实验 2——Mini CAD

一. 问题介绍

本实验要求使用 Java 的 AWT 和 Swing，完成简单的绘图软件 Mini CAD。软件的具体功能如下。

绘制新的图形。可以绘制直线、矩形、圆形和字符串四种基本图形。按下鼠标左键拖动，直到松开鼠标左键，可以确定图形的起始位置和大小，颜色、粗细等属性使用默认值。如果绘制类型为字符串，拖动后会出现 Dialog 对话框，用户可以输入字符串。

删除画板上的图形。首先选中画板上的一个图形，按“R”键删除该图形。

拖动图形。选中画板上的某个图形，用鼠标拖动图形位置。

改变图形粗细。选中画板上的某个图形，按“>”键可以使该图形变粗，按“<”键变细。

改变图形大小。选中画板上的某个图形，按“+”键可以使该图形变大，按“-”键变小。

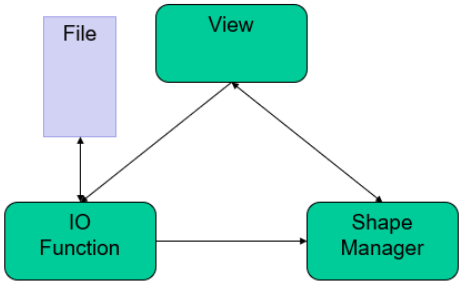
改变图形颜色。选中画板上的某个图形，点击工具栏的颜色按钮，该图形颜色变为颜色按钮上的颜色。

将画板上的图形导出到文件。鼠标右键单机 file 按钮，保存到.cad 文件。

将文件内容导入到画板上。鼠标左键单击 file 按钮，选择文件导入，导入到画板上的图形依然可以改变。

二. 问题分析

本实验是一个 GUI 程序，主要涉及 Shape Manager、View 和 IO Function 三个主要方面，由于程序较为简单，省略了 API 这一层。三者的关系如下图所示，箭头指向被调用方法的模块或资源流到的模块。



View 中 Component，如 JButton、JPanel 等，可以调用 Shape Manager 相关类的方法，从而改变相关类成员变量的内容，也可以调用 IO Function 模块的相关方法。IO Function 模块会调用 Shape Manager 模块的相关方法。在本实验中，Shape Manager 和 IO Function 的相关类与它们的方法如下图所示。

```
public class ShapeManager {
    private static List<Element> ls = new ArrayList<Element>();

    public static List<Element> getList() {

    public static void add(Element e) {

    public static void remove(Element e) {

    public static void removeAll() {
}

public class IOFunction {
    //读文件，处理文件中的字符串
    //通过文件构造相应对象存储进ShapeManager中
    //文件格式见下方save()函数
    public static void open(String path, String file) throws FileNotFoundException, IOException {

    //写文件
    //文件格式如下
    //Line x1 y1 x2 y2 color stroke isSelected
    //Rectangle x y width height color ... ...
    //...
    //StringText要记录内容，可能有空格，此时我们用2个空格代表1个空格（类似于转义符号'\'）
    public static void save(String path, String file) throws IOException {
}
```

三. 数据结构与算法

ArrayList，是 Java 语言自带的一种可以自动增加容量的容器，它由多个数组连接成的链表组成。在实验中，我们定义了 ArrayList 用来存储图形信息。

Generic Container，泛型容器，程序中使用 ArrayList<Type>。

四. 实验关键步骤

1. 定义图形及其管理模块

首先定义抽象类 Element，是四种图形的父类。Element 类定义颜色 eColor，默认为黑色；粗细 strokeWidth，默认为 2.0f；是否被选中的布尔型变量 select，默认为 false。该类还定义共有函数和每个图形不同的抽象函数，其中 draw()函数只执行各种图形通用的代码，各种图形的 draw()方法都需要首先调用 super.draw()方法。

在 Element 类的基础上，定义 Line、Rectangle、Oval 和 StringText 四个子类，分别拥有自己的成员变量和方法。其中，前三个类的操作比较相似，都是通过几何计算完成所需要的功能，比如直线的 prolong()方法是先算斜率 k，根据斜率 k 算出直

线右端点的新值(x+delta, y+k*delta)，而 StringText 通过 Font 类和 FontMetrics 类实现相关的功能。

```
public abstract class Element {
    public Color eColor = Color.black;
    public float strokeWidth = 1.0f;
    public boolean select = false;
    private Stroke s;

    public void draw(Graphics2D g2d) {

    public void setColor(Color c) {

    public abstract void moveTo(int dx, int dy);

    public abstract void prolong();

    public abstract void detract();

    public void wider() {

    public void thinner() {
}
```

管理模块的功能在程序中由 ShapeManager 类实现。图形都存储在 ArrayList 中，可以通过 Element 指针添加、删除，也可以清空，用于文件读取。这里直接实现了 getList()函数，而并没有像 Java 标准的 ArrayList 那样通过 size()和 get()一个一个元素获取。此外，整个程序只需要 1 个 ShapeManager，而且它的耦合度和其他类并不高，所以它的全部方法都设计为 static 方法。

```
public class ShapeManager {
    private static List<Element> ls = new ArrayList<Element>();

    public static List<Element> getList() {

    public static void add(Element e) {

    public static void remove(Element e) {

    public static void removeAll() {
}
```

2. 创建 GUI 界面并添加相关事件监听器

由演示视频我们可以使用 Border Layout 作为最上层的布局，中心为画板，东侧为工具栏，工具栏可以使用 Grid Layout，最后一行为多个颜色按钮，又可以在工具栏的 Grid Layout 中嵌套一个 4 行 3 列的 Grid Layout。

```
public Starter() {
    ToolBar t = new ToolBar();
    frm = new JFrame("MiniCAD");
    frm.setSize(wSize, wSize);
    frm.setResizable(false);
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.setLayout(new BorderLayout());
    frm.add(t, BorderLayout.EAST);
    frm.add(t.cadPanel, BorderLayout.CENTER);
    frm.setLocationRelativeTo(null);
    frm.setVisible(true);
}
```

工具栏是最主要的容器，在程序中被定义为 ToolBar 类，它包含众多按钮和画板类 CADPanel。之所以把画板作为 ToolBar 的成员变量，是因为这样比较容易实现先选中画板上的图形，再通过颜色按钮改变图形的颜色。在 ToolBar 类中，我们又定义了 ButtonListener 类、IOListener 类和 CADColorJPanel 类，分别对应选择绘制图形的类型的按钮、实现文件 Export 和 Import 的按钮和决定被选中图形颜色的按钮。

```
public class ToolBar extends JPanel {
    private static final long serialVersionUID = 1L;
    private static JButton[] jbtn;
    private static Dimension pSize;
    private static int number = 0;
    private static Color color = Color.black;
    private static final String[] btnHint = { "line", "rectangle", "text", "circle", "file" };
    public static final Color[] clr = { Color.black, Color.blue, Color.cyan, Color.darkGray,
            Color.gray, Color.green, Color.lightGray, Color.pink, Color.orange,
            Color.red, Color.white, Color.yellow };
    protected CADPanel cadPanel = new CADPanel();

    // 设置颜色按钮
    class CADColorJPanel extends JPanel {

    class ButtonListener implements ActionListener {

    class IOListener implements MouseListener {

    // 设置四种按钮并添加相应的监听器
    // 添加颜色按钮所在的Component
    public ToolBar() {

    public static int getButton() {
}
```

CADPanel 类主要负责画板上的相关操作，包括鼠标事件、键盘事件和绘制三个主要事件。paintComponent()方法负责把 ShapeManager 中所有图形画在画板上（调用各自的 draw()方法，从而获得粗细、大小、位置和颜色不同的图形。addElement()方法根据 CADPanel 成员变量 type 的不同，把新绘制的图形存入 ShapeManager 中，接着重新绘制画板。CADPanel()是构造函数，获取焦点（类似于输入文本时闪烁的竖线）并添加鼠标和键盘事件。值得注意的是，每次鼠标单击后，需要让 CADPanel 重新获取焦点，所以在 mouseClicked()方法最后要加上 requestFocus()这一命令。

```
public class CADPanel extends JPanel {
    private static final long serialVersionUID = 1L;
    protected static final int delta = 3;
    private static Point start, end;
    private static int type = 0;
    private static boolean select = false;

    public CADPanel() {

    //把新的图形按照种类进行构造，并添加进ShapeManager中
    //type为0代表直线，1代表矩形，2代表字符串，3代表椭圆（实际上只支持圆形）
    //最后重新绘制JPanel
    public void addElement() {

    //把ShapeManager所管理的图形绘制在JPanel上
    public void paintComponent(Graphics g) {
}
```

3. 文件操作

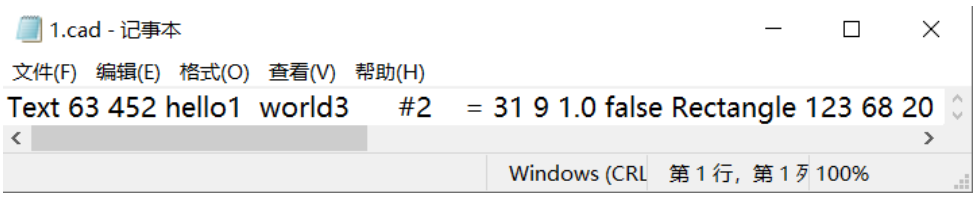
文件操作主要由 IOFunction 和 Starter 两个类完成。

Starter 类有 save()和 open()两个静态方法，都是先弹出 FileDialog，再调用 IOFunction 里面的方法，实现真正的文件读写。此外，这两个方法还负责处理一些文件读写的异常，比如 FileNotFoundException 等。

IOFunction 类只有 open()和 save()两个方法。首先是 open()方法。该方法首先清空 ShapeManager 中的内容，然后使用 BufferedReader 的 readLine()先把文件内容读到临时字符串变量中，再通过空格分隔字符串，放进字符串的 ArrayList 中。上述步骤需要注意的是，StringText 类的 Element 会把字符串内容存储进文件中，用来区分的方法类似于转义符，即每 1 个空格在文件里被存储为 2 个，所以从文件读进来的时候就需要注意这一点，再把 2 个空格变成 1 个，并且不在此处把字符串断开。在成功把原始字符串按正确方法断开后，我们可以根据类型提示构造相应的 Element，存储在 ShapeManager 中。下面是 save()方法。该方法需要注意存储格式，具体格式如下方代码图片注释和文件图片所示。其中 Text 的 hello 和 world 后面的数字分别代表输入时的空格数 1 和 3，可以验证 open()时所说的用 2 个空格代替 1 个空格的算法是可行的。

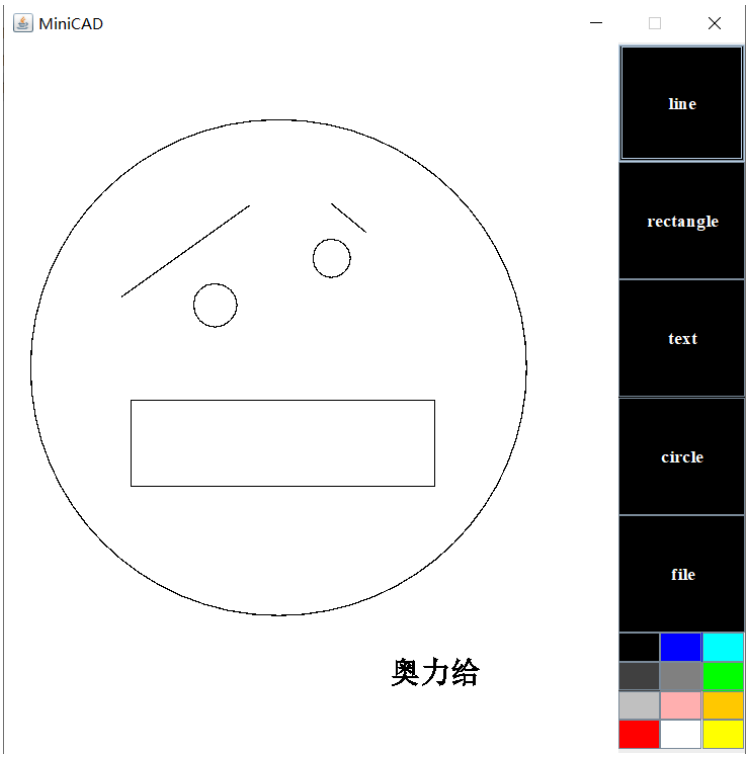
```
public class IOFunction {
    //读文件，处理文件中的字符串
    //通过文件构造相应对象存储进ShapeManager中
    //文件格式见下方save()函数
    public static void open(String path, String file) throws FileNotFoundException, IOException {

    //写文件
    //文件格式如下
    //Line x1 y1 x2 y2 color stroke isSelected
    //Rectangle x y width height color ... ...
    //...
    //StringText要记录内容，可能有空格，此时我们用2个空格代表1个空格（类似于转义符'\'）
    public static void save(String path, String file) throws IOException {
}
```

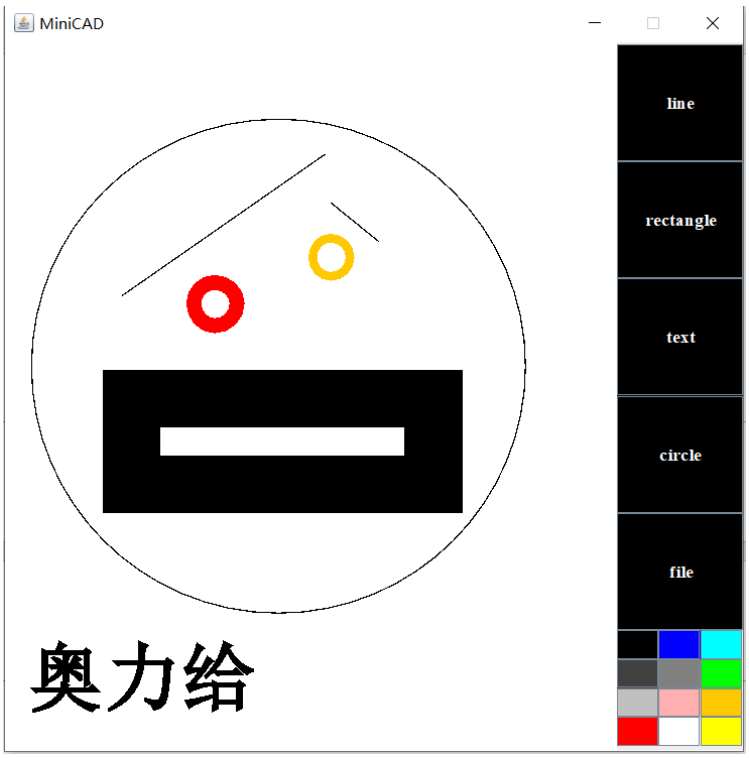


五. 结果展示

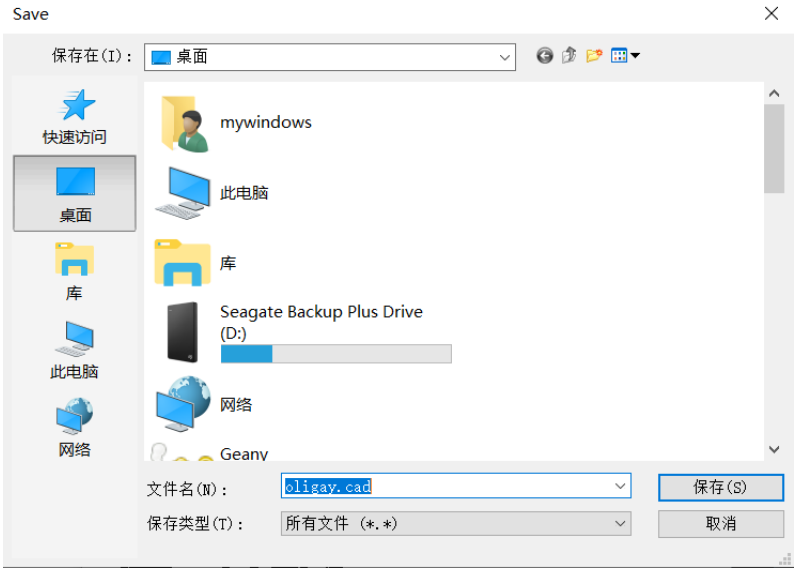
1. 绘制图形

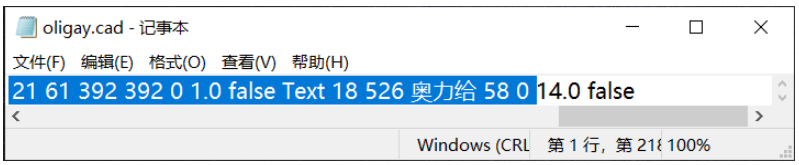


2. 拖动图形位置（奥力给）、改变颜色（眼睛）、改变大小（眼眉与奥力给）、改变粗细（嘴与眼睛）

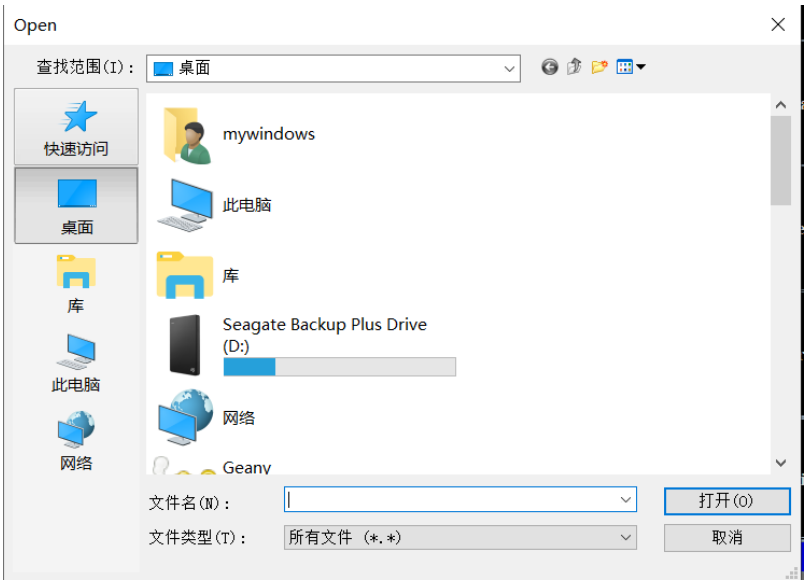


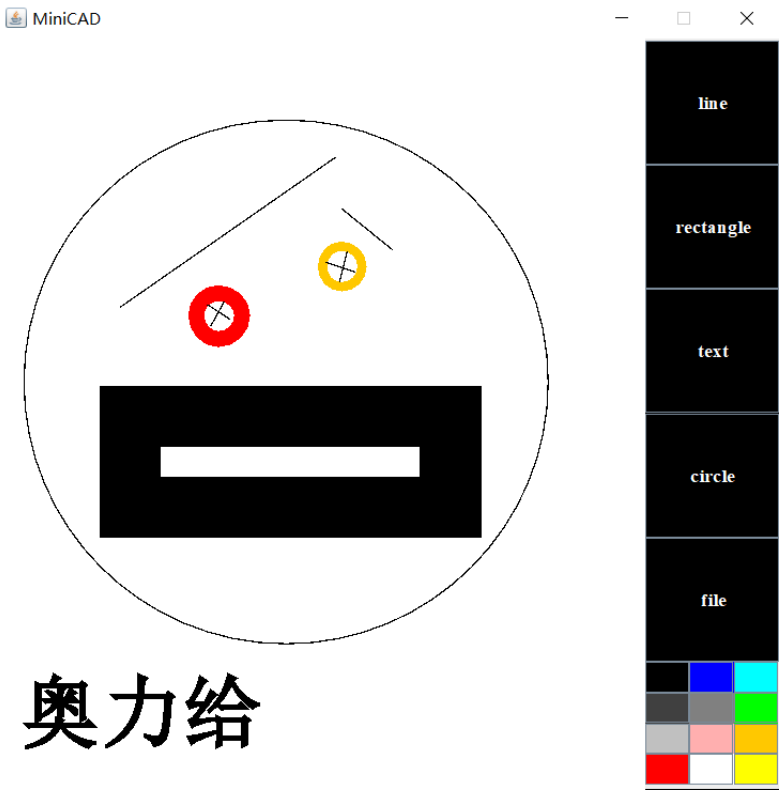
3. 存储文件





4. 读取文件，读入的内容仍然可以改变（眼睛）




MiniCAD
line
rectangle
text
circle
file
奥力给