上海电力学院

高级语言程序设计（JAVA）
课程设计

题　　目：计算数学表达式的程序

学　　号：＿＿＿＿＿＿　姓　　名：＿＿＿＿＿＿

院　　系：计算机与信息工程学院

专业年级：软件工程 2011 级

2012 年 7 月 6 日

目录

第一章　系统需求与分析

1.1　系统设计目的

计算机刚发明的时候，其主要目的，顾名思义——用于计算的机器。

计算机发展至今，其功能有了日新月异的发展，但是，其所有的功能都依赖于其强大的计算能力。

现在各平台上计算器软件可以说是数不胜数，但是，我们发现，很少有可以输入数学公式直接计算的。于是，我们开始有意向开发这样一款软件。

以往的计算器总是从机器的角度计算，让人适应机器。本系统的主要目标即为：按照人类书写习惯计算。

其可以实现直接输入一串算式，然后直接得出结果。

1.2 系统设计意义

这是我们学习了一年 Java 语言之后，第一次独立开发应用。这套系统的开发成功，标志着我们对 Java 语言的基本掌握，对语言有了一定的运用能力。在开发过程中遇到的问题，在解决的过程中，很好的锻炼了我们的思维和差错能力。

这套系统的开发，思想上的最大进步是：该程序在平台的允许下，完全按照人类习惯，不再要求人类适应机器，关于这一点，后面将会详细表述。

1.3 系统设计功能

1.3.1 屏幕按钮计算

计算器窗口下方有 5×6 的按钮，计算的时候可以直接通过按钮输入算式进行计算。

图 1-1 屏幕按钮计算演示

1.3.2 键盘输入计算

计算器同样可以由用户自行输入算式计算，需注意：

*乘除运算符号（×、÷）对映键盘上的 / 和 * ，在键盘的主键盘区和小键盘区输入皆可，程序会自动识别为×或÷显示在计算窗口中。*

1.3.3 计算功能

本计算器可进行以下计算：

- 四则运算 如：（34+67）×50%÷3.8π
- 乘方，开方运算 如：3^（2+5）、2√2
- 三角函数计算 如：cos(sin(5π+6)) （包括角度制和弧度制）
- 绝对值与求余计算 如：abs(7e-7^2) 、 17mod5

1.3.4 自然语言输入

本节举例说明自然语言输入

- 在数字和（）之间可以省略乘号；

- 在数字和π、e之间可以省略乘号；
- 数字位于sin、cos、tan之前可以省略乘号；

乘除号一律用标准的×、÷表示，为方便键盘输入，在键盘的主键盘区和小键盘区输入皆可，程序会自动识别为×或÷显示在计算窗口中；

- √使用和自然情况完全相同；
- 输入π、PI、pi、e皆可参加运算，系统自动识别。

1.3.5 系统特色功能

- 在设置中可以转换三角函数的角度制和弧度制，并且在下一次打开的时候会记忆上一次选择的模式；
- 算式输入完成后，按下回车或者点击“=”按钮，系统会自动填充缺少的右括号“)”；
- 保存的打开系统，可以快速保存当前的计算结果，以便在下一次使用时调用。
- 可以方便的转换三角函数的弧度制以及角度制，并且可以在再次打开时恢复到上次的选择。

第二章 系统设计

2.1 GUI 界面设计

2.1.1 GUI 界面结构

图 2-1 软件界面 UI 界面

软件借鉴了 Windows 自带计算器的界面，以简洁大方清晰为主要设计思路。

2.1.1 GUI 菜单设计

在 JMenuBar 上放置了 menufile, menuedit, menuhelp, menuset 四个菜单按钮，菜单中分别有如下按钮：

图 2-2 软件菜单设置

2.1.2 GUI 计算窗口设计

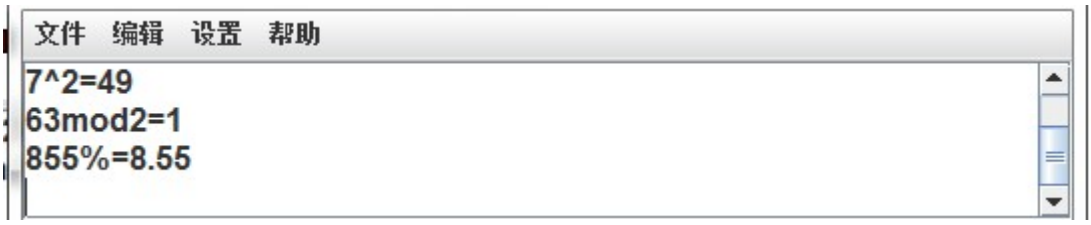


图 2-3 计算窗口设置

计算窗口使用了一个 4 行的 JTextArea，加入了滚动条，可以记录历史计算数据，输入数据和显示结果位于同一界面，如果算式过长，可以自动换行

JTextArea 位于 BorderLayout 的 NORTH。

2.1.3 GUI 按钮设计

刚开始的时候使用 FlowLayout 设计按钮，但是无法保持顺序和美观度，后改用 GridLayout，自动表格排版，简单美观。

设计按钮顺序的时候参考了 Windows 计算器的布局，根据自己的习惯安排，现在一个 Excel 文件中规划，如下：

	A	B	C	D	E	F
1	sin	根	CE	π	e	/
2	cos	^2	7	8	9	*
3	tan	^	4	5	6	-
4	abs	(	1	2	3	+
5	mod	)	0	.	%	=

图 2-4 计算按钮原始设计

然后根据原始设计，安排按钮布局，add 按钮。如下：

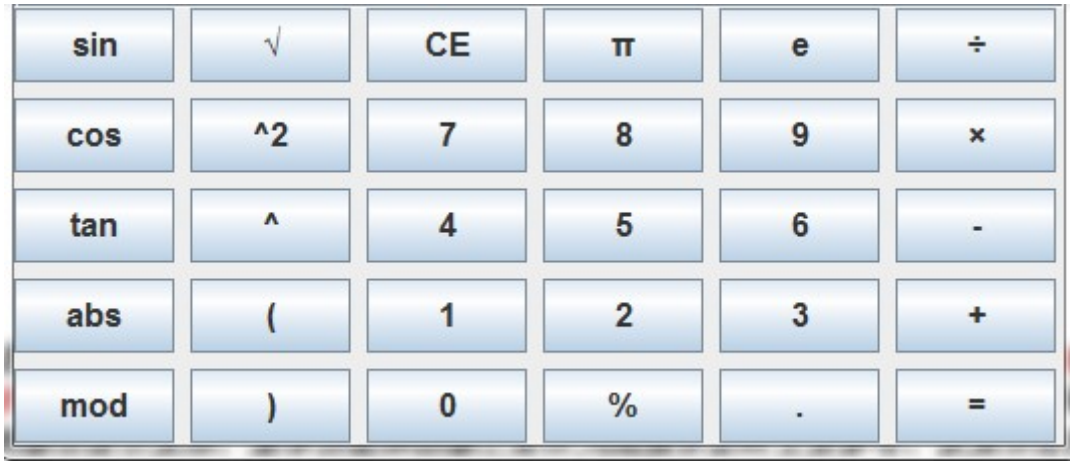


图 2-5 计算按钮设计

开始尝试按+-×÷顺序拜访，后来注意到因为+用的最多，所以大多数设备（计算器、小键盘）顺序皆为÷×-+。于是加以修改。

2.2 运算核心功能设计

2.2.1 设计计算数学表达式的基本过程

以流程图表示：

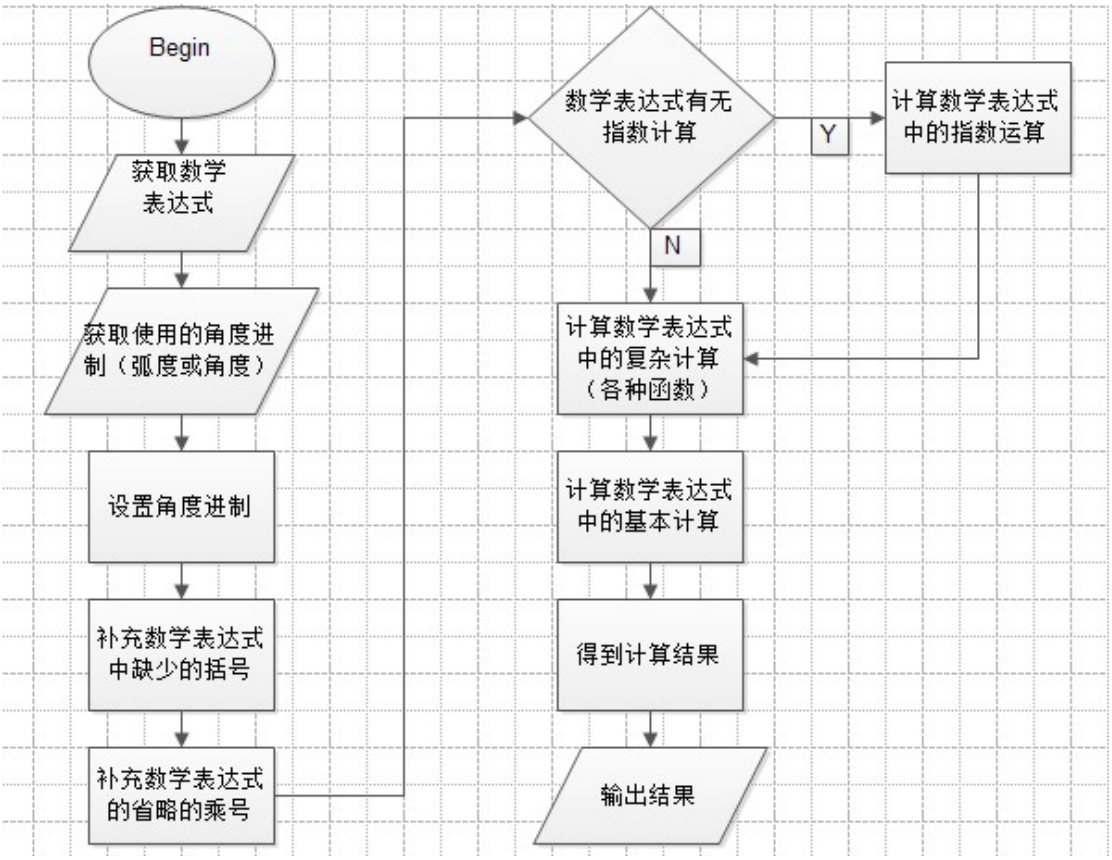


图 2-6 计算过程流程图

2.2.2 计算数学表达式的功能要求：

1)由用户输入一个简单的四则运算表达式，求出其计算结果后显示。
如计算表达式：23+56/(102-100)*((36-24)/(8-6))
2)允许在表达式中出现常用的数学函数，如绝对值、取整、三角函数、倒数、平方根、平方、立方等。
3)尽量使用图形界面实现，要符合日常软件使用规范来设计菜单和界面。
4)如果无法实现图形界面，则在命令行方式下也需要提供菜单，方便用户操作。

2.2.3 核心实现功能：

1） 找到数学表达式中的符号，然后找到符号前后的数字进行计算
2） 找到数学表达式中的函数，如果是单目运算找到后面的数字计算，如果是双目的计算找到前后数值进行计算

2.2.4 附加功能：

1） 如果数学表达式中遗漏括号，补充在最后
2） 符合一般数学书写，即在一些函数 如 sin，cos 等以及 π，e，括号之前可以省略乘号
3）计算三角函数时可以转换弧度制和角度制

2.3 GUI 与核心 API 适配设计

本程序 GUI 与核心功能由两人分别完成，在 GUI 中输入输出，核心提供 API 以完成计算。

核心提供的 API 有以下 3 个：

```
public void setPattern(String Pattern)
//此 API 实现表达式从 UI 传递到核心运算类。

public double solvePattern(boolean mode)
//此 API 返回计算结果

public int fillParenthese()
//此 API 返回需要补充的右括号“）”数量
```

通过如上述 API，GUI 编程人员和运算核心编程人员可以互不熟知代码而完成工作，我们使用了这种方案完成工作，有效的提高了编程效率。

此外，GUI 编程人员在获得核心代码之后，完成黑盒测试，找寻双方 Bug，然后分配 DeBuging。

第三章 系统关键模块技术实现

3.1 得到数学表达式后的处理函数

```
public double solvePattern(boolean mode) {// 参数表示角/弧制 true为弧度
false为角度
    double result = 0;
    this.mode = mode;
    fillParenthese();
    signTone();
    smartTip();
    if (hasPow(OPattern))
        solvePow(OPattern);
    result = solveComplex(OPattern);
    // result = solveSimple(OPattern);
    return result;
}
private boolean hasParenthese(StringBuffer Pattern) {// 判断表达式中是否有
括号
    for (int i = 0; i < Pattern.length(); i++) {
        if (Pattern.charAt(i) == '(' || Pattern.charAt(i) == '（')
            return true;
    }
    return false;
}
private boolean hasPow(StringBuffer Pattern) {// 判断表达式中是否有指数
    if (Pattern.indexOf(pow) != -1)
        return true;
    else
        return false;
}
```

要求输入一个 boolean 参数确定使用的角度进制。定义一个 double 记录结果设定角度进制

fillParenthese()函数用来补充表达式中缺少的括号

signTone()函数把表达式中的符号统一

smartTip()函数用来补充省略的乘号

hasPow(OPattern)函数判断表达式中是否有指数运算

solvePow(OPattern)函数用来把表达式中的指数计算先完成

solveComplex(OPattern)函数用来表达式中的函数运算完成，在之后会自动调用 solveSimple(OPattern)来计算表达式中的普通计算，即加减乘除以及取余

然后返回结果

3.2 处理函数运算的计算函数的关键代码

```
    if (start - 4 >= 0 && Pattern.substring(start - 4, start - 1)
        .equalsIgnoreCase(sin)) {// 计算sin
        if (mode) {
        Pattern.replace(
        start - 4,
        end + 1,
        Math.sin(solveComplex(new StringBuffer(Pattern
                .substring(start, end)))) + "");
        } else {
            Pattern.replace(start - 4,end + 1,
                Math.sin(Math.toRadians(solveComplex(new StringBuffer(
                Pattern.substring(start, end)))))+ "");
        }
    } else {
                    Pattern.replace(
                            start - 1,
                            end + 1,
                            solveComplex(new StringBuffer(Pattern.substring(
                                    start, end))) + "");
    }
    result = solveSimple(Pattern);
        // System.out.println(result);
        return result;
    }
}
```

判断表达式中的是哪个函数，来确定用哪个函数的运算处理，然后递归函数里面的式子，这样就可以计算嵌套的函数运算。直到函数内部没有复杂表达式，用 solveSimple()计算出结果返回，将原来表达式中的式子用计算出来的结果代替原来的式子，然后返回上一级运算，以此类推，将最终的式子转化为简单计算，再运行 solveSimple()得到最终结果。

如果表达式包含括号也是以相似的思路，将括号中的被容递归，先计算出括号中的内容，
替换掉原来的式子，将表达式化为最简单的数学表达式，得出最终的结果。

第四章　系统测试与部署

【注意：对系统主要部分进行测试。系统部署运行截取部分界面并做简单描述】

4.1 GUI 方面数据测试

我们设计了如下数据进行黑盒测试：

1、 sin(cos(tan(cos(sin(abs(14-5^2

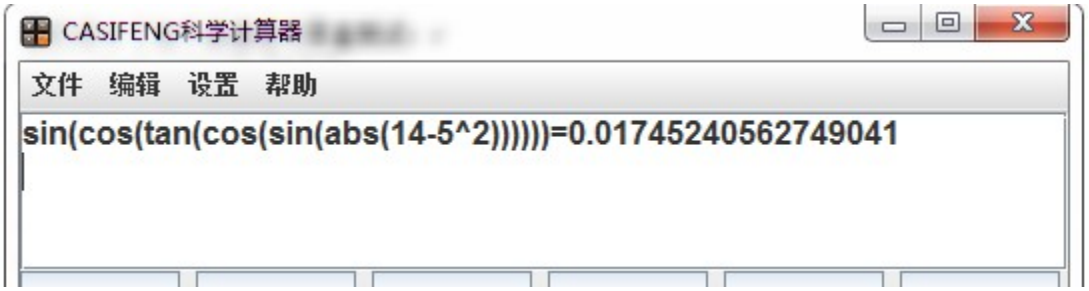


图 4-1 测试结果

测试通过！

2、8^2π-999mod7e

图 4-2 测试结果

测试通过！

3、2^√3^5

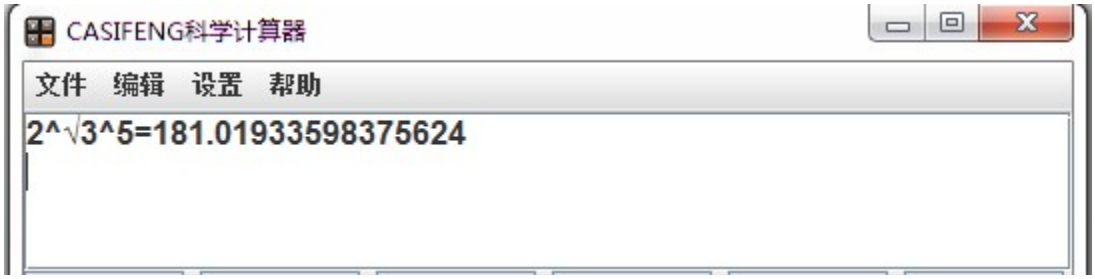


图 4-3 测试结果

测试通过！

4.2 核心方面数据测试

4.2.1

```
private double solveComplex(StringBuffer Pattern) {
        int num = 0;
        int start = 0, end = 0;
        double result = 0;
        while (hasParenthese(Pattern) || hasFunction(Pattern)) {
            System.out.println(Pattern);
            for (int i = 0; i < Pattern.length(); i++) {
                if (Pattern.charAt(i) == '(' || Pattern.charAt(i) == '（') {
                    if (num == 0)
                        start = i + 1;
                    num++;
                }
                if (Pattern.charAt(i) == ')' || Pattern.charAt(i) == '）') {
                    num--;
                    if (num == 0)
                        end = i;
                }
            }
            if (start == 1) {
                Pattern.replace(0, end + 1, solveComplex(new StringBuffer(
                        Pattern.substring(start, end))) + "");
            }
```

测试表达式：

sin(cos(1)+3)

得到的 sin 函数之后的数值为：

cos(1)+3

说明程序运行正常，可以完成想要的功能

```
sin(cos(1)+3)
cos(1)+3
1.0
3.5403023058681398
-0.388229529493152
```

4.2.2

还未找到好的解决办法的问题，但是不影响一般功能的代码：

```
System.out.println(Pattern);
            location = Pattern.indexOf("^");
            start = 0;
            end = 0;
            for (int i = location - 1; i >= 0; i--) {
                if (!Character.isDigit(Pattern.charAt(i))
                        && Pattern.charAt(i) != '.') {
                    start = i + 1;
                    break;
                } else if (i == 0) {
                    start = 0;
                    break;
                }
            }
            Pattern.replace(
                    start,
                    end,
                    Math.pow(
                            Double.valueOf(Pattern.substring(start,
location)),
                            Double.valueOf(Pattern.substring(location + 1,
end)))
                            + "");
        }
        // System.out.println(result);
        return result;
```

测试表达式:

2^2^2^2^2^2^2^2^2

测试结果：
0

```
65536.0^2^2^2^2
4.294967296E9^2^2^2
4.294967296E81.0^2^2
4.294967296E6561.0^2
0.0
```

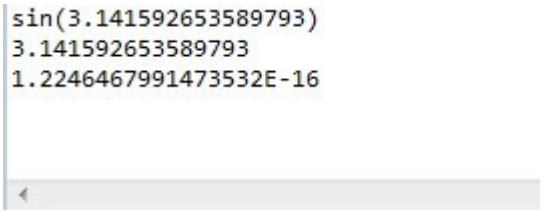
BUG 描述：结果为 0，不正确
原因：
指数数值的增长率太大，很容易超过 double 的最大范围，超过之后就得到了 0

4.2.3

```
if (start == 1) {
                Pattern.replace(0, end + 1, solveComplex(new StringBuffer(
                        Pattern.substring(start, end))) + "");
            } else {
                if (start - 4 >= 0
                        && Pattern.substring(start - 4, start - 1)
                                .equalsIgnoreCase(sin)) {// 计算sin
                    if (mode) {
                        Pattern.replace(
                                start - 4,
                                end + 1,
                                Math.sin(solveComplex(new
StringBuffer(Pattern
                                        .substring(start, end)))) + "");
        } else {
                Pattern.replace(start - 4,end + 1,
                Math.sin(Math.toRadians(solveComplex(newStringBuffer(
                Pattern.substring(start,end)))))+ "");
        }
```

测试表达式：
sin(3.141592653589793)

测试结果：
1.2246467991473532E-16



BUG 描述：sin(π)的理论值为 0，结果虽然非常接近 0 但是始终不正确
原因：
JAVA 中的 π 的数值始终有定值，会对最后的结果造成微小的差别，但是却与一般数学事实不符

第五章　系统开发总结与心得体会

5.1 GUI 开发总结与心得体会

5.1.1　GUI 开发心得

这是第一次写大型程序的 UI，之前写过 Android 的 UI，但是那个可以用 XML 写布局文件。相比而言，这次的 UI 则是完全通过手写代码生成，刚开始的时候没有按照一定的模块，导致添加菜单、按钮都很混乱，发现错误也不好修改。

后来，我才用分块注释的方法，把不同功能的代码区分开来，这样，不仅添加代码方便了，也更容易发现错误。

5.1.2　角度弧度问题解决心得

发现问题： 当我第一次得到核心代码运行的时候，我就发现，程序中三角函数是以弧度来计算的。而对于计算器来说，因为很难输入 π，而使得三角函数难以使用。

思考解决方法：解决方案分两部分，一是要能输入 π；而是想办法使得可以使用角度制。

解决过程：我和核心代码编写者经过沟通，他表示可以提供一个角度弧度转化接口，只要我给他提供一个 boolean 变量，返回值即可自动转化。于是我就想到加入一个单选菜单，并且定义一个 boolean 变量记录。
在实际使用中，我又发现，在设置过后，再次打开程序，又会恢复到初始状态，所以，我用了一个文件：Settings 来记录，使得程序可以自动按照上次设定的形式设置角度制。

5.1.3　×÷号自动转化心得

发现问题：刚开始的时候，乘除号就是用的计算机中标准的*和/。但是我觉得按照自然书写习惯，应该用×÷来代替，所以在做按钮的时候就用了×÷。但是在和核心程序编写人员沟通的时候，他提出，在键盘输入的时候，如果不能统一输入成×÷，这种风格不统一造成了软件的不专业。于是我听取了他的意见着手修改。

思考解决方法：解决方案很简单，用按键监听器监听键盘，当键盘输入*/的时候，拦截下来，并且自动补充×÷。

解决过程：这段代码看上去简单，但是我着实是废了两天的功夫才完美解决，因为我是笔记本键盘，没有数字小键盘区，所以*是通过 Shift+8 来输入的，但是无

论如何，系统都不识别这个*。
最终，我发现原来在 Java 中 小键盘的*/和主键盘的*/的键值是不一样的，而主键盘的*实际应该监听的就是 Shift+8，至此问题完美解决。

5.2 核心计算代码开发总结与心得体会

1）Q：如何将数学表达式中的计算提取出来
A：通过数学符号和函数的符号作为分割点，将各个部分提取计算

1） Q:如何计算更接近人们平时的数学表达式
A:补充人们平时经常会省略的括号和乘号，然后再计算

2） Q:将括号中的式子用结果代替，经常因为起始点和结束点的位置而出错
A:用 System.out.println()将每次截取的式子打印在控制台上，查看是不是写的有问题

3） Q:如何使程序可以解决嵌套计算
A:使用递归算法一部分一部分解决

5.3 GUI 与核心计算代码分离开发总结与心得体会

这个项目由两人完成，我们几乎是自然而然的选择了一方开发 GUI 一方开发核心计算代码的模式，这正巧和 Java 这种面向对象程序语言相适应。从实际情况看来，一方负责 GUI 和 Bug 查询，另一方负责核心计算代码和 DeBug，确实是非常有效率，双方都只要关心自己的代码就可以了，而不需要关注对方的具体代码。把有限的时间都用来调试好自己的代码，这无形给双方都减少了很多的工作量。

我们觉得，这种工作模式其实才是我们这次课程设计的最大收获吧。

参考文献

A 期刊
无

B 专著
无