浙江大學

Project 1 - miniCAD

授课教师	:	翁恺
姓　　名	:	韩恺荣
学　　号	:	3200105385
日　　期	:	2022.11.19

一、实验内容

（1）实验配置：

- java版本java 18及以上
- IDE:idea2022

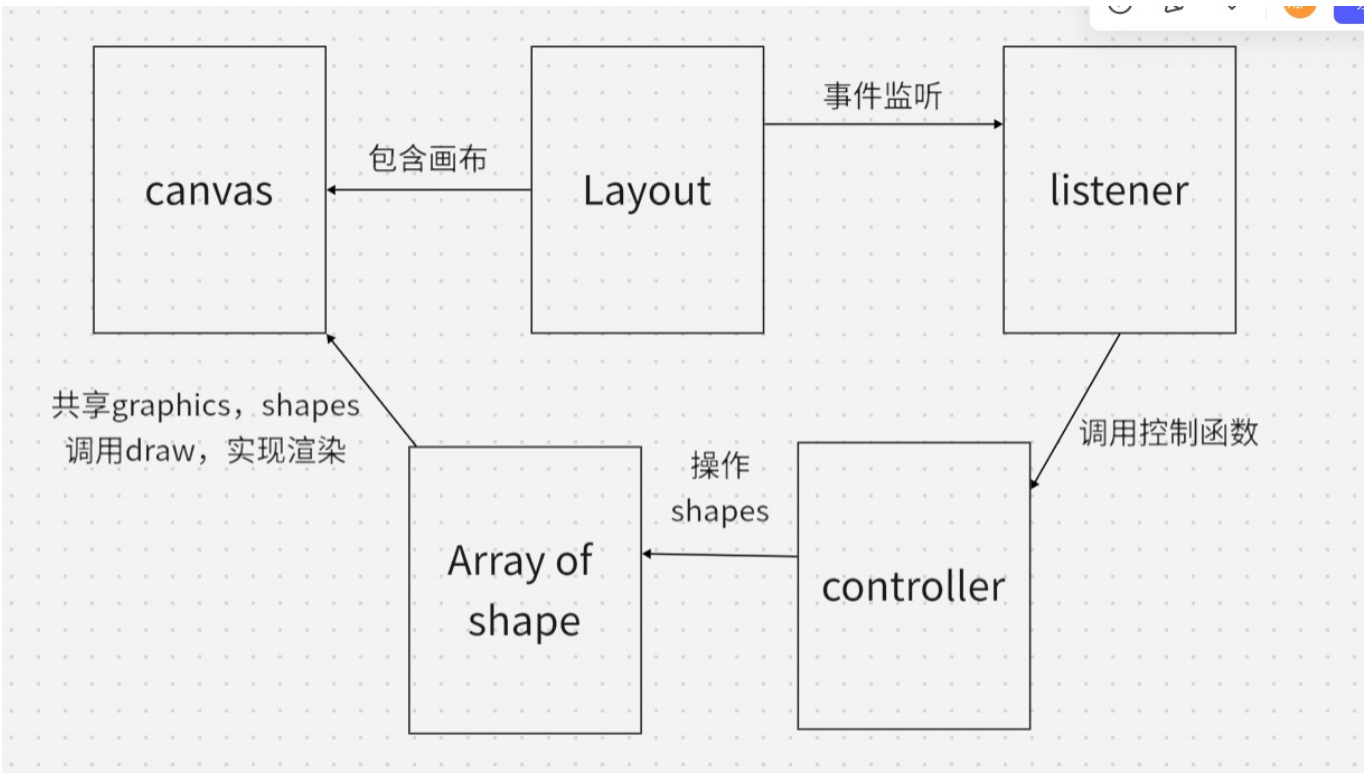
(2)实验目标：

用Java的awt和swing做一个简单的绘图工具，以CAD的方式操作，能放置直线、矩形、圆和文字，能选中图形，修改参数，如颜色等，能拖动图形和调整大小，可以保存和恢复。功能请参考视频演示。

二、实验设计：

1. 实验类SVC模型设计：

本次实验采用类SVC模型设计实现miniCAD程序，主要涉及到的类共有三个，分别是：MyController，Mylistener，Layout_Ctrl，其中，MyController是整个程序的逻辑控制单元，内含有包括shape的Arraylist，Mylistener是整个程序的事件监听单元，对不同的鼠标以及键盘事件进行监听的同时，做出对应的响应，而Layout_Ctrl是整个程序的页面展示单元，集成了Jframe，将各种组件有机的结合在了一起，最终实现了整体的控制逻辑。



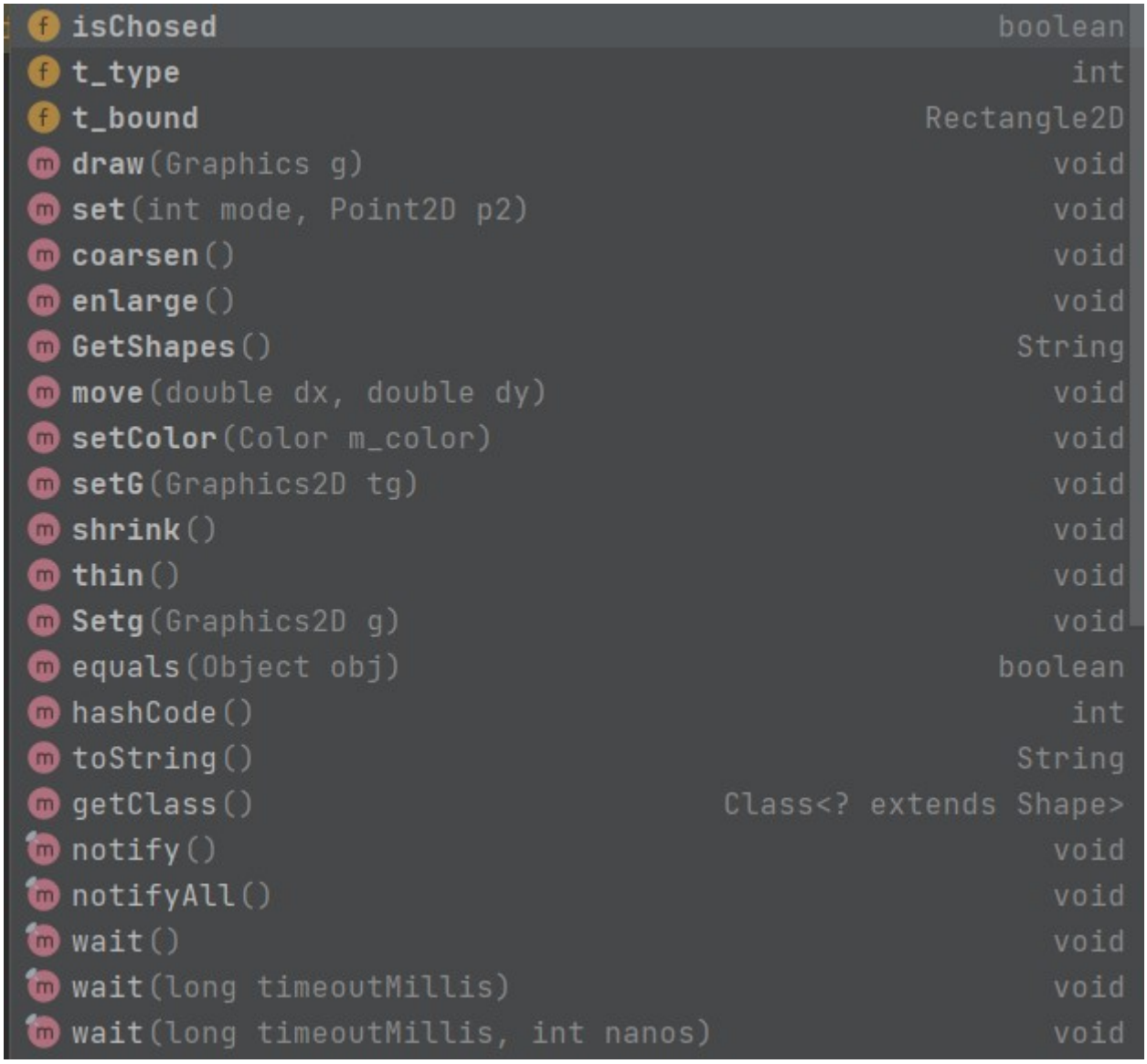
2. Shape类：

实验操作对象的主体，其内部含有的成员如下：

```
public static final int LINE = 0;
    public static final int RECTANGLE = 1;
    public static final int CIRCLE = 2;
```

```
    public static final int TEXT = 3;
    private Color t_color;
    private Rectangle2D rect;
    private Ellipse2D circle;
    private Line2D line;
    private String text;
    private FontMetrics t_fontMetrics;
    private Font t_font;
    Rectangle2D t_bound;
    private Graphics2D tg;
    boolean isChosed = false;
    private Stroke dash = new
BasicStroke(1.0f,BasicStroke.CAP_BUTT,BasicStroke.JOIN_ROUND,3.5f,new float[]
{15,10,},0f);
    public int t_type;
    private float t_weight;
    private Point2D p1;
```

简言之，这些变量记录了shape的坐标，类型，以及粗细颜色等基本的信息，此外，shape类为上层提供了大量的接口，这些接口大多都被controller使用，并提供了有关shape的初始化，移动，改变大小，改变颜色，改变粗细，设置Graphics等方法：



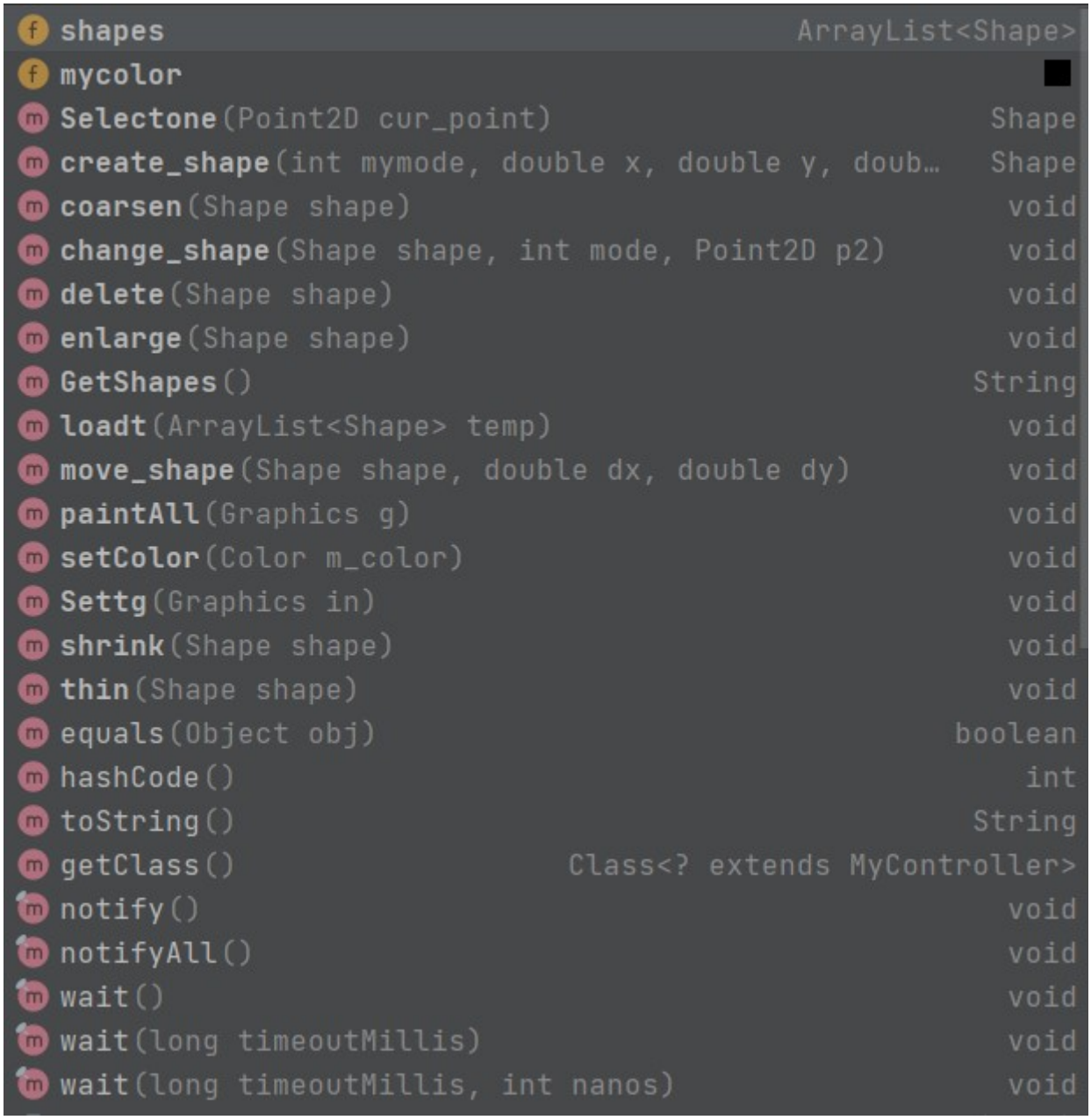
3. MyController类

这个类是程序的全局逻辑控制类，其内部成员变量如下：

```
    ArrayList<Shape> shapes = new ArrayList<>();
    Color mycolor = Color.BLACK;
    private static Graphics2D tg;

    static final int LINE = 0;
    static final int CIRCLE = 1;
    static final int RECTANGLE = 2;
    static final int TEXT = 3;
```

该类将shapes封装在了ArrayList中，并从canvas获取到了Graphics2D，使得shapes，MyController类和canvas类的Graphics2D指针指向同一个对象，这样我们就可以在我们的shape中直接调用draw来实现在canvas上绘画，让canvas和shapes解耦，是实现MVC模型的关键所在。

此外，这个类想listener提供了大量关于shapes的操作方法：



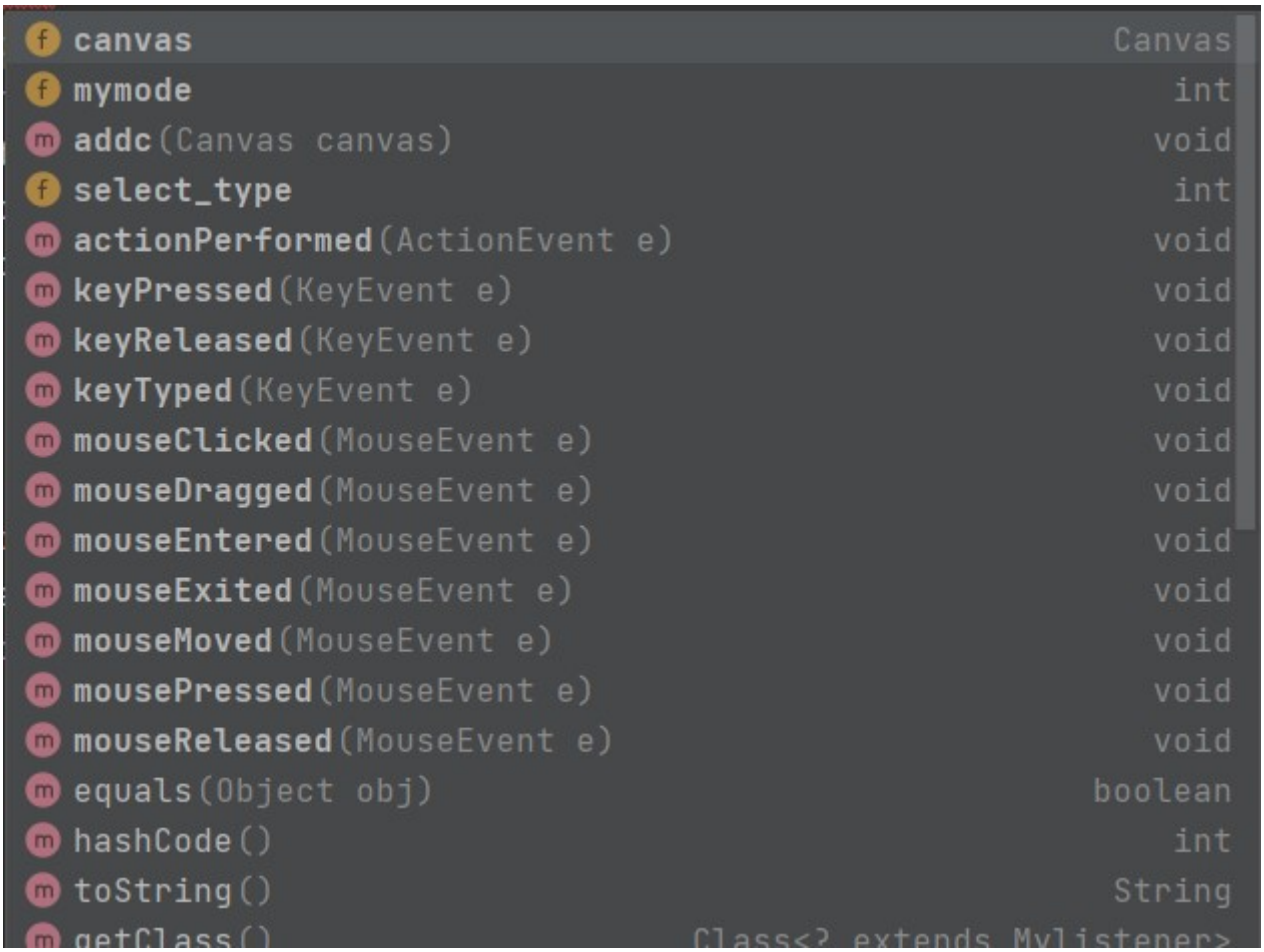
4. Mylistener类：

这个类对我们的画面的全局监视类，能够监视在画布上操作的一举一动，获取键盘和鼠标的动作，实现绘画等基本功能，该类含有的成员变量如下：

```
static final int LINE = 0;
    static final int CIRCLE = 1;
    static final int RECTANGLE = 2;
    static final int TEXT = 3;
    static final int SELECTED = 4;
    static final int FREE = 5;
    static final int COLOR = 6;

    static final int SAVE = 10;
    static final int OPEN = 11;
    int mymode = FREE;
    private static Color m_color = Color.BLACK;
    private Point2D cur_point = new Point2D.Double();
```

```
    private Shape shape = null;
    private MyController myController;
    private String text;
    int select_type;
    Canvas canvas;
```

为了实现对鼠标和键盘事件的监听和响应，该类继承了ActionListener, MouseListener, MouseMotionListener,KeyListener ，并override了需要实现了函数，具体的，listener提供了如下的方法：



5. canvas类：

全局画布类，继承了Jpanel，实现了整体的画面操作区域：

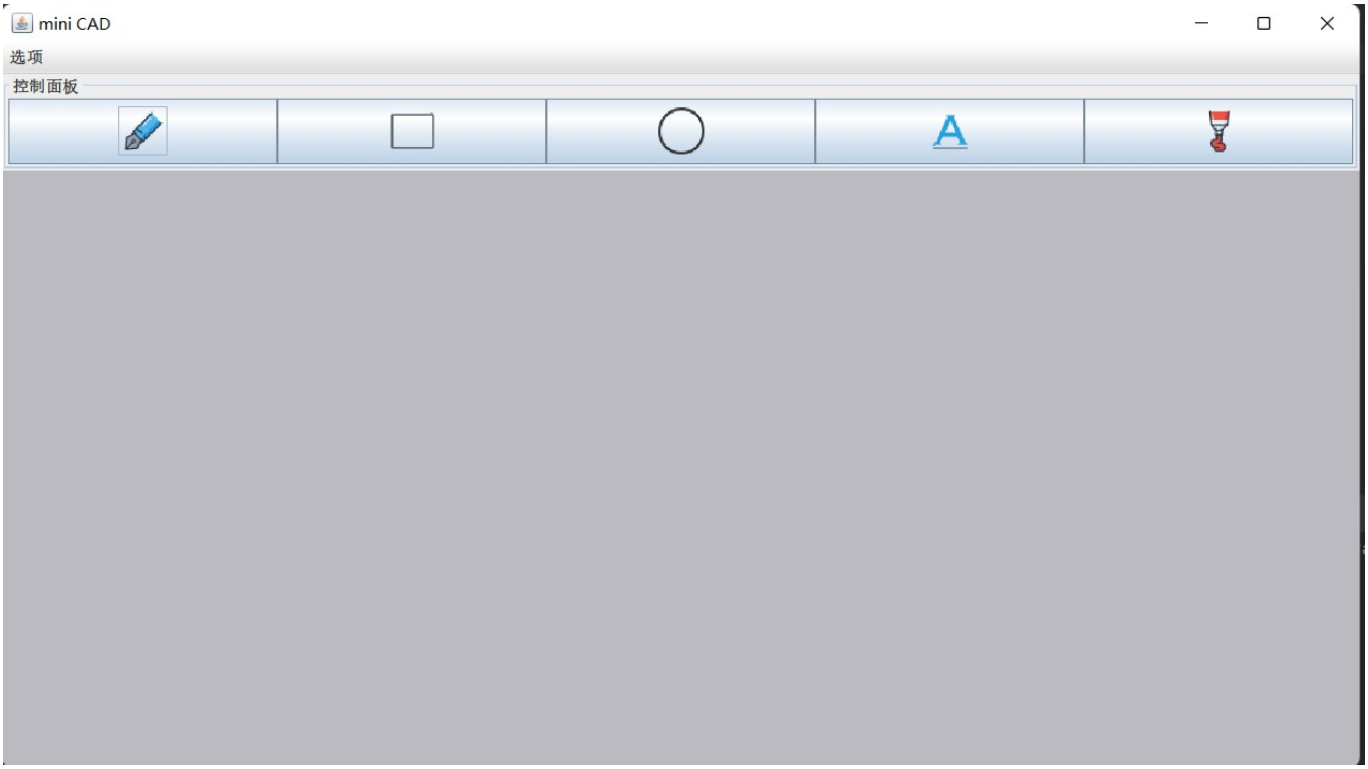
```
public class Canvas extends JPanel {
    MyController myController;
    Mylistener mylistener;
    public Canvas(MyController col,Mylistener lis){
        myController = col;
        mylistener = lis;
        this.setBackground(new Color(186, 186, 191, 255));
        this.addMouseListener(mylistener);
        this.addMouseMotionListener(mylistener);
    }

    @Override
```

```
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        myController.paintAll(g);
    }
}
```

该类提供了paintComponent方法，从而实现了在这里调用该函数也可以由于继承和多态等面向对象的性质，调用到shape类中已经实现的draw方法，从而将画布与将要画的对象分离，在将要画的对象中调用画布的函数。

6. Layout_Ctrl类

整体的画面布局类，在这个类中，我们为不同的button设置不同的点击之后的命令参数，并通过jpanel的组合，实现了界面的布局：



第一行的各种button被封装在了一个Layout是Grid布局的JPanel中，而下面的画布是另一个Jpanel，两个Jpanel之间采用的是默认的BorderLayout布局，canvas位于center，而上面的控制面板位于north。

7. Main类：

```
public class Main {

    public static void main(String[] args) {
        MyController myController = new MyController();
        Mylistener mylistener = new Mylistener(myController);
        Layout_Ctrl layout_ctrl = new Layout_Ctrl(mylistener,myController);
        layout_ctrl.start();
        myController.Settg(layout_ctrl.canvas.getGraphics());
        mylistener.addc(layout_ctrl.canvas);
    }
}
```

程序的入口类.

三、程序编译方式：

程序是基于idea的IDE直接编译实现的，如果要在vscode或者命令行中编译，需要进行如下的操作：

当是在命令行中编译运行：

1. **进入MiniCAD_idea文件夹所在的文件夹**
2. 执行javac MiniCAD_idea/Main.java
3. 运行java MiniCAD_idea.Main

使用vscode类似，不过关键在于，需要在MiniCAD_idea文件夹所在的文件夹打开项目，然后直接使用vscode自带的java插件编译运行即可。

需要强调的是，为了正常的展示出程序的图标，当是命令行运行时，icon文件夹需要在**MiniCAD_idea文件夹所在的文件夹**

```
|-icon
|   |-circle.jpg
|   |-line.jpg
|   ……
|
|-MiniCAD_idea
    |-Main.java
    |-Canvas.java
    ……
```

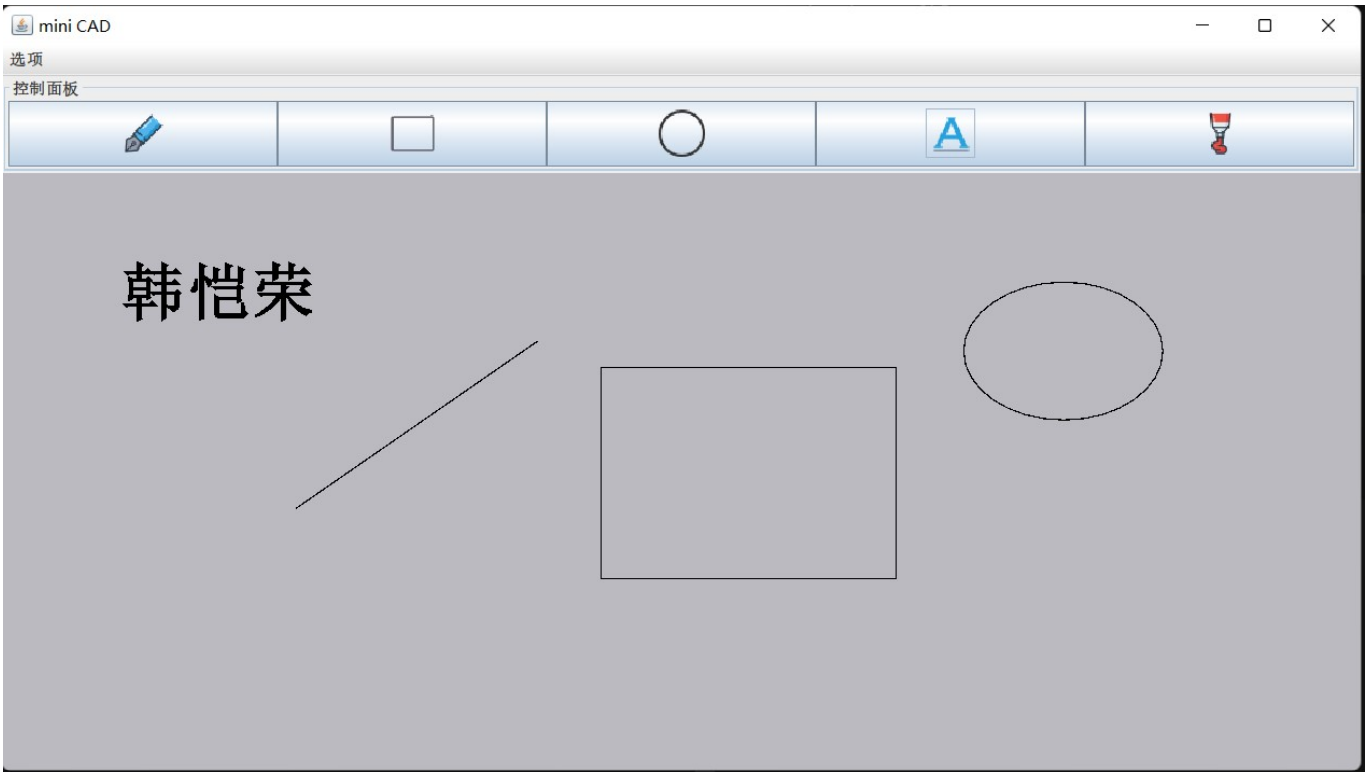
当是用jar包直接运行时，需要保证jar和icon在用一个目录：

```
|-icon
|   |-circle.jpg
|   |-line.jpg
|   ……
|
|-MiniCAD_idea.jar
```

四、程序功能演示：

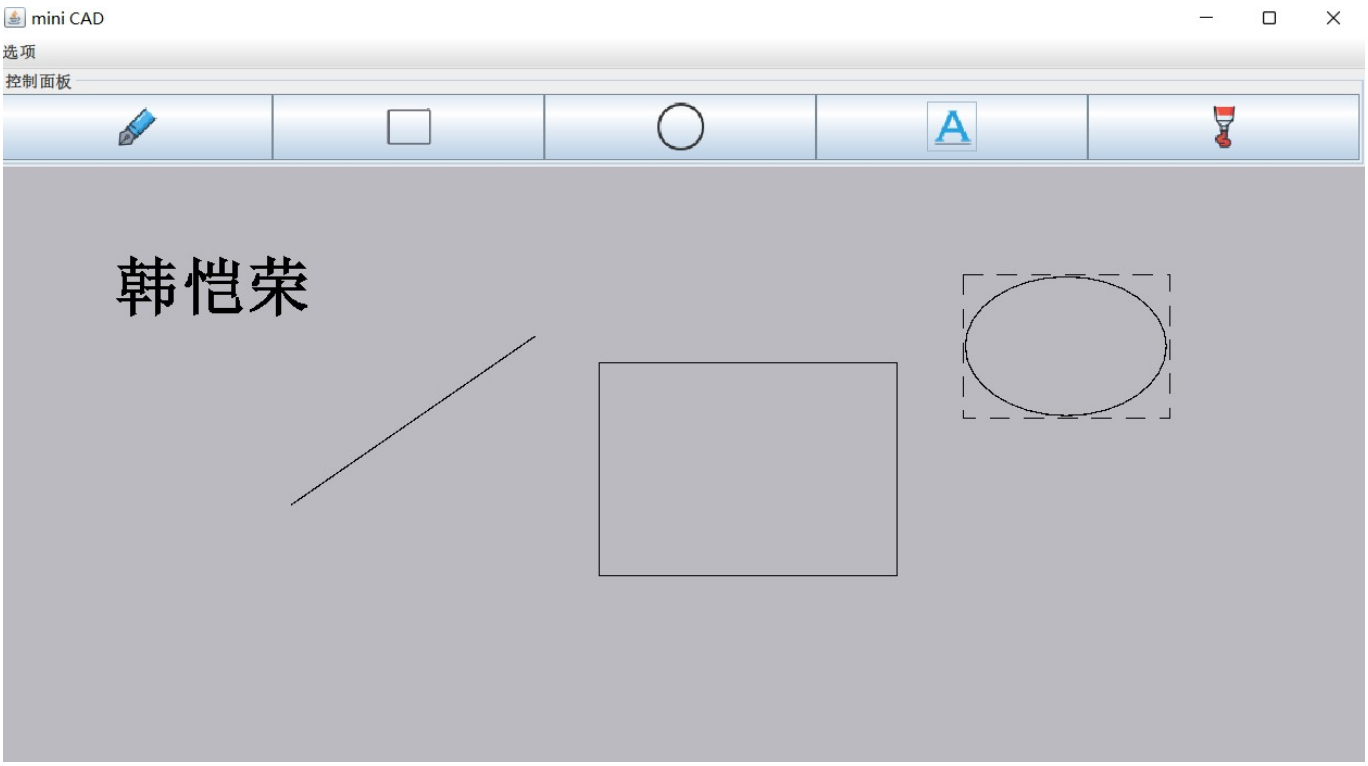
1. 创建图像：

首先点击想要创建图像的对应图标，然后再画布上点击后拖拽调整大小即可：

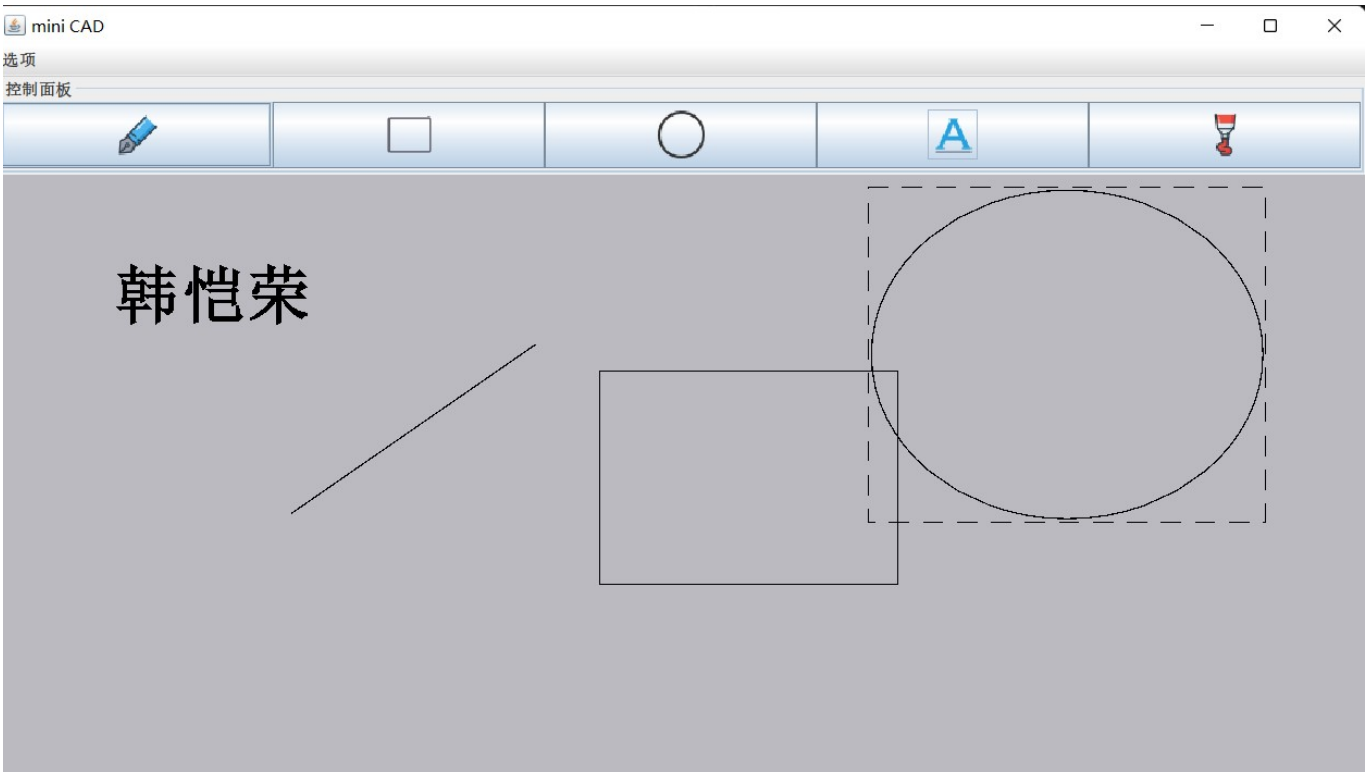


2. 改变图像大小：

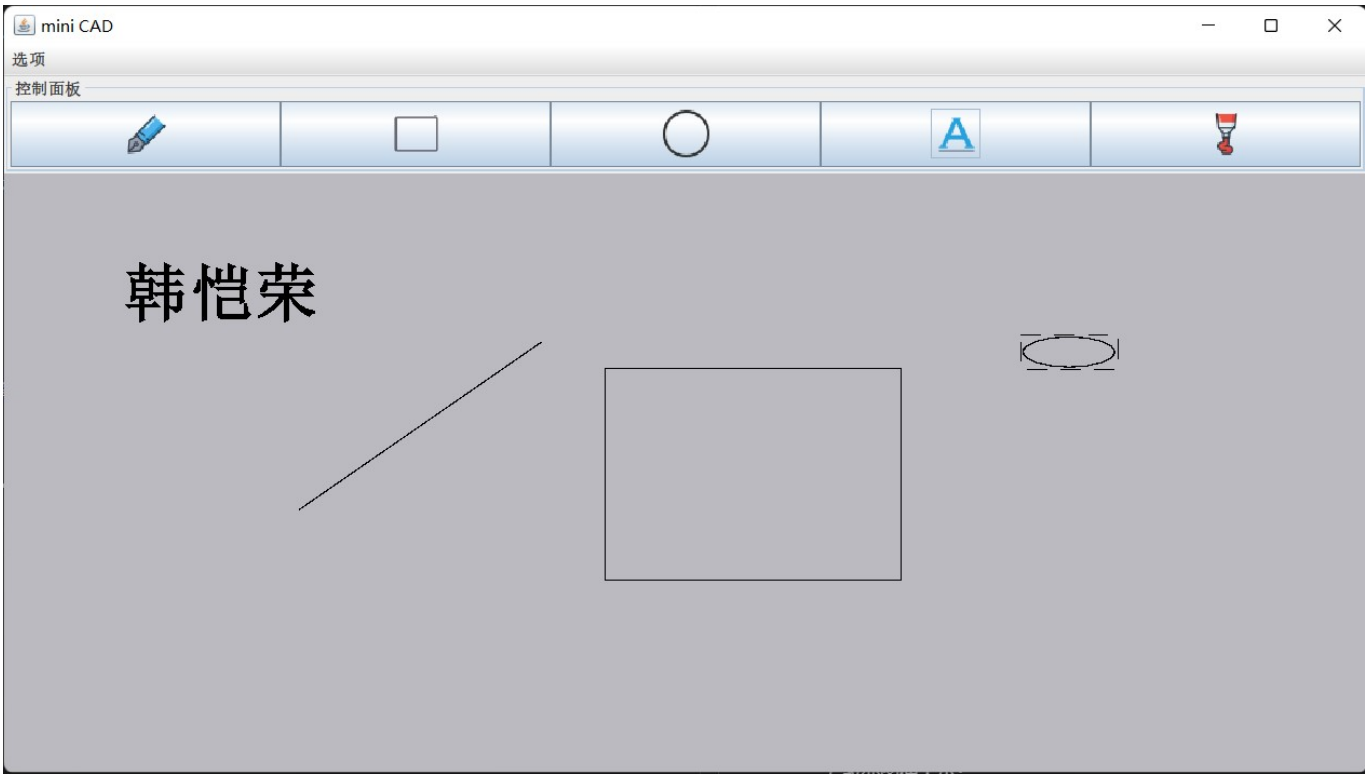
首先选中想要操作的对象，以圆为例：



然后键盘输入"+"

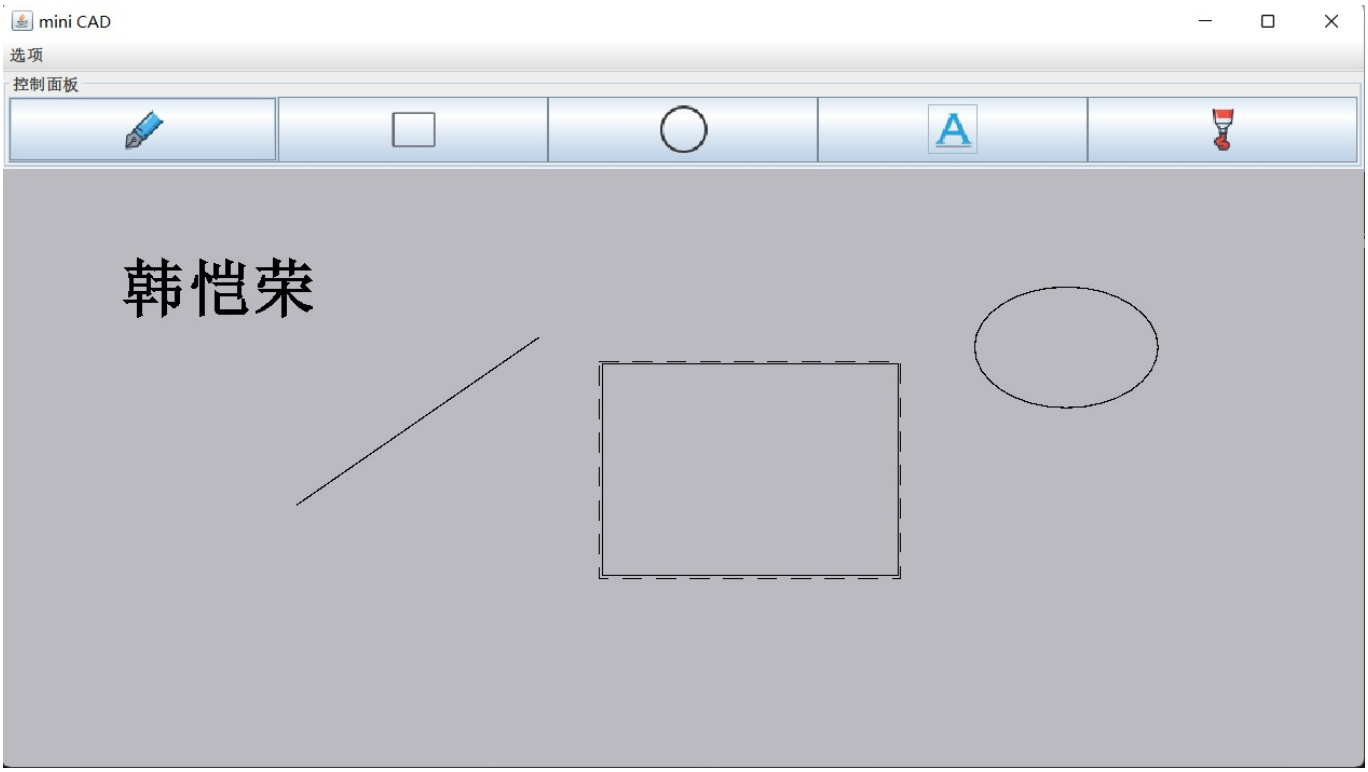


想要缩小类似的，输入"-"即可：

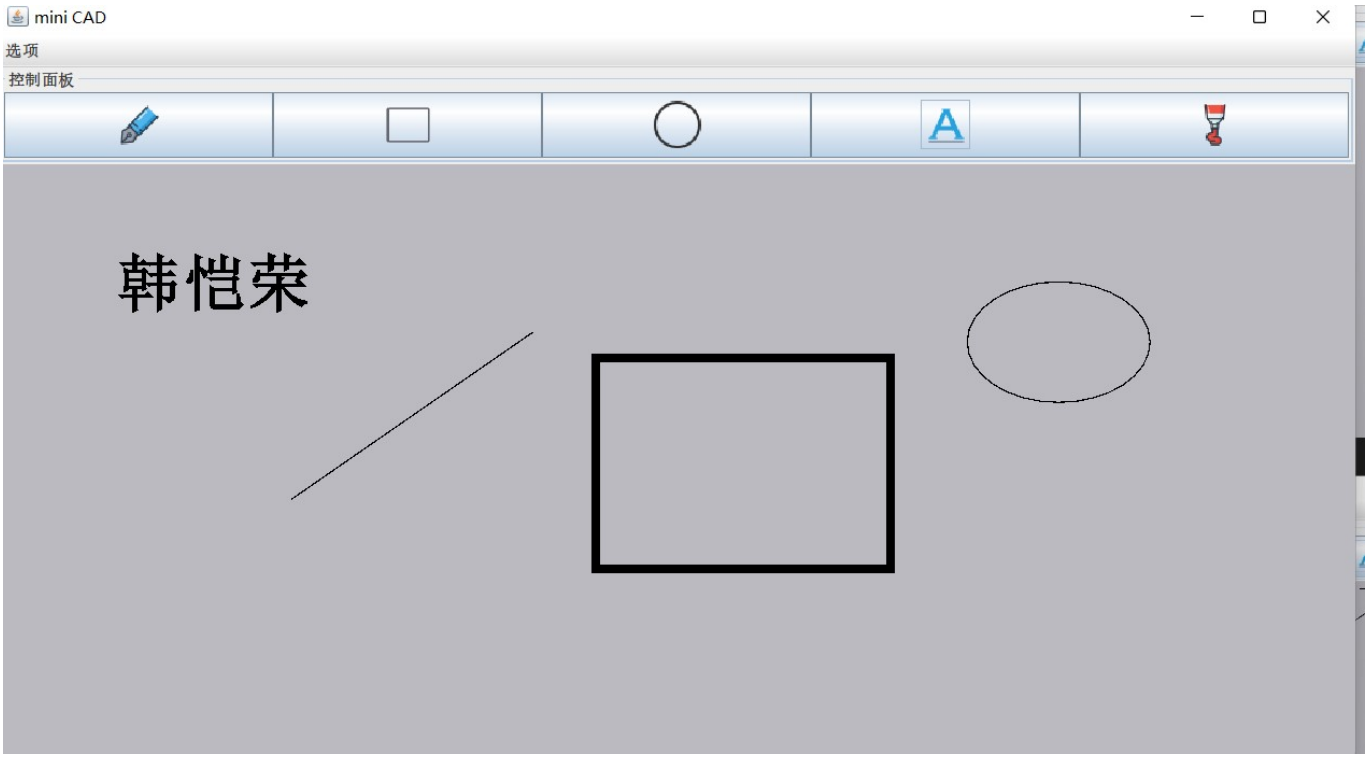


3. 改变粗细：

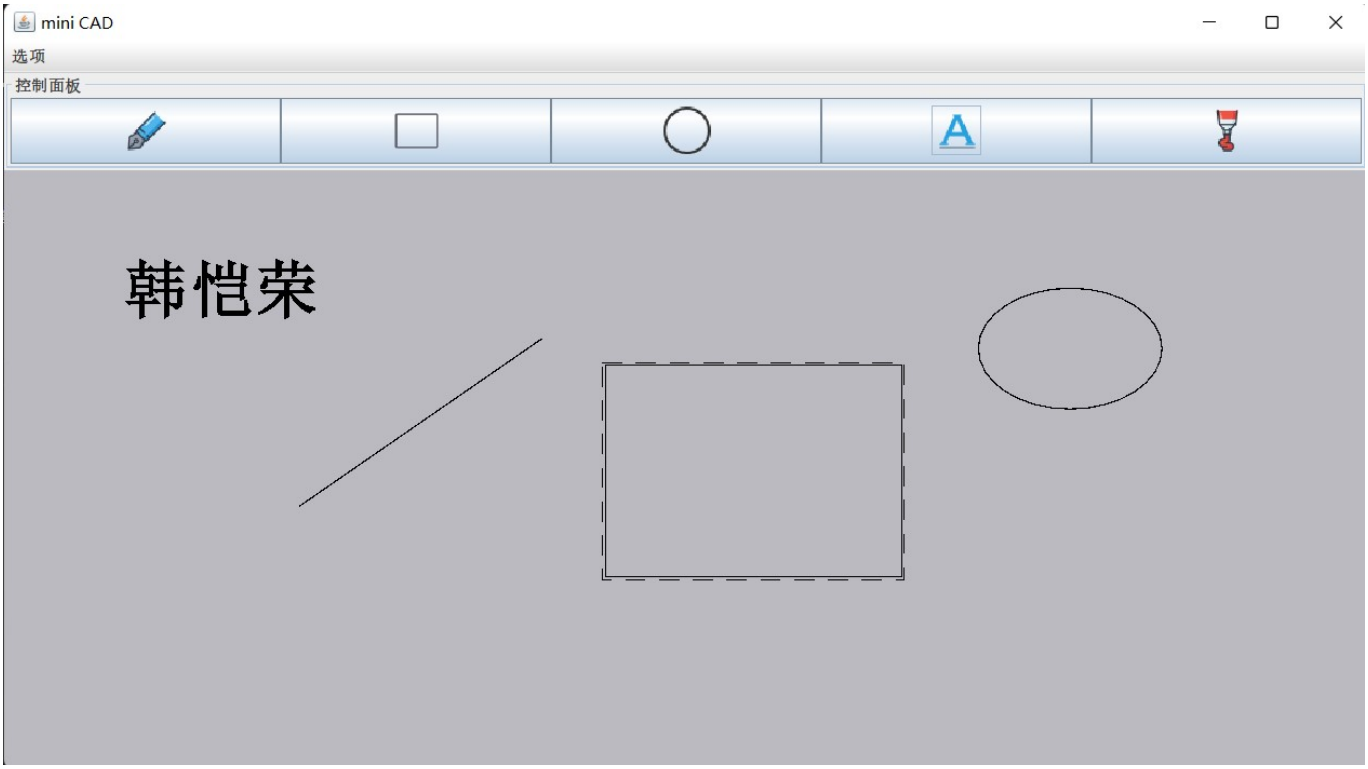
这个功能只对我们的直线，圆，矩形有效，对文字无效，类似的，首先选中我们的图像，以矩形为例，其他类似：



然后输入"<",实现变粗,注意中英文区别:

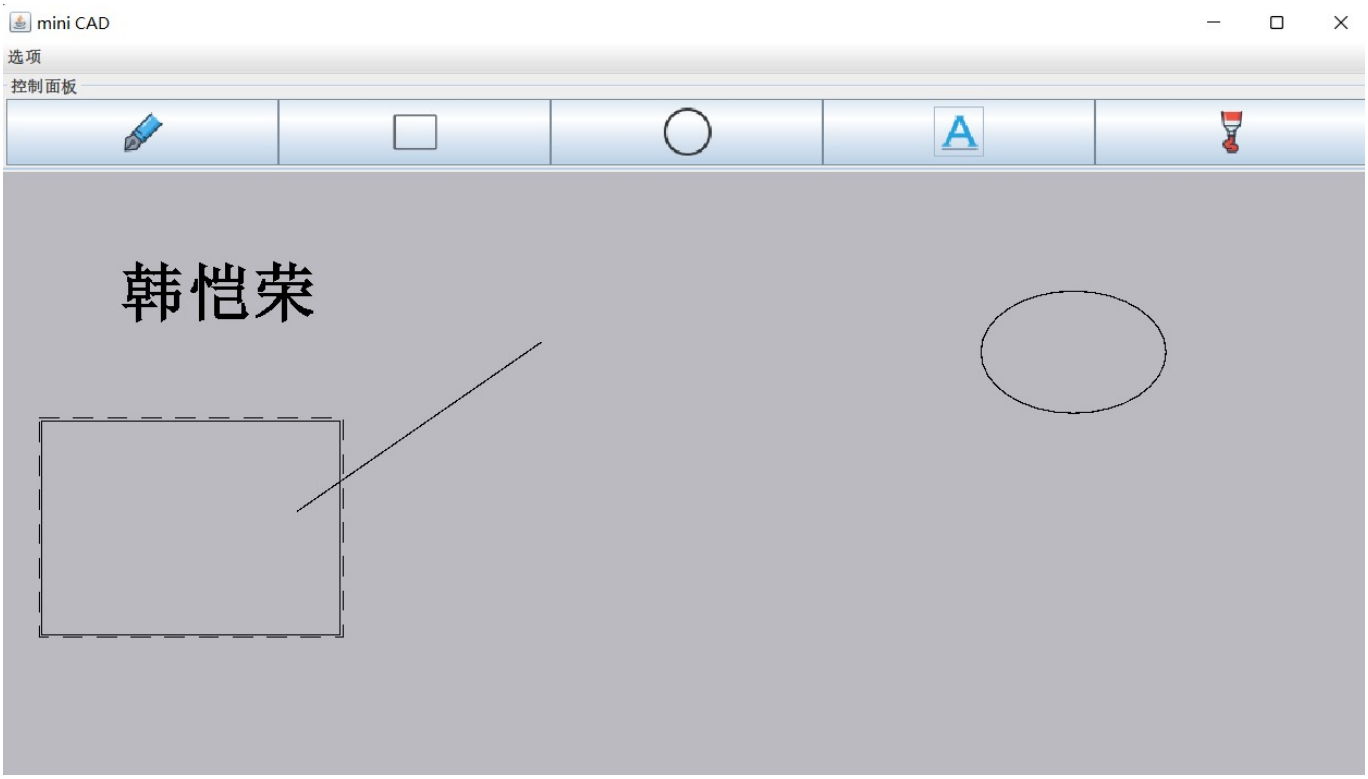


然后输入">"实现变细,注意中英文的区别:



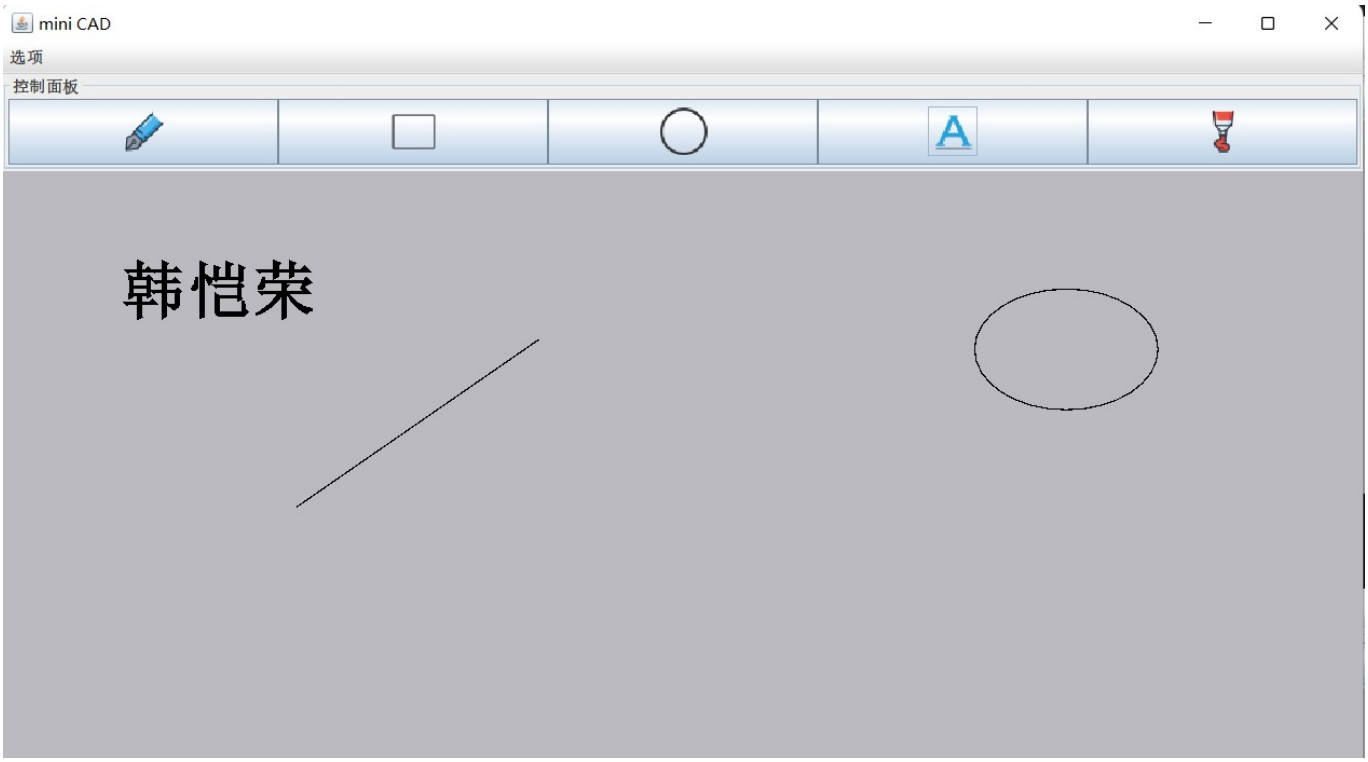
1. 改变位置

当我们选中某个图像后，默认的再次点击画布并拖拽就算改变位置操作：



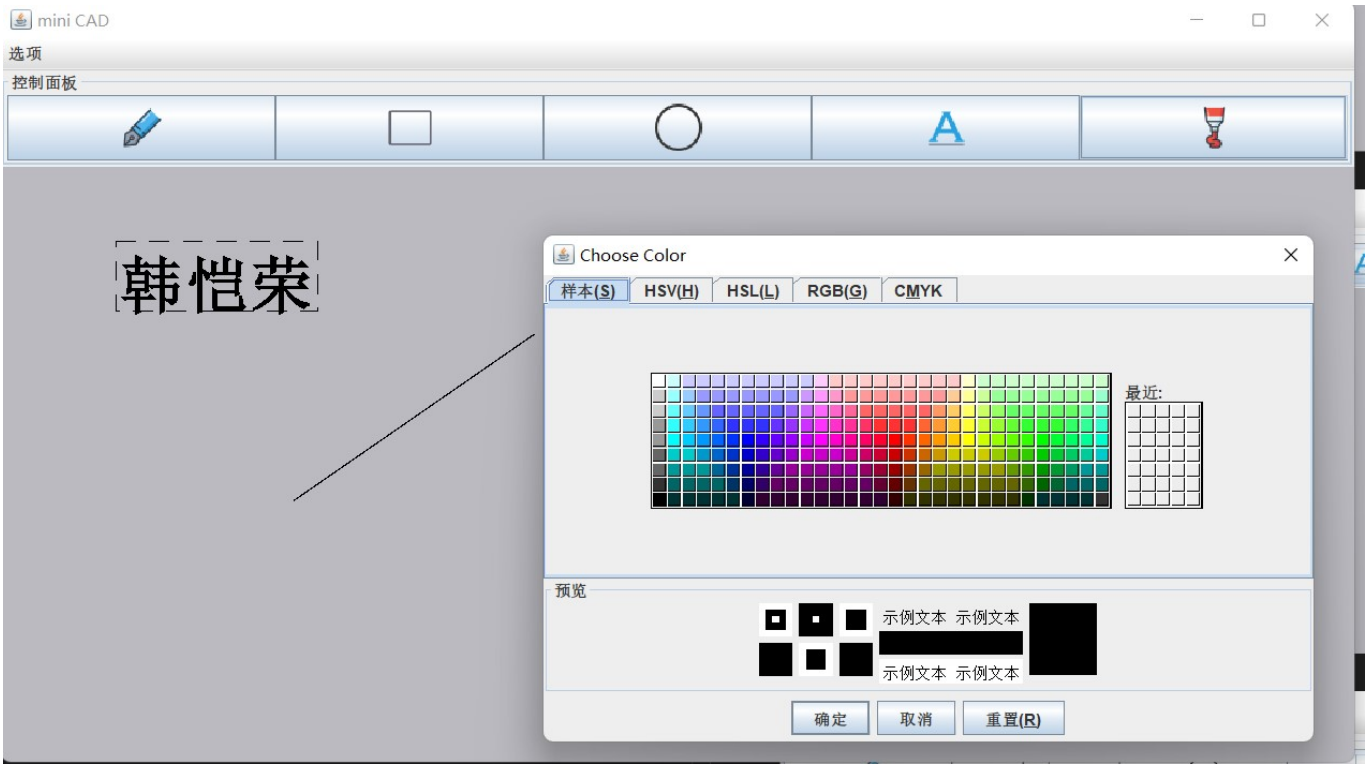
5. 删除一个图像：

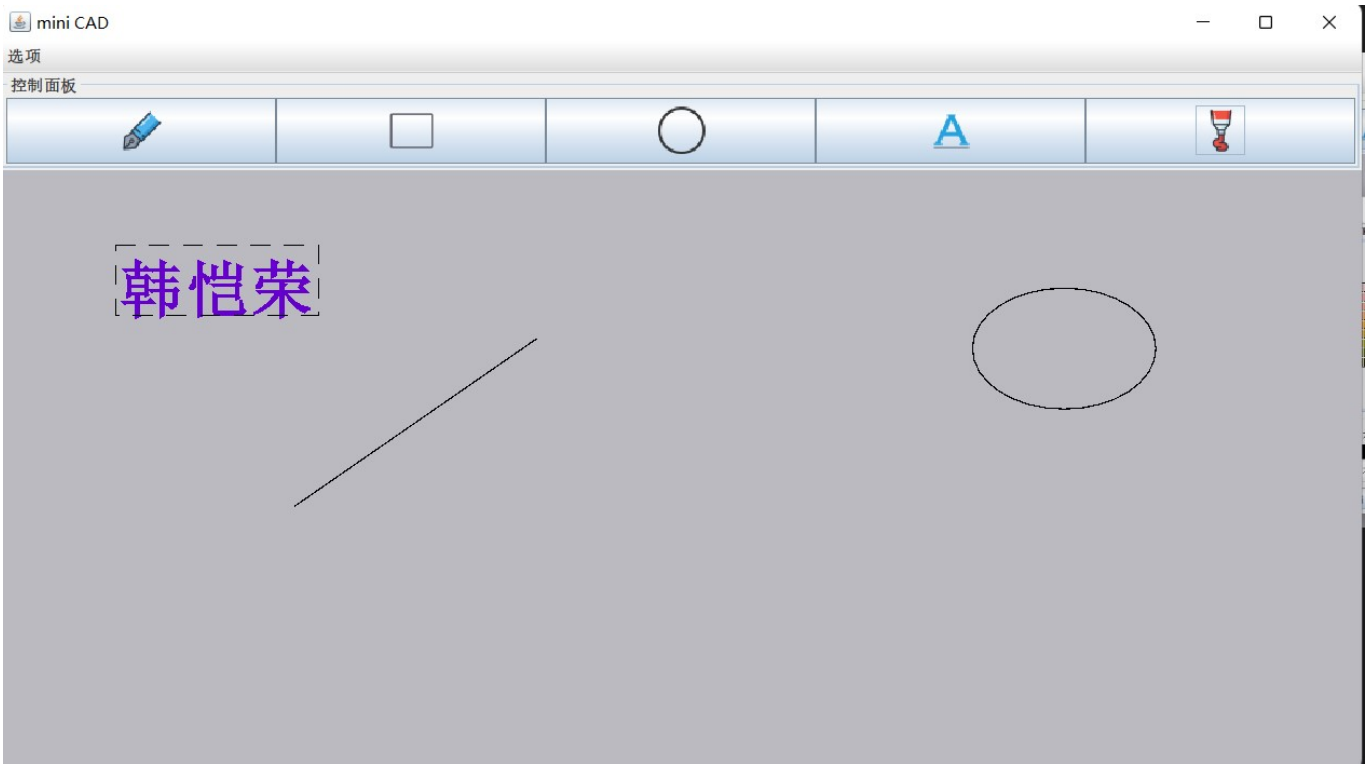
选中图像后，键盘输入delete键即可：



6. 改变颜色

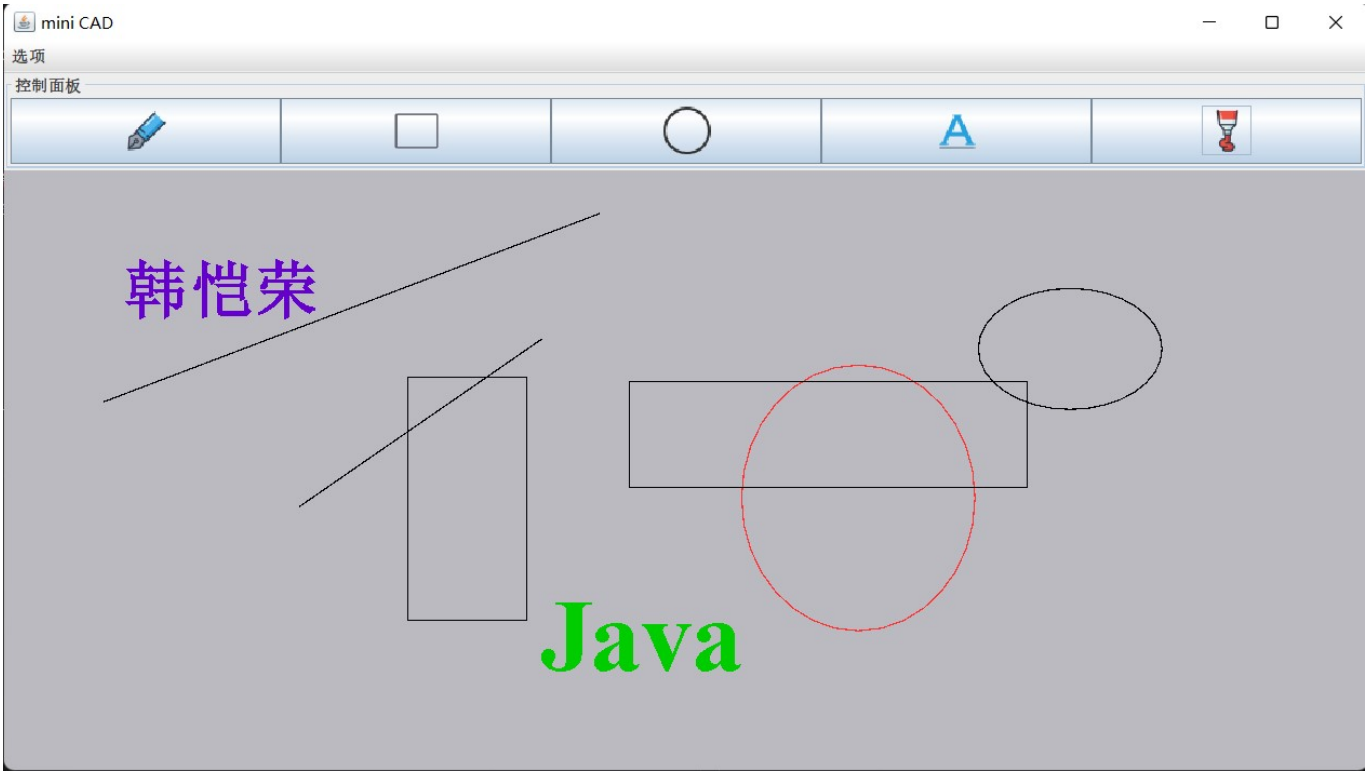
选中图像，然后点击控制面板最后一个图标，选择颜色即可修改图像颜色，当然如果在为选中任何图像时修改颜色，默认是对全局的修改，这时，当操作者再次画任意图像时都会显示刚才的图像，直到操作者再次改变颜色。



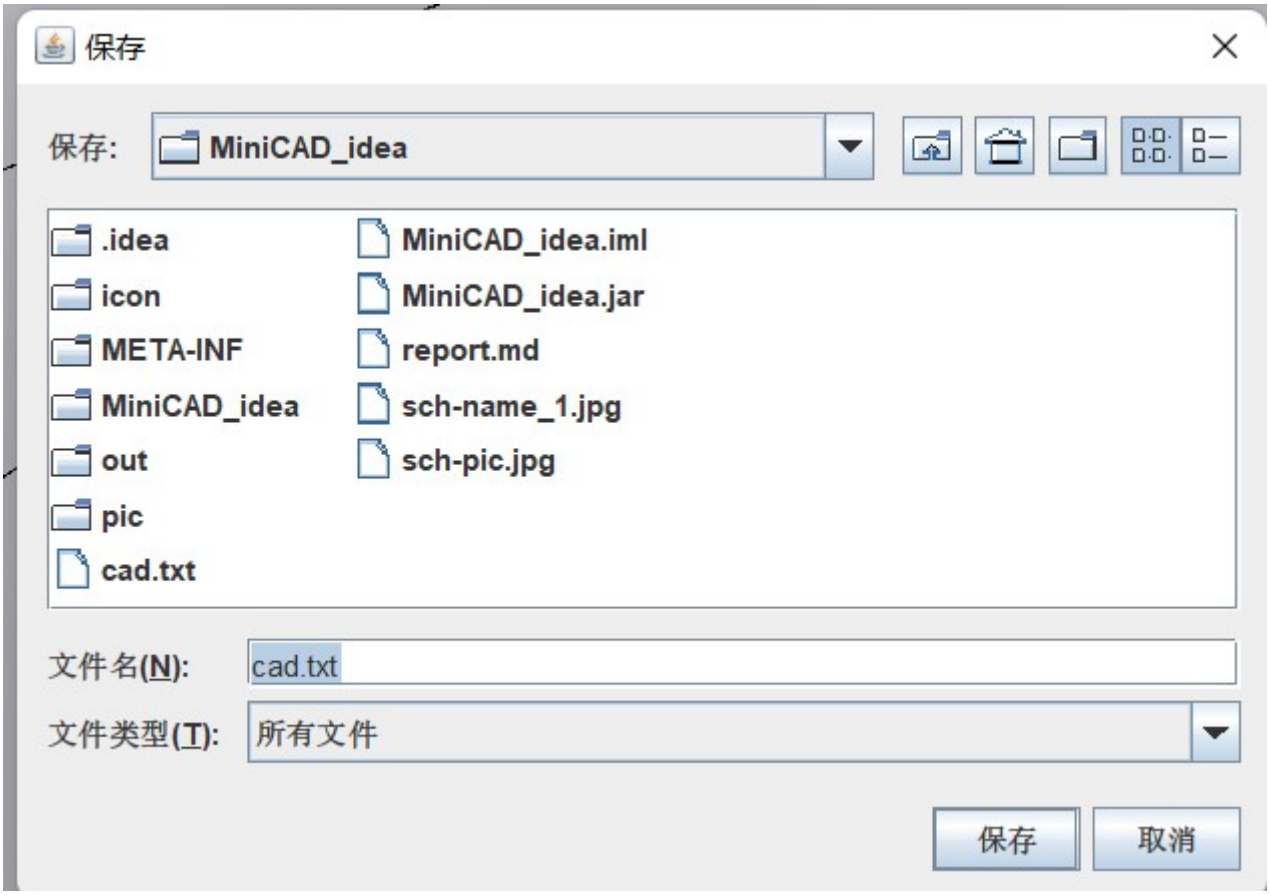


7. 保存文件

文件默认是会保存为txt文件，我们首先再多次创建一些图像：



点击菜单栏选中保存即可：

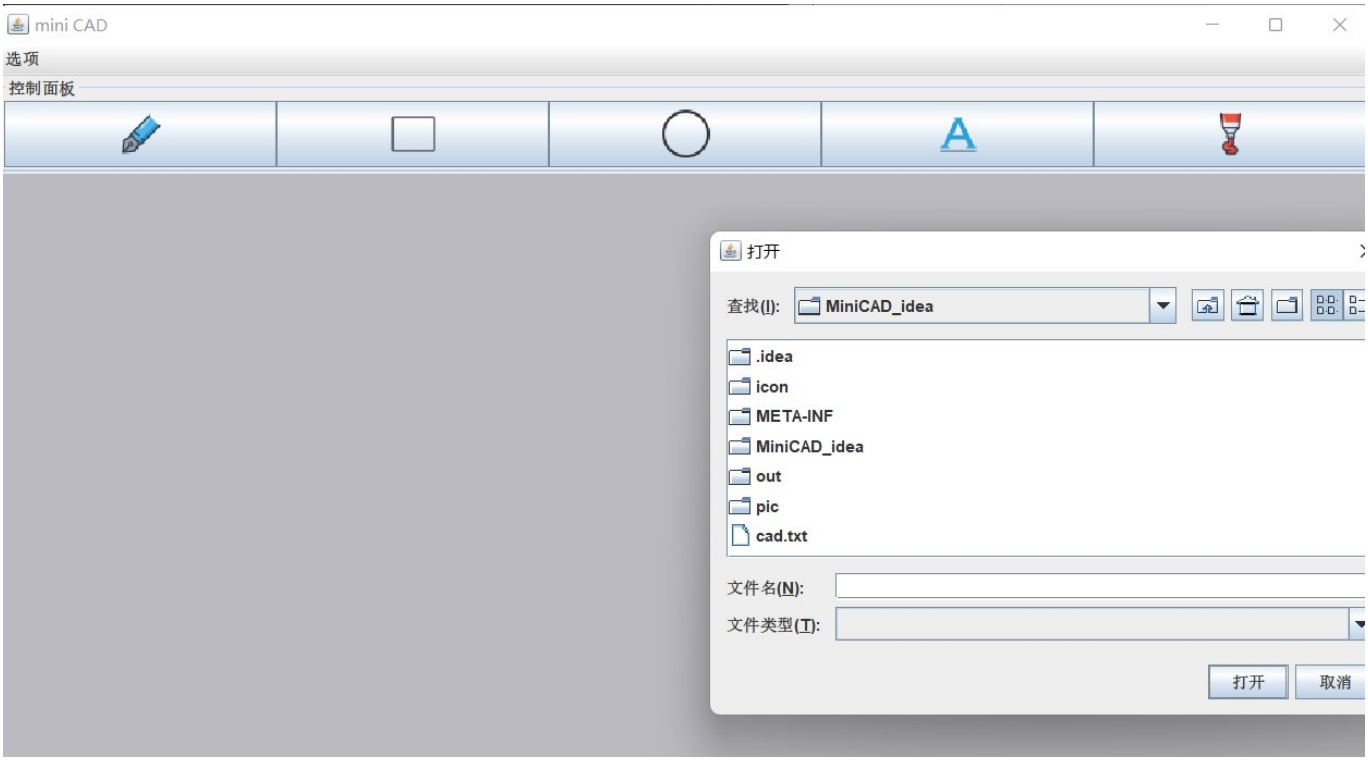


txt的内容如下：

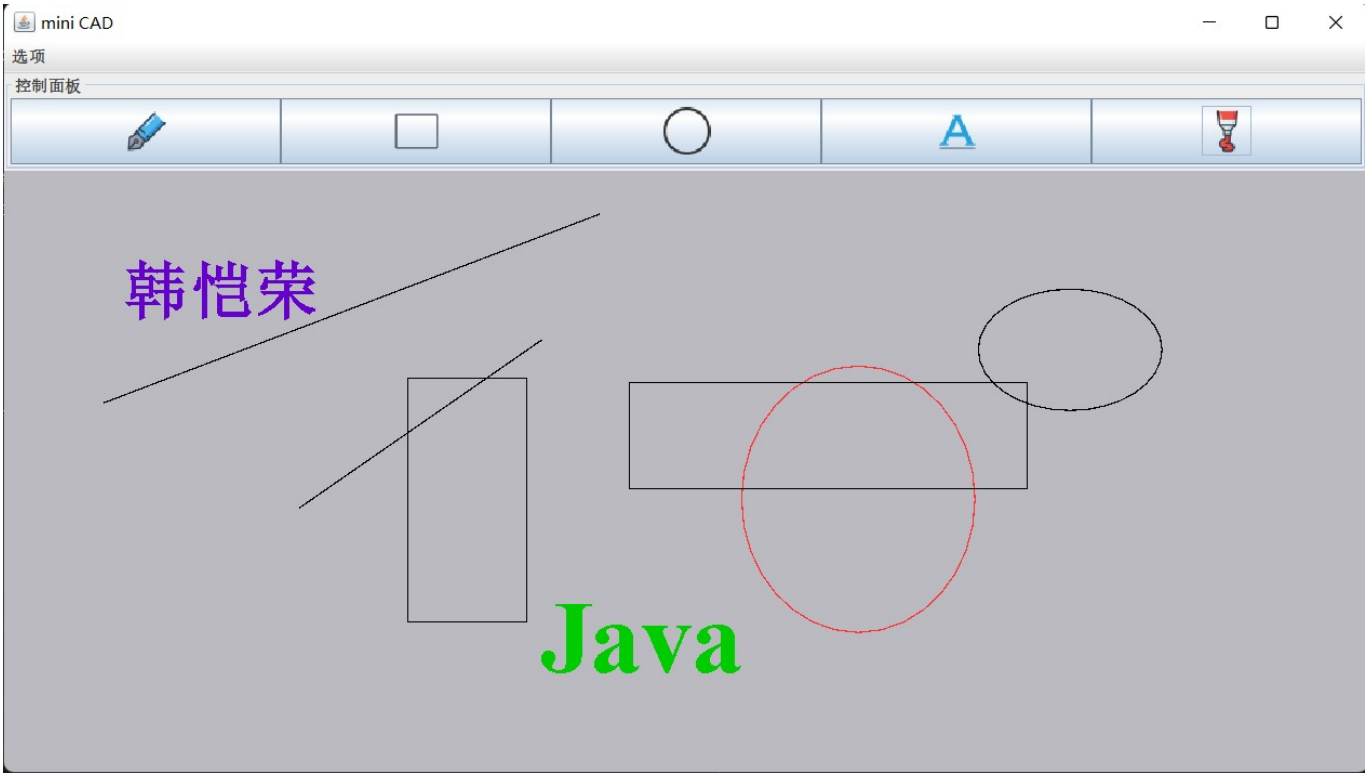
```
0,0,0,0,255,1.0,219.0,249.0,398.0,125.0
2,0,0,0,255,1.0,721.0,87.0,857.0,177.0
3,102,0,204,255,1.0,韩恺荣,88.0,57.0,234.0,105.0
2,255,51,51,255,1.0,546.0,144.0,719.0,341.0
1,0,0,0,255,1.0,463.0,156.0,757.0,234.0
1,0,0,0,255,1.0,299.0,153.0,387.0,333.0
0,0,0,0,255,1.0,74.0,171.0,440.0,32.0
3,0,204,0,255,1.0,Java,397.0,294.0,547.0,370.0
```

8. 打开文件

在7的基础上，我们首先删除所有的图像，然后选择菜单栏的打开，选中我们刚才的txt文件即可



然后我们就可以看到刚才的画面了：



当然，对于非法的无法解析的txt，我们会在内部抛出异常，并不会影响到操作者后续的操作。

五、心得体会

本次实验，我熟悉的Swing和Awt，对Java构建GUI有了较为深刻的认识和理解，并对面向对象和SVC的设计模式有了全面的认识。此外，通过这次作业，锻炼了我使用java编写代码量较多的工程的能力，并让我学会使用idea和命令行编译，让我对package有了实践和锻炼，总得来看，收获颇丰。