山东大学

信息科学与工程学院

2020－2021 学年第二学期

实　验　报　告

课程名称：　Java 编程技术

实验名称：　实验五

专　业　班　级　　物 联 网

学　生　学　号

学　生　姓　名

实　验　时　间　2020 年 12 月 03 日

实验报告

【实验要求】

1. 要写代码段注释，对于代码段要分段注释，说明代码用途， 如果这个程序是自己完成的，代码功能注释应该毫无压力，如果 是参考网络上的部分代码，必须要看懂程序，必须学会分析程序；

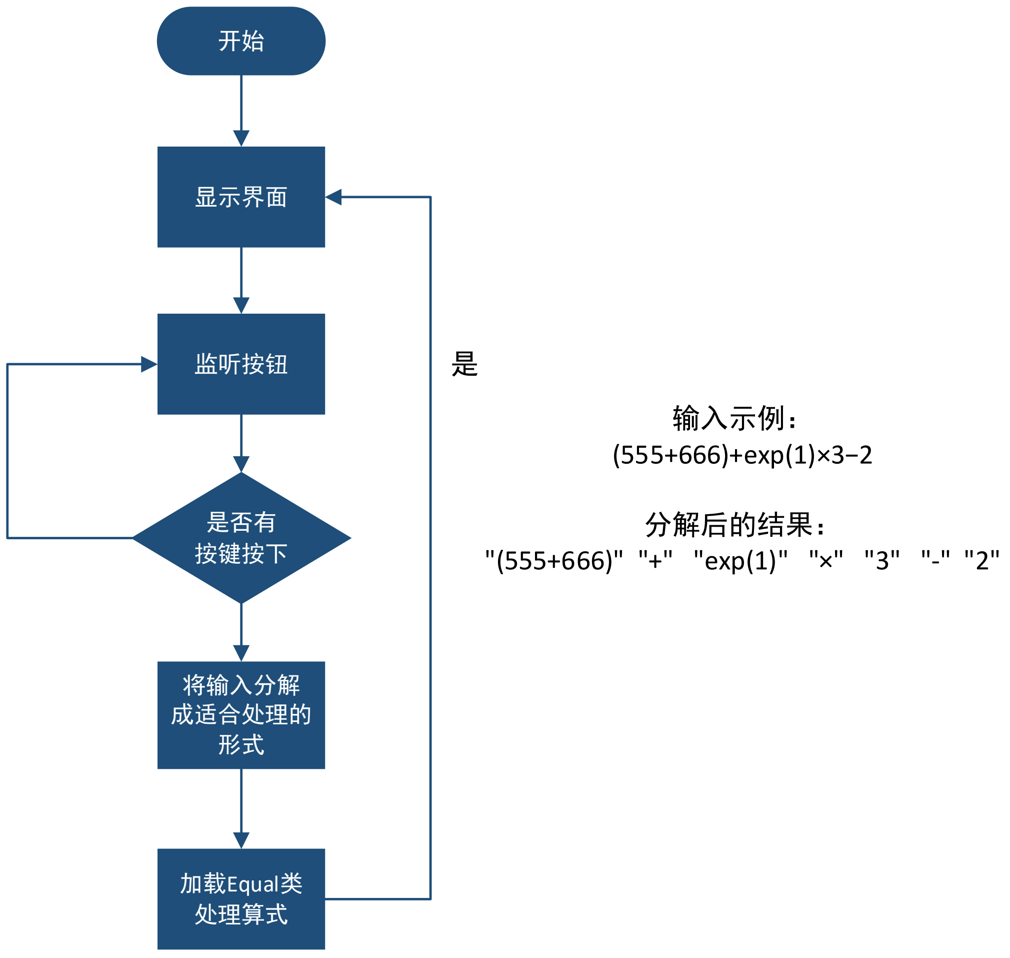
2. 要写实验心得，这个实验你完成后一定能有收获，如果你没有写实验心得，可能你是写程序大神，不屑于这么简单的实验，或者你没有认真做这个实验，所以才可能没有什么好说的。

【实验具体内容】

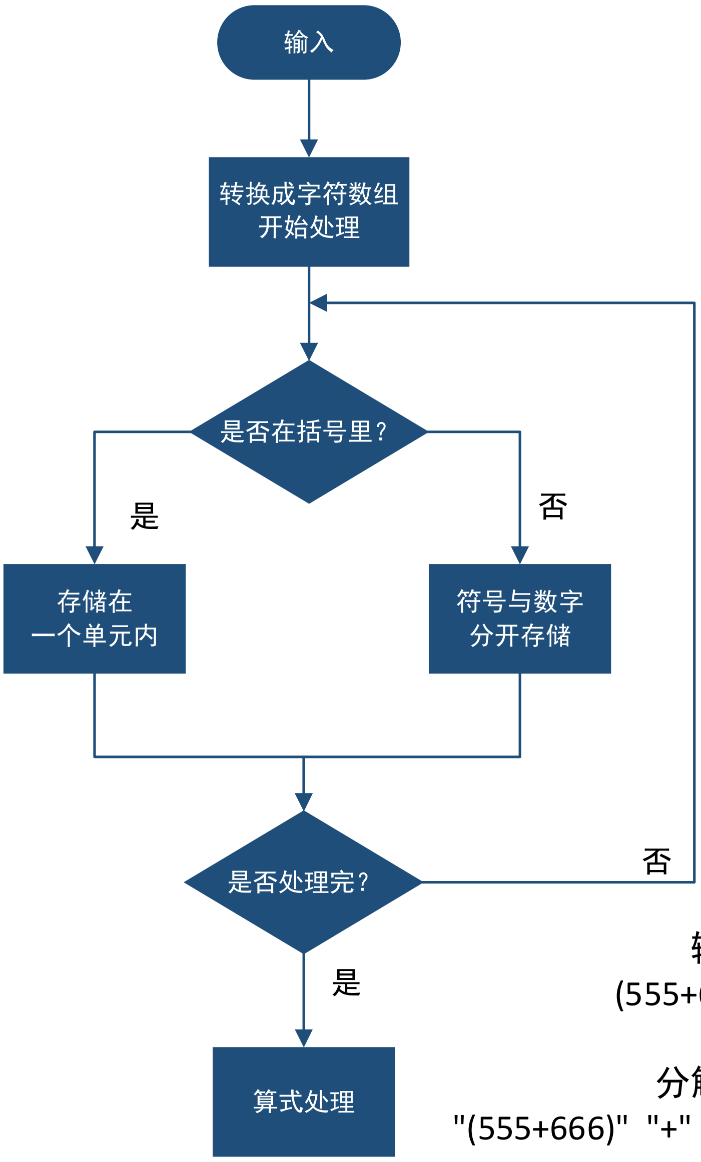
如果没有，请老师先阅读使用指南

（1）实验流程图（非必须，根据实际要求来定，算法题建议画流程图）：

1、总体程序流程：



2. 算式转换流程：



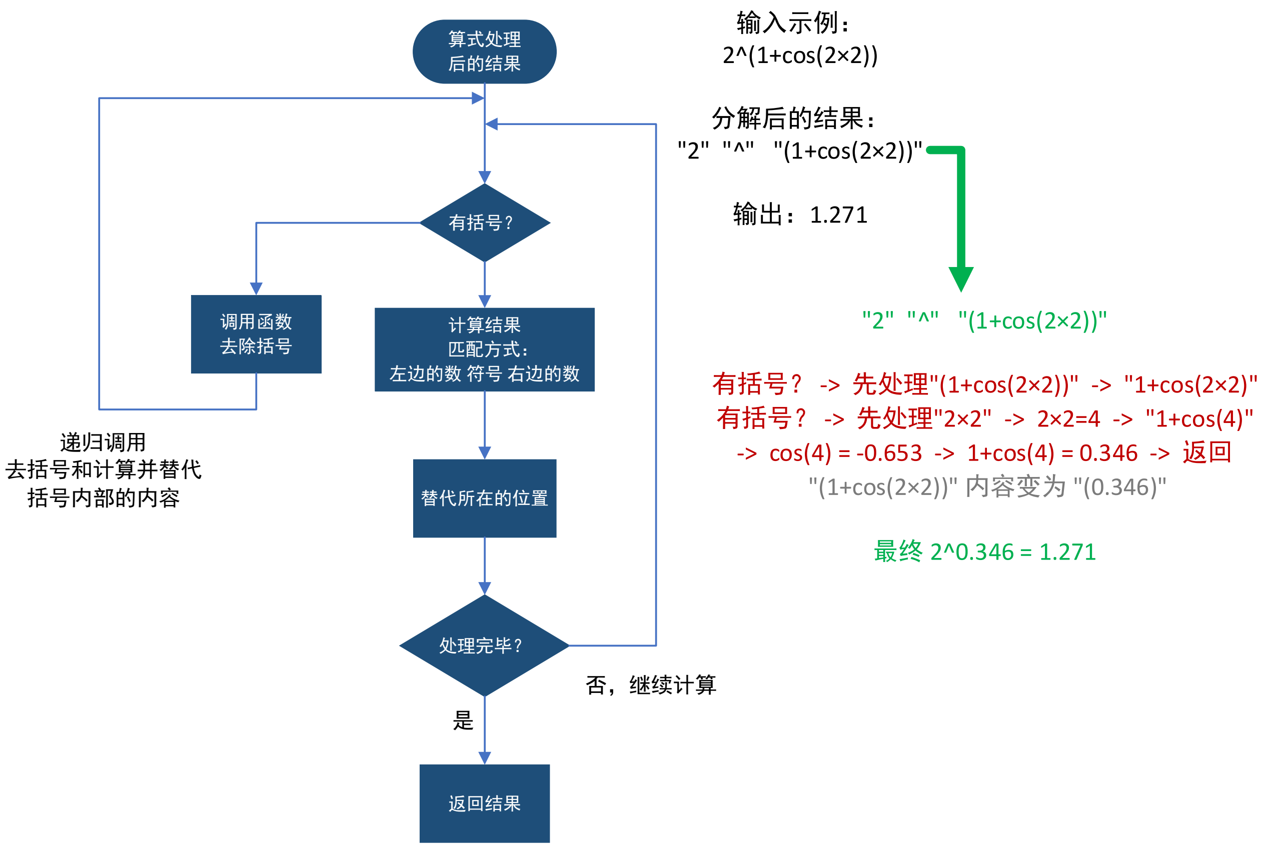
输入示例：

(555+666)+exp(1)×3−2

分解后的结果：

"(555+666)" "+" "exp(1)" "×" "3" "-" "2"

3. 算式计算流程：



程序主要思想：将输入的算式变换成容易处理的形式来处理并且输出。

（2）实验源代码（粘贴源代码）：

目录：

1. 文件 MainGUI.java 中：

```
package calculator;
```

```
import com.wolfram.alpha.WAEngine;
import com.wolfram.alpha.WAQuery;

import javax.imageio.ImageIO;
import javax.swing.*;
import javax.swing.plaf.basic.BasicBorders;
import java.awt.*;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.Objects;


/* WARNING */
/* IF NOT ESSENTIAL, DO NOT CHANGE THE CONTENT OF COM.WOLFRAM.AL
PHA PACKAGE */
/* IT HAS BEEN MODIFIED BY VELA YANG TO BE USED MORE EASILY IN TH
IS PROJECT */

public class MainGUI implements Runnable {

    // Tag for whether should new Thread do
    private enum DoWhat {
        GET_KNOWLEDGE, CALCULATE
    }

    // Tag for using ONLINE mode or not
    private enum CalculatorMode {
        ALPHA, LOCAL
    }
    // Enum the action what should do when click the button
    private static DoWhat shouldDoWhat = DoWhat.GET_KNOWLEDGE;
    // if it is online
    private static CalculatorMode ifONLINE = CalculatorMode.ALPH
A;

    // jTextArea for Calculator page
    protected static final JTextArea jTextArea = new JTextArea("
  ", 5, 30);
    // for WolframAlpha page
    static final JTextArea jTextArea2 = new JTextArea("  ", 5, 3
0);
    // APPID of WolframAlpha API
    private static final String appid = "XWKV46-AAUVUJX388";
    // Contains button of calculator
    private static final JPanel buttonPanel1 = new JPanel();
    private static final JPanel buttonPanel2 = new JPanel();
    // CardLayout for different windows
    private static final CardLayout cardLayout = new CardLayout
();
    private static final JButton calculateButton = new JButton("
=");
    private static final JButton acquireButton = new JButton("In
put and Acquire ANYTHING!!");
    // initialize the GUI components
    public static JFrame jFrame = new JFrame("Simple Calculator
By Vela Yang");
    private static final Container container = jFrame.getContent
Pane();
    private static JLabel resultImage;
    // The panel contains first page
    private final JPanel topPanel1 = new JPanel();
    // The panel contains the second page
    private final JPanel topPanel2 = new JPanel();
    private final Color borderColor = new Color(224, 224, 224);
    // the result of calculator
    private double resultNumber;

    // Main function
    public static void main(String[] args) throws ClassNotFoundE
xception, UnsupportedLookAndFeelException, InstantiationExcepti
on, IllegalAccessException {

        // Set Layout of jFrame
        container.setLayout(cardLayout);
        // Set the gridLayout of JPanel
        GridLayout gridLayout = new GridLayout(4, 5, 0, 0);
        buttonPanel1.setLayout(gridLayout);
        buttonPanel2.setLayout(gridLayout);
        // Set the UI looks like
        UIManager.setLookAndFeel(UIManager.getCrossPlatformLookA
ndFeelClassName());

        // Initialize
        new MainGUI().addListenerForAcquireAndCalculateButton();
        new MainGUI().setjTextAreas();
        new MainGUI().setWolframAlphaPage();
        new MainGUI().setContainer();
        new MainGUI().addMenuBar();        jFrame.setIconImage(ne
w ImageIcon(Objects.requireNonNull(MainGUI.class.getClassLoader
().getResource("icon.png"))).getImage());
        jFrame.setVisible(true);
        jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jFrame.setSize(400, 600);
    }

    private void addListenerForAcquireAndCalculateButton() {
        // Equal
        // Calculate
        calculateButton.addActionListener(e -> {
            shouldDoWhat = DoWhat.CALCULATE;
            new Thread(new MainGUI()).start();
        });
        // Processing the input content
        acquireButton.addActionListener(e -> {
            shouldDoWhat = DoWhat.GET_KNOWLEDGE;
            new Thread(new MainGUI()).start();
        });
    }

    // Set MenuBar of JFrame
    private void addMenuBar() {

        // JMenuBar
        final JMenuBar jMenuBar = new JMenuBar();
        JMenu jMenu = new JMenu("Change");
        jMenu.setFont(new Font("Segoe UI", Font.PLAIN, 20));
        JLabel modeLabel = new JLabel(" 科学");
        modeLabel.setFont(new Font("微软雅黑", Font.BOLD, 20));


        // MenuItem 1
        JMenuItem calculatorMenu = new JMenuItem("Calculator");
        calculatorMenu.addActionListener(e -> {
            modeLabel.setText(" 科学");
            jTextArea.setText(" ");
            cardLayout.last(container);
            jFrame.setSize(400, 600);
        });
        calculatorMenu.setBackground(Color.white);
        calculatorMenu.setFont(new Font("Segoe UI", Font.PLAIN,
20));
```

```
        // MenuItem 2
        JMenuItem wolframAlphaMenu = new JMenuItem("Click Here!
");
        wolframAlphaMenu.setBackground(Color.white);
        wolframAlphaMenu.setFont(new Font("Segoe UI", Font.PLAIN,
 20));
        wolframAlphaMenu.addActionListener(e -> {
            modeLabel.setText("  Alpha");
            jTextArea2.setText(" ");
            cardLayout.first(container);
            jFrame.setSize(650, 840);
        });

        jMenu.add(calculatorMenu);
        jMenu.add(wolframAlphaMenu);
        jMenuBar.add(jMenu);
        jMenuBar.add(modeLabel);
        jMenuBar.setBackground(Color.white);

        // Set size of MenuBar
        jMenuBar.setPreferredSize(new Dimension(200, 40));
        jFrame.setJMenuBar(jMenuBar);
    }

    // The panel contains buttonPanel
    private void setContainer() {
        // Set Layout of jFrame
        container.setLayout(cardLayout);
        // Set the gridLayout of JPanel
        GridLayout gridLayout = new GridLayout(4, 5, 0, 0);
        buttonPanel1.setLayout(gridLayout);
        buttonPanel2.setLayout(gridLayout);
        // Set the gridLayout of JFrame
        topPanel1.setLayout(new GridLayout(3, 1, 0, 0));

        topPanel1.add(jTextArea);

        // Add functional buttons
        new MainGUI().addButtons();

        // Set the font of buttons
        for (int i = 0; i < buttonPanel1.getComponentCount(); i+
+) {
            buttonPanel1.getComponent(i).setFont(new Font("Segoe
UI", Font.PLAIN, 20));
            buttonPanel1.getComponent(i).setBackground(new Color
(244, 244, 244));
            buttonPanel2.getComponent(i).setFont(new Font("Segoe
UI", Font.PLAIN, 20));
            if ((0 < i && i < 4) || (5 < i && i < 9) || (10 < i && i
 < 14) || (15 < i && i < 19))
                buttonPanel2.getComponent(i).setBackground(Color.
white);
            else buttonPanel2.getComponent(i).setBackground(new C
olor(244, 244, 244));
            if (i == 19) buttonPanel2.getComponent(i).setBackgrou
nd(new Color(75, 210, 172));
        }

        // Add the jPanels
        topPanel1.add(buttonPanel1);
        topPanel1.add(buttonPanel2);
        container.add(topPanel1);
        // Set the frame size
        jFrame.setSize(650, 840);
```

```
    }

    private void setWolframAlphaPage() {
        // Set the font of jTextArea
        jTextArea2.setFont(new Font("Microsoft YaHei UI", Font.P
LAIN, 30));
        jTextArea2.setSelectedTextColor(Color.black);

        // Reset the size of Frame
        jFrame.setSize(650, 840);
        // Set topPanel2 with GridBagLayout
        topPanel2.setLayout(new GridBagLayout());

        // Set Acquire Button

        acquireButton.setFont(new Font("Segoe UI", Font.BOLD, 2
0));
        acquireButton.setBackground(Color.white);

        GridBagConstraints gbc = new GridBagConstraints();

        // Add jTextArea to GridBagLayout
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 100;
        gbc.gridheight = 200;
        gbc.weightx = 10;
        gbc.weighty = 10;
        gbc.anchor = GridBagConstraints.NORTH;
        gbc.fill = GridBagConstraints.BOTH;
        gbc.insets = new Insets(0, 0, 0, 0);
        gbc.ipadx = 100;
        gbc.ipady = 100;
        topPanel2.add(jTextArea2, gbc);
        // Add jTextArea to GridBagLayout
        gbc.gridy = 200;
        gbc.gridwidth = 100;
        gbc.gridheight = 400;
        gbc.ipadx = 650;
        gbc.ipady = 600;
        JScrollPane jScrollPane = new MainGUI().getScrollPane();

        // Set velocity of MouseWheel rolls
        jScrollPane.getVerticalScrollBar().setUnitIncrement(35);
        jScrollPane.addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if (e.getKeyChar() == '\n') {

                    // Ignore the default result which appends a '\
n' to jTextArea2
                    e.consume();
                    acquireButton.doClick();
                }
            }
        });
        topPanel2.add(jScrollPane, gbc);

        // Add jTextArea to GridBagLayout
        gbc.gridy = 600;
        gbc.gridwidth = 100;
        gbc.gridheight = 200;
        gbc.ipadx = 600;
        gbc.ipady = 30;
        topPanel2.add(acquireButton, gbc);
        container.add(topPanel2);
    }
    // Return a JScrollPane by given image
    private JScrollPane getScrollPane() {
        resultImage = new JLabel();
        JPanel imagePanel = new JPanel();
        imagePanel.setBackground(Color.white);
        imagePanel.add(resultImage);
        imagePanel.setPreferredSize(new Dimension(600, 2500));
        return new JScrollPane(imagePanel);
    }

    private String getResultFromWolframAlpha() throws IOExceptio
n {
        String input = jTextArea.getText();
        WAEngine engine = new WAEngine();

        // These properties will be set in all the WAQuery object
s created from this WAEngine.
        engine.setAppID(appid);
        //engine.addFormat("plaintext");
        // Create the query.
        WAQuery query = engine.createQuery();
        // Set properties of the query.
        query.setInput(input);

        // Get result
        String result = "default";
        BufferedReader bufferedReader = new BufferedReader(new In
putStreamReader(new URL(engine.toURL(query, "result")).openStre
am()));
        String inputLine;
        while ((inputLine = bufferedReader.readLine()) != null) r
esult = inputLine;
        if (result.equals("default")) throw new VoidResultExcept
```

```
ion();

        // If can't get result from server, then throw the Except
ion
        return result;
    }

    private void setjTextAreas() {
        // Make jTextArea change lines automatically
        jTextArea.setLineWrap(true);
        jTextArea.setWrapStyleWord(true);
        jTextArea2.setLineWrap(true);
        jTextArea2.setWrapStyleWord(true);
        // Set the font of jTextArea
        jTextArea.setFont(new Font("Microsoft YaHei UI", Font.PL
AIN, 30));
        jTextArea.setSelectedTextColor(Color.black);

        // Add Listener for jTextAreas
        jTextArea.addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if (e.getKeyChar() == '\n') {

                    // Ignore the default result which appends a '\
n' to jTextArea
                    e.consume();
                    calculateButton.doClick();
                }
                if (e.getKeyChar() == '*') {

                    // Ignore the default result which appends a '\
n' to jTextArea2
                    jTextArea.insert("×", jTextArea.getCaret().get
Dot());
                        jTextArea.setEditable(false);
                    }
                    if (e.getKeyChar() == '/') {

                        // Ignore the default result which appends a '\
n' to jTextArea2
                        e.consume();
                        jTextArea.insert("÷", jTextArea.getCaret().get
Dot());
                        jTextArea.setEditable(false);
                    }
                    if (e.getKeyChar() != '/' && e.getKeyChar() != '*')
 jTextArea.setEditable(true);
                }
            });
            jTextArea2.addKeyListener(new KeyAdapter() {
                @Override
                public void keyPressed(KeyEvent e) {
                    if (e.getKeyChar() == '\n') {
                        // Ignore the default result which appends a '\
n' to jTextArea2
                        e.consume();
                        acquireButton.doClick();
                    }
                }
            });
        }

        // Finish the run function of Runnable Interface
        // Only acquiceButton and calculateButton call this RUN METH
OD(keyListener use JButton.doClick() to call this)
        // In this Calculator Program, RUN METHOD will process the In
put and give the output to the user when it is being called
```

```
    @Override
    public void run() {

        switch (shouldDoWhat) {

            // Actions for calculaor Page
            case CALCULATE:
                // If is Online , get result from Alpha
                if (ifONLINE == CalculatorMode.ALPHA) {

                    String result = "default";
                    try {
                        result = new MainGUI().getResultFromWolfram
Alpha();
                        jTextArea.append(" = " + result);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                    // If can't get result from WolframAlpha, then
inform users of change the input
                    if (result.equals("default"))
                        try {
                            throw new VoidResultException();
                        } catch (VoidResultException e) {
                            e.printStackTrace();
                        }

                } else {
                    // get the input content
                    char[] input = jTextArea.getText().toCharArray
();

                    // Split the input content to a easily processe
```

```
d one
                 StringBuffer[] segmentOfInput = new StringBuff
er[100];
                 for (int i = 0; i < segmentOfInput.length; i++)
 {
                     segmentOfInput[i] = new StringBuffer();
                 }

                 // Split the input content to a easily processe
d one
                 // after this for sentence, the input content s
uch as :
                 // 1.5+2×3+9÷3 will be stored in the given Stri
ngBuffer collection like the follows
                 // "1.5" "+" "2" "×" "3" "+" "9" "÷" "3"
                 int inBracket = 0;
                 for (int i = 0, j = 0; i < input.length; i++) {
                     if (Character.isDigit(input[i]) || input[i]
 == '.' || input[i] == '!' || input[i] == 'π' || input[i] == 'e'
|| inBracket != 0) {
                         if (i >= 1 && !Character.isDigit(input[i
 - 1]) && input[i - 1] != '(' && input[i - 1] != '.' && input[i]
 != '!' && inBracket == 0)
                             j++;
                         segmentOfInput[j].append(input[i]);
                     }
                     // store the operator
                     else {
                         if (i >= 1 && !Character.isLetter(input
[i - 1]) && input[i] != ')') j++;
                         else if (i >= 1 && (Character.isDigit(in
put[i - 1]) || input[i - 1] == 'π' || (input[i - 1] == 'e' && inp
ut[i] != 'x'))
                                 && input[i - 1] != ')' && input[i]
 != ')')
                             j++;
                         segmentOfInput[j].append(input[i]);
                     }
                     if (input[i] == '(') inBracket++;
                     if (input[i] == ')') inBracket--;
                 }

                 // 如果括号数量不等，抛出异常
                 if (inBracket != 0) try {
                     throw new NoCorrespondingBracketException();
                 } catch (NoCorrespondingBracketException e) {
                     e.printStackTrace();
                 }

                 try {
                     // get result
                     resultNumber = Equal.calculateResult(segmen
tOfInput);
                     // if resultNumber would be more likely to a
n integer number, then print the integer
                     long intResultNumber = (long) resultNumber;
                     if  (Math.abs(resultNumber  -  intResultNumbe
r) <= 1e-7) {
                         jTextArea.setText(jTextArea.getText()  +
 " = " + intResultNumber);
                     }  else  jTextArea.setText(jTextArea.getText
() + " = " + String.format("%.8f", resultNumber));
                     // If result is too large that can not prese
nt, then throw InfinityException
                 }  catch  (InfinityException  infinityException)
{
                     infinityException.printStackTrace();
                 }
```

```
                }
                break;

            // Actions for wolframAlpha Page
            case GET_KNOWLEDGE: {
                String inputString = jTextArea2.getText();
                jTextArea2.append("\n  Good  things  deserve  waitin
g...");

                // Get correct API URL from WolframAlpha API
                WAEngine engine = new WAEngine();
                engine.setAppID(appid);
                WAQuery query = engine.createQuery();
                query.setInput(inputString);
                try {
                    // Get Image from API
                    URL  url  =  new  URL(engine.toURL(query,  "simple
"));

                    BufferedImage image1 = ImageIO.read(url);
                    resultImage.setIcon(new ImageIcon(image1));
                } catch (IOException malformedURLException) {
                    jTextArea2.setText("\nCan't get result, try to
 rearrange the INPUT content");
                    malformedURLException.printStackTrace();
                }
            }
        }
    }

    // Add buttons and set the function of buttons
    // Intotal, 8lines for buttons
    private void addButtons() {

        // Line 1
        {
            // Resize the window
            JButton modeButton = new JButton("Alpha");
            modeButton.setBorder(new BasicBorders.ButtonBorder(bo
rderColor, borderColor, borderColor, borderColor));
            modeButton.addActionListener(e -> {
                ifONLINE = modeButton.getText().equals("Local") ?
CalculatorMode.ALPHA : CalculatorMode.LOCAL;
                modeButton.setText(modeButton.getText().equals("L
ocal") ? "Alpha" : "Local");
            });
            buttonPanel1.add(modeButton);

            // Button presents Grecian alphabet π which is approxi
mately to 3.1415926536 on jTextArea
            JButton piButton = new JButton("π");
            piButton.setBorder(new BasicBorders.ButtonBorder(bord
erColor, borderColor, borderColor, borderColor));
            piButton.addActionListener(e -> jTextArea.insert("π",
 jTextArea.getCaret().getDot()));
            buttonPanel1.add(piButton);

            // Button presents natural exponent e which is approxi
mately to 2.7182818285 on jTextArea
            JButton eButton = new JButton("e");
            eButton.addActionListener(e -> jTextArea.insert("e",
jTextArea.getCaret().getDot()));
            buttonPanel1.add(eButton);
            eButton.setBorder(new BasicBorders.ButtonBorder(borde
rColor, borderColor, borderColor, borderColor));

            // Button for Clearing the content of jTextArea
            // More precisely, make the jTextArea to the initial s
tatus which has the String content "  "
```

```
          JButton clearButton = new JButton("C");
          clearButton.setBorder(new BasicBorders.ButtonBorder(b
orderColor, borderColor, borderColor, borderColor));
          clearButton.addActionListener(e  ->  jTextArea.setText
("  "));
          buttonPanel1.add(clearButton);

          // Button for Delete a character of jTextArea
          // Unfinished at now
          JButton delButton = new JButton("Del");
          delButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
          delButton.addActionListener(e -> {
             StringBuilder tempString = new StringBuilder(jText
Area.getText());
             tempString.deleteCharAt(jTextArea.getCaret().getD
ot() - 1);
             jTextArea.setText(tempString.toString());
          });

          buttonPanel1.add(delButton);

       }

       // Line 2
       {
          // sin function
          JButton sinButton = new JButton("sin");
          sinButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
          sinButton.addActionListener(e -> jTextArea.insert("si
n(", jTextArea.getCaret().getDot()));
          buttonPanel1.add(sinButton);
```

```
            // cos function
            JButton cosButton = new JButton("cos");
            cosButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
            cosButton.addActionListener(e -> jTextArea.insert("co
s(", jTextArea.getCaret().getDot()));
            buttonPanel1.add(cosButton);

            // tan function
            JButton tanButton = new JButton("tan");
            tanButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
            tanButton.addActionListener(e -> jTextArea.insert("ta
n(", jTextArea.getCaret().getDot()));
            buttonPanel1.add(tanButton);

            // arcsin function
            JButton asinButton = new JButton("asin");
            asinButton.setBorder(new BasicBorders.ButtonBorder(bo
rderColor, borderColor, borderColor, borderColor));
            asinButton.addActionListener(e -> jTextArea.insert("a
rcsin(", jTextArea.getCaret().getDot()));
            buttonPanel1.add(asinButton);

            // arctan function
            JButton atanButton = new JButton("atan");
            atanButton.setBorder(new BasicBorders.ButtonBorder(bo
rderColor, borderColor, borderColor, borderColor));
            atanButton.addActionListener(e -> jTextArea.insert("a
rctan(", jTextArea.getCaret().getDot()));
            buttonPanel1.add(atanButton);

        }
```

```
       // Line 3
       {

          // Integer
          JButton intButton = new JButton("∫");
          intButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
          intButton.addActionListener(e -> jTextArea.insert("∫
", jTextArea.getCaret().getDot()));
          buttonPanel1.add(intButton);

          // Differential
          JButton diffButton = new JButton("y'");
          diffButton.setBorder(new BasicBorders.ButtonBorder(bo
rderColor, borderColor, borderColor, borderColor));
          diffButton.addActionListener(e -> jTextArea.insert("'
", jTextArea.getCaret().getDot()));
          buttonPanel1.add(diffButton);

          // Button presents ^2 on jTextArea
          JButton squareButton = new JButton("<html>X<sup>2</su
p>");
          squareButton.setBorder(new BasicBorders.ButtonBorder
(borderColor, borderColor, borderColor, borderColor));
          squareButton.addActionListener(e -> jTextArea.insert
("^2", jTextArea.getCaret().getDot()));
          buttonPanel1.add(squareButton);



          // Exp()
          JButton expButton = new JButton("exp");
          expButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
          expButton.addActionListener(e -> jTextArea.insert("ex
p(", jTextArea.getCaret().getDot()));
          buttonPanel1.add(expButton);

          // Mod
          JButton modButton = new JButton("mod");
          modButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
          modButton.addActionListener(e -> jTextArea.insert("mo
d", jTextArea.getCaret().getDot()));
          buttonPanel1.add(modButton);
      }

      // Line 4
      {
          // Square root
          JButton squareRootButton = new JButton("<html>√<sup>
—</sup>");
          squareRootButton.setBorder(new BasicBorders.ButtonBor
der(borderColor, borderColor, borderColor, borderColor));
          squareRootButton.addActionListener(e -> jTextArea.ins
ert("sqrt(", jTextArea.getCaret().getDot()));
          buttonPanel1.add(squareRootButton);

          // Left bracket
          JButton leftBracketButton = new JButton("(");
          leftBracketButton.setBorder(new BasicBorders.ButtonBo
rder(borderColor, borderColor, borderColor, borderColor));
          leftBracketButton.addActionListener(e -> jTextArea.in
sert("(", jTextArea.getCaret().getDot()));
          buttonPanel1.add(leftBracketButton);

          // Right bracket
          JButton rightBracketButton = new JButton(")");
          rightBracketButton.setBorder(new BasicBorders.ButtonB
order(borderColor, borderColor, borderColor, borderColor));
          rightBracketButton.addActionListener(e -> jTextArea.i
nsert(")", jTextArea.getCaret().getDot()));
          buttonPanel1.add(rightBracketButton);

          // Factorial
          JButton factorialButton = new JButton("n!");
          factorialButton.setBorder(new BasicBorders.ButtonBord
er(borderColor, borderColor, borderColor, borderColor));
          factorialButton.addActionListener(e -> jTextArea.inse
rt("!", jTextArea.getCaret().getDot()));
          buttonPanel1.add(factorialButton);

          // Divide
          JButton divideButton = new JButton("÷");
          divideButton.setBorder(new  BasicBorders.ButtonBorder
(borderColor, borderColor, borderColor, borderColor));
          divideButton.addActionListener(e  ->  jTextArea.insert
("÷", jTextArea.getCaret().getDot()));
          buttonPanel1.add(divideButton);
      }

      // Line 5
      {
          // Power
          JButton exponentialXAndYButton = new JButton("<html>X
<sup>y</sup>");
          exponentialXAndYButton.setBorder(new BasicBorders.But
tonBorder(borderColor, borderColor, borderColor, borderColor));
          exponentialXAndYButton.addActionListener(e -> jTextAr
ea.insert("^", jTextArea.getCaret().getDot()));
          buttonPanel2.add(exponentialXAndYButton);

          // Seven
          JButton sevenButton = new JButton("7");
          sevenButton.setBorder(new BasicBorders.ButtonBorder(b
orderColor, borderColor, borderColor, borderColor));
          sevenButton.addActionListener(e -> jTextArea.insert("
7", jTextArea.getCaret().getDot()));
          buttonPanel2.add(sevenButton);

          // Eight
          JButton eightButton = new JButton("8");
          eightButton.setBorder(new BasicBorders.ButtonBorder(b
orderColor, borderColor, borderColor, borderColor));
          eightButton.addActionListener(e -> jTextArea.insert("
8", jTextArea.getCaret().getDot()));
          buttonPanel2.add(eightButton);

          // Nine
          JButton nineButton = new JButton("9");
          nineButton.setBorder(new BasicBorders.ButtonBorder(bo
rderColor, borderColor, borderColor, borderColor));
          nineButton.addActionListener(e -> jTextArea.insert("9
", jTextArea.getCaret().getDot()));
          buttonPanel2.add(nineButton);

          // Multiply
          JButton multiplyButton = new JButton("×");
          multiplyButton.setBorder(new BasicBorders.ButtonBorde
r(borderColor, borderColor, borderColor, borderColor));
          multiplyButton.addActionListener(e -> jTextArea.inser
t("×", jTextArea.getCaret().getDot()));
          buttonPanel2.add(multiplyButton);
      }

      // Line 6
      {
```

```
        // Power of ten
        JButton exponentialOfTenButton = new JButton("<html>1
0<sup>x</sup>");
        exponentialOfTenButton.setBorder(new BasicBorders.But
tonBorder(borderColor, borderColor, borderColor, borderColor));
        exponentialOfTenButton.addActionListener(e -> jTextAr
ea.insert("10^(", jTextArea.getCaret().getDot()));
        buttonPanel2.add(exponentialOfTenButton);

        // Four
        JButton fourButton = new JButton("4");
        fourButton.setBorder(new BasicBorders.ButtonBorder(bo
rderColor, borderColor, borderColor, borderColor));
        fourButton.addActionListener(e -> jTextArea.insert("4
", jTextArea.getCaret().getDot()));
        buttonPanel2.add(fourButton);

        // Five
        JButton fiveButton = new JButton("5");
        fiveButton.setBorder(new BasicBorders.ButtonBorder(bo
rderColor, borderColor, borderColor, borderColor));
        fiveButton.addActionListener(e -> jTextArea.insert("5
", jTextArea.getCaret().getDot()));
        buttonPanel2.add(fiveButton);

        // Six
        JButton sixButton = new JButton("6");
        sixButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
        sixButton.addActionListener(e  ->  jTextArea.insert("6
", jTextArea.getCaret().getDot()));
        buttonPanel2.add(sixButton);

        // Subtract
```

```
            JButton subtractButton = new JButton("-");
            subtractButton.setBorder(new BasicBorders.ButtonBorde
r(borderColor, borderColor, borderColor, borderColor));
            subtractButton.addActionListener(e -> jTextArea.inser
t("-", jTextArea.getCaret().getDot()));
            buttonPanel2.add(subtractButton);
        }

        // Line 7
        {
            // Logarithm
            JButton logarithmOfTenButton = new JButton("<html>log
<sub>10</sub>");
            logarithmOfTenButton.setBorder(new BasicBorders.Butto
nBorder(borderColor, borderColor, borderColor, borderColor));
            logarithmOfTenButton.addActionListener(e  ->  jTextAre
a.insert("log(", jTextArea.getCaret().getDot()));
            buttonPanel2.add(logarithmOfTenButton);

            // One
            JButton oneButton = new JButton("1");
            oneButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
            oneButton.addActionListener(e  ->  jTextArea.insert("1
", jTextArea.getCaret().getDot()));
            buttonPanel2.add(oneButton);

            // Two
            JButton twoButton = new JButton("2");
            twoButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
            twoButton.addActionListener(e  ->  jTextArea.insert("2
", jTextArea.getCaret().getDot()));
            buttonPanel2.add(twoButton);
            // Three
            JButton threeButton = new JButton("3");
            threeButton.setBorder(new BasicBorders.ButtonBorder(b
orderColor, borderColor, borderColor, borderColor));
            threeButton.addActionListener(e -> jTextArea.insert("
3", jTextArea.getCaret().getDot()));
            buttonPanel2.add(threeButton);

            // Add
            JButton addButton = new JButton("+");
            addButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
            addButton.addActionListener(e  ->  jTextArea.insert("+
", jTextArea.getCaret().getDot()));
            buttonPanel2.add(addButton);
        }

        // Line 8
        {
            // Logarithm of e
            JButton logarithmOfEButton = new JButton("ln");
            logarithmOfEButton.setBorder(new BasicBorders.ButtonB
order(borderColor, borderColor, borderColor, borderColor));
            logarithmOfEButton.addActionListener(e -> jTextArea.i
nsert("ln(", jTextArea.getCaret().getDot()));
            buttonPanel2.add(logarithmOfEButton);

            // Change the signal
            JButton positiveButton = new JButton("+/-");
            positiveButton.setBorder(new BasicBorders.ButtonBorde
r(borderColor, borderColor, borderColor, borderColor));
            positiveButton.addActionListener(e -> jTextArea.setTe
xt("  " + (resultNumber = resultNumber == 0 ? 0 : 0 - resultNumbe
r)));
            buttonPanel2.add(positiveButton);

            // Zero
            JButton zeroButton = new JButton("0");
            zeroButton.setBorder(new BasicBorders.ButtonBorder(bo
rderColor, borderColor, borderColor, borderColor));
            zeroButton.addActionListener(e -> jTextArea.insert("0
", jTextArea.getCaret().getDot()));
            buttonPanel2.add(zeroButton);

            // Decimal dot
            JButton dotButton = new JButton(".");
            dotButton.setBorder(new BasicBorders.ButtonBorder(bor
derColor, borderColor, borderColor, borderColor));
            dotButton.addActionListener(e -> jTextArea.insert(".
", jTextArea.getCaret().getDot()));
            buttonPanel2.add(dotButton);


            calculateButton.setBorder(new BasicBorders.ButtonBord
er(borderColor, borderColor, borderColor, borderColor));
            buttonPanel2.add(calculateButton);
        }
    }

}
```

2. 文件 Equal.java 中：

```
package calculator;

public class Equal {

    // "(123)" -> "123"
    private static StringBuilder deleteBracket(StringBuilder tem
p) {
        if (temp.lastIndexOf("(") == -1) return temp;
        return new StringBuilder(temp.substring(temp.lastIndexOf
("(") + 1, temp.indexOf(")")));
    }

    // Get result and return
    protected static double calculateResult(StringBuilder[] segm
ents) throws InfinityException {

        // split the segment and stored in segmentOfInput
        StringBuilder[] segmentOfInput = segments;

        // the temporary calculating result will be stored in tem
pCell
        // and will replace a element in segmentOfInput
        double tempCell;

        // Processing the Bracket
        for (int i = 0; i < segmentOfInput.length; i++) {
            if (segmentOfInput[i].toString().equals("")) continu
e;
            if (segmentOfInput[i].toString().lastIndexOf("(") !=
-1) {
                String tempString = segmentOfInput[i].substring(se
gmentOfInput[i].indexOf("(") + 1, segmentOfInput[i].lastIndexOf
(")"));
                tempString = String.valueOf(new Equal().elicitBrac
ket(tempString));
                segmentOfInput[i] = segmentOfInput[i].replace(seg
mentOfInput[i].indexOf("(") + 1, segmentOfInput[i].lastIndexOf
(")"), tempString);
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                //i = 0;
            }
        }
        // calculate the top priority inputs
        for (int i = 0; i < segmentOfInput.length; i++) {
            if (segmentOfInput[i].toString().equals("")) continu
e;
            tempCell = 0;
            // replace the consistent number pi
            if (segmentOfInput[i].toString().lastIndexOf("π") !=
-1) {
                tempCell += Math.PI;
                segmentOfInput[i].replace(segmentOfInput[i].toStr
ing().indexOf("π"), segmentOfInput[i].toString().indexOf("π") +
 1, String.valueOf(tempCell));
                i = 0;
            }
            // replace the consistent number E
            if (segmentOfInput[i].toString().lastIndexOf("e") !=
-1 && segmentOfInput[i].toString().lastIndexOf("exp") == -1) {
                tempCell += Math.E;
                segmentOfInput[i].replace(segmentOfInput[i].toStr
ing().indexOf("e"), segmentOfInput[i].toString().indexOf("e") +
 1, String.valueOf(tempCell));
                i = 0;
            }
            // calculate the ln()
            if (segmentOfInput[i].toString().lastIndexOf("ln") !=
 -1) {
                String tempString = segmentOfInput[i].substring(se
gmentOfInput[i].indexOf("(") + 1, segmentOfInput[i].indexOf(")
"));
                tempCell += Math.log(Double.parseDouble(tempStrin
g));
                segmentOfInput[i] = new StringBuilder(String.value
Of(tempCell));
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the log()
            if (segmentOfInput[i].toString().lastIndexOf("log") !
= -1) {
                String tempString = segmentOfInput[i].substring(se
gmentOfInput[i].indexOf("(") + 1, segmentOfInput[i].indexOf(")
"));
                tempCell += Math.log10(Double.parseDouble(tempStr
ing));
                segmentOfInput[i] = new StringBuilder(String.value
Of(tempCell));
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the exp()
            if (segmentOfInput[i].toString().lastIndexOf("exp") !
= -1) {
                String tempString = segmentOfInput[i].substring(se
gmentOfInput[i].indexOf("(") + 1, segmentOfInput[i].indexOf(")
"));
                tempCell += Math.exp(Double.parseDouble(tempStrin
g));
```

```
                segmentOfInput[i] = new StringBuilder(String.value
Of(tempCell));
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the sqrt()
            if (segmentOfInput[i].toString().lastIndexOf("sqrt")
 != -1) {
                String tempString = segmentOfInput[i].substring(se
gmentOfInput[i].indexOf("(") + 1, segmentOfInput[i].indexOf(")
"));
                tempCell += Math.sqrt(Double.parseDouble(tempStri
ng));
                segmentOfInput[i] = new StringBuilder(String.value
Of(tempCell));
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the arcsin()
            if (segmentOfInput[i].toString().lastIndexOf("arcsin
") != -1) {
                String tempString = segmentOfInput[i].substring(se
gmentOfInput[i].indexOf("(") + 1, segmentOfInput[i].indexOf(")
"));
                tempCell += Math.asin(Double.parseDouble(tempStri
ng));
                segmentOfInput[i] = new StringBuilder(String.value
Of(tempCell));
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the arctan()
            if (segmentOfInput[i].toString().lastIndexOf("arctan
") != -1) {
                String tempString = segmentOfInput[i].substring(se
gmentOfInput[i].indexOf("(") + 1, segmentOfInput[i].indexOf(")
"));
                tempCell += Math.atan(Double.parseDouble(tempStri
ng));
                segmentOfInput[i] = new StringBuilder(String.value
Of(tempCell));
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the sin()
            if (segmentOfInput[i].toString().lastIndexOf("sin") !
= -1) {
                String tempString = segmentOfInput[i].substring(se
gmentOfInput[i].indexOf("(") + 1, segmentOfInput[i].indexOf(")
"));
                tempCell += Math.sin(Double.parseDouble(tempStrin
g));
                segmentOfInput[i] = new StringBuilder(String.value
Of(tempCell));
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the cos()
            if (segmentOfInput[i].toString().lastIndexOf("cos") !
= -1) {
                String tempString = segmentOfInput[i].substring(se
gmentOfInput[i].indexOf("(") + 1, segmentOfInput[i].indexOf(")
"));
```

```
                tempCell  +=  Math.cos(Double.parseDouble(tempStrin
g));
                segmentOfInput[i] = new StringBuilder(String.value
Of(tempCell));
                segmentOfInput  =  Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the tan()
            if (segmentOfInput[i].toString().lastIndexOf("tan") !
= -1) {
                String tempString = segmentOfInput[i].substring(se
gmentOfInput[i].indexOf("(")  +  1,  segmentOfInput[i].indexOf(")
"));
                tempCell  +=  Math.tan(Double.parseDouble(tempStrin
g));
                segmentOfInput[i] = new StringBuilder(String.value
Of(tempCell));
                segmentOfInput  =  Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the factorial
            if (segmentOfInput[i].toString().lastIndexOf("!") !=
-1) {
                String tempString = segmentOfInput[i].substring(0,
 segmentOfInput[i].indexOf("!"));
                tempCell += new Equal().calculateFact(Integer.pars
eInt(tempString));
                segmentOfInput[i] = new StringBuilder(String.value
Of(tempCell));
                segmentOfInput  =  Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }


        }

        // calculate the power
        // We should notice that the "^" operator combining from
right to left
        for (int i = segmentOfInput.length - 1; i > 0; i--) {
            if  (segmentOfInput[i].toString().equals(""))  continu
e;
            tempCell = 0;
            // calculate the power
            if (segmentOfInput[i].toString().equals("^")) {
                tempCell  +=  Math.pow(Double.parseDouble(Equal.del
eteBracket(segmentOfInput[i - 1]).toString()),
                        Double.parseDouble(Equal.deleteBracket(seg
mentOfInput[i + 1]).toString()));
                segmentOfInput[i - 1] = new StringBuilder(String.v
alueOf(tempCell));
                segmentOfInput[i] = segmentOfInput[++i] = new Stri
ngBuilder();
                segmentOfInput  =  Equal.compactStrings(segmentOfIn
put);
                i = segmentOfInput.length - 1;
            }

        }

        // calculate the multiplying and dividing
        for (int i = 0; i < segmentOfInput.length; i++) {
            if  (segmentOfInput[i].toString().equals(""))  continu
e;
            tempCell = 0;
            // calculate the multiplying
            if (segmentOfInput[i].toString().equals("×")) {
                tempCell += Double.parseDouble(Equal.deleteBracke
t(segmentOfInput[i - 1]).toString()) *
                        Double.parseDouble(Equal.deleteBracket(seg
mentOfInput[i + 1]).toString());
                segmentOfInput[i - 1] = new StringBuilder(String.v
alueOf(tempCell));
                segmentOfInput[i] = segmentOfInput[++i] = new Stri
ngBuilder();
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the dividing
            if (segmentOfInput[i].toString().equals("÷")) {
                tempCell += Double.parseDouble(Equal.deleteBracke
t(segmentOfInput[i - 1]).toString()) /
                        (Double.parseDouble(Equal.deleteBracket(se
gmentOfInput[i + 1]).toString()) + 0.0);
                segmentOfInput[i - 1] = new StringBuilder(String.v
alueOf(tempCell));
                segmentOfInput[i] = segmentOfInput[++i] = new Stri
ngBuilder();
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
        }

        // calculate the adding and subtracting
        for (int i = 0; i < segmentOfInput.length; i++) {
            if (segmentOfInput[i].toString().equals("")) continu
e;
```

```
            tempCell = 0;
            // calculate the adding
            if (segmentOfInput[i].toString().equals("+")) {
                tempCell += Double.parseDouble(Equal.deleteBracke
t(segmentOfInput[i - 1]).toString()) +
                        Double.parseDouble(Equal.deleteBracket(seg
mentOfInput[i + 1]).toString());
                segmentOfInput[i + 1] = new StringBuilder(String.v
alueOf(tempCell));
                segmentOfInput[i - 1] = segmentOfInput[i] = new St
ringBuilder();
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the subtracting
            if (segmentOfInput[i].toString().equals("-")) {
                tempCell += Double.parseDouble(Equal.deleteBracke
t(segmentOfInput[i - 1]).toString()) -
                        Double.parseDouble(Equal.deleteBracket(seg
mentOfInput[i + 1]).toString());
                segmentOfInput[i + 1] = new StringBuilder(String.v
alueOf(tempCell));
                segmentOfInput[i - 1] = segmentOfInput[i] = new St
ringBuilder();
                segmentOfInput = Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
            // calculate the Mod
            if (segmentOfInput[i].toString().equals("mod")) {
                tempCell += Double.parseDouble(Equal.deleteBracke
t(segmentOfInput[i - 1]).toString()) %
                        Double.parseDouble(Equal.deleteBracket(seg
mentOfInput[i + 1]).toString());
                segmentOfInput[i + 1] = new StringBuilder(String.v
alueOf(tempCell));
                segmentOfInput[i - 1] = segmentOfInput[i] = new St
ringBuilder();
                segmentOfInput  =  Equal.compactStrings(segmentOfIn
put);
                i = 0;
            }
        }

        segmentOfInput = compactStrings(segmentOfInput);
        // the result will be stored in resultNumber
        double resultNumber = 0;
        // store the result of calculating
        for (StringBuilder strings : segmentOfInput) {
            if (strings.toString().equals("Infinity")) {
                throw new InfinityException();
            }
            if ((!strings.toString().equals("")) && (Character.is
Digit(strings.charAt(0)) || strings.charAt(0) == '-') || string
s.indexOf("(") != -1) {
                if (strings.indexOf("(") != -1) {
                    new Equal();
                    resultNumber  =  Double.parseDouble(deleteBracke
t(strings).toString());
                } else {
                    resultNumber  =  Double.parseDouble(strings.toSt
ring());
                }
            }
        }
        return resultNumber;
    }
    // "delete" the void element in tempStringBuilders and retur
n
    static StringBuilder[] compactStrings(StringBuilder[] tempSt
ringBuilders) {
        int i;
        int j;
        StringBuilder[] newStringBuilders = new StringBuilder[te
mpStringBuilders.length];
        for (i = 0; i < newStringBuilders.length; i++) newStringB
uilders[i] = new StringBuilder();

        // "delete" the void element in tempStringBuilders and re
turn
        for (j = 0, i = 0; j < newStringBuilders.length; j++) {
            if (!tempStringBuilders[j].toString().equals("")) {
                newStringBuilders[i] = tempStringBuilders[j];
                i++;
            }
        }
        return newStringBuilders;
    }

    // Calculate the fact
    private int calculateFact(int n) {

        if (n <= 1) {
            return 1;
        } else {
            return n * calculateFact(n - 1);
        }
    }

    // Elicit the content of bracket
    // 采用反复递归调用的方法来解决这个问题
    // 这里用到的字符串处理方法和 MainGUI 类中的 313 行的方法差不多
    Double elicitBracket(String tempString) throws InfinityExcep
tion {
        char[] input = tempString.toCharArray();
        StringBuilder[] segmentOfInput = new StringBuilder[100];
        for (int i = 0; i < segmentOfInput.length; i++) segmentOf
Input[i] = new StringBuilder();
        // Split the input content to a easily processed one
        // after this for sentence, the input content such as :
        // 1.5+2×3+9÷3 will be stored in the given StringBuilder
collection like the follows
        // "1.5" "+" "2" "×" "3" "+" "9" "÷" "3"
        int inBracket = 0;
        for (int i = 0, j = 0; i < input.length; i++) {
            if (Character.isDigit(input[i]) || input[i] == '.' ||
 input[i] == '!' || input[i] == 'π' || input[i] == 'e' || inBrack
et != 0) {
                if (i >= 1 && !Character.isDigit(input[i - 1]) && i
nput[i - 1] != '(' && input[i - 1] != '.' && input[i] != '!' && i
nBracket == 0)
                    j++;
                tempStringBuilders[j].append(input[i]);
            }
            // store the operator
            else {
                if (i >= 1 && !Character.isLetter(input[i - 1]) &&
 input[i] != ')') j++;
                else if (i >= 1 && (Character.isDigit(input[i - 1])
 || input[i - 1] == 'π' || (input[i - 1] == 'e' && input[i] != 'x
')) && input[i - 1] != ')' && input[i] != ')')
                    j++;
                tempStringBuilders[j].append(input[i]);
            }
            if (input[i] == '(') inBracket++;
            if (input[i] == ')') inBracket--;
        }       return calculateResult(segmentOfInput);
    }

}
```

3. 文件 InfinityException.java 中：

```
package calculator;

// If the num is Infinity, then throw this exception
public class InfinityException extends Exception{
    InfinityException(){
        MainGUI.jTextArea.append(" = Infinity");
    }
}
```

4. 文件 NoCorrespondingBracketException.java 中：

```
package calculator;

public class NoCorrespondingBracketException extends Exception
{
    NoCorrespondingBracketException() {
        MainGUI.jTextArea.append("\nError!!\nCheck the NUMBER of
 \"(\" and \")\" ");
    }
}
```

5. 文件 VoidResultException.java 中：

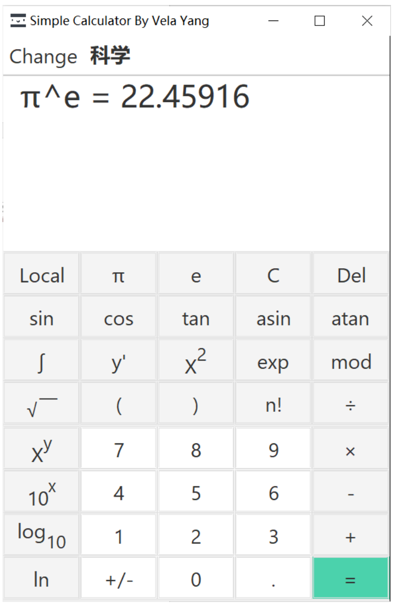
```
package calculator;
import java.io.IOException;
public class VoidResultException extends IOException {
    VoidResultException() {
        MainGUI.jTextArea.append("\nCan't get result, try to rear
range the INPUT content");
    }
}
```

6. 修改过的 WAEngine 中：

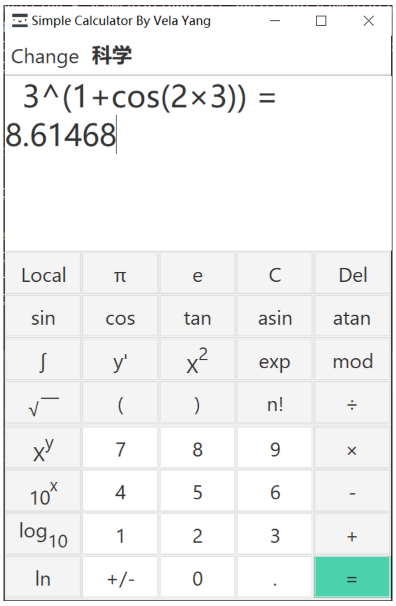
```
// Edited By Vela Yang
    public String toURL(WAQuery query, String accessType) {
        this.path += accessType;
        return "http://" + server + path + "?" + "appid=" + appid
+ query + "&fontsize=18";
    }
```

（3）实验代码、过程、相应结果（截图）并对实验进行说明和分析：

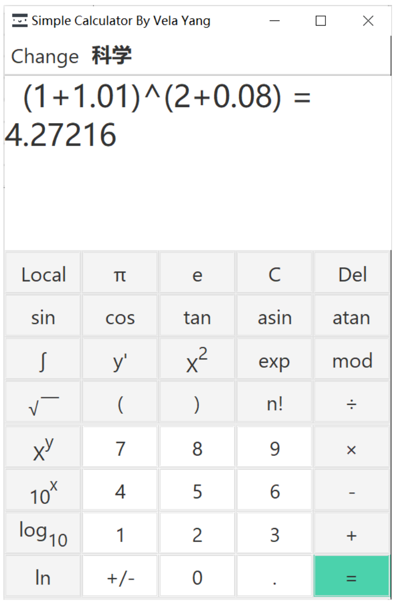
1. 科学计算器，Local 模式：


Simple Calculator By Vela Yang
Change 科学

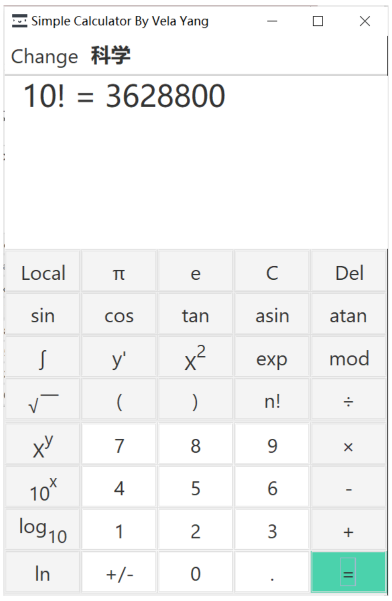
```
π^e = 22.45916
```


Simple Calculator By Vela Yang
Change 科学

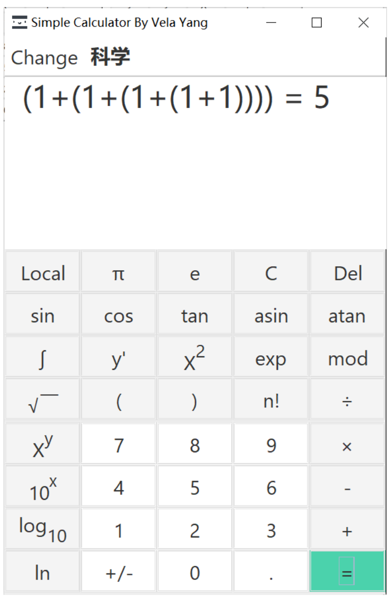
```
3^(1+cos(2×3)) =
8.61468
```


Simple Calculator By Vela Yang
Change 科学

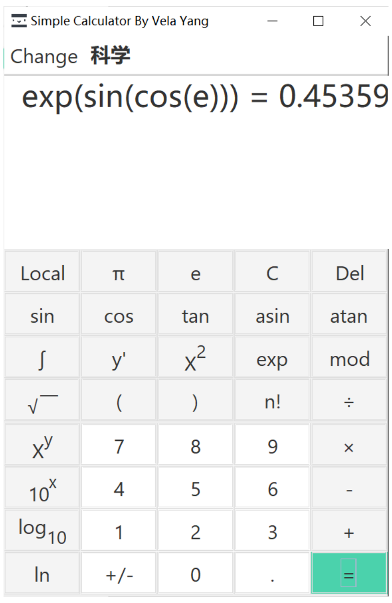
```
(1+1.01)^(2+0.08) =
4.27216
```


Simple Calculator By Vela Yang
Change 科学

```
10! = 3628800
```


Simple Calculator By Vela Yang
Change 科学

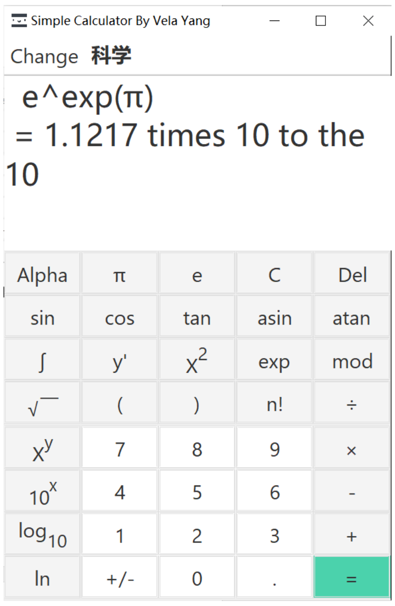
```
(1+(1+(1+(1+1)))) = 5
```


Simple Calculator By Vela Yang
Change 科学

```
exp(sin(cos(e))) = 0.45359
```

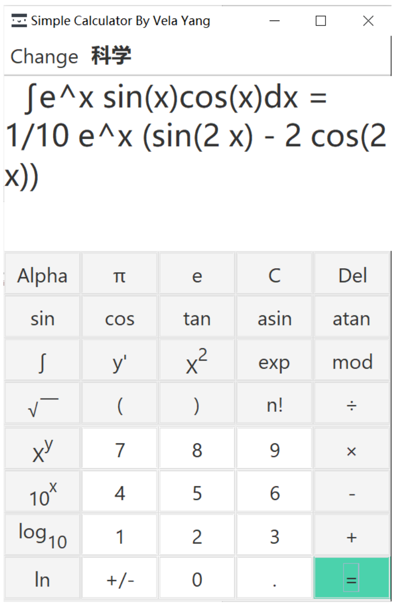
以上结果均经过验证，结果正确。可以得到，我的计算器，由于末尾使用了整数的判断，所以输出结果时整数就是整数，1 就是 1，不是什么 1.0 之类的。考虑到 double 类型的计算可能产生的误差，只取了小数点之后 5 位，保证精确性。

2. 科学计算器，Alpha 模式：


Simple Calculator By Vela Yang

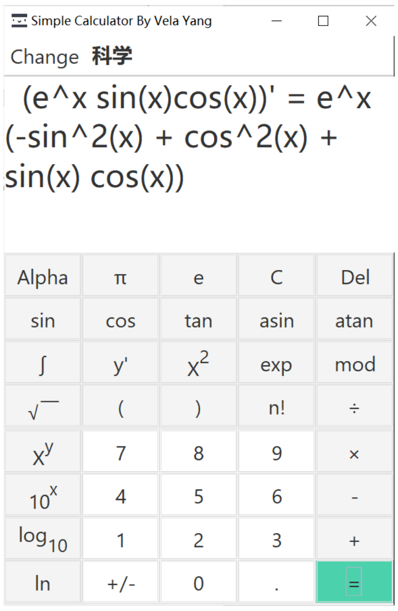
Change 科学

e^exp(π)
= 1.1217 times 10 to the 10


Simple Calculator By Vela Yang

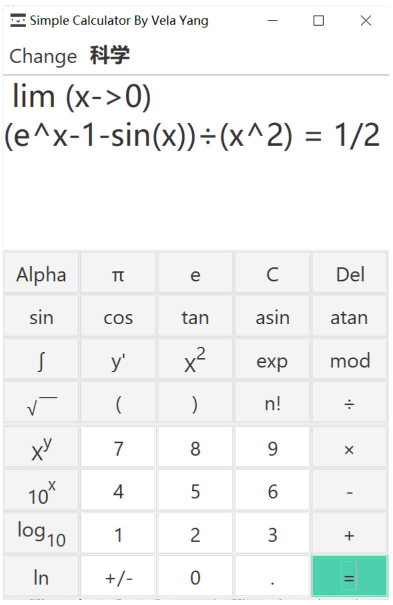
Change 科学

∫e^x sin(x)cos(x)dx = 1/10 e^x (sin(2 x) - 2 cos(2 x))


Simple Calculator By Vela Yang

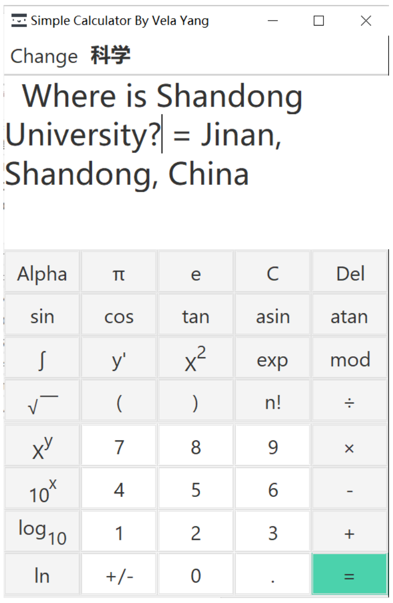
Change 科学

(e^x sin(x)cos(x))' = e^x (-sin^2(x) + cos^2(x) + sin(x) cos(x))


Simple Calculator By Vela Yang

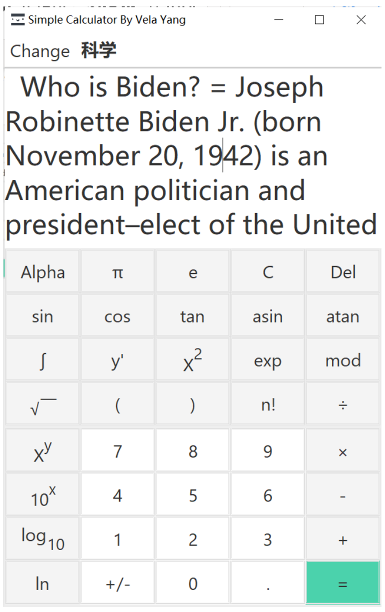
Change 科学

lim (x->0) (e^x-1-sin(x))÷(x^2) = 1/2


Simple Calculator By Vela Yang

Change 科学

Where is Shandong University? = Jinan, Shandong, China

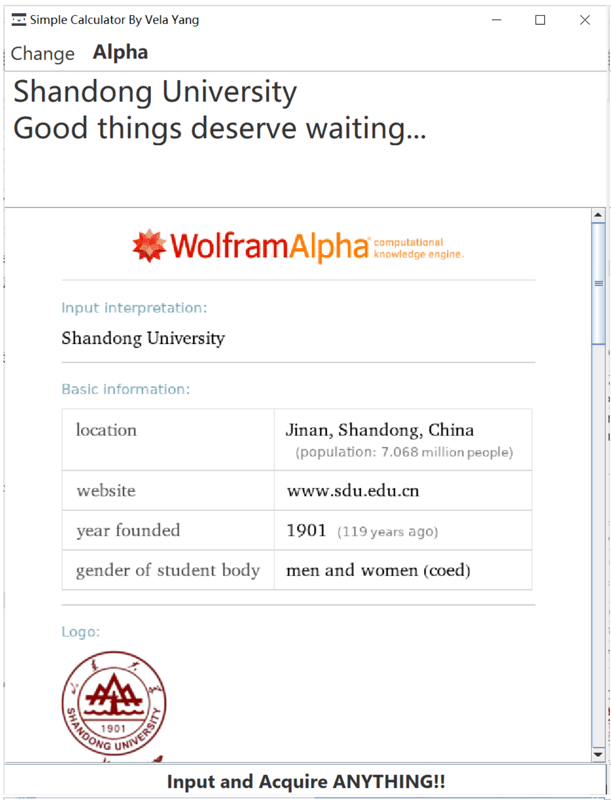

Simple Calculator By Vela Yang

Change 科学

Who is Biden? = Joseph Robinette Biden Jr. (born November 20, 1942) is an American politician and president–elect of the United

可以看到 Alpha 模式具有极其丰富的功能，并且服务器响应快(大都在 1s 中之内)，于是这是我计算器的默认模式，当用户离线时可以轻松切换成 Local 本地计算模式计算所需要的算式。

3. 探索模块(点击左上角 Change 即可切换):


Simple Calculator By Vela Yang

Change Alpha

Shandong University
Good things deserve waiting...

WolframAlpha computational knowledge engine.

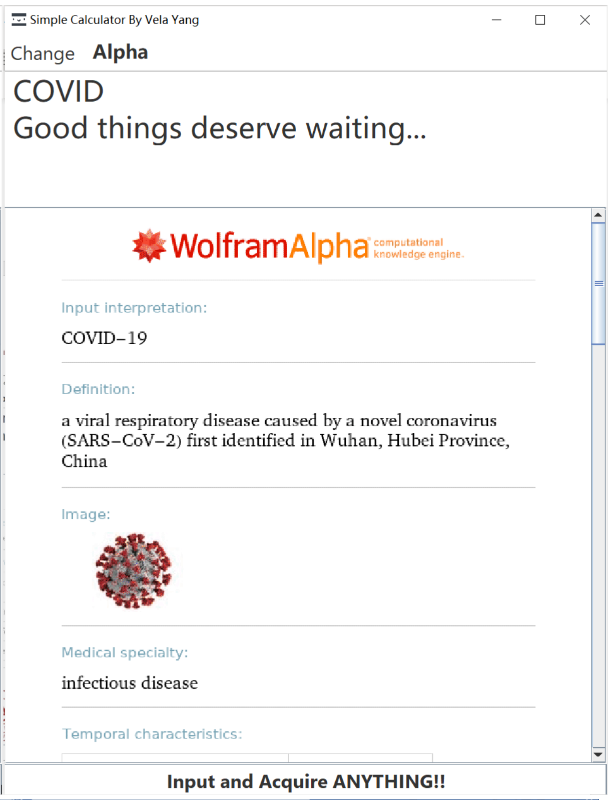
Input interpretation:
Shandong University

Basic information:

location	Jinan, Shandong, China (population: 7.068 million people)
website	www.sdu.edu.cn
year founded	1901 (119 years ago)
gender of student body	men and women (coed)

Logo:

Input and Acquire ANYTHING!!


Simple Calculator By Vela Yang

Change Alpha

COVID
Good things deserve waiting...

WolframAlpha computational knowledge engine.

Input interpretation:
COVID−19

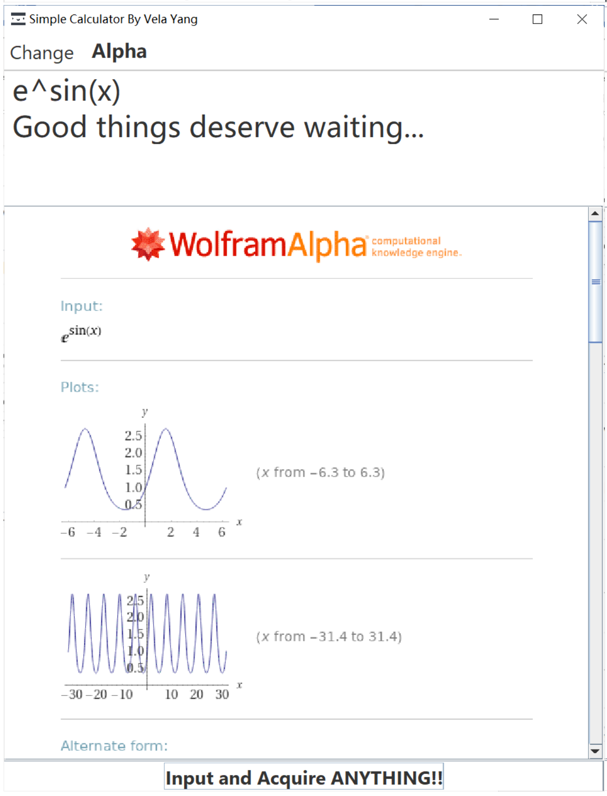
Definition:
a viral respiratory disease caused by a novel coronavirus (SARS−CoV−2) first identified in Wuhan, Hubei Province, China

Image:

Medical specialty:
infectious disease

Temporal characteristics:

Input and Acquire ANYTHING!!


Simple Calculator By Vela Yang

Change Alpha

e^sin(x)
Good things deserve waiting...

WolframAlpha computational knowledge engine.

Input:
$e^{\sin(x)}$

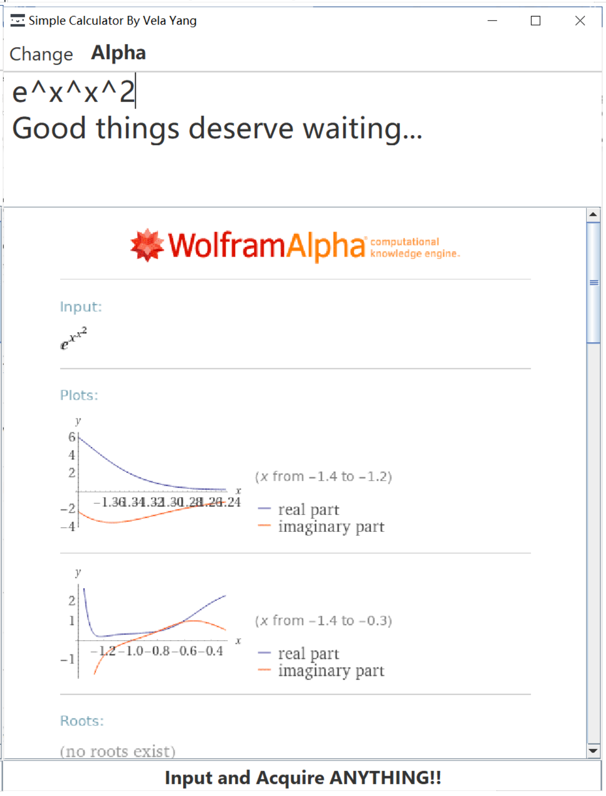
Plots:

(x from −6.3 to 6.3)

(x from −31.4 to 31.4)

Alternate form:

Input and Acquire ANYTHING!!


Simple Calculator By Vela Yang

Change Alpha

e^x^x^2
Good things deserve waiting...

WolframAlpha computational knowledge engine.

Input:
$e^{x^{x^2}}$

Plots:

(x from −1.4 to −1.2)
real part
imaginary part

(x from −1.4 to −0.3)
real part
imaginary part

Roots:
(no roots exist)

Input and Acquire ANYTHING!!

可以看到 WolframAlpha 的强大之处，不仅借此实现了高清函数图像的显示，并且还在我的计算器中相当于嵌入了一个“知识搜索引擎”，让用户可以了解很多很多事情，上到天文，下到地理，可谓无所不能。（需要等待约 5s）

【实验心得】

为了打造出这个计算器，我利用了很多在 Java 课上学习到的数据，例如 GUI 设计、权限修饰符、匿名类、static 修饰符、异常抛出、自定义异常、枚举类型、多线程(MainGUI 继承 Runnable 接口)等等。

首先就是算式的处理，毕竟这是计算器程序的核心。网上有很多高级的方法来处理，例如正则表达式和栈的联合使用。但作为一位 Java 初学者，我想自己造个轮子，何况也听说过那句话“初学编程语言，不要害怕自己造轮子”。不过，这轮子一造起来就停不下了，花了我将近一周的时间才得以完成。不过，最终事实证明，我成功造出来了属于自己的轮子，而且性能很不错，让我很开心（嘿嘿），并且，私以为使用递归来处理括号是一种很巧妙的方法。不过需要承认的是，If 语句判断的还是比较复杂的，可能也会让别人头疼，不过不要紧，我的最根本的思路就是把数字和操作符分开来处理，就是这么简单，但实现起来稍微复杂了一点。此外，我发现我自己造的小轮子实现了时间复杂度 $O(n)$，和别人使用波兰表达式等高级方法的结果一样，也让我感到欣慰，自己造的轮子也不差嘛！

接着是程序结构，刚开始学习 Java 时，老师就跟我们说要重视程序的结构规范。于是我写代码时也时不时检查下是不是方便后面人员调试，就比如这个主界面的 Button，我没有使用数组，而是先用语句块标记出是哪一行的按钮，然后再一个一个地创建、添加监听按钮，而且每个 Button 都见名知义，我认为，这样对后面人员的维护、修改来说十分方便，就比如说修改一个 Button，直接在对应的地方修改就可以了，不用去根据数组来计算它所在的位置，然后又在数组的范围内慢慢修改、调试。

其次是 WolframALpha API 的调用，这个只有英文版的介绍和文档，访问 API 时有需要用我不会的 URL 解码，就很难受，不过好在仔细琢磨了 Demo 和 WolframAlpha API 官网 https://products.wolframalpha.com/api/ 之后，也学会了使用它的方法，但是此时得到的东西又很复杂，然后发现调用 simple API 获取图片是一种比较好的方法，又查了很多资料（感谢 StackOverflow 这个网站，太赞了）发现了 ImageIO 中的从网址读取图片的方法，成功将 WolframAlpha 完美移植到我的计算器中（需要说明，我使用 API 中主要就是其中 WolframAlpha 的 url 编码方法，自己也修改了这一部分，使其更适合我的程序），哈哈，真的开心。

再其次是 GUI，为此，我恶补了很多关于 Java GUI 的知识，计算器中要做出这么“复杂”（对我来说，哈哈）的界面，需要用到多种布局相互嵌套，例如，我的 Button 使用的是 4×5 GridLayout 布局，而两个 Button Panel 和 TextArea 又是使用的 3×1 GridLayout 布局；探索界面使用的是 GridBagLayout 布局，而在计算器和探索界面之间又是在 container 之内的 CardLayout 布局以便于切换，总体来说还是有点复杂，占用内存也比较大，但好处是切换快，不用每次都重新加载。

设计完 GUI，这 Jdk 生成的也太难看了，字体全都模模糊糊的，很影响体验。所以又花了大概一下午时间查找 Java 怎样支持高 DPI，结果没啥成果。再次感谢万能的 StackOverflow，在上面我发现了 Java 从 Java 9 开始支持了高 DPI，效果还很不错，比 Java 8 生成的模模糊糊的界面好很多，也美观很多，字体清晰锐利。

然后是权限修饰符，其实这个还算比较简单吧，主要就是分清楚哪些东西需要让外部类来改变，哪些不需要，哪些不能在外部改变（例如最终结果）等。

接下来是 static 修饰符，想必老师刚开始打开我的程序的时候也会想我怎么会用了这么多的 static 修饰符，从内存优化的角度来说，太多的 static 变量和方法是让人忌讳的；从程序设计的角度来说，全局变量不利于调试。所以，我在使用 static 变量时，也会考虑到这些问题，我也对不需要 static 修饰的变量进行了处理，节省内存。在这个计算器程序中，我所使用 static 变量的特征如下：

1. 在内存中(Data Segment)不经常改变的，且长存于内存之中(例如 CardLayout 和其中容器的组件)
2. 声明后大多为 final 型变量，不会随处更改它的内容。
3. 方便添加监听函数时的匿名类的重写方法中对 MainGUI 中变量的调用。
4. 有些需要在匿名类重写方法中使用的变量，特别是枚举类型变量。
5. 实现 MainGUI.run()方法中便于使用的变量（毕竟存放在 Data Segment 内）。

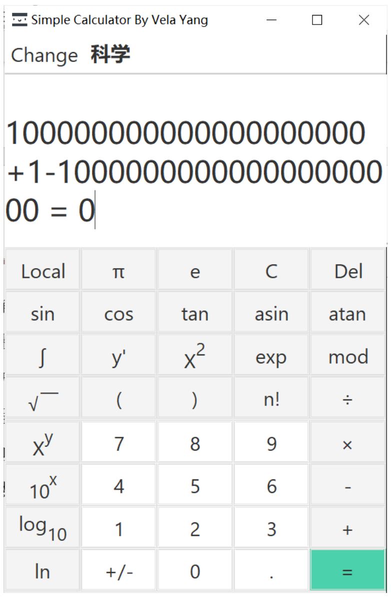
应该来说，我使用 static 型变量的理由是比较充分的。

关于异常抛出和自定义异常，其实我写的也不是很多，发现这玩意确实能很好的帮助我 Debug，确实是一个很好的东西，但还没能实现想系统中常见异常的那种方式（不过好像调用父类的 printStackTrace 方法就行）

枚举类型还是很方便的一个东西，主要就是因为是自己定义的，作用清晰明了，而且目的特别明确，使用起来也挺方便，和自定义异常一样，是个很好的东西。

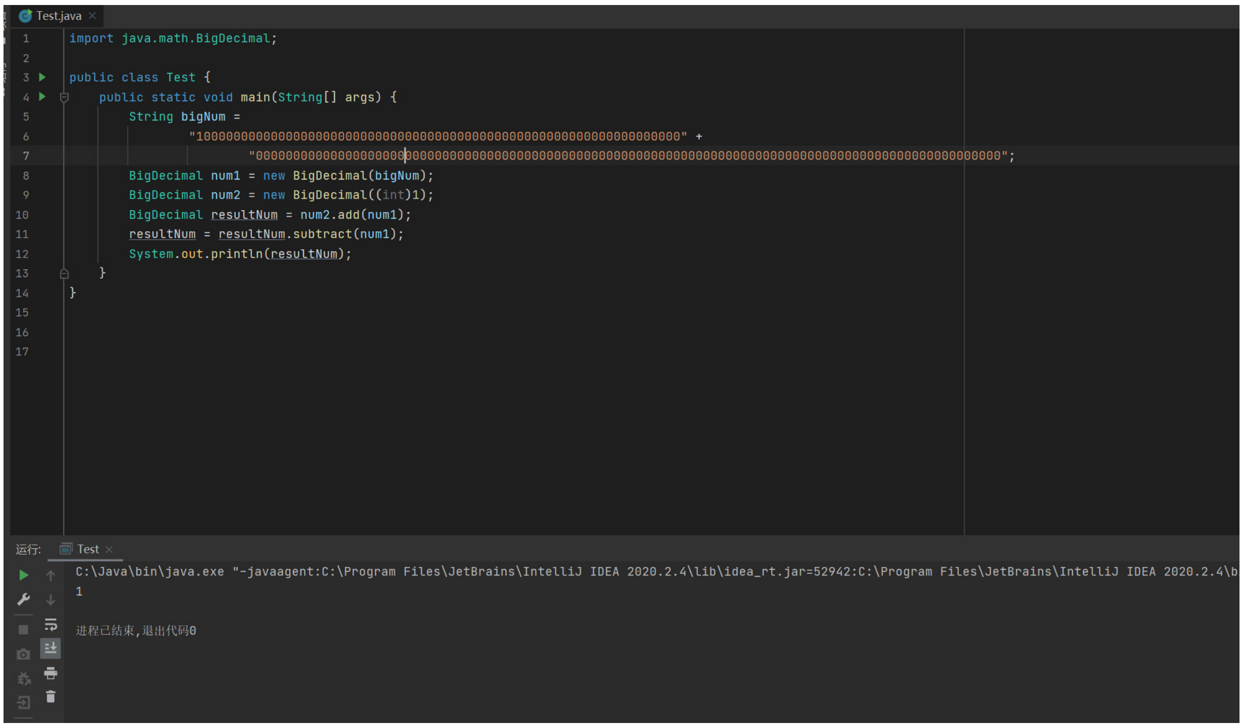
多线程在GUI设计中的重要性不言而喻,应该没有用户会想在程序处理的时候卡住,无法操作吧，这里我使用的是 MainGUI 继承 Runnable 接口并实现 run 方法的方式来实现了多线程，让程序更加有活力。也让用户省心，不会在 Query 结果时卡住。也使用了 synchronized 修饰符保护线程（不过这个程序似乎也不应该一次运行太多个线程）。

当然，该计算器也不是十全十美的，例如由于 Double 类型固有的计算误差，虽然在输出时已经取前八位来保持精度，但在大数处理时会产生可能比较大的误差。例如下面经典的体现浮点数计算误差的结果：



```
100000000000000000000
+1-10000000000000000000
00 = 0
```

不过也有一种解决的方法，就是使用 BigDecimal 类，如下图所示：



```
import java.math.BigDecimal;

public class Test {
    public static void main(String[] args) {
        String bigNum =
                "100000000000000000000000000000000000000000000000000000000000000000" +
                        "00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
        BigDecimal num1 = new BigDecimal(bigNum);
        BigDecimal num2 = new BigDecimal((int)1);
        BigDecimal resultNum = num2.add(num1);
        resultNum = resultNum.subtract(num1);
        System.out.println(resultNum);
    }
}
```

```
C:\Java\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2020.2.4\lib\idea_rt.jar=52942:C:\Program Files\JetBrains\IntelliJ IDEA 2020.2.4\b
1

进程已结束,退出代码0
```

但由于 Math 中的许多方法（如 Math.sin()、Math.cos()等）均不支持 BigDecimal 类，抑或是需要转换成有损的 Double 类型然后再计算，于是我没有使用 BigDecimal 类。

总的来说，本次计算器实验我采用自己的算法，自己造轮子解决了计算算式的问题，成就感还挺强的，而且使用了很多老师教我们、推荐我们使用的东西，也很感谢老师的敦敦教诲。