上使用 JAVA 支持的任意面板。

但需要注意的是，虽然 Swing 提供的组件可以方便开发 Java 应用程序，但是 Swing 并不能取代 AWT，在开发 Swing 程序时通常要借助与 AWT 的一些对象来共同完成应用程序的设计。

2 分析与设计

2.1 UML建模



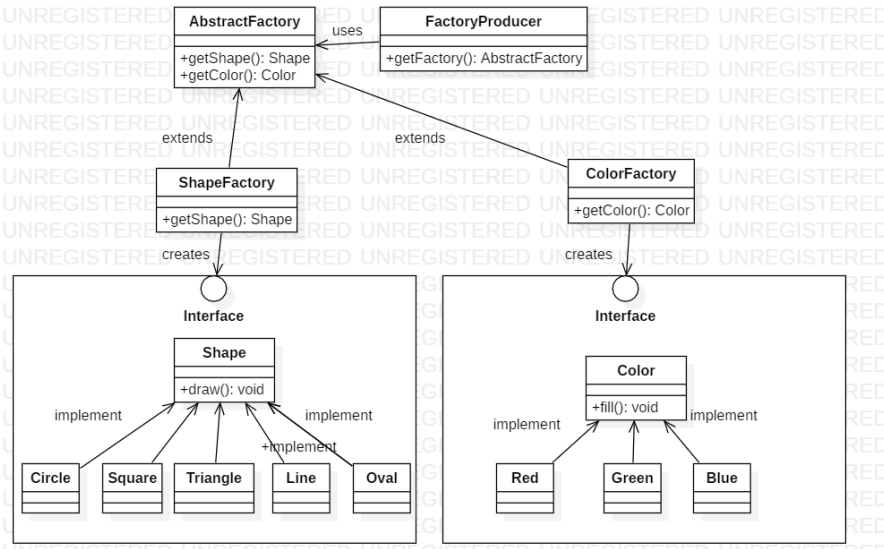
主界面 MyFrame 类继承 JFrame 类，其中调用菜单栏 MyMenu 类、工具栏 MyToolbar 类、画板 MyPanel 类（继承 JPanel）、调色板 ColorPanel 类（继承 JPanel）。

将菜单栏分装成 MyMenu 类，主要实现文件的打开、保存、新建和退出，以及部分的使用说明功能；将工具栏封装成 MyToolbar 类，主要实现图形绘制按钮，文本绘制、文本字体大小和字体的设置；将调色板封装成 ColorPanel 类，主要用来设置画笔的颜色；将画板封装成 DrawPanel 类，主要实现图形以及文本的绘制。

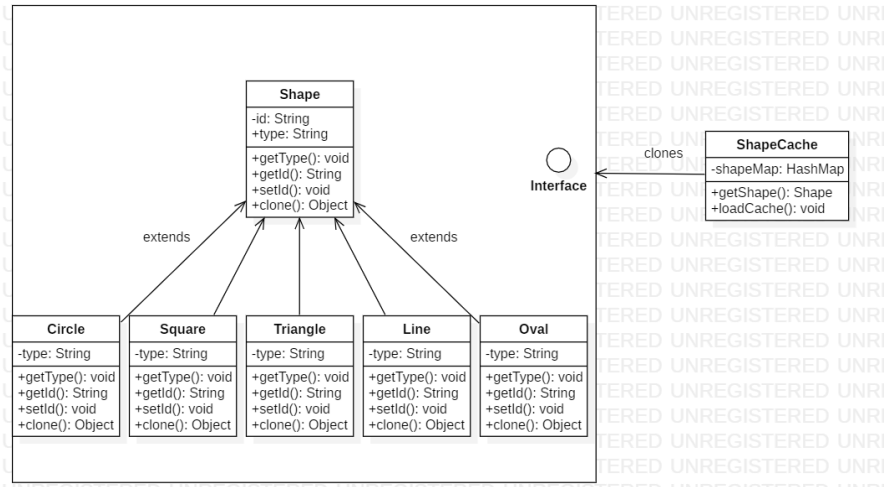
对于图形绘制，设计一个抽象 Shape 类，包含图形绘制的基本属性值和一个抽象 draw(Graphics2D g)方法；其他图形类对象均继承 Shape 类，并实现抽象方法。

2.2 设计模式

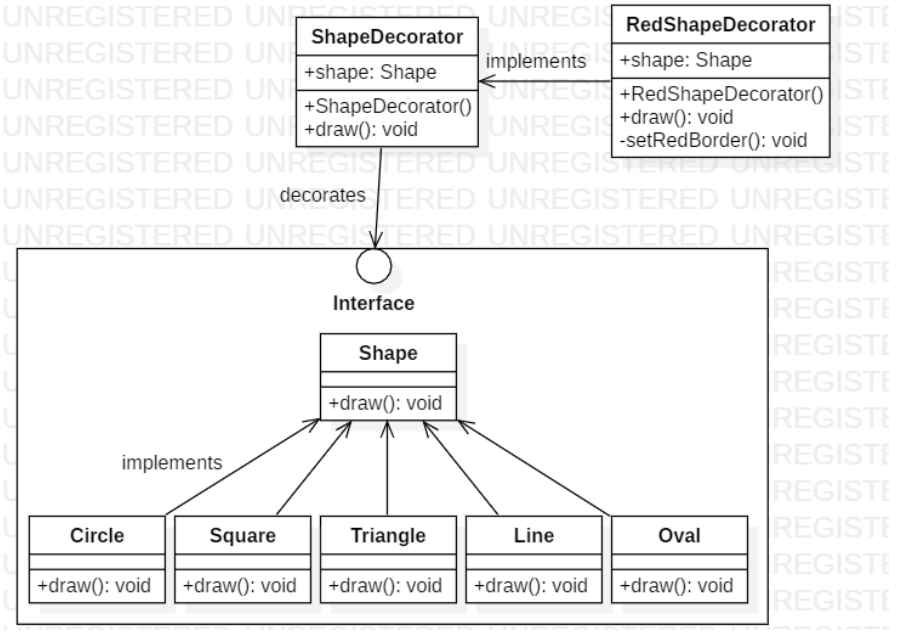
1.抽象工厂模式：



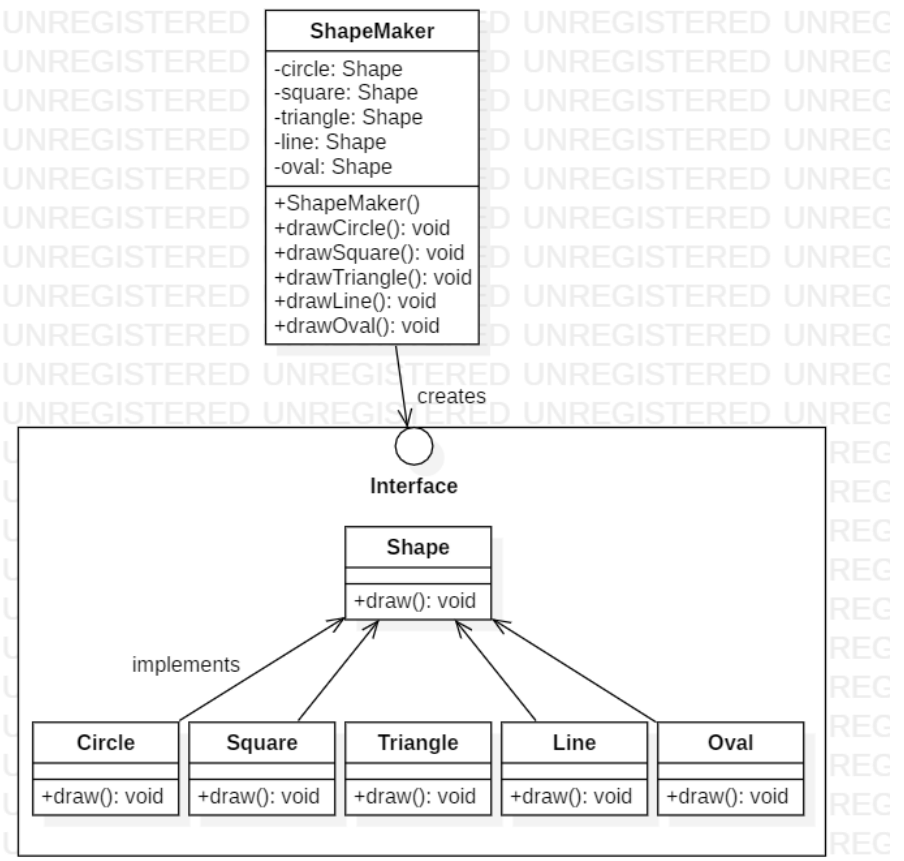
2.原型模式：



3.装饰者模式：


ShapeDecorator
+shape: Shape
+ShapeDecorator()
+draw(): void
implements
RedShapeDecorator
+shape: Shape
+RedShapeDecorator()
+draw(): void
-setRedBorder(): void
decorates
Interface
Shape
+draw(): void
implements
Circle
+draw(): void
Square
+draw(): void
Triangle
+draw(): void
Line
+draw(): void
Oval
+draw(): void

4.外观模式：


ShapeMaker
-circle: Shape
-square: Shape
-triangle: Shape
-line: Shape
-oval: Shape
+ShapeMaker()
+drawCircle(): void
+drawSquare(): void
+drawTriangle(): void
+drawLine(): void
+drawOval(): void
creates
Interface
Shape
+draw(): void
implements
Circle
+draw(): void
Square
+draw(): void
Triangle
+draw(): void
Line
+draw(): void
Oval
+draw(): void

3 实现方案

MyFrame.java

```java
public class MyFrame extends JFrame {

    public static int saved = 0;
    // 声明颜色属性，并赋默认值
    public static Color c = Color.black;
    public static int lengthCount = 0; // 铅笔或橡皮擦图形的存储长度
    public static String fontName = new String( original: "宋体");
    public static int fSize = 16;
    public static int index = 0;// 图形形状的标记
    public static Shape[] itemList = new Shape[5000];// 图形存储单元
    public static int stroke = 1;// 画笔粗细
    public static Color color = Color.black;// 用于存放当前颜色
    public static int currentChoice = 3; // 初始状态是画笔
    public static JLabel statusBar;// 鼠标状态
    public static MyPanel drawingArea; // 画图区域
    // 菜单类
    MyMenu menu;

    // 工具条
    MyToolbar myToolbar;

    // 调色板
    ColorPanel colorPanel;

    public MyFrame() {

    }
```

```java
    public MyFrame(String s) {
        this.setTitle(s);
        this.setSize( width: 1100,  height: 800);// 设置窗口大小
        this.setLocationRelativeTo(null);// 居中显示

        menu = new MyMenu();//菜单

        myToolbar = new MyToolbar();
        colorPanel = new ColorPanel();
        drawingArea = new MyPanel();
        add(colorPanel, BorderLayout.SOUTH);
        add(drawingArea, BorderLayout.CENTER);
        statusBar = new JLabel();//标签组件
        statusBar.setBackground(new Color( r: 195,  g: 195,  b: 195));
        statusBar.setOpaque(true);// 设置该组件为透明
        statusBar.setHorizontalAlignment(SwingConstants.CENTER);

        drawingArea.createNewGraphics();

        this.setVisible(true);
    }
```

```
class MyMenu {
    /**
     * 菜单初始化部分
     */

    private JMenuBar jMenuBar;// 菜单条
    private JMenuItem file_item_new, file_item_open, file_item_save, file_item_exit;// 定义文件菜单的菜单项
    private JMenuItem help_item_about;
    private JMenuItem help_item_manual;
    private JMenu file_menu,help_menu;// 定义文件、帮助菜单
    public MyMenu() {

        jMenuBar = new JMenuBar();
        // 实例化菜单对象
        file_menu = new JMenu( s: "File");
        help_menu = new JMenu( s: "Help");
        file_item_new = new JMenuItem( text: "new");
        file_item_open = new JMenuItem( text: "open");
        file_item_save = new JMenuItem( text: "save");
        file_item_exit = new JMenuItem( text: "exit");
        help_item_manual = new JMenuItem( text: "manual");
        help_item_about = new JMenuItem( text: "about");


        // 给文件菜单设置监听
        file_item_new.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) { menu.newFile(); }
        });
```

```
        help_item_about.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {...}
        });

        // 添加菜单项到菜单
        file_menu.add(file_item_new);
        file_menu.add(file_item_open);
        file_menu.add(file_item_save);
        file_menu.add(file_item_exit);
        help_menu.add(help_item_manual);
        help_menu.add(help_item_about);
        // 添加菜单到菜单条
        jMenuBar.add(file_menu);
        jMenuBar.add(help_menu);
        // 添加菜单条
        setJMenuBar(jMenuBar);
    }
```

```
    // 保存图形文件
    private void saveFile() {
        // 文件选择器
        JFileChooser fileChooser = new JFileChooser();
        // 设置文件显示类型为仅显示文件
        fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
        fileChooser.setMultiSelectionEnabled(false);
        // 返回当前的文本过滤器，并设置成当前的选择
        fileChooser.setFileFilter(fileChooser.getFileFilter());
        // 弹出一个 文件选择器对话框
        int result = fileChooser.showSaveDialog( parent: MyFrame.this);
        if (result == JFileChooser.CANCEL_OPTION) {
            return;
        }
        File fileName = fileChooser.getSelectedFile();
        String t = fileName.getPath() + ".png";
        fileName = new File(t);
        fileName.canWrite();
        BufferedImage image = createImage(drawingArea);
        try {
            ImageIO.write(image, formatName: "png", fileName);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

MyPanel.java

```
// 鼠标事件mouseAction类，继承了MouseAdapter，用来完成鼠标相应事件操作
class mouseAction extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        itemList[index].x1 = e.getX();
        itemList[index].x2 = e.getX();
        itemList[index].y1 = e.getY();
        itemList[index].y2 = e.getY();
        // 如果当前选择的图形是画笔或者橡皮檫，则进行下面的操作
        if (currentChoice == 3 || currentChoice == 13 || currentChoice == 14) {
            index++;
            lengthCount=1;
            createNewGraphics();
        }
    }
```

```
    public void mouseReleased(MouseEvent e) {

        if (currentChoice == 3 || currentChoice == 13 || currentChoice == 14) {
            itemList[index].x1 = e.getX();
            itemList[index].y1 = e.getY();
            lengthCount++;
            itemList[index].length = lengthCount;
        }
        itemList[index].x2 = e.getX();
        itemList[index].y2 = e.getY();
        repaint();
        index++;
        createNewGraphics();
    }
```

```
    public void mouseEntered(MouseEvent e) {

    }

    public void mouseExited(MouseEvent e) {

    }
}

// 鼠标事件mouseMotion类继承了MouseMotionAdapter,用来完成鼠标拖动和鼠标移动时的响应操作
class MouseMotion extends MouseMotionAdapter {
    public void mouseDragged(MouseEvent e) {
        itemList[index].x2 = e.getX();
        itemList[index].y2 = e.getY();
        if (currentChoice == 3 || currentChoice == 13 || currentChoice == 14) {
            itemList[index - 1].x1 = e.getX();
            itemList[index - 1].y1 = e.getY();
            itemList[index].x1 = e.getX();
            itemList[index].y1 = e.getY();
            index++;
            lengthCount++;
            createNewGraphics();
        }
        repaint();
    }
    public void mouseMoved(MouseEvent e) {

    }
}
```

Shape.java

```
public abstract class Shape implements Serializable {

    public int x1, y1, x2, y2;// 绘制图形的坐标
    public Color color;// 画笔颜色
    public int width;// 画笔粗细
    public int length; // 铅笔或橡皮擦的笔迹长度
    public BufferedImage image; // 存放待打开图片
```

```
    public JPanel board; // 绘画的画板
    public int  fontSize;//字体大小
    public String fontName;//字体
    public String s;//文本
    //抽象方法
    public abstract void draw(Graphics2D g);

}
```

4 实验结果

4.1 实现功能介绍

1.良好的图形用户界面，界面中提供铅笔、直线、矩形、椭圆、圆、三角形、涂鸦、橡皮擦等工具，并可以对对应的图形进行颜色填充、边界加粗。

2.允许用户添加文字描述，并可以对文字进行字体大小、颜色、字体（如“宋体”）调整。

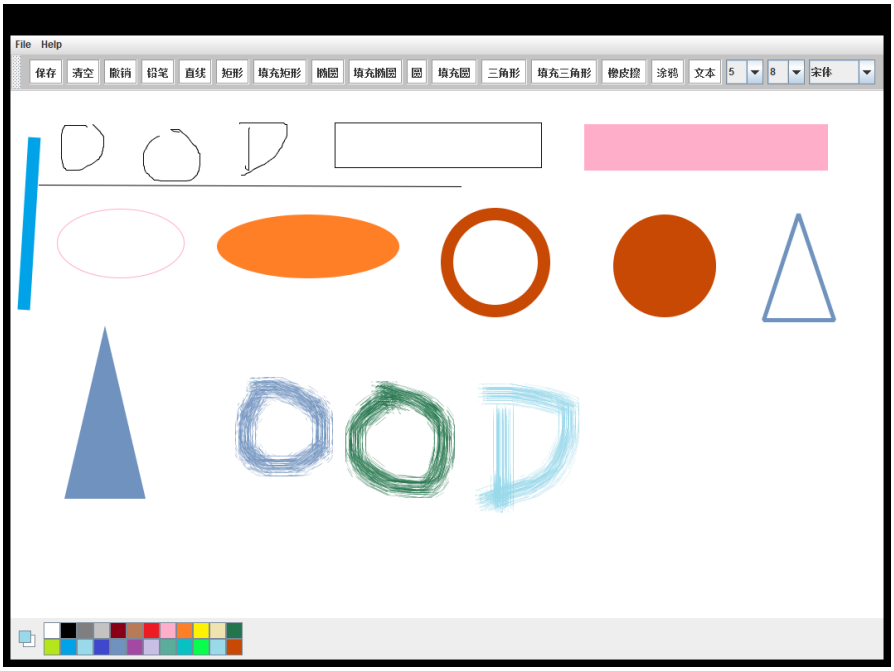
3.支持撤销上一步操作的功能，以及清空画布的功能。

4.提供打开、新建、保存、退出文件的操作。其中打开的文件类型可以是 jpg、png、bmp 格式，保存的文件类型是 png 格式。

5.提供部分功能的 HELP，如查看使用手册、ABOUT。

4.2 界面展示

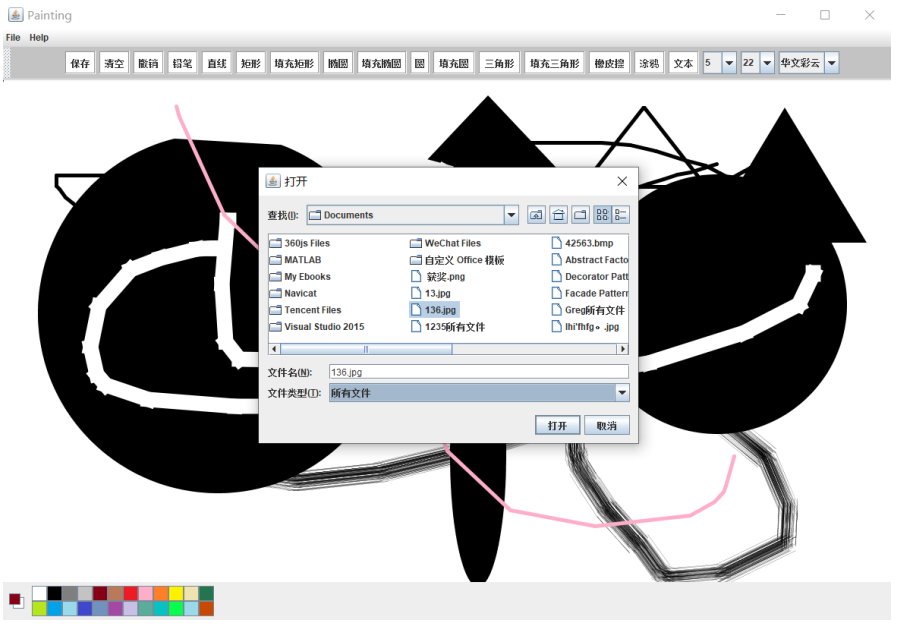
1.绘制功能：



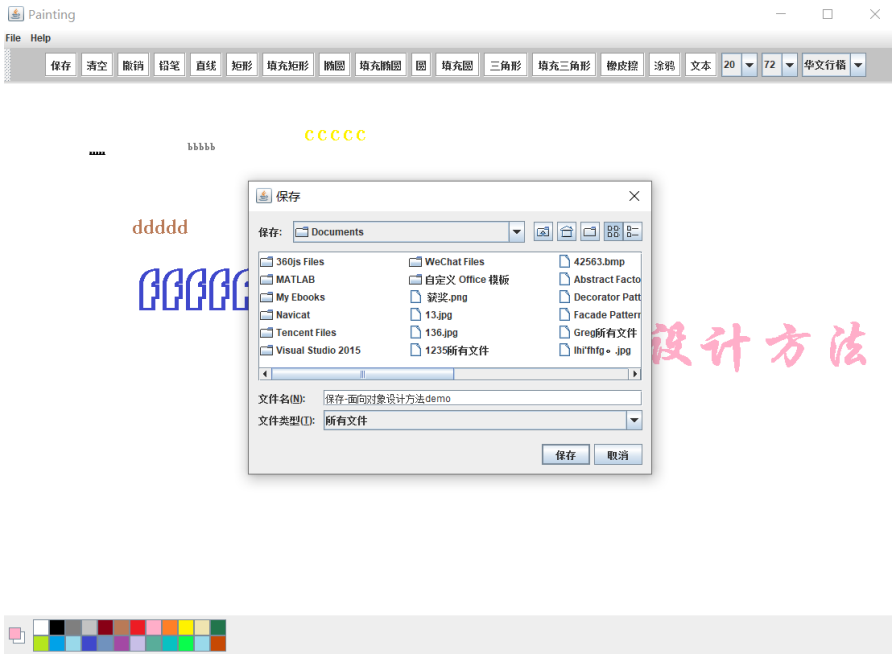
2.文字描述：

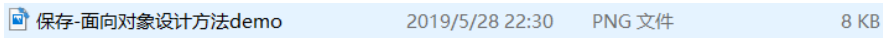

Painting
File Help
保存 清空 撤销 铅笔 直线 矩形 填充矩形 椭圆 填充椭圆 圆 填充圆 三角形 填充三角形 橡皮擦 涂鸦 文本 20 72 华文行楷
ccccc
bbbbb
eeeee
ddddd
面向对象设计方法
输入
请输入文字
确定 取消

3.文件操作：


Painting
File Help
保存 清空 撤销 铅笔 直线 矩形 填充矩形 椭圆 填充椭圆 圆 填充圆 三角形 填充三角形 橡皮擦 涂鸦 文本 5 22 华文彩云
打开
查找(I): Documents
360js Files
MATLAB
My Ebooks
Navicat
Tencent Files
Visual Studio 2015
WeChat Files
自定义 Office 模板
获奖.png
13.jpg
136.jpg
1235所有文件
42563.bmp
Abstract Facto
Decorator Patt
Facade Pattern
Greg所有文件
文件名(N): 136.jpg
文件类型(T): 所有文件
打开 取消

12





保存-面向对象设计方法demo 2019/5/28 22:30 PNG 文件 8 KB

4.其他

5 小结

功能点较少，对于知识的掌握还是不够。

此外，在舒适性和便捷性而言，java FX 应该比 Swing 和 AWT 更好。