

**信息科学与工程学院**

**2022**－**2023** 学年第一学期

实 验 报 告

课程名称： Java 编程技术

实验名称： 第五次实验内容

专 业 班 级 通信三班

学 生 学 号 202000120202

学 生 姓 名 李鑫

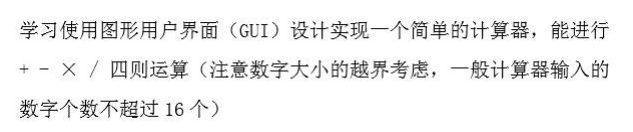
实 验 时 间 2022 年 11 月 18 日

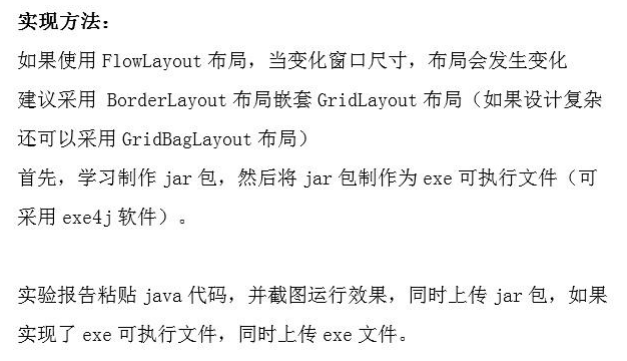
实验报告

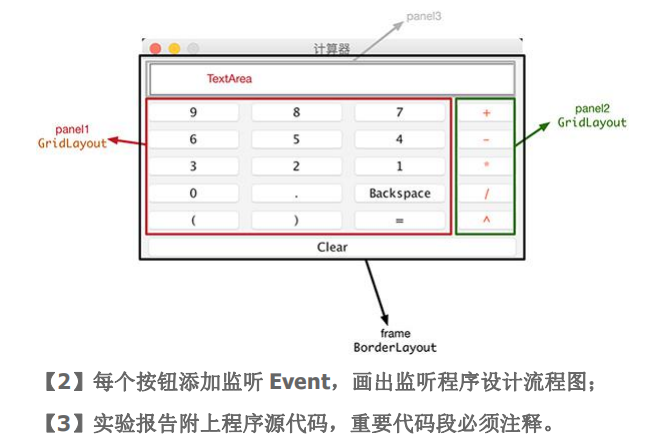
【实验目的】

1. 熟悉、了解GUI控件的使用
2. 利用GUI实现界面设计及功能实现

【实验要求】



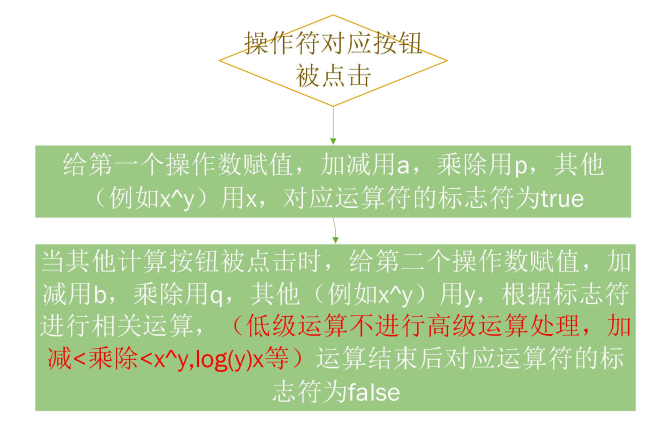
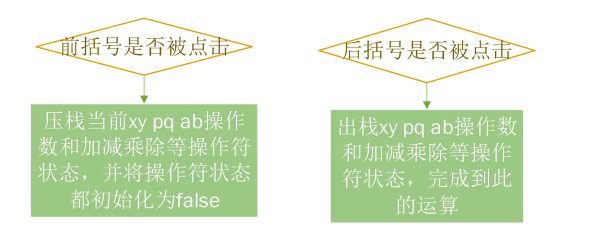
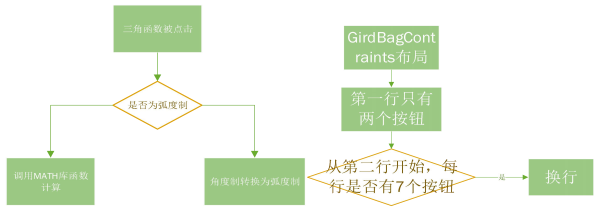




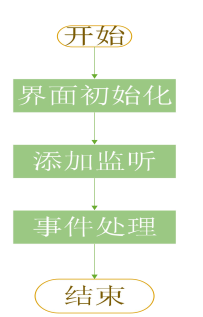
【实验具体内容】

1. 实验流程图

A.科学计算器



1. 标准计算器



1. 实验源代码

自己实现的科学计算器：

|  |
| --- |
| /\*  \* @author:李鑫  \* @学号：202000120202  \* time：2022.11.18  \* \*/  import java.awt.\*;  import javax.swing.JTextArea;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.Color;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JMenu;  import javax.swing.JMenuBar;  import javax.swing.JMenuItem;  import javax.swing.JPanel;  import javax.swing.JTextField;  import java.util.Stack;  public class ScienceCalculator extends JFrame implements ActionListener {  //创建面板  private JPanel p1 = new JPanel();  private JPanel p2 = new JPanel();  private JPanel p3 = new JPanel();  //常规功能键  private String name[] = {"RAD", "Exp", "(", ")", "|X|", "MC", "MR", "M+", "M-", "x² ", "x^y", "x^3", "+", "-", "×", " ÷ ", " √x", "y√x", "3√x", "7", "8", "9", "<-", "x!", "1/x", "2^x", "4", "5", "6", "CE", "e", " π ", "e^x", "1", "2", "3", "C", "ln", "log", "log(y)x", "±", "0", ".", "="};  //按钮  private JButton keys[] = new JButton[name.length];  //显示文本框  private JTextField resultText = new JTextField("0");  private JTextArea resultArea = new JTextArea(4, 5);  //运算参数,y 用于两个参数计算  double x, y, z, memory, rusult;  //用于计算+ -  double a, b;  //用于计算\* ÷  double p, q;  //判断运算符情况  int n;  //括号数  int front = 0;  int back = 0;  // 标志用户按的是否是整个表达式的第一个数字,或者是运算符后的第一个数字  private boolean firstDigit = true;  //三角函数默认弧度制  private boolean radians\_Deg = true;  //x^Y  private boolean operate1 = false;  //y√x  private boolean operate2 = false;  //log(y)x  private boolean operate3 = false;  //+  private boolean operate4 = false;  //-  private boolean operate5 = false;  //\*  private boolean operate6 = false;  //÷  private boolean operate7 = false;  //科学计算法  private boolean exp = false;  //存储数字的栈  Stack<Double> nums = new Stack<>();  //存储运算符的栈  Stack<Boolean> operates = new Stack<>();  //构造函数  public ScienceCalculator() {  //初始化计算器  super("李鑫的计算器");  //创建内容面板对象  Container c = getContentPane();  //设置窗口大小  this.setSize(400, 350);  //显示到中间  this.setLocationRelativeTo(null);  //设置窗口关闭动作  this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  JMenuBar menuBar = new JMenuBar();  setJMenuBar(menuBar);  JMenu menu1 = new JMenu("三角函数");  JMenu menu2 = new JMenu("反三角函数");  menuBar.add(menu1);  menuBar.add(menu2);  JMenuItem menuItem\_3 = new JMenuItem("sin");  menu1.add(menuItem\_3);  JMenuItem menuItem\_4 = new JMenuItem("cos");  menu1.add(menuItem\_4);  JMenuItem menuItem\_5 = new JMenuItem("tan");  menu1.add(menuItem\_5);  JMenuItem menuItem\_6 = new JMenuItem("cot");  menu1.add(menuItem\_6);  JMenuItem menuItem\_7 = new JMenuItem("sec");  menu1.add(menuItem\_7);  JMenuItem menuItem\_8 = new JMenuItem("csc");  menu2.add(menuItem\_8);  JMenuItem menuItem\_9 = new JMenuItem("sin^(-1)");  menu2.add(menuItem\_9);  JMenuItem menuItem\_10 = new JMenuItem("cos^(-1)");  menu2.add(menuItem\_10);  JMenuItem menuItem\_11 = new JMenuItem("tan^(-1)");  menu2.add(menuItem\_11);  //sin  menuItem\_3.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent arg0) {  x = Double.parseDouble(resultText.getText().trim());  resultArea.append("sin");  firstDigit = true;  //判断是角度制还是弧度制  if (radians\_Deg) {  resultText.setText("" + (Math.sin(x)));  } else {  resultText.setText("" + (Math.sin(Math.toRadians(x))));  }  }  }  );  //cos  menuItem\_4.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent arg0) {  x = Double.parseDouble(resultText.getText().trim());  resultArea.append("cos");  firstDigit = true;  //判断是角度制还是弧度制  if (radians\_Deg) {  resultText.setText("" + (Math.cos(x)));  } else {  resultText.setText("" + (Math.cos(Math.toRadians(x))));  }  }  }  );  //tan  menuItem\_5.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent arg0) {  x = Double.parseDouble(resultText.getText().trim());  resultArea.append("tan");  firstDigit = true;  //判断是角度制还是弧度制  if (radians\_Deg) {  //tan 函数分母不能为零  if (Math.cos(x) == 0) {  resultText.setText("无效输入");  } else {  resultText.setText("" + (Math.tan(x)));  }  } else {  if (Math.cos(Math.toRadians(x)) == 0) {  resultText.setText("无效输入");  } else {  resultText.setText("" + (Math.tan(Math.toRadians(x))));  }  }  }  });  //cot  menuItem\_6.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent arg0) {  x = Double.parseDouble(resultText.getText().trim());  resultArea.append("cot");  firstDigit = true;  //判断是角度制还是弧度制  if (radians\_Deg) {  //cot 函数分母不能为 0  if (Math.sin(x) == 0) {  resultText.setText("无效输入");  } else {  resultText.setText("" + (Math.cos(x) / Math.sin(x)));  }  } else {  if (Math.sin(Math.toRadians(x)) == 0) {  resultText.setText("无效输入");  } else {  resultText.setText("" + (Math.cos(Math.toRadians(x)) / Math.sin(Math.toRadians(x))));  }  }  }  }  );  //sec  menuItem\_7.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent arg0) {  x = Double.parseDouble(resultText.getText().trim());  resultArea.append("sec");  firstDigit = true;  //判断是角度制还是弧度制  if (radians\_Deg) {  if (Math.cos(x) == 0) {  resultText.setText("无效输入");  } else {  resultText.setText("" + (1 / Math.cos(x)));  }  } else {  if (Math.cos(Math.toRadians(x)) == 0) {  resultText.setText("无效输入");  } else {  resultText.setText("" + (1 / Math.cos(Math.toRadians(x))));  }  }  }  }  );  //csc  menuItem\_8.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent arg0) {  x = Double.parseDouble(resultText.getText().trim());  resultArea.append("csc");  firstDigit = true;//判断是角度制还是弧度制  if (radians\_Deg) {  if (Math.sin(x) == 0) {  resultText.setText("无效输入");  } else {  resultText.setText("" + (1 / Math.sin(x)));  }  } else {  if (Math.sin(Math.toRadians(x)) == 0) {  resultText.setText("无效输入");  } else {  resultText.setText("" + (1 / Math.sin(Math.toRadians(x))));  }  }  }  }  );  //sin^(-1)  menuItem\_9.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent arg0) {  x = Double.parseDouble(resultText.getText().trim());  resultArea.append("sin^(-1)");  firstDigit = true;  if (x <= 1) {  //判断是角度制还是弧度制  if (radians\_Deg) {  resultText.setText("" + (Math.asin(x)));  } else {  //转换为角度制输出  resultText.setText("" + (Math.toDegrees(Math.asin(x))));  }  } else {  resultText.setText("无效输入");  }  }  }  );  //cos^(-1)  menuItem\_10.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent arg0) {  x = Double.parseDouble(resultText.getText().trim());  resultArea.append("cos^(-1)");  firstDigit = true;  if (x <= 1) {  //判断是角度制还是弧度制  if (radians\_Deg) {  resultText.setText("" + (Math.acos(x)));  } else {  //转换为角度制输出  resultText.setText("" + (Math.toDegrees(Math.acos(x))));  }  } else {  resultText.setText("无效输入");  }  }  }  );  //tan^(-1)  menuItem\_11.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent arg0) {  x = Double.parseDouble(resultText.getText().trim());  resultArea.append("tan^(-1)");  firstDigit = true;  if (x <= 1) {  //判断是角度制还是弧度制  if (radians\_Deg) {  resultText.setText("" + (Math.atan(x)));  } else {  //转换为角度制输出  resultText.setText("" + (Math.toDegrees(Math.atan(x))));  }  } else {  resultText.setText("无效输入");  }  }  }  );  //设置为可见  this.setVisible(true);  //禁止调整框架大小  this.setResizable(false);  //初始化显示器  // 文本框中的内容采用右对齐方式  resultText.setHorizontalAlignment(JTextField.RIGHT);  // 不允许修改结果文本框  resultText.setEditable(false);  resultArea.setEditable(false);  //自动换行  resultArea.setLineWrap(true);  //边界布局管理器  p1.setLayout(new BorderLayout());  p3.setLayout(new BorderLayout());  //将文本框加入面板并位于中央  p1.add("Center", resultText);  p3.add("Center", resultArea);  GridBagLayout gbl = new GridBagLayout();  p2.setLayout(gbl);  //初始化其它按钮  JPanel panel = new JPanel();  for (int i = 0; i < name.length; i++) {  keys[i] = new JButton(name[i]);  //监听  keys[i].addActionListener(this);  keys[i].setForeground(Color.white);  keys[i].setBackground(Color.blue);  }  p2.add(keys[0]);  p2.add(keys[1]);  p2.add(panel);  for (int i = 2; i < name.length; i++) {  p2.add(keys[i]);  }  //定义一个 GridBagConstraints，是用来控制添加进的组件的显示位置  GridBagConstraints gbs = new GridBagConstraints();  //使组件完全填满其显示区域  gbs.fill = GridBagConstraints.BOTH;  //该方法是设置组件水平所占用的格子数，如果为 0，就说明该组件是该行的最后一个  gbs.gridwidth = 2;  //该方法设置组件水平的拉伸幅度，如果为 0 就说明不拉伸，不为 0 就随着窗口增大进行拉伸，0 到 1 之间  gbs.weightx = 0;  //该方法设置组件垂直的拉伸幅度，如果为 0 就说明不拉伸，不为 0 就随着窗口增大进行拉伸，0 到 1 之间  gbs.weighty = 0;  //设置组件  gbl.setConstraints(keys[0], gbs);  gbs.gridwidth = 2;  gbs.weightx = 0;  gbs.weighty = 0;  gbl.setConstraints(keys[1], gbs);  //用 Panel 填充本行剩余部分，然后换行  gbs.gridwidth = 0;  gbs.weightx = 0;  gbs.weighty = 0;  gbl.setConstraints(panel, gbs);  //放置其他按钮  for (int j = 2; j < name.length; j++) {  //每七个按钮换一次行  if (((j - 1) % 7) == 0) {  gbs.gridwidth = 0;  } else {  gbs.gridwidth = 1;  }  gbl.setConstraints(keys[j], gbs);  }  // 整体布局  //整体布局采用边界布局管理器，组件水平间隔 3 个像素，上下间隔 5 个像素  c.setLayout(new BorderLayout(3, 5));  //从上到下一次是显示框、M 键、其他键  c.add("Center", p1);  c.add("North", p3);  c.add("South", p2);  }  public void actionPerformed(ActionEvent e) {  //用于存储运算数学式  StringBuffer[] stringBuffers = new StringBuffer[10];  //用于数字输入  String num = e.getActionCommand();  //角度变换  if (e.getSource() == keys[0]) {  radians\_Deg = !radians\_Deg;  if (radians\_Deg) {  keys[0].setText("RAD");  } else {  keys[0].setText("DEG");  }  n = 0;  } else if (e.getSource() == keys[1]) {  //科学计数法输入  resultArea.append("E");  x = Double.parseDouble(resultText.getText().trim());  resultText.setText(" " + x + ".E");  firstDigit = true;  exp = true;  n = 1;  } else if (e.getSource() == keys[2]) {  // (  resultArea.append("(");  handleExp(x);  front++;  //当遇到前括号时对前面的运算结果和过程进行压栈  nums.push(x);  nums.push(y);  nums.push(a);  nums.push(b);  nums.push(p);  nums.push(q);  operates.push(operate1);  operates.push(operate2);  operates.push(operate3);  operates.push(operate4);  operates.push(operate5);  operates.push(operate6);  operates.push(operate7);  //暂停之前其他的运算  operate1 = false;  operate2 = false;  operate3 = false;  operate4 = false;  operate5 = false;  operate6 = false;  operate7 = false;  firstDigit = true;  resultText.setText("(");  n = 2;  firstDigit = true;  } else if (e.getSource() == keys[3]) {  // )  //将括号内计算完成  resultArea.append(")");  handleOperate();  handleOperate2();  handleOperate3();  back++;  //弹出原来操作数和运算符，恢复原状态  q = nums.pop();  p = nums.pop();  b = nums.pop();  a = nums.pop();  y = nums.pop();  x = nums.pop();  operate7 = operates.pop();  operate6 = operates.pop();  operate5 = operates.pop();  operate4 = operates.pop();  operate3 = operates.pop();  operate2 = operates.pop();  operate1 = operates.pop();  //完成压栈之前的运算  handleOperate();  handleOperate2();  handleOperate3();  n = 3;  firstDigit = true;  } else if (e.getSource() == keys[4]) {  //|x|  resultArea.append("Absolute");  handleExp(x);  x = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  resultText.setText("" + (Math.abs(x)));  handleOperate();  firstDigit = true;  n = 4;  } else if (e.getSource() == keys[5]) {  //MC  memory = 0;  n = 5;  } else if (e.getSource() == keys[6]) {  //MR  resultText.setText("" + (memory));  n = 6;  } else if (e.getSource() == keys[7]) {  //M+  memory += Double.parseDouble(resultText.getText().trim());  n = 7;  } else if (e.getSource() == keys[8]) {  //M- memory-=Double.parseDouble(resultText.getText().trim());  n = 8;  } else if (e.getSource() == keys[9]) {  //x^2  resultArea.append("^2");  handleExp(x);  x = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  resultText.setText("" + (x \* x));  n = 9;  handleOperate();  firstDigit = true;  } else if (e.getSource() == keys[10]) {  //y^x  resultArea.append("^");  handleExp(x);  y = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  n = 10;  resultText.setText(" ");  operate1 = true;  handleOperate();  firstDigit = true;  } else if (e.getSource() == keys[11]) {  //x^3  resultArea.append("^3");  handleExp(x);  n = 11;  x = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  resultText.setText("" + (x \* x \* x));  handleOperate();  firstDigit = true;  } else if (e.getSource() == keys[12]) {  //+  resultArea.append("+");  handleExp(x);  //运算顺序遵循运算优先级，两位操作数>乘除>加减  handleOperate();  handleOperate2();  handleOperate3();  a = Double.parseDouble(resultText.getText().trim());  operate4 = true;  n = 12;  firstDigit = true;  } else if (e.getSource() == keys[13]) {  //-  resultArea.append("-");  handleExp(x);  handleOperate();  handleOperate2();  handleOperate3();  a = Double.parseDouble(resultText.getText().trim());  operate5 = true;  n = 13;  firstDigit = true;  } else if (e.getSource() == keys[14]) {  //\*  resultArea.append("\*");  handleExp(x);  //加减优先级低，不进行运算  handleOperate();  handleOperate2();  p = Double.parseDouble(resultText.getText().trim());  operate6 = true;  n = 14;  firstDigit = true;  } else if (e.getSource() == keys[15]) {  //÷  resultArea.append("÷");  handleExp(x);  //加减优先级低，不进行运算  handleOperate();  handleOperate2();  p = Double.parseDouble(resultText.getText().trim());  operate7 = true;  firstDigit = true;  n = 15;  } else if (e.getSource() == keys[16]) {  //"sqrt"  resultArea.append("sqrt");  handleExp(x);  x = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  //操作数不能为负数  if (x < 0) {  resultText.setText("无效输入");  } else {  resultText.setText("" + (Math.sqrt(x)));  }  n = 16;  handleOperate();  firstDigit = true;  } else if (e.getSource() == keys[17]) {  //y√x  resultArea.append("√");  handleExp(x);  y = Double.parseDouble(resultText.getText().trim());  resultText.setText(" ");  firstDigit = true;  //两位操作数运算用到标志符，点击时为 true，运算结束后为 false  operate2 = true;  n = 17;  handleOperate();  firstDigit = true;  } else if (e.getSource() == keys[18]) {  //3√x  resultArea.append(")3√");  handleExp(x);  x = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  resultText.setText("" + (Math.pow(x, 1.0 / 3.0)));  n = 18;  handleOperate();  } else if (e.getSource() == keys[22]) {  //退格  handleBackspace();  n = 22;  } else if (e.getSource() == keys[23]) {  //阶乘  resultArea.append("!");  handleExp(x);  x = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  //阶乘不能为负数  if (x < 0) {  resultText.setText("无效输入");  } else {  resultText.setText("" + (factorialNumber(x)));  }  n = 23;  handleOperate();  } else if (e.getSource() == keys[24]) {  //"1/x"  resultArea.append("^(-1)");  handleExp(x);  x = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  if (x == 0) {  resultText.setText("分母不能为零");  } else {  resultText.setText("" + (1 / x));  }  n = 24;  handleOperate();  } else if (e.getSource() == keys[25]) {  //2^X  resultArea.append(")2^");  handleExp(x);  x = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  resultText.setText("" + (Math.pow(2, x)));  n = 25;  handleOperate();  } else if (e.getSource() == keys[29]) {  //CE  resultText.setText("0");  firstDigit = true;  n = 29;  } else if (e.getSource() == keys[30]) {  //e  resultArea.append("e");  resultText.setText("" + (Math.E));  n = 100;  } else if (e.getSource() == keys[31]) {  //PI  resultArea.append("PI");  resultText.setText("" + (Math.PI));  n = 100;  firstDigit = true;  } else if (e.getSource() == keys[32]) {  //e^X  resultArea.append(")e^");  handleExp(x);  x = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  resultText.setText("" + (Math.pow(Math.E, x)));  n = 32;  } else if (e.getSource() == keys[36]) {  //c  //显示框全部都清零  resultText.setText("0");  resultArea.setText(" ");  //操作数、括号数都归零  a = 0;  b = 0;  p = 0;  q = 0;  front = 0;  back = 0;  //操作符标志符全部为 false  operate1 = false;  operate2 = false;  operate3 = false;  operate4 = false;  operate5 = false;  operate6 = false;  operate7 = false;  firstDigit = true;  n = 36;  } else if (e.getSource() == keys[37]) {  //ln  resultArea.append("ln");  handleExp(x);  x = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  resultText.setText("" + (Math.log(x)));  n = 37;  handleOperate();  } else if (e.getSource() == keys[38]) {  //lg  //Java 中没有直接计算以 10 为底对数的方法，需要利用 logy(x) =loge(x) /loge(y)，换底公式  resultArea.append("lg");  handleExp(x);  x = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  resultText.setText("" + (Math.log(x) / Math.log(10)));  n = 38;  handleOperate();  } else if (e.getSource() == keys[39]) {  //log(y)x  resultArea.append("log base");  handleExp(x);  y = Double.parseDouble(resultText.getText().trim());  firstDigit = true;  operate3 = true;  resultText.setText(" ");  handleOperate();  n = 39;  } else if (e.getSource() == keys[40]) {  x = Double.parseDouble(resultText.getText().trim());  resultText.setText("" + ((-1) \* x));  n = 100;  } else if (e.getSource() == keys[43]) {  //=  resultArea.append("=");  handleExp(x);  handleOperate();  handleOperate2();  handleOperate3();  //对操作数和操作符都回归为初始状态  a = 0;  b = 0;  p = 0;  q = 0;  operate4 = false;  operate5 = false;  operate6 = false;  operate7 = false;  n = 43;  //括号数归零  //front = back = 0;  } else {  handleNumber(num);  //n=100 表示此时进行是下输入框数字输入  n = 100;  }  }  //退格处理  private void handleBackspace() {  String text = resultText.getText();  int i = text.length();  if (i > 0) {  // 退格，将文本最后一个字符去掉  text = text.substring(0, i - 1);  if (text.length() == 0) {  // 如果文本没有了内容，则初始化计算器的各种值  resultText.setText("0");  firstDigit = true;  //operator = "=";  } else {  // 显示新的文本  resultText.setText(text);  }  String text2 = resultArea.getText();  int j = text2.length();  if (j > 0) {  // 退格，将文本最后一个字符去掉  text2 = text2.substring(0, i - 1);  if (text2.length() == 0) {  // 如果文本没有了内容，则初始化计算器的各种值  resultArea.setText(" ");  firstDigit = true;  } else {  // 显示新的文本  resultArea.setText(text);  }  }  }  }  //阶乘  private double factorialNumber(double num) {  double sum = 0;  if (num == 1) {  return 1;//跳出循环  } else {  sum = num \* factorialNumber(num - 1);//递归  return sum;  }  }  //数字输入处理  private void handleNumber(String num) {  // 输入的第一个数字  if (firstDigit) {  resultText.setText(num);  // resultArea.setText(num);  } else if ((num.equals(".")) && (resultText.getText().indexOf(".") < 0)) {  // 输入的是小数点，并且之前没有小数点，则将小数点附在结果文本框的后面  resultText.setText(resultText.getText() + ".");  // resultArea.append(".");  } else if (!num.equals(".")) {  // 如果输入的不是小数点，则将数字附在结果文本框的后面  resultText.setText(resultText.getText() + num);  //resultArea.append(num);  }  if ((num.equals(".")) && (resultText.getText().indexOf(".") < 0)) {  resultArea.append(".");  } else {  resultArea.append(num);  }  // 以后输入的肯定不是第一个数字了  firstDigit = false;  }  //对两位操作数运算符计算  private void handleOperate() {  if (operate1) {  handleOperate1(y);  }  //y√x  if (operate2) {  handleOperate2(y);  }  //log(y)x  if (operate3) {  handleOperate3(y);  }  }  // \* ÷  private void handleOperate2() {  //\*  if (operate6) {  handleOperate6(p);  }  //÷  if (operate7) {  handleOperate7(p);  }  }  //+ -  private void handleOperate3() {  //+  if (operate4) {  handleOperate4(a);  }  //-  if (operate5) {  handleOperate5(a);  }  }  //y^x 处理  private void handleOperate1(double y) {  x = Double.parseDouble(resultText.getText().trim());  resultText.setText(" " + (Math.pow(y, x)));  resultText.setText("x^y");  operate1 = false;  }  //y√x 处理  private void handleOperate2(double y) {  x = Double.parseDouble(resultText.getText().trim());  resultText.setText(" " + (Math.pow(y, 1 / x)));  operate2 = false;  }  //log(y)x 处理  private void handleOperate3(double y) {  x = Double.parseDouble(resultText.getText().trim());  resultText.setText("" + (Math.log(y) / Math.log(x)));  operate3 = false;  }  //+  private void handleOperate4(double a) {  b = Double.parseDouble(resultText.getText().trim());  resultText.setText("" + (a + b));  operate4 = false;  }  //-  private void handleOperate5(double a) {  b = Double.parseDouble(resultText.getText().trim());  resultText.setText("" + (a - b));  operate5 = false;  }  //\*  private void handleOperate6(double p) {  q = Double.parseDouble(resultText.getText().trim());  resultText.setText("" + (p \* q));  operate6 = false;  }  //÷  private void handleOperate7(double p) {  q = Double.parseDouble(resultText.getText().trim());  resultText.setText("" + (p / q));  operate5 = false;  }  private void handleExp(double x) {  //科学计算法表示输入  //判断是否按钮点击  if (exp) {  y = Double.parseDouble(resultText.getText().trim());  resultText.setText("" + (x \* Math.pow(10, y)));  }  exp = false;  }  public static void main(String[] args) {  ScienceCalculator scienceCalculator = new ScienceCalculator();  }  } |

网上参考的标准计算器

|  |
| --- |
| import java.util.\*;  import java.awt.\*;  import java.awt.BasicStroke;  import javax.swing.\*;  import javax.script.\*;  import java.awt.event.\*;  import java.awt.Color;  import java.awt.Font;  import java.awt.Graphics;  import java.awt.Graphics2D;  import java.awt.GridLayout;  import java.math.BigDecimal;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  /\*创建主类E\_calculator\*/  public class E\_calculator {  JButton st; // st的作用类似于寄存器  /\* 以下i\_1至i\_5为状态开关 \*/  int i\_1 = 1;  int i\_2 = 1; // 开关2用于记录符号键的次数  int i\_3 = 1; // 开关3用于标识 str1 是否可以被清 0  int i\_4 = 1; // 开关4用于标识 str2 是否可以被清 0  int i\_5 = 1; // 开关5用于控制小数点可否被录入  String str1 = "0"; // 操作数1，为了程序的安全，初值设置为0  String str2 = "0"; // 操作数2  String signal = "+"; // 运算符  String result = ""; // 运算结果  Vector vt = new Vector(20, 20);  // 创建一个 JFrame 对象并初始化  JFrame frame = new JFrame("简易计算器");  // JTextField用于显示操作和计算结果的文本框  JTextField TextField = new JTextField(result, 30);  JButton clear\_Button = new JButton("清 除"); // 清除按钮  /\* 数字键0到9 \*/  JButton button0 = new JButton("0");  JButton button1 = new JButton("1");  JButton button2 = new JButton("2");  JButton button3 = new JButton("3");  JButton button4 = new JButton("4");  JButton button5 = new JButton("5");  JButton button6 = new JButton("6");  JButton button7 = new JButton("7");  JButton button8 = new JButton("8");  JButton button9 = new JButton("9");  // 计算命令按钮，加减乘除以及小数点等  JButton button\_jia = new JButton("+");  JButton button\_jian = new JButton("—");  JButton button\_cheng = new JButton("\*");  JButton button\_chu = new JButton("/");  JButton button\_dy = new JButton("=");  JButton button\_Dian = new JButton(".");  //创建E\_calculator()方法中编写数字键的响应逻辑  public E\_calculator() {  button0.setMnemonic(KeyEvent.VK\_0);  TextField.setHorizontalAlignment(JTextField.RIGHT);  JPanel panel = new JPanel();// 创建一个 Jpanel 对象并初始化  // 设置该容器的布局为四行四列，边距为10像素  panel.setLayout(new GridLayout(4, 4, 10, 10));  panel.setSize(500, 800);  panel.setBackground(Color.blue);  panel.setForeground(Color.white);  /\* 将用于计算的按钮添加到容器内 \*/  panel.add(button7);  panel.add(button8);  panel.add(button9);  panel.add(button\_cheng);  panel.add(button4);  panel.add(button5);  panel.add(button6);  panel.add(button\_chu);  panel.add(button1);  panel.add(button2);  panel.add(button3);  panel.add(button\_jian);  panel.add(button0);  panel.add(button\_Dian);  panel.add(button\_dy);  panel.add(button\_jia);  // 设置panel对象的边距为16  panel.setBorder(BorderFactory.createEmptyBorder(16, 16, 16, 16)); JPanel panel2 = new JPanel(); // 按照同样的方式设置第二个JPanel  panel2.add(TextField, BorderLayout.WEST);  panel2.add(clear\_Button, BorderLayout.EAST);  frame.setLocation(400, 200); // 设置主窗口出现在屏幕上的位置  frame.setResizable(true); // 设置窗体可以调大小  frame.getContentPane().setLayout(new BorderLayout());  // 窗体中可以放置 JPanel，这里我们将面板panel和面板panel2让如窗体  frame.getContentPane().add(panel2, BorderLayout.NORTH);  frame.getContentPane().add(panel, BorderLayout.CENTER);  frame.pack();  frame.setVisible(true);  /\* Listener类中编写的是数字键的响应逻辑\*/  class Listener implements ActionListener {  public void actionPerformed(ActionEvent test) {  String ss = ((JButton) test.getSource()).getText();  st = (JButton) test.getSource();  vt.add(st);  if (i\_1 == 1) {  if (i\_3 == 1) {  str1 = "";  i\_5 = 1;  }  str1 = str1 + ss;  i\_3 = i\_3 + 1;  TextField.setText(str1);  } else if (i\_1 == 2) {  if (i\_4 == 1) {  str2 = "";  i\_5 = 1;  }  str2 = str2 + ss;  i\_4 = i\_4 + 1;  TextField.setText(str2);  }  }  }  /\* Listener\_signal类中编写了运算符号键的响应逻辑 \*/  class Listener\_signal implements ActionListener {  public void actionPerformed(ActionEvent e) {  String ss2 = ((JButton) e.getSource()).getText();  st = (JButton) e.getSource();  vt.add(st);  if (i\_2 == 1) {  i\_1 = 2;  i\_5 = 1;  signal = ss2;  i\_2 = i\_2 + 1;  } else {  int a = vt.size();  JButton c = (JButton) vt.get(a - 2);  if (!(c.getText().equals("+")) && !(c.getText().equals("—")) && !(c.getText().equals("\*"))&& !(c.getText().equals("/"))) {  cal();  str1 = result;  i\_1 = 2;  i\_5 = 1;  i\_4 = 1;  signal = ss2;  }  i\_2 = i\_2 + 1;  }  }  }  /\*Listener\_clear类中编写了清除键的响应逻辑\*/  class Listener\_clear implements ActionListener {  public void actionPerformed(ActionEvent test) {  st = (JButton) test.getSource();  vt.add(st);  i\_5 = 1;  i\_2 = 1;  i\_1 = 1;  i\_3 = 1;  i\_4 = 1;  str1 = "0";  str2 = "0";  signal = "";  result = "";  TextField.setText(result);  vt.clear();  }  }  /\*Listener\_dy类中编写的是等于号键的响应逻辑\*/  class Listener\_dy implements ActionListener {  public void actionPerformed(ActionEvent e) {  st = (JButton) e.getSource();  vt.add(st);  cal();  i\_1 = 1;  i\_2 = 1;  i\_3 = 1;  i\_4 = 1;  str1 = result;  }  }  /\*Listener\_xiaos类中编写的是小数点键的相应逻辑\*/  class Listener\_xiaos implements ActionListener {  public void actionPerformed(ActionEvent e) {  st = (JButton) e.getSource();  vt.add(st);  if (i\_5 == 1) {  String ss2 = ((JButton) e.getSource()).getText();  if (i\_1 == 1) {  if (i\_3 == 1) {  str1 = "";  i\_5 = 1;  }  str1 = str1 + ss2;  i\_3 = i\_3 + 1;  TextField.setText(str1);  } else if (i\_1 == 2) {  if (i\_4 == 1) {  str2 = "";  i\_5 = 1;  }  str2 = str2 + ss2;  i\_4 = i\_4 + 1;  TextField.setText(str2);  }  }  i\_5 = i\_5 + 1;  }  }  // 监听等于键  Listener\_dy jt\_dy = new Listener\_dy();  // 监听数字键  Listener jt = new Listener();  // 监听符号键  Listener\_signal jt\_signal = new Listener\_signal();  // 监听清除键  Listener\_clear jt\_c = new Listener\_clear();  // 监听小数点键  Listener\_xiaos jt\_xs = new Listener\_xiaos();  button7.addActionListener(jt);  button8.addActionListener(jt);  button9.addActionListener(jt);  button\_chu.addActionListener(jt\_signal);  button4.addActionListener(jt);  button5.addActionListener(jt);  button6.addActionListener(jt);  button\_cheng.addActionListener(jt\_signal);  button1.addActionListener(jt);  button2.addActionListener(jt);  button3.addActionListener(jt);  button\_jian.addActionListener(jt\_signal);  button0.addActionListener(jt);  button\_Dian.addActionListener(jt\_xs);  button\_dy.addActionListener(jt\_dy);  button\_jia.addActionListener(jt\_signal);  clear\_Button.addActionListener(jt\_c);  // 窗体关闭事件的响应程序  frame.addWindowListener(new WindowAdapter() {  public void windowClosing(WindowEvent e) {  System.exit(0);  }  });  }  // cal()方法中编写了计算逻辑的实现。  public void cal() {  double a2,b2; //定义两个变量a2,b2  String c = signal;  double result2 = 0; //初始化  if (c.equals("")) {  TextField.setText("Please input operator");  } else {  if (str1.equals("."))  str1 = "0.0";  if (str2.equals("."))  str2 = "0.0";  a2 = Double.valueOf(str1).doubleValue();  b2 = Double.valueOf(str2).doubleValue();  if (c.equals("+")) {  result2 = a2 + b2;  }  if (c.equals("—")) {  result2 = a2 - b2;  }  if (c.equals("\*")) {  BigDecimal m1 = new BigDecimal(Double.toString(a2));  BigDecimal m2 = new BigDecimal(Double.toString(b2));  //乘法运算,只需导入组件（import java.math.BigDecimal）  result2 = m1.multiply(m2).doubleValue();  }  if (c.equals("/")) {  if (b2 == 0) {  result2 = 0;  } else {  result2 = a2 / b2;  }  }  result = ((new Double(result2)).toString());  TextField.setText(result);  }  }  public static void main(String[] args) {  E\_calculator CT = new E\_calculator();  }  } |

1. 实验运行截图：
2. 科学计算器

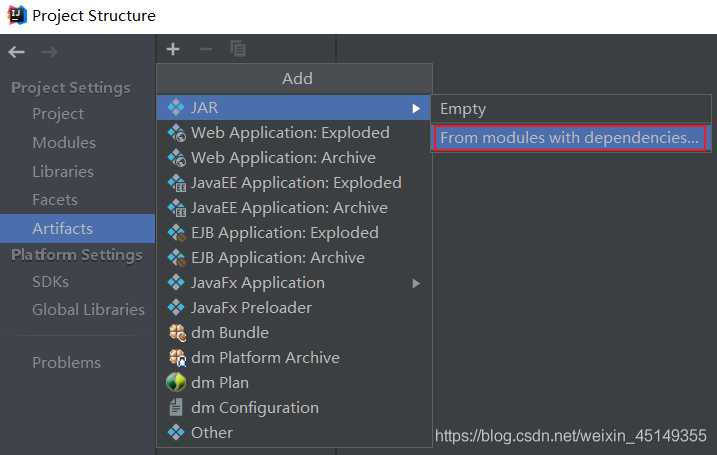
|  |
| --- |
|  |

1. 标准计算器

|  |
| --- |
| 4+5的实现        输入1/0报错 |

【打包为exe文件】

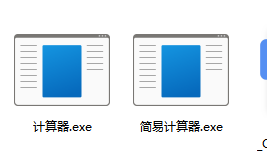
在几周的学习中，我一直在选择使用通过IDEA编写代码，而不是老师推荐的Eclipse，所以我在打包为EXE文件的时候通过上网查询，发现需要右键项目–>Open Module Settings 进入Project Structure，然后按下图创建一个module

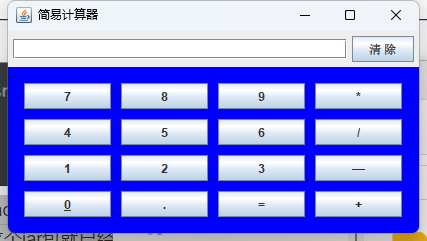


Main Class就选择你启动项目的main函数所在的类，下面的路径一般都是放在artifacts路径下，设置好了之后点击ok

点击idea工具栏上的Build–>Build Artifacts… ，之后就会弹出一个小东西，点击build就会在前面设置的路径下生成我们的jar包

之后通过使用老师教程中的exe4j文件，最终打包为了两个计算器，分别为简易计算器.exe和计算器.exe

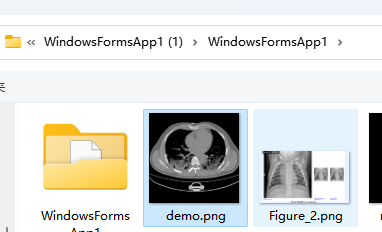
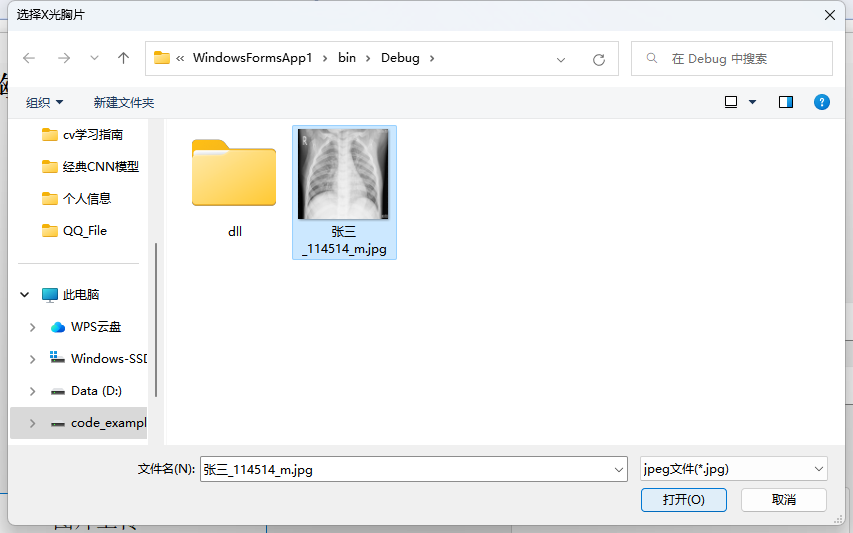


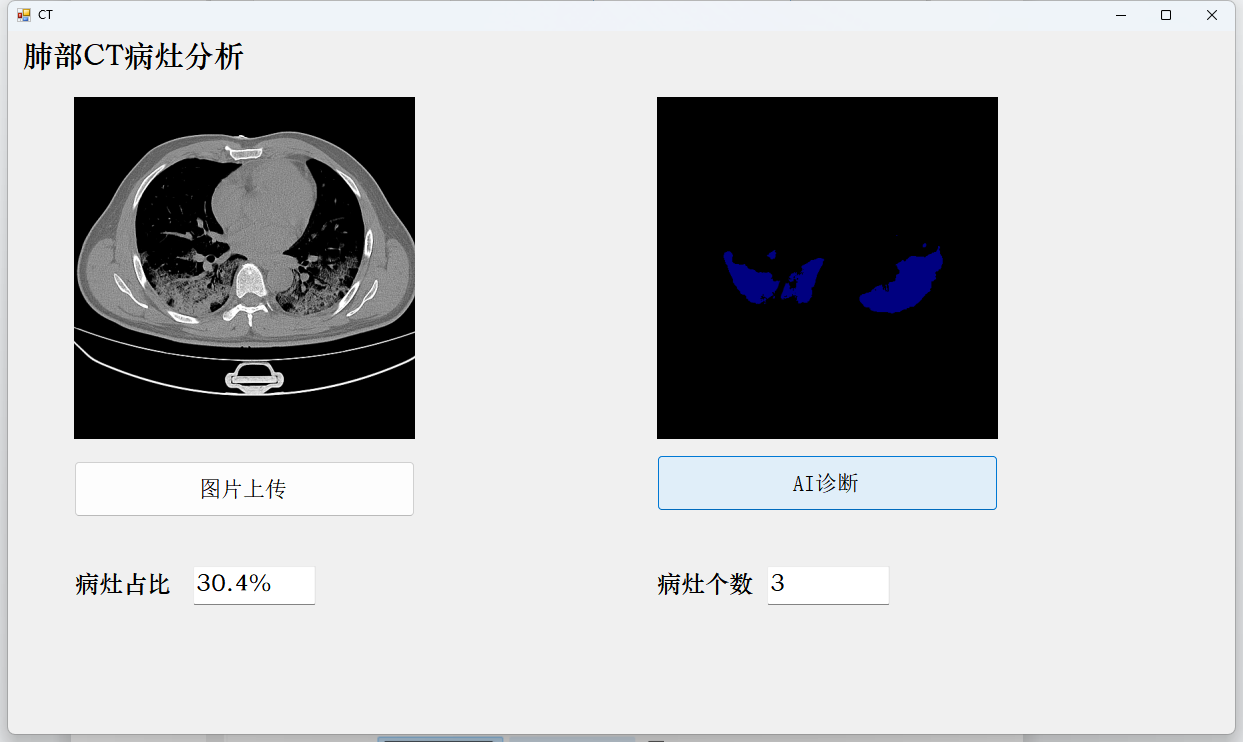


【实验心得】

本次实验收获颇多，对GUI图形化设计有了一个更深的了解。这个实验陆陆续续花了我将近两个多周的时间，从界面设计到事件监听梳理，一步步修改bug，收获颇丰。

对界面的设计要提前有所规划。有什么组件以及各组件的布局需要提前在纸上大体画出一个模型，根据自己设计的模型考虑自己需要几个JPanel、JButton等等，自己的界面需要什么样的布局方式，是Bord？Gird还是GirdBag？他们各有优势，我们应该根据需求选择。本实验中对于标准计算器按钮采用网格布局方式，三个JPanel是边界布局方式。科学计算器中因为第一行只有两个按钮，布局可能更复杂一点所以采用的GirdBag布局方式。在放置两个按钮后，利用一个JPanel，让JPanel的gridwidth为0，设置为本行的最后一个组件，让其填充完本行剩余部分。同理在后面按钮的换行的也是采用每行第七个按钮的gridwidth为0进行换行。设置容器的大小也很重要，如果太大则会导致较大区域的空白，如果太小则导致组件重叠被挡住。这就像我们最近在做的一个自主无人消毒小车和做的新冠肺炎检测平台，最开始的时候，我们设计的亚克力板在Solidworks中组合没有问题，但是最后成品出来的时候就有很多螺丝错位或者孔径太大的问题，还有亚克力板材质不够硬导致放上消毒水以后，小车车板发生了部分侧弯，所以在一开始我们就需要把这些问题考虑完善，毕竟良好的开始是成功的一半。(下图为我们第八周四天开发的新冠肺炎检测平台，程序采用C#进行GUI界面设计与神经网络部署)





这是我们最近设计的基于ROS系统的无人消毒小车，外形设计上存在缺憾



注意对 Container 属性的设置，为了保证界面布局不被破坏，我选择了让其大小 不可改变，当时也要注意常用的设置，例如为可见、设置窗口关闭动作等。

为了能够更加便捷，我在计算器上设置两个显示框，上显示框用于显示全部过 程的计算公式，下显示框显示当前输入数以及结果。上显示框需要不断在补充延长字符 串，我选择了 JTextArea ，因为它可以 append ，而这是 JTextField 所缺乏，利用 append 方法可以快速简洁的补充延长字符串，实现完整计算公式的输入。

两位数的操作符实验借鉴了一位学姐的博客：用不同运算符进行不同级的运算，用不同的运算符分别表示就可以把这三个等级的运算分开，互不干扰。以科学计算器为例，double a,b 用于表示加减运算的第一、二为操作数，p、q 分别表示乘除运算的第一、二为操作数，x、y 表示其他两位操作数运算符的第一、二为操作数 (包括x^y、 x^3 、y √x 、log(y)x ) ，优先级依次提高。在设计运算符标记符，当时对应的按钮被点击时对应的标记符为 true，此时 JTextField 中的数字为第一操作数存入。当其他运算符被点击时，调用一个标记符判断方法对算术 运算符的标记符进行判断，当 true 时再调用对应的计算方法，此时 JTextField 显示的内容为第二个操作数，完成操作符对应的运算，完成运算后将标记符返回为false ，关闭相关运算。加减运算的判断方法为handleOperate1 ，乘除为 handleOperate2，其余为 handleOperate 。高级运算不对低级运算标记符进行判断，即高级运算符调用时不考虑低 级运算，按照运算级由高到低调用他的判断方法，在=按钮的监听事件中，依次从高级 到低级调用判断方法，进行运算。这样高级运算执会优先执行，高级运算执行结果作为 优先级低的操作数，实现优先级。

括号的实现依赖于堆栈，前括号需要 压栈，后括号则是出栈。其实再把上一个运算优先级问题解决以后，这个问题其实就很 好实现了，实例化两个栈，一个存储操作数，一个存放标记符状态。压栈就是对操作数 x y a b p q 以及运算符的标记符压栈，并且将标记全部恢复到false 。出栈则是弹出操作 数和标记符，并且按照优先级高低调用标记符判断方法实现运算。

JAVA 课程的是短暂而又痛苦的，一周一节课但是每次实验自己课下都需要耗费巨大的时间、精力、心血去完成，经过这三个月的“折磨”，自己对 JAVA 有了一个认识和了解。最后一次大作业十分考验自己的耐心和逻辑思维、全局设计，对自己的能力有了一个极大的提升。

感谢老师十周以来的辛苦付出。