Java 编写简单计算器

目录

请用 Word 打开 用 WPS 打开会导致部分乱码

1. 系统简介

1.1 设计背景

在当今社会，随着科技的迅速发展，从学生到专业人士，每个人都面临着日常生活和工作中进行快速而准确计算的需求。常规计算器虽然能满足基本需求，但在处理复杂运算和提供额外功能方面往往显得不够高效。因此，开发一个具备更多功能和更高效率的计算器应用变得尤为重要。

利用 Java 语言的跨平台特性和丰富的库支持，可以实现高效计算器的设计。我设计了这款简易计算器，它不仅支持基本的加、减、乘、除四则运算，还包括清除功能、平方与开平方等额外功能。这些功能旨在提供更为全面的计算服务，满足用户在更广泛场景下的计算需求。通过图形用户界面（GUI），该计算器提供了一个直观易用的操作界面，使得用户能够快速上手并有效执行各种计算任务。

1.2 开发工具及环境

(1) 开发工具及介绍

IntelliJ IDEA 是一个智能的 Java 集成开发环境（IDE），提供了强大的代码编辑、调试和部署功能。作为一个开放源代码项目，IntelliJ IDEA 专注于为开发者提供高效的工具和平台，使他们能够快速、轻松地开发各种类型的应用程序。IntelliJ IDEA 的特点包括智能代码提示、代码重构、版本控制集成等。

(2) 开发环境

操作系统：Windows 11

IDE：IntelliJ IDEA Community Edition 2024.1.1

JDK 版本：Azul Zulu 13.0.14

以上是我所选择的开发环境，它们将为我提供一个稳定、高效的开发平台，以便于开发出高质量的计算器应用程序。

需要注意代码中有些库在高版本的 JDK 中已经不适配，如果需要运行代码，需要下载对应版本的 JDK。

2. 系统分析与设计

2.1 设计目的

本项目旨在使用 Java Swing 的 GUI 图形用户界面编程，设计并实现一个简易计算器程序。用户可以通过鼠标输入数值和运算符，实现简单的四则运算，如加、减、乘、除。此外，该计算器还提供了平方和开平方功能，以支持更广泛的数学计算需求，使得用户能够处理更复杂的运算任务，从而增强计算器的实用性和功能性。

2.2 功能需求

该计算器程序具有以下功能：

⑴用户可以通过鼠标或键盘输入数字、运算符，进行加、减、乘、除等混合运算。

⑵提供加、减、乘、除四则运算功能。

⑶提供清除单个数字和清除全部数字的功能。

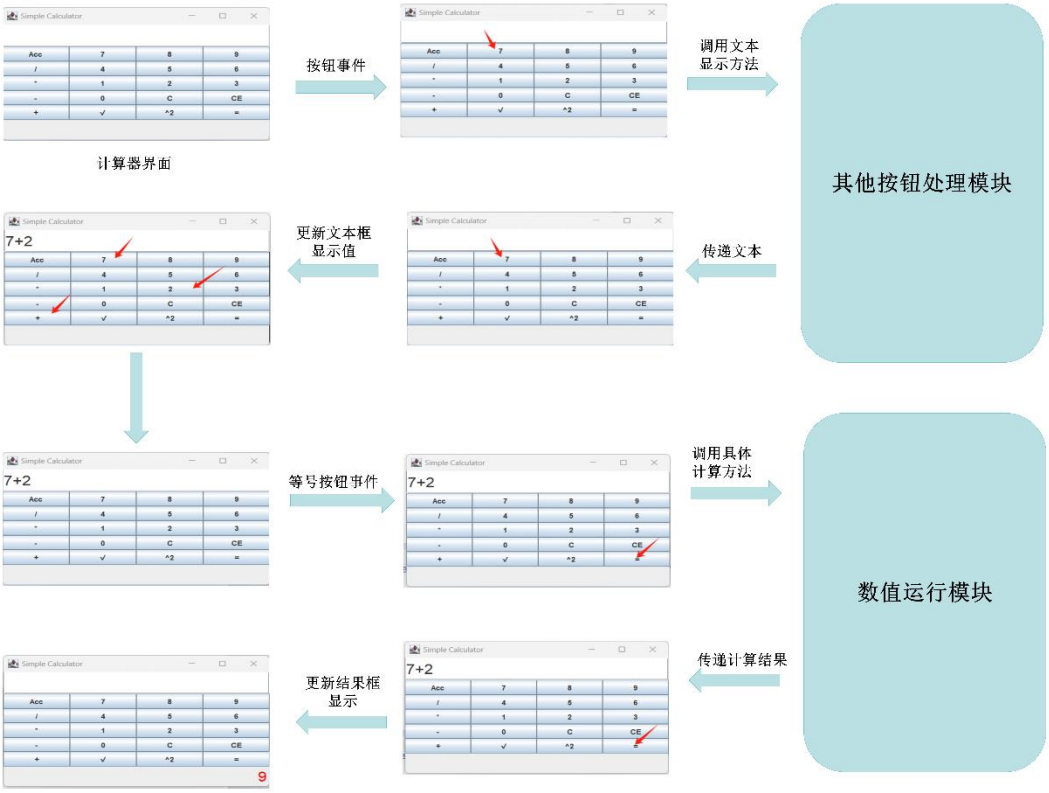
⑷提供开平方和平方功能。

⑸结果保留四位小数。

⑹特别地，添加了 Acc 功能，该功能允许用户调用上一次计算的结果，便于进行连续计算，增强了计算器的实用性和灵活性。

2.3 系统运行结构图

简易图形化界面计算器的运行结构图如下：



2.4 系统功能代码设计

2.4.1 包和类的说明

在这个简易计算器项目中，我主要使用了以下 Java 类和包来实现所需的功能：

包说明：

默认包：在这个项目中，所有的类都位于默认包下，简化了包的管理和类之间的交互。

类说明：

Calculator：这是主类，继承自 JFrame 并实现了 ActionListener 接口。该类负责创建和管理计算器的图形用户界面（GUI），处理事件，并执行相关的数学运算。

主要成员变量：

buttons：一个 JButton 类型的数组，存储所有计算器上的按钮。

expressionBuilder：一个 StringBuilder 对象，用于构建和存储用户输入的表达式。

history：用于记录上一次计算的结果，便于再次使用。

showResult：布尔值，用于控制是否在文本字段中显示计算结果。

previousResult：双精度浮点数，存储上一次计算的结果。

expressionField：JTextField 对象，用于显示用户输入的表达式。

resultField：JTextField 对象，设置为只读，用于显示计算结果。

df：DecimalFormat 对象，用于格式化数字，以保留四位小数。

主要方法：

Calculator()：构造函数，用于初始化计算器的界面和各组件。

actionPerformed(ActionEvent e)：实现了 ActionListener 接口的方法，用于处理来自按钮的动作事件。

updateDisplay()：更新计算器的显示字段，根据当前状态显示表达式或结果。

evaluateExpression(String expression)：解析和计算用户输入的数学表达式，并返回结果。

2.4.2 简易计算器的源代码清单

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Stack;
import java.text.DecimalFormat;

public class Calculator extends JFrame implements ActionListener {
    // 界面组件
    private JButton[] buttons;
    private JTextField expressionField;
    private JTextField resultField;

    // 内部变量
    private StringBuilder expressionBuilder;
```

```
    private String history;
    private boolean showResult;
    private double previousResult;
    private DecimalFormat df;

    // 构造函数
    public Calculator() {
        // 设置窗体属性
        setTitle("Simple Calculator");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // 创建按钮面板
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(5, 4));

        // 初始化按钮并添加事件监听器
        String[] buttonLabels = {"Acc", "7", "8", "9", "/", "4", "5", "6", "*", "1", "2", "3", "-", "0",
"C", "CE", "+", "√", "^2", "="};
        buttons = new JButton[buttonLabels.length];
        for (int i = 0; i < buttonLabels.length; i++) {
            buttons[i] = new JButton(buttonLabels[i]);
            buttons[i].addActionListener(this);
            panel.add(buttons[i]);
        }
        add(panel, BorderLayout.CENTER);

        // 初始化表达式和结果文本框
        expressionField = new JTextField(20);
        expressionField.setPreferredSize(new Dimension(20, 40));
        expressionField.setFont(new Font(expressionField.getFont().getName(), Font.PLAIN, 24));
        resultField = new JTextField(20);
        resultField.setEditable(false);
        resultField.setForeground(Color.RED);
        resultField.setFont(new Font(resultField.getFont().getName(), Font.PLAIN, 24));
        resultField.setHorizontalAlignment(SwingConstants.RIGHT);
        add(expressionField, BorderLayout.NORTH);
        add(resultField, BorderLayout.SOUTH);

        // 设置窗体可见
        setVisible(true);
        pack();

        // 初始化内部变量
```

```
            showResult = false;
            expressionBuilder = new StringBuilder();
            history = "0";
            previousResult = 0;
            df = new DecimalFormat("#.####");
        }

        // 事件监听器实现
        @Override
        public void actionPerformed(ActionEvent e) {
            // 处理按钮点击事件
            JButton source = (JButton) e.getSource();
            String buttonText = source.getText();
            // 省略其他逻辑...
        }

        // 其他方法实现...

        // 主程序入口
        public static void main(String[] args) {
            new Calculator();
        }
}
```

以上是简易计算器的源代码清单，包括界面组件的初始化、事件监听器的实现以及主程序的入口。

3. 系统调试

3.1 编写源程序界面

源程序界面是基于 Java Swing 开发的，设计为用户友好的图形用户界面。界面包括多个按钮，分别表示数字键、运算符、以及特殊功能键（如清除和历史计算结果）。每个按钮的功能都与传统计算器相对应，使得用户能够直观地进行计算操作。

3.2 测试用例

为了确保计算器程序的准确性和可靠性，设计了以下测试用例：

基本运算测试： 测试加、减、乘、除运算，确保输出正确。

边界条件测试：包括除数为零的情况，验证程序是否能正确处理异常。

连续计算测试：进行一系列运算，确保计算器的连续运算能力。

特殊功能测试：测试清除功能、平方和开平方功能，确保它们的正确实现。

3.3 运行结果

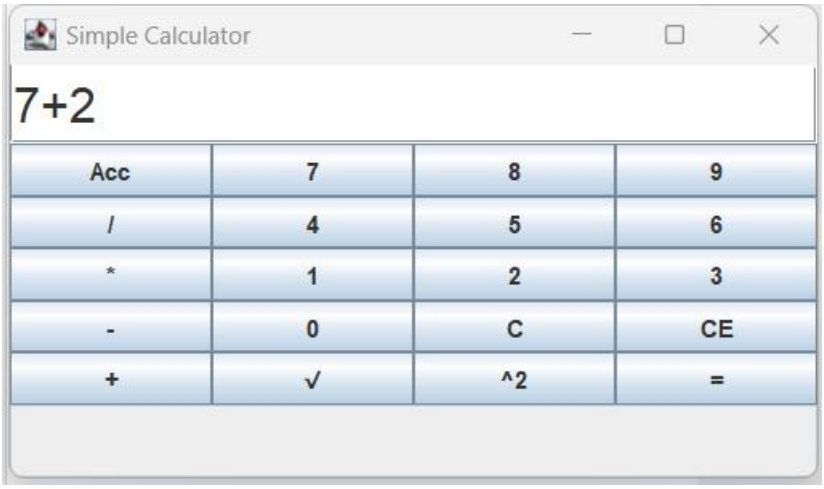
运行测试用例后，计算器应用基本能够正确执行预定功能。特别是处理边界条件和特殊功能时，程序能够正确响应，没有发生错误或异常。但是对于少量极端测试，计算器仍有改进空间。

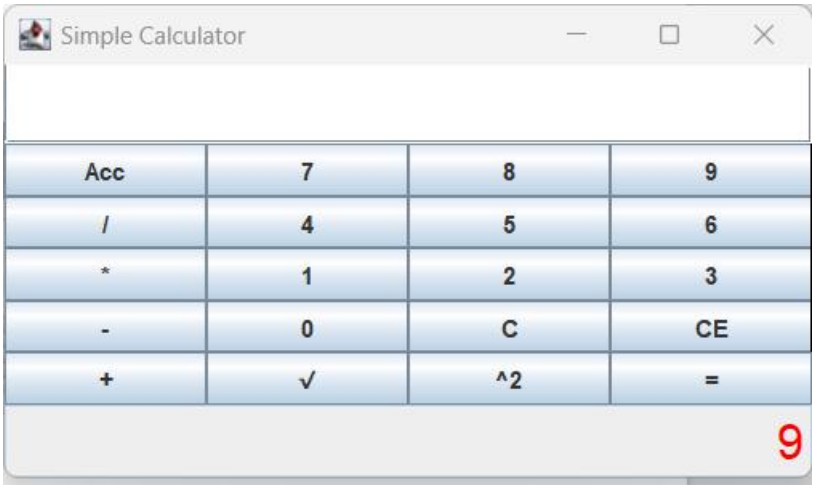
运算结果附图

测试用例 1：7+2=

测试目的：检查计算器是否能实现基本逻辑运算

测试结果：



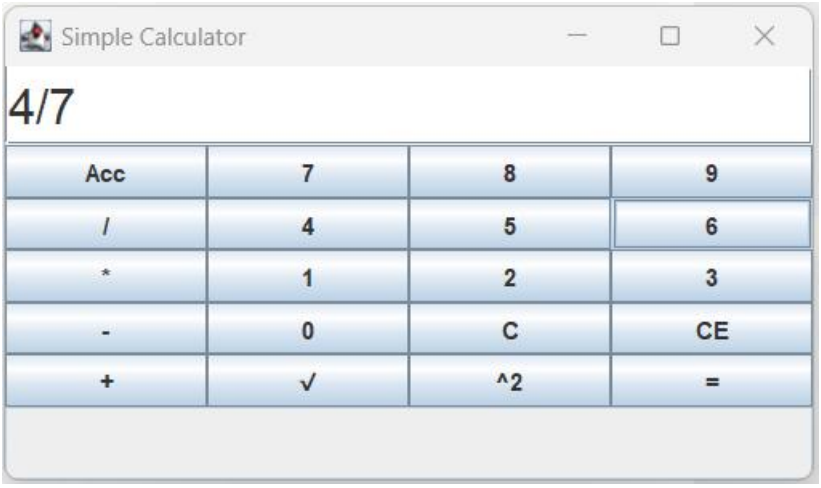


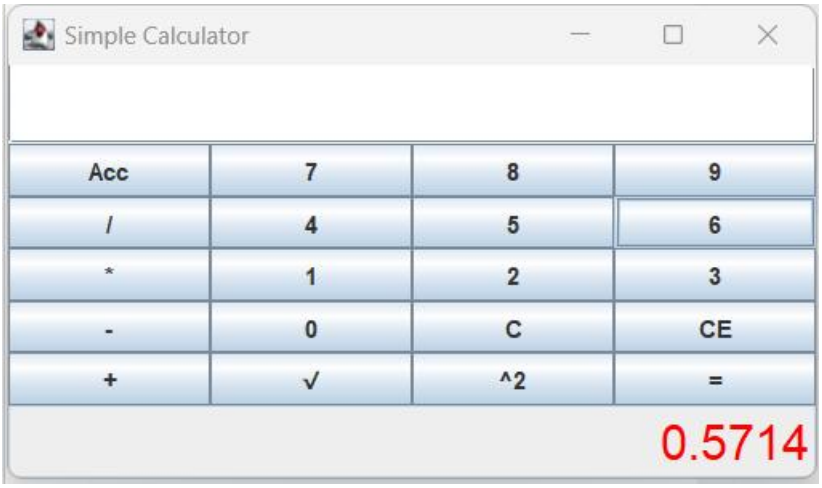
符合预期

测试用例 2：4/7=

测试目的：检查计算器是否能实现基本逻辑运算，并且保留四位有效数字

测试结果：



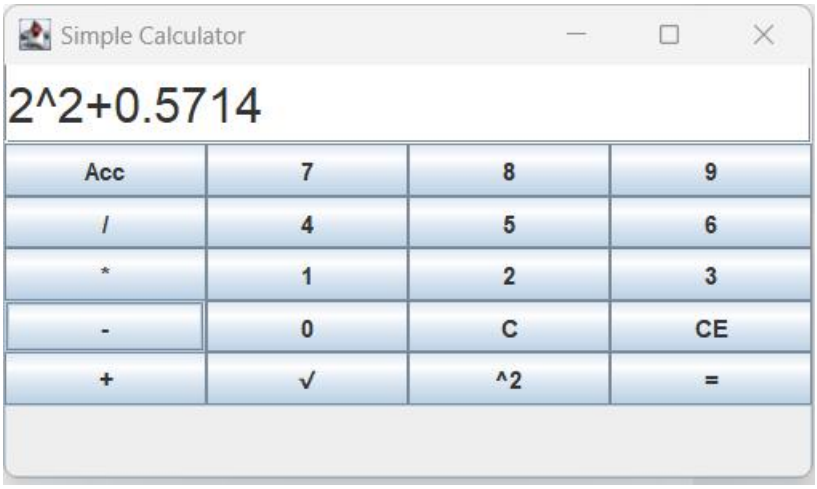


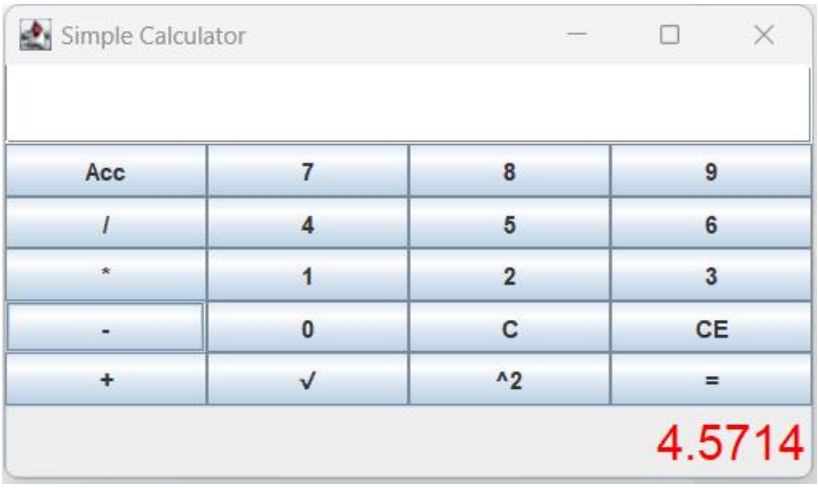
符合预期

测试用例 3：2^2+Acc=

测试目的：检查计算器是否能实现平方运算，Acc 能否正确记录上一次的结果

测试结果：



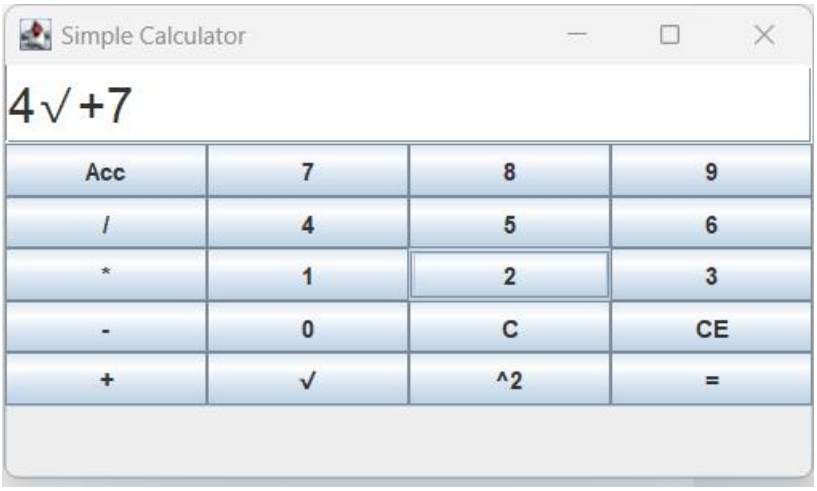


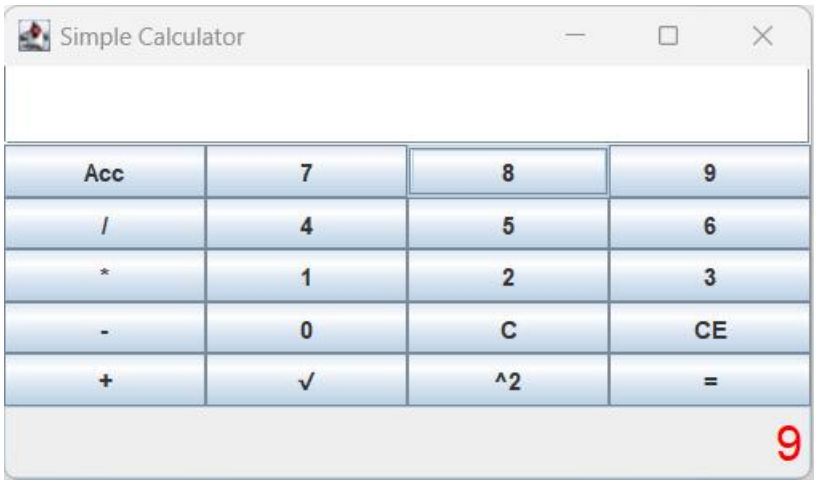
符合预期

测试用例 4：√4+7=

测试目的：检查计算器是否能实现开方运算

测试结果：



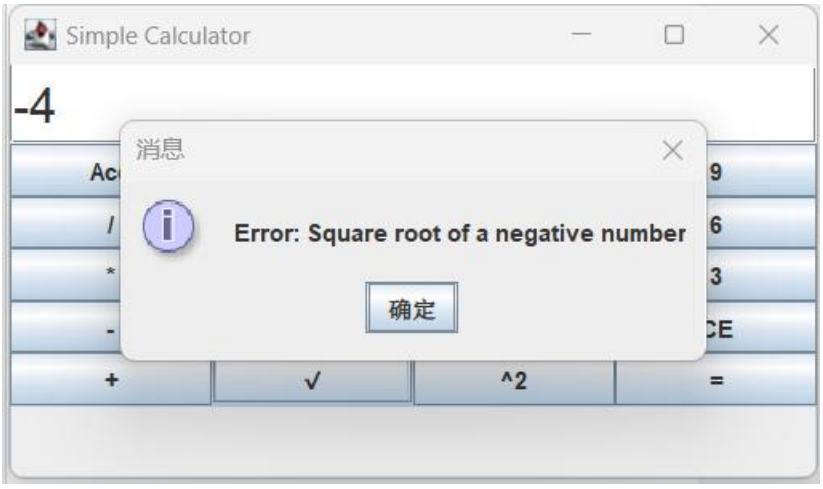


大体符合预期，但是根号输入在数值之后，可能与书写习惯不符，后续可以考虑改变根号处理逻辑

测试用例 5：√-4=

测试目的：检查计算器是否能实现开方运算，对于负数开方是否能判定报错

测试结果：

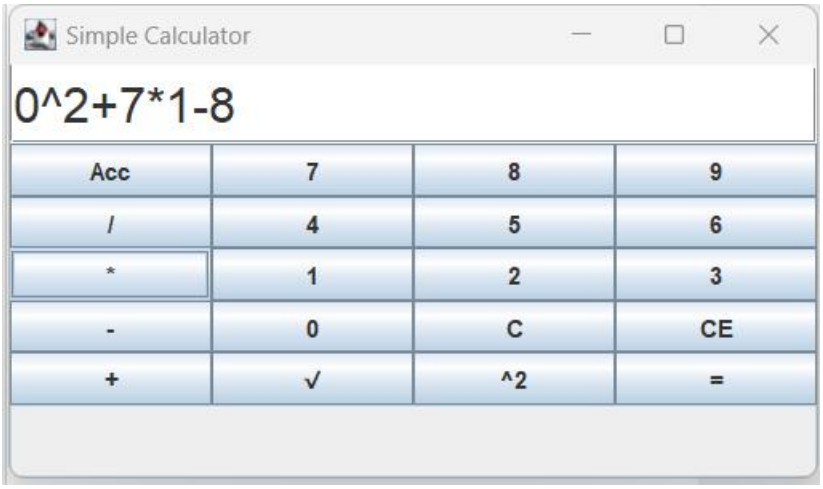


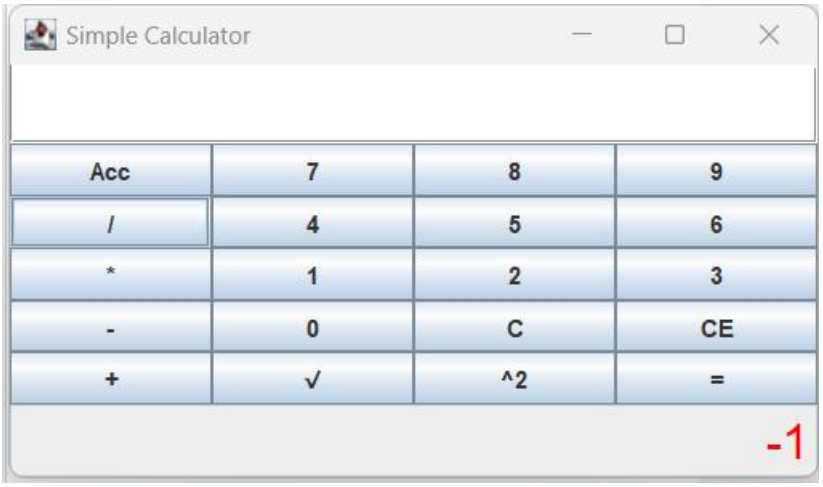
符合预期

测试用例 6：0^2+7*1-8=

测试目的：检查计算器是否能实现复杂逻辑运算，涉及到不同优先级的顺序，能否正确处理

测试结果：



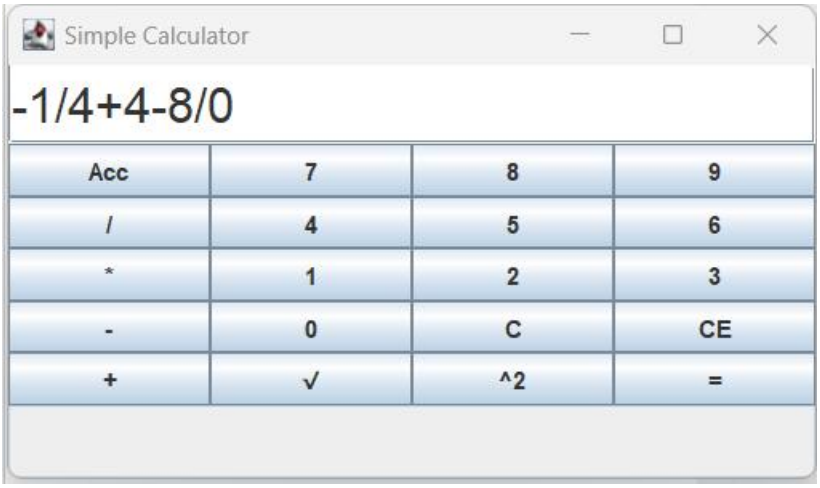


符合预期

测试用例 7：-1/4+4-8/0=

测试目的：检查计算器是否能处理第一个为负数的情况

测试结果：

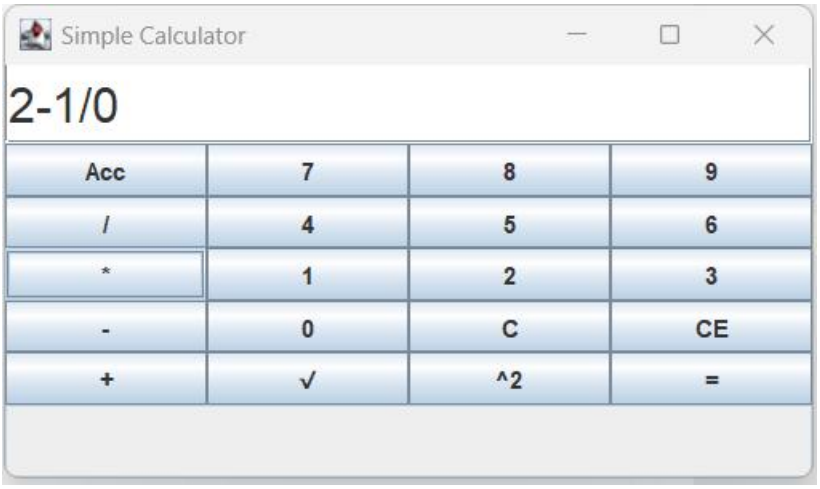


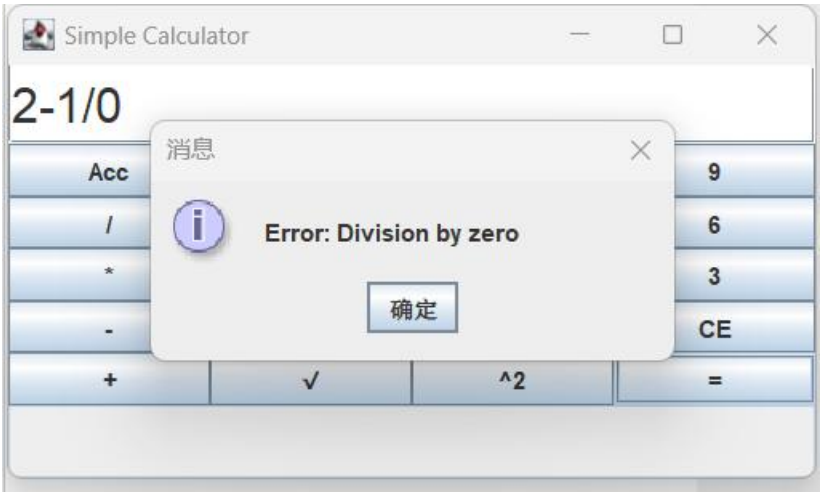
未能实现预期效果，程序没有报错也没崩溃，之后可以考虑在情况文本框之后添加一个默认的 0 来解决第一个数字就是负数导致的无法处理问题

测试用例 8：2-1/0=

测试目的：检查计算器是否能处理除 0 的情况并报错

测试结果：



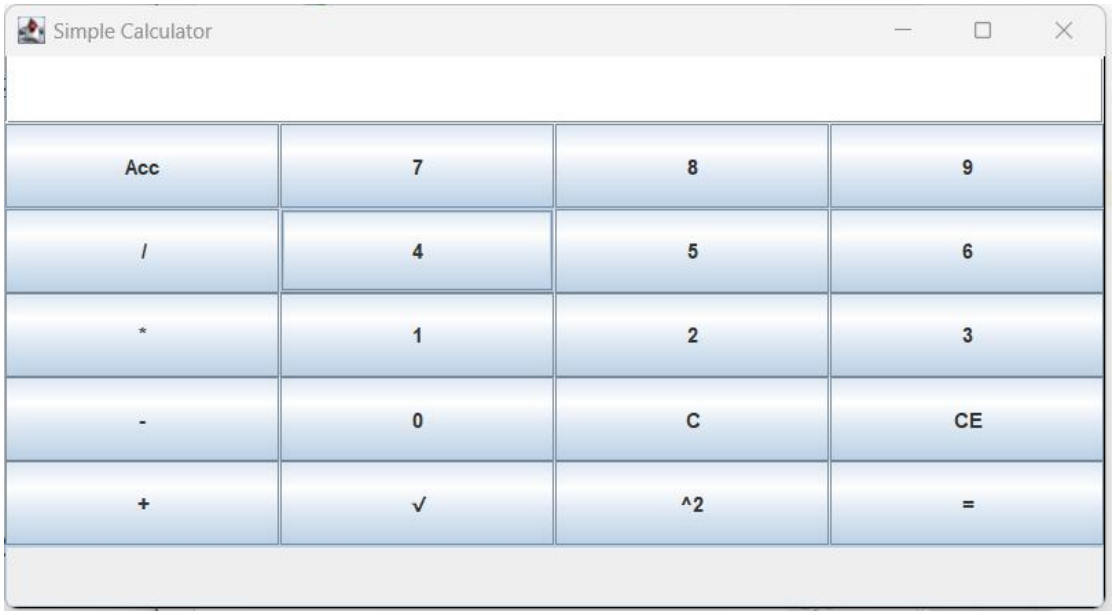


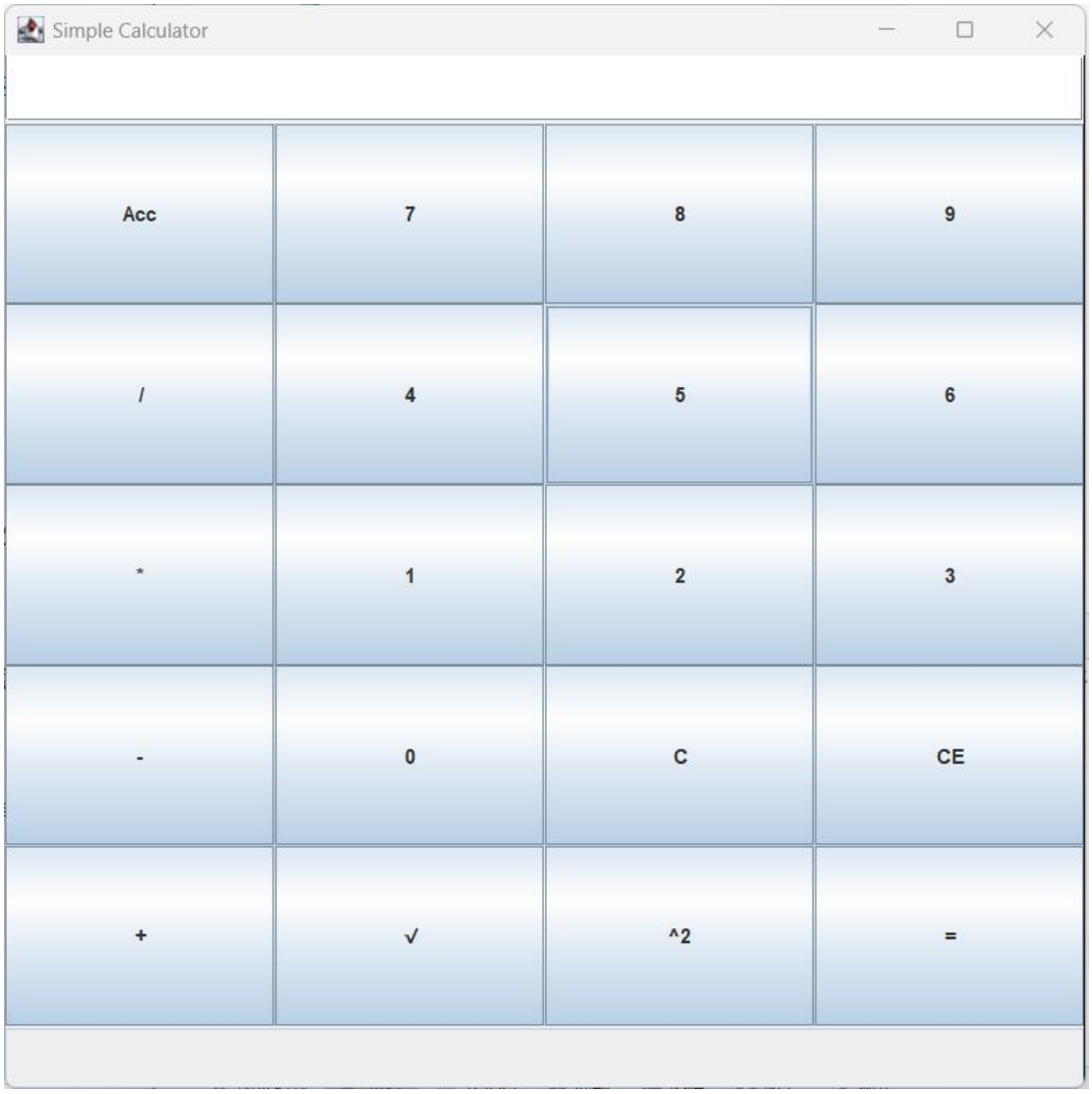
符合预期

测试用例 9：拉动计算器界面

测试目的：检查计算器是否能够实现界面缩放时，所有组件同步缩放，保持界面整体风格不变。

测试结果：





基本符合预期，按钮都同步缩放了，文本框个人认为可以不缩放，因为把字体大小设置固定了，当然后续也可以考虑修改。

4. 设计总结

在这次课程设计中，我深刻体会到了综合运用所学知识来发现、分析、解决实际问题的重要性。这不仅是对我的编程技能的一次锻炼，也是对我的问题解决能力的一次考验。通过这次实践，我更加理解了 Java 面向对象编程的特性，并体会到它与 C 语言的不同。

这次设计中，我引入了一个特别的功能键“Acc”，该功能键能够记录上一次的运算结果，并在下一次计算时被调用，极大地增强了计算器的实用性。这一

创新点不仅提高了计算器的功能性，也让我在设计中学习到了如何在实际应用中添加实用功能。

在调试过程中，我遇到了许多挑战，例如如何有效地保存和调用输入的数字，以及如何处理连续计算过程中的数据传递。通过不断的试验和错误，我逐步优化了代码，并成功实现了所有预定功能。

总的来说，这次课程设计不仅加深了我对 Java 图形用户界面编程的了解，也增强了我将理论知识应用于实际问题解决的能力。我期待在未来的学习和工作中，能将这次的学习经历转化为更大的能力，为解决更复杂的问题提供支持。