面向对象程序设计（Java）

实验报告

实验三 GUI实验

|  |  |
| --- | --- |
| 学院名称 | 计算机科学与技术学院 |
| 专业班级 | 信安1802 |
| 学生姓名 | 侯天铭 |
| 学号 | 5120182377 |
| 指导教师 | 刘涌 |

2019年 4月

**一、实验目的**

练习面向对象的图形用户界面设计、界面布局、事件设计方法。

**二、实验内容**

设计一个类似Windows计算器的Application应用程序。

**三、概要设计**

1. **需求分析过程**
2. **用户对界面的需求：**

首先，作为一个计算器，必须得有一个输出框显示输入的相关内容，然后其还应该具备基本的按键，即0-9的数字，以及‘+’‘-’ ‘\*’ ‘/’四种运算符和‘=’，当然还需要一个点‘.’来做小数的运算。有时候，还需要一个按键使得用户能够通过点击这一个按键来清空所有输入的数据。至于窗体上的菜单栏对于本次设计的简单计算器倒并不是很重要。

1. **用户对使用过程的需求：**

在设计这个计算器之前，我也只是一个使用计算器程序的用户，所以至少对于我这个用户而言，使用一个计算器的过程应该是：输入第一个计算的数，屏幕上显示出这个数，输入运算符，屏幕上显示这个运算符，然后输入第二个数，屏幕上照样显示出这个数，然后点击等号后，屏幕上出现计算的结果，点击‘C’按键，将数据重置为0。因此这一次我设计的计算器程序也是按照这个思路来进行的。

1. **总体设计方案**
2. **界面设计**

我本次设计的计算器界面主要分为四部分：**整个窗体frame：**首先设置窗体名称即该应用程序的名字：基本整数计算器，显示在窗体左上角，同时还应包括该应用程序的图标。**菜单栏：**使用MenuBar创建，包括了选项，编辑，帮助的Menu，每一项Menu下可包含多项MenuItem。**显示框**：在ContentPane内使用TextFiled构建，用以显示用户输入的操作（默认为0），**操作框**：即一系列的按钮，同样在ContentPane中，TextFiled以下，使用Botton创建，包括了0-9的数字按键，小数点，加减乘除四则运算符号以及清空键C。整体布局从上到下依次为：窗体名称，菜单栏，显示框，操作框，整体布局分明。用户能够使用此计算器进行简单的整数四则运算，并且能够看到自己每一步进行的操作是什么。

1. **事件设计**

（从用户使用的角度出发，分析需要设计哪些事件、各个事件应完成怎样的功能）

对于用户而言，使用计算器的时候，每点击一个按键，程序都应该作出相应的反应回馈，因此我的设计思路是：对于每一个按键均设置相应的点击事件，用户点击0-9的数字按键以及加减乘除运算符的时候，均在显示框内显示出用户点击即输入的内容。另外，当用户点击‘=’时，不再显示‘=’，而是直接显示出计算的结果。当然，用户点击C后，显示框内应重置为0。举个例子，当用户点击1时，屏幕上就显示出1，点击‘+’，屏幕显示‘+’，再点击1，屏幕又出现一个‘1’，最后点击‘=’，屏幕上输出计算结果‘2’，最后点击C，屏幕上输出内容又重置为0。当然也可以不使用C清空，再此输入别的数即可覆盖结果继续进行下次运算。总的来说，最重要的事件如下：对于0-9的每一个按键，在点击的时候，屏幕上应输出相应的数字；对于运算符来讲，点击一次，首先需要一个变量储存当前屏幕上的数，然后再在屏幕上显示出当前输入的运算符，然后输入下一个数；最后点击‘=’时，应再使用一变量储存当前屏幕上即用于计算的第二个数，并将其与第一个数进行上一步输入的运算符代表的运算，然后将结果输出到屏幕上。

**四、详细设计**

**1、类的设计**

在Eclipse中选中一个项目内的一个包，右键点击new一个other，在下面的菜单中选中WindowBuilder中的Swing Designer内的Application Window（也可选择JPanel），在此以Application Window为例。接着输入这个类的名称，注意首字母大写，然后点击finish即可。新建的类中会自动填充一些代码，这些代码的功能就是创建了一个窗体对象，这个窗体具体长什么样，点击Design即可进入一个可视化界面，看到刚刚创建的窗体对象。再次点击Source可回到查看源代码。此后在Design内进行的可视化操作会反应到Source中生成或修改相应的代码；同样，在Source中写入或修改一些代码也会改变Design中窗体对象外观。

**2、界面设计**

整个计算器程序是一个frame窗体，但这个窗体最初是空白的，内部面板上也什么都没有，我们要做的就是在这个窗体里面增加一些东西，使其成为一个计算器程序。首先，设置窗体的名字：基本整数计算器。接着在窗体顶部下方独立于内部面板使用Menubar构建一个菜单栏，包括选项，编辑，帮助等可选项，每一项下还可包括若干个MenuItem。接着在ContentPane面板上紧贴菜单栏使用TextField创建一个文本输出框，用以显示用户输入的相关数据，运算符等，并且默认显示为0。剩下的就是设置按键，使用Button在面板内部，输出框下方构建若干个按钮有规律地分布开来，修改每一个按键显示的文本从而使其分别代表数字，运算符，等号等一系列需要使用的按键。

1. **事件设计**

在上文中已经初步说明了各个按键所对应的事件达到的效果，接下来将深入分析各个事件的工作流程。首先，在设计一切事件之前需要做一些准备：

**（1）**定义一个整型的num1变量用于存储进行两个数的运算时的第一个运算数；

**（2）**定义一个整型的oper变量来代表当前进行的操作，并将其初始化为0表示未进行任何操作，其值1，2，3，4，5分别对应输入‘+’‘-’‘\*’‘/’‘=’；

**（3）**定义一个boolean类型的变量operJustPressed用于判断是否输入了运算符，并初始化为false，当输入了运算符后，将其值置为true。接下来我会对每一类事件进行详细地描述：

1、**点击0-9的数字：**当用户点击数字按键的时候将触发该事件，步骤主要分为：

**（1）判断是否输入了运算符。**即判断此时点击输入的数字是运算的第一个数还是第二个数。由于之前使用了变量operJustPressed来表示是否输入了运算符，因此可用if(operJustPressed)语句来完成条件判断。以数字1为例，如果是第一次输入，则首先获取当前屏幕上的值即0，储存在变量i里，然后将屏幕上的输出值置为(i\*10)+1，、于是此时屏幕上应显示1，第二次点击1，获取当前屏幕上的值1储存在i中，然后按规则(i\*10)+1屏幕上会显示11，由此往后类推**；**如果条件成立，说明这是输入的第二个数也就是说在此之前刚刚输入了一个操作符并且这个操作符显示在了屏幕上，那么当用户再次点击1时，屏幕上应该开始显示用户输入的第二个数，也就是应该先输出一个1，并把oper置为0，意味着当用户下次点击1时将按照没输入运算符时的规则进行操作，即再点击一次1，屏幕上会显示11，再点击会显示111等。

**（2）判断是否输入了等号。**由于之前使用了oper这一变量代表用户输入的符号，oper==5代表用户刚刚点击了‘=’即刚刚完成了一次运算输出了结果，那么使用else if(oper == 5)判断条件成立的话意味着用户在进行下一次的运算了，即屏幕上本次运算的结果应被覆盖，并且显示用户点击的用于下次运算的数值。同样以1为例，用户完成一次运算后再次点击‘1’时，首先将屏幕上显示的数值设为1，然后将oper置为0，意味着接下来用户再次点击1时将按前述的基本规则进行操作，即先获取值1储存在i内，然后设置输出的值为(i\*10)+1。这一步判断的主要目的是当用户点击‘=’输出计算结果后再次点击数字，可以实现对上一步计算结果的清空，以及新输入数的显示。

**（3）**点击数字0之后，操作大部分与其余数字相同，只不过在if(operJustPressed)判断条件不成立即未输入运算符时，不再是像1一样显示(i\*10)+1的结果，而是直接显示(i\*10)的结果。

**2、点击运算符：**当用户点击运算符时触发该事件。以加法为例，为了实现连续的计算，在用户每次点击‘+’后都需判断此前是否输入过运算符，如果用户此前已经输入过了‘+’，说明刚刚已经完成了一次加法的运算，按照先前的思路，第一个数存储在变量num1里，第二个数存储在变量i里，由于这是第二次输入‘+’，最终计算结果我也是以num1表示，因此首先需要对num1的值进行改变即令num1 = num1 + i，举个例子，就是说如果用户已经输入了“1+1”，那么当用户再次点击‘+’时，会首先完成一次加法运算，使得此时的num1 = 2，然后再将刚点击的‘+’显示在屏幕上，当然最后需要把oper置为1，operJustPressed置为true。那么对于其他的运算也是一样的，在用户点击每种运算符时都会判断oper的值，如果是1，则说明先进行过加法运算，则使num1 = num1 + i；若i = 2，则使num1 = num1 – i；若i = 3，则使num1 = num1 \* i；若i = 4，则使num1 = num1 / i。最后总结一下，总体思路是把连续的计算转化为两个数之间的运算，使用num1来存储用于计算的第一个数，变量i存储用于计算的第二个数，当用户每次点击运算符时都要判断此前是否输入过运算符，输入的是什么运算符，从而先进行一次相关的运算并把得到的值赋给num1，实现计算数的“合二为一”，然后再输入下一个数，再输入运算符，又再次进行相关运算把结果赋给num1这一个变量，然后再输入数……以此类推，实现连续的运算。

**3、点击等号：**当用户点击‘=’时，说明需要运算的数以及运算符已经输入完毕了，现在需要得到结果，那么在此我使用了一个switch语句来判断最后输入的运算符是什么。根据上述对运算符事件的设计我们知道，每当用户点击一个运算符都会完成一次运算，也就是说当用户点击‘=’时实际上就只剩两个需要进行运算的数了，即之前运算的结果和最后输入的一个数。使用switch(oper)根据oper不同的值进行最后一次运算（还是把当前屏幕上的数存储在变量i里）：若oper =1，num1 = num1 + i；若oper =2，num1 = num1 - i；若oper =3，num1 = num1 \* i；若oper =4，num1 = num1 / i。最终将num1转化为字符串显示在屏幕上即可。

**以上所有操作最重要的有两点：**

**（1）**获取屏幕框中的值后（此时为String类型）需将其转换为int型，使用到了Interger类的parseInt()方法，方法要求传入一个String类型的字符串，结果返回一个与该字符串对应的整型数字。具体操作为i = Interger.parseInt(txtview.getText());

**（2）**以点击1为例，操作**(**i\*10)+1的结果为整型，要将其显示在输出框内需要先将其转换为String字符串类型，只需用其加上一个空的字符串即可（基本类型+String最终都会转换为String类型）。具体实现为：txtview.setText((i\*10+1)+ '' '');

**五、实验结果**

在本次实验结果的测试过程中，我首先分别对加减乘除运算单独验证结果的正确性，然后再对多种混合运算验证其正确与否。**（结果均只考虑整数且不考虑运算优先级）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入表达式 | 应得结果 | 实得结果 | 正误判断 | 结论 |
| 25+35+46= | 106 | 106 | 正确 | 该应用程序能正确完成整数加法的计算 |
| 12-45-32= | -65 | -65 | 正确 | 该应用程序能正确完成整数减法的计算 |
| 4\*3\*8= | 96 | 96 | 正确 | 该应用程序能正确完成整数乘法的计算 |
| 560/5/4/2= | 14 | 14 | 正确 | 该应用程序能正确完成整数除法的计算 |
| 58+20-30+2= | 37.5 | 35 | 不考虑小数情况下正确 | 该应用程序能正确完成整数加减混合运算 |
| 18\*2\*3/4= | 27 | 27 | 正确 | 该应用程序能正确完成整数乘除混合运算 |
| 19+2\*2-4+20/4= | 14.5 | 14 | 不考虑小数情况下正确 | 该应用程序能正确完成整数四则混合运算 |
| 23-12=+2\*3/2 | 19.5 | 19 | 不考虑小数情况下正确 | 该应用程序能正确完成整数四则混合运算 |
| 120\*5=-10/2=+50= | 345 | 345 | 正确 | 该应用程序能正确完成整数四则混合运算 |

由一系列的测试数据得到最后的实验结果：**本次设计的计算器程序能满足用户进行整数四则运算的需求。（不考虑优先级）**

**六、讨论与分析**

1. **对课程知识点的理解：**

本次实验的主题就是利用WindowBuilder创建一个具有图形用户界面的应用程序。那么在实验过程中，首先是学到了利用WindowBuilder来创建类的过程（包括了JFrame类和Application Window类两种）。也认识到由于利用JFrame来创建对象是将JFrame作为了父类，因此其子类本身就是一个窗体类，所以可以直接new一个该类的实例对象，然后在构造方法里面实现对该窗体的初始化，并且再创建一个JPanel对象（窗体内部的面板）完成初始化，就完成了整体窗体的搭建，之后就是往内部再填充别的东西了。但是对于Application Window，以此创建的类并没有继承自任何类，因此要创建一个窗体就需要先在类体中定义一个JFrame类型的属性frame，然后在创建本类对象window的时候同时创建一个JFrame对象frame，即创建了一个窗体，并完成相关大小的初始化。然后可利用window对象访问其属性frame实现对窗体的操作。

然后说说Source和Design的关系。本身所有的操作都可以在Source界面里面通过写代码来完成，但是WindowBuilder为我们提供了一个可视化的Design界面，大大方便了图形用户界面的设计。同时需要明白的是，二者之间是互通的，改变源代码，Design界面中的窗体会发生变化，同时在Design界面中对窗体进行可视化操作后的结果也会反应到Source中的源代码上。

1. **对设计过程的理解：**

通过本次实验，在程序设计过程上最大的感触就是，当我们在设计一个程序的时候，一定要从用户的角度考虑，那么这时候就需要把自己想象成一个在使用这个程序的用户，如果我是用户，我会想要一个什么样的界面，我点击窗体上的各个按键又会分别触发什么样的效果。具体到计算器，如果我是用户，首先我想要一个清楚明了的界面，上方有个菜单栏，下面有一个屏幕显示数据，再往下是一个个的按键，因此我们需要首先设计出这个窗体的外形。有了清楚的界面后，接着我想要点哪个数字，屏幕上就出现哪个数字，点‘+’就做加法，点‘-’就做减法，因此我们又需要去思考如何设计每一个按键对应的事件，当用户点击每个按键时会触发什么效果，屏幕上又会显示什么东西。所以整个设计过程实际上就是在跟着用户的想法走，考虑用户的需求，实现用户的需求。

1. **实验结果的分析：**

实验结果在第**五**点中已经详细地进行了分析：输入任意长度的表达式，在不考虑运算优先级的情况下，能够正确完成整数的四则运算并在屏幕上显示正确的结果。同时可以先进行一部分计算，点击‘=’显示出结果后再在此结果的基础上点击运算符继续运算也是可以的，同样能得到正确的结果。

1. **需要引起注意的问题：**
2. 由于本计算器只能进行整数的四则运算，因此需要保证输入的数以及计算的过程中不出现小数，并且做除法的时候被除数不能小于除数，否则会出现错误的结果。后期会对此进行改进，实现小数的计算。
3. 本计算器目前能够实现用户点击什么，屏幕上就显示什么，但是显示的仅仅是单个的字符，也就是说如果先后点击1+1，屏幕上并不会把1+1这整个表达式显示出来，因此如果输入的表达式过长就会使得整个计算过程不直观。

**七、附录：关键代码**

**（1）首先是最重要的三个变量：**

**int** num1 = 0; //储存第一个操作数

**int** oper = 0; //代表操作符(0,1,2,3,4,5分别代表无，加减乘除，等号)

**boolean** operJustPressed = **false**; //判断是否输入运算符

**（2）对于运算符事件，在此只展示加法，其余均类似：**

JButton btnplus = **new** JButton("+");

btnplus.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**if**(oper == 1) { //之前输入过加号

**int** i = Integer.*parseInt*(txtview.getText()); //获取当前屏幕上的值，转化为整型赋给变量i（第二个操作数）

num1 = num1 + i; //num1原是第一个操作数，为了实现连续计算，先将前两个操作数合并

txtview.setText("+"); //屏幕上显示加号

operJustPressed = **true**; //输入了运算符，该变量置为true

}

**else** **if**(oper == 2) { //之前输入过减号

**int** i = Integer.*parseInt*(txtview.getText());

num1 = num1 - i; //与上同理

txtview.setText("+");

operJustPressed = **true**;

}

**else** **if**(oper == 3) { //之前输入过乘号

**int** i = Integer.*parseInt*(txtview.getText());

num1 = num1 \* i;

txtview.setText("+");

operJustPressed = **true**;

}

**else** **if**(oper == 4) { //之前输入过除号

**int** i = Integer.*parseInt*(txtview.getText());

num1 = num1 / i;

txtview.setText("+");

operJustPressed = **true**;

}

**else**{ //第一次输入运算符

num1 = Integer.*parseInt*(txtview.getText());//获取当前屏幕上的值，转化为整型赋给变量num1（第一个操作数）

operJustPressed = **true**;//表示刚刚点击了操作符

txtview.setText("+");//屏幕上显示加号

}

oper = 1; //表示刚刚点击了加号

}

});

**（3）接下来是数字按键事件，在此只展示数字1，其余均类似：**

JButton btn1 = **new** JButton("1");

btn1.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**int** i;

**if**(operJustPressed) { //表示刚按了运算符

txtview.setText("1"); //屏幕上显示1

operJustPressed = **false**; //已经进行了操作后将该变量置为false，后续再次输入运算符时再置为true，防止干扰。再次输入数字按第一次输入判断

}

**else** **if**(oper == 5) { //oper == 5 意为此前输入了等号，说明已经完成了一次运算

txtview.setText("1"); //清空结果，显示1

oper = 0; //oper置为0，往后再次输入运算符时再改变oper的值，防止干扰。再次输入数字按第一次输入判断

}

**else** { //没按运算符

i = Integer.*parseInt*(txtview.getText()); //说明输入的是第一个数

txtview.setText((i\*10 + 1)+""); //由于界面上初始化为0，（i\*10 + 1）可实现点击1显示1，再次点击1，显示11，以此类推

}

}

});

**由于0不能作为除数，因此需要多一步判断（若在输入了‘/’的前提下，再次输入0，则报错）：**

**if**(oper == 4) {

txtview.setText("除数不能为0！");

}

**（4）最后是等号的设计，思路是利用switch语句：**

JButton btnequal = **new** JButton("=");

btnequal.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**int** i;

String s;

**switch**(oper) {

**case** 1: //表示进行加法

i = Integer.*parseInt*(txtview.getText());//获取当前屏幕上的值，转化为整型赋给变量i

num1 = num1 + i; //因为进行的是加法，因此num1 = num1 + i

s = num1+""; //输出的结果

txtview.setText(s);//输出字符串s

**break**;

**case** 2: //表示进行减法

i = Integer.*parseInt*(txtview.getText());

num1 = num1 - i; //因为进行的是减法，因此num1 = num1 - i

s = num1+"";

txtview.setText(s);

**break**;

**case** 3: //表示进行乘法

i = Integer.*parseInt*(txtview.getText());

num1 = num1 \* i; //因为进行的是乘法，因此num1 = num1 \* i

s = num1+"";

txtview.setText(s);

**break**;

**case** 4: //表示进行除法

i = Integer.*parseInt*(txtview.getText());

num1 = num1 / i; //因为进行的是除法，因此num1 = num1 / i

s = num1+"";

txtview.setText(s);

**break**;

}

oper = 5; //表示点击了等号

num1 = 0; //num1既是操作数，也是结果数，因此每完成一次计算后应将其置为0

}

});