简易手持计算器

开发者文档

一、开发任务描述与功能需求分析

设计一个简单的手持计算器，实现功能：

1、图形界面；

2、程序支持算术运算加、减、乘、除和等号（用户点击等号后显示结果）以及清除（清除最后输入的一个字符）和全清除（清除所有输入）；

3、用户可以连续输入，如1+2ⅹ3 =，系统要输出计算结果，即7；

4、用户还可以输入（），如（1+2）ⅹ3 =，系统要输出计算结果9；

5、如果用户输入的算式有错误，要给用户提示；

6、记录每次使用计算器时所有的正确的计算并可以查看；

7、每次使用计算器后能够将所有的正确的计算保存到文本文件中。

二、总体方案设计说明

1.开发环境

使用IntelliJ IDEA集成开发环境在操作系统macOS Ventura 13.4.1下开发，开发语言为Java，JDK为Oracle OpenJDK version 20。GUI部分使用了Nimbus风格的LookAndFeel使界面更加美观，详细介绍见官方说明文档：

<https://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/nimbus.html#:~:text=Permanently%20set%20the%20default%20look%20and%20feel%20to,not%20yet%20exist%2C%20you%20need%20to%20create%20it>。

2.总体结构和模块划分

系统模块划分为GUI启动器模块（class Starter），运算模块（class Calculator），有效性校验模块（class Validator）和字符串处理模块（class Reader），各模块功能如下：

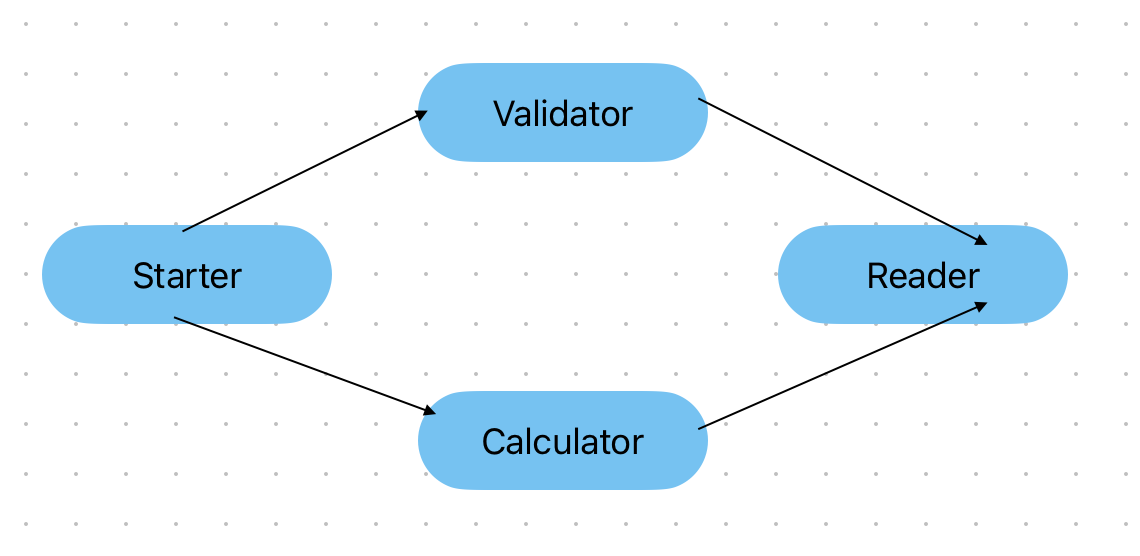
class Starter：创建计算器图形界面；通过按钮事件构建表达式；为Calculator和Validator提供需处理的表达式；显示运算结果；将运算结果存入本地文件。

class Calculator：计算给定表达式的值，若在计算中出错则提示“算数错误”。

class Validator：检验给定的表达式是否具有正确的语法（是否可被计算），若表达式存在语法错误则提示“语法错误”。

class Reader：包含在给定的字符串的指定位置读取一个double类型的浮点数或读取一个运算符的方法。

3.各模块调用关系：



三、各模块设计说明

1.class Starter

实例变量：

|  |  |
| --- | --- |
| 实例变量 | 描述 |
| private JTextField displayField; | 计算器显示区域 |
| Date date; | 用于获取当前系统时间 |

方法：

|  |  |
| --- | --- |
| 方法 | 描述 |
| public Starter()； | 同名构造器，创建计算器图形界面窗口，在窗口中添加各组件及其事件监听器 |
| public void actionPerformed(ActionEvent e)； | 处理事件 |
| private void backSpace()； | 退格，删除显示区域的最后一个字符 |
| private void evaluateExpression()； | 计算显示区域中的表达式，并更新显示区域 |
| private void updateDisplay(String text)； | 更新结果显示区域，在原显示内容后衔接一个字符串 |
| public void addToLog(String str) throws IOException； | 将指定字符串写入文档中 |
| public void initSubWindow()； | 创建一个子窗口，显示使用计算器时的所有正确计算 |
| public static void main(String[] args)； | 程序的main方法，启动整个程序 |

核心算法描述：

显示区域：计算器包含一个JTextField来显示输入字符串和显示结果backSpace、evaluateExpression等函数都可以根据具体情况修改显示区域的内容。

按钮：计算器图形界面共有20个按钮，其中包含10个数字（0～9）、小数点（‘.’）运算符（‘+’、‘-’、‘\*’、‘/’）、括号（‘（’、‘）’）、退格（DEL）、清空（AC）和等于（‘=’）。所有按钮被放置在一个Jpanel上，该Jpanel使用4\*5的Grid Layout。所有按钮共用一个事件监听器，监听鼠标是否被按下，actionPerformed根据被按下按钮的不同来执行不同功能。

“Results”子窗口：计算器图形界面底部Results按钮被按下时，唤起一个Jdialog子窗口。子窗口创建一个输入流并从本地文件中读取所有的正确计算。被读取的计算内容呈现在一个JTextArea中。同时Jdialog子窗口底部设置一个close按钮，按下即可关闭子窗口。

写入文件：首先用Date类获取系统事件。然后addToLog方法创建一个输出流，将每次正确计算和该计算发生的时间写入本地文件。

2.class Calculator

方法：

|  |  |
| --- | --- |
| 方法 | 描述 |
| public static double evaluateExpression(String expression) throws ArithmeticException； | 计算表达式的结果，如果出现除数为零的错误则抛出 |
| private static void processOperator(Stack<Double> operandStack, Stack<Character> operatorStack) throws ArithmeticException； | 处理运算符 |
| private static int stateTransition(int state,char input)； | 有限状态机的状态转移 |
| private static boolean isOpeningParenthesis(char ch)； | 判断字符是否为开括号 |
| private static boolean hasHigherOrEqualPrecedence(char operator1, char operator2)； | 判断第一个运算符的优先级是否大于等于第二个运算符的优先级 |
| private static int getOperatorPrecedence(char operator)； | 获取运算符的优先级 |

核心算法描述：

采取两个栈（Stack）分别存储运算数和运算符。采取有限状态自动机进行计算，0表示初态，1表示数字，2表示运算符，3表示'('，4表示')'。

开始时状态为0，若输入数字则状态转为1，若输入‘（’则状态转为3。

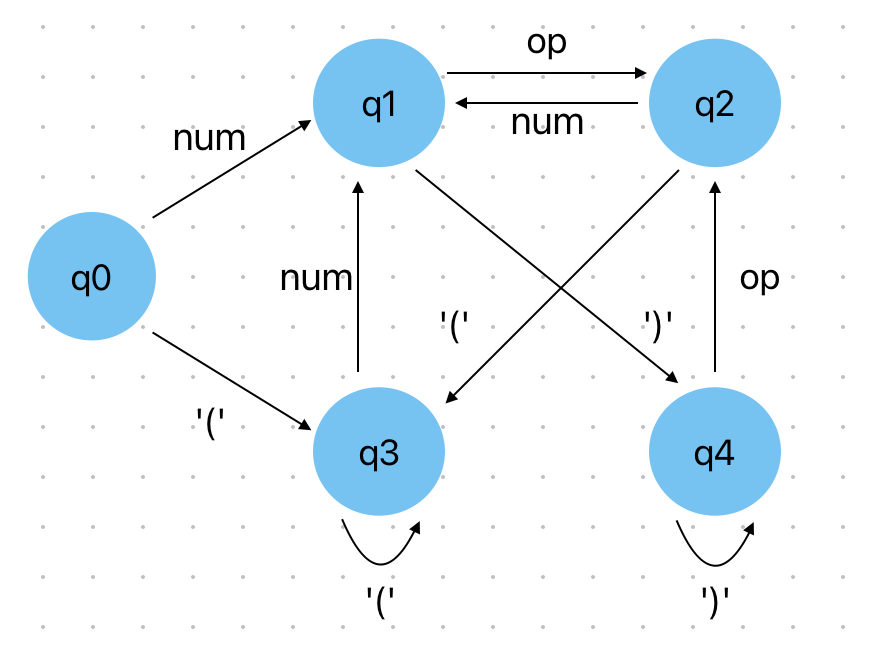
在状态1下，程序读入一个数字并将其压入运算数栈中。

在状态2下，程序读入一个操作符，并判断栈顶的运算符是否具有比该运算符更高的优先级。若栈顶运算符优先级更高，则从操作符栈中取出运算符op，从操作数栈中依次取出操作数operand2和operand1，计算结果operand1 op operand2，并压入操作数栈，然后将读入的运算符压入运算符栈；否则直接将读入的运算符压入运算符栈。

在状态3下，程序读入一个‘（’，并直接压入操作符栈。

在状态4下，程序不断从操作数栈和操作符栈中取出元素并进行运算，直到操作符栈栈顶元素为‘（’。

有限状态机状态转移图如下：



同时需要注意的是，在任何情况下，如果Calculator遇到了“除0”操作，则抛出一个

ArithmeticException。

3.class Validator

方法：

|  |  |
| --- | --- |
| 方法 | 描述 |
| public static boolean isValidExpression(String expression)； | 判断输入字符串是否是合法的 |
| private static int stateTransition(int state,char input)； | 计算有限状态机的状态转移 |

核心算法描述：

首先给出合法字符串的递归定义：

1. “数字”是合法的
2. “合法串op 合法串”是合法的，其中op为+，-，\*或/
3. “（合法串）”是合法的

通过对合法串递归定义的分析，可以得到判断一个表达式是否合法的经验规则：

数字后只能连接操作符或‘）’；操作符后只能连接数字或‘（’；‘（’后只能连接数字或‘（’；‘）’后只能连接操作符或‘）’。同时，整个表达式的‘（’与‘）’的数量相等，且表达式的任意前缀中，‘（’的数量大于等于‘）’的数量。

所以仍采取有限状态机的方式进行表达式的合法性验证，0表示初态，1表示数字，2表示运算符，3表示'('，4表示')'。

开始时状态为0，若输入数字则状态转为1，若输入‘（’则状态转为3。

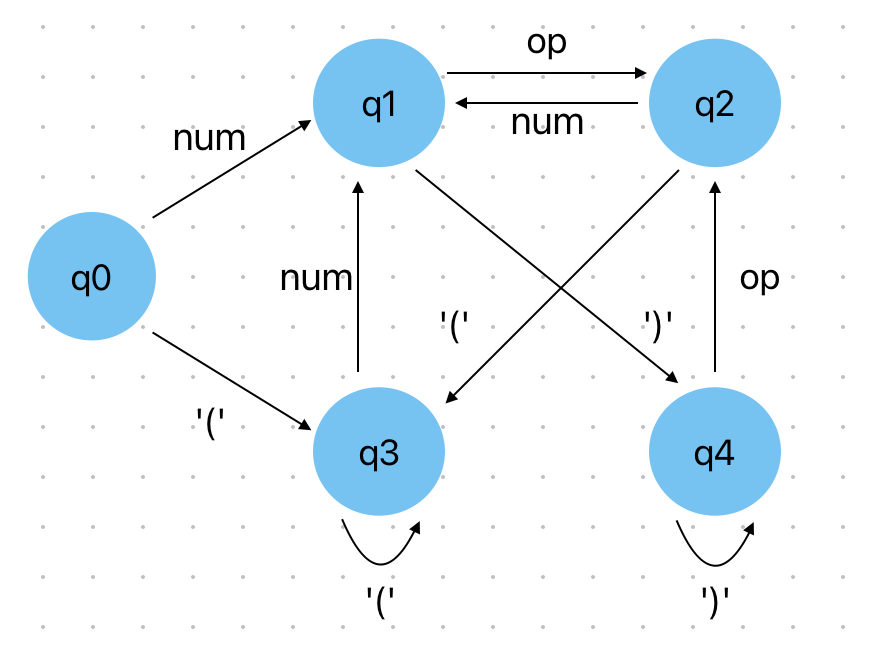
在状态1下，程序读入一个数字。

在状态2下，程序读入一个操作符。

在状态3下，程序读入一个‘（’，同时括号计数器加一。

在状态4下，程序读入一个‘）’，同时括号计数器减一。

有限状态机的状态转移图如下：



在任何情况下，如果程序读到一个不合规的串，则有限自动机因没有合规输入而停机，同时Validator的isValidExpression方法抛出一个IllegalArgumentException。

4.class Raeder

方法：

|  |  |
| --- | --- |
| 方法 | 描述 |
| public String readNumber(String expression,int index) throws IllegalArgumentException； | 在字符串的index位置读取一个数字 |
| public String readOperator(String expression,int index) throws IllegalArgumentException； | 在字符串的index位置读取一个操作符 |
| public boolean isOperator(char ch)； | 判断字符是否为运算符 |
| public boolean isParenthesis(char ch)； | 判断字符是否为括号 |

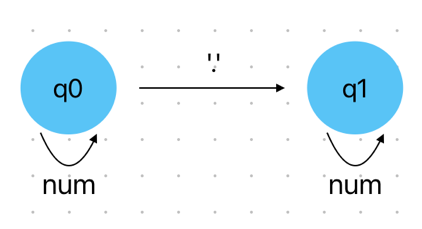
核心算法描述：

readNumber方法首先判断读取的第一个字符是否为‘-’，如为‘-’则读取的数字为小数。

然后该方法使用有限状态自动机来读取一个double类型数字（以字符串的方式读取，然后计算时用parseDouble方法将字符串转化为double类型变量）。

有限状态自动机中，0为(-)(整数)，1为"(-)(整数).(小数)"，状态0读取‘.’后进入状态1。状态0中读取的为数据的整数部分，状态1中读取的为数据的小数部分。

有限状态机状态转移图如下：



在有限状态自动机中，如果读到运算符‘+’、‘-’、‘\*’、‘/’，或括号，则读取数字完毕，返回读取的字符串；若读到其他非法字符，或在状态1下读到小数点‘.’，则抛出IllegalArgumentException，并在异常的参数中包含错误出现的位置。

readNumber方法直接读入一个字符，如果是运算符则返回它，否则抛出IllegalArgumentException并在异常的参数中包含错误出现的位置。

四、系统使用说明

1.启动简易手持计算器

如果电脑安装了IntelliJ IDEA集成开发环境，则直接用IntelliJ IDEA打开工程文件，编译运行即可。若没有安装IntelliJ IDEA集成开发环境，则应先确保电脑安装了较新版本的Java JDK。

打开包含源文件的文件夹，使用

javac \*.java

编译所有文件，可以看到文件夹中生成了多个class文件，然后使用

java Starter

运行启动器，即可启动该简易手持计算器。

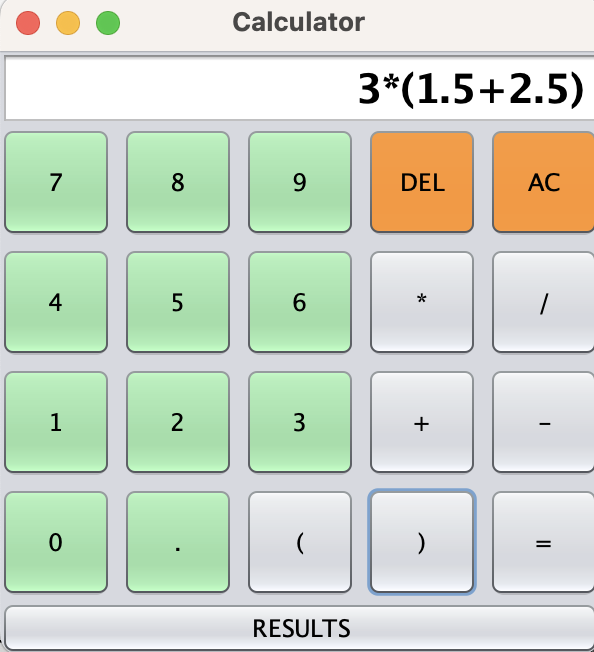
2.计算功能

启动简易手持计算器后，可以看到如下界面：



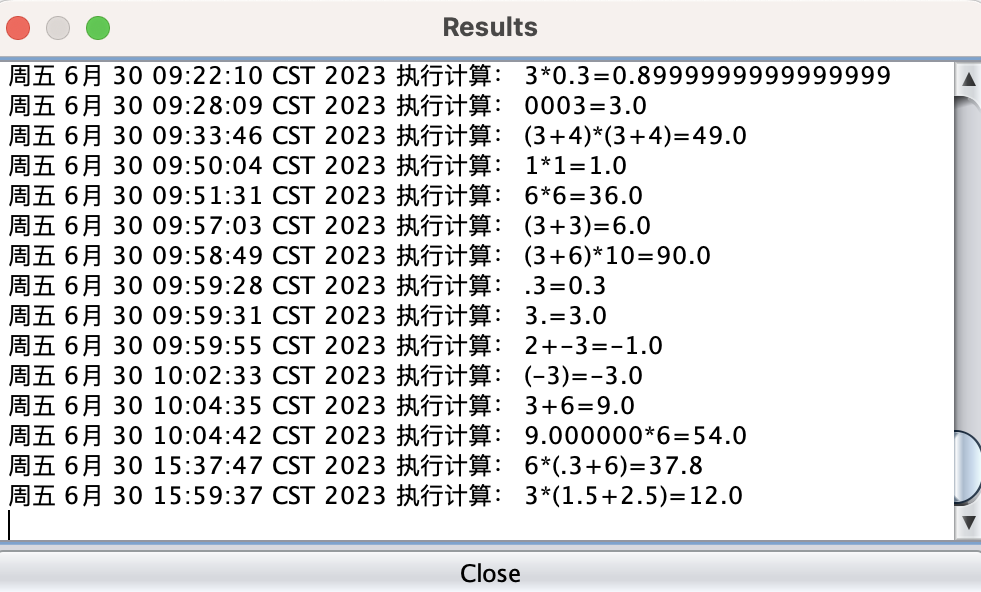
其中绿色的按钮为数字按钮，包括0～9和小数点；橙色为删除按钮，包括退格（DEL）和清空（AC）；灰色为符号按钮。

按照正常计算器的使用方式使用即可：先输入需要计算的表达式，按下等号后显示区域即可显示运算结果。效果展示如下：

3.查看日志功能

点击计算器图形界面下方的Results按钮即可查看计算日志：



PS：如果日志无法正常显示，请编辑Starter.java，并将addToLog函数和initSubWindow中的filename改为电脑中可以访问的路径。

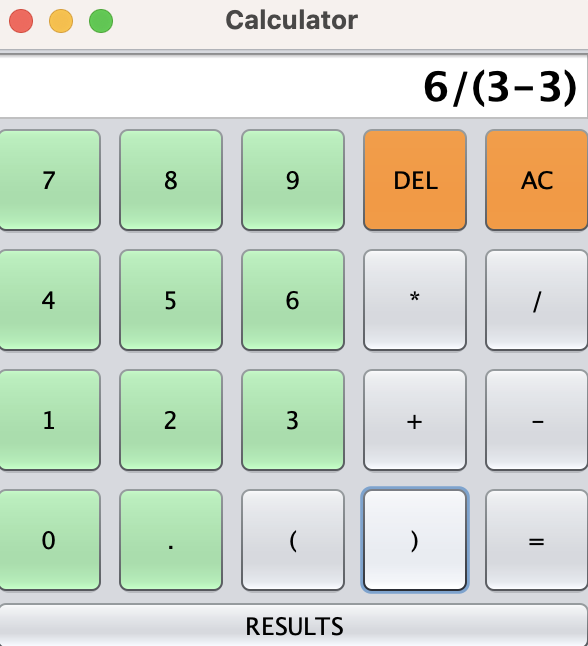
4.异常

该计算器有两种异常提示，“语法错误”、“算数错误”。

当用户输入的字符串不是正确的算数运算表达式时，会提示“语法错误”：

当用户提供的计算式在运算过程中出现了“除数为0”错误时，会提示“算数错误”：

当出现异常提示时，用户必须检查输入的表达式中哪里存在错误，然后输入正确的表达式，才能进行计算。