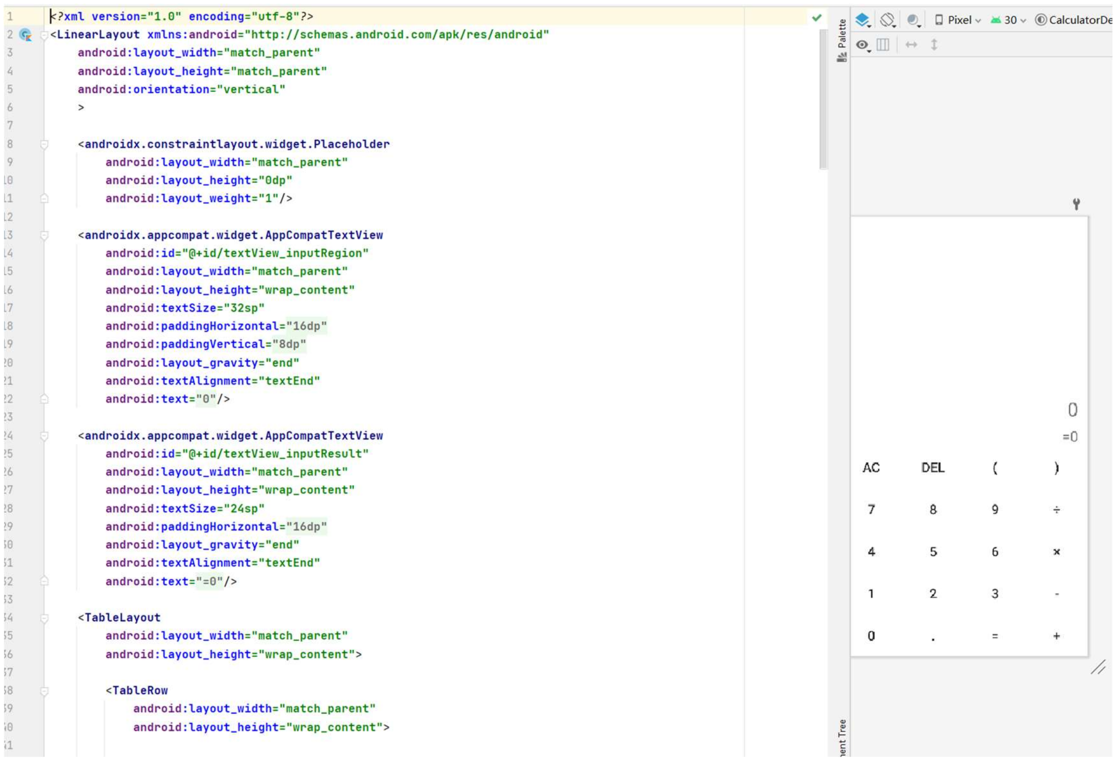
安卓项目第二次实验：计算器

布局说明：

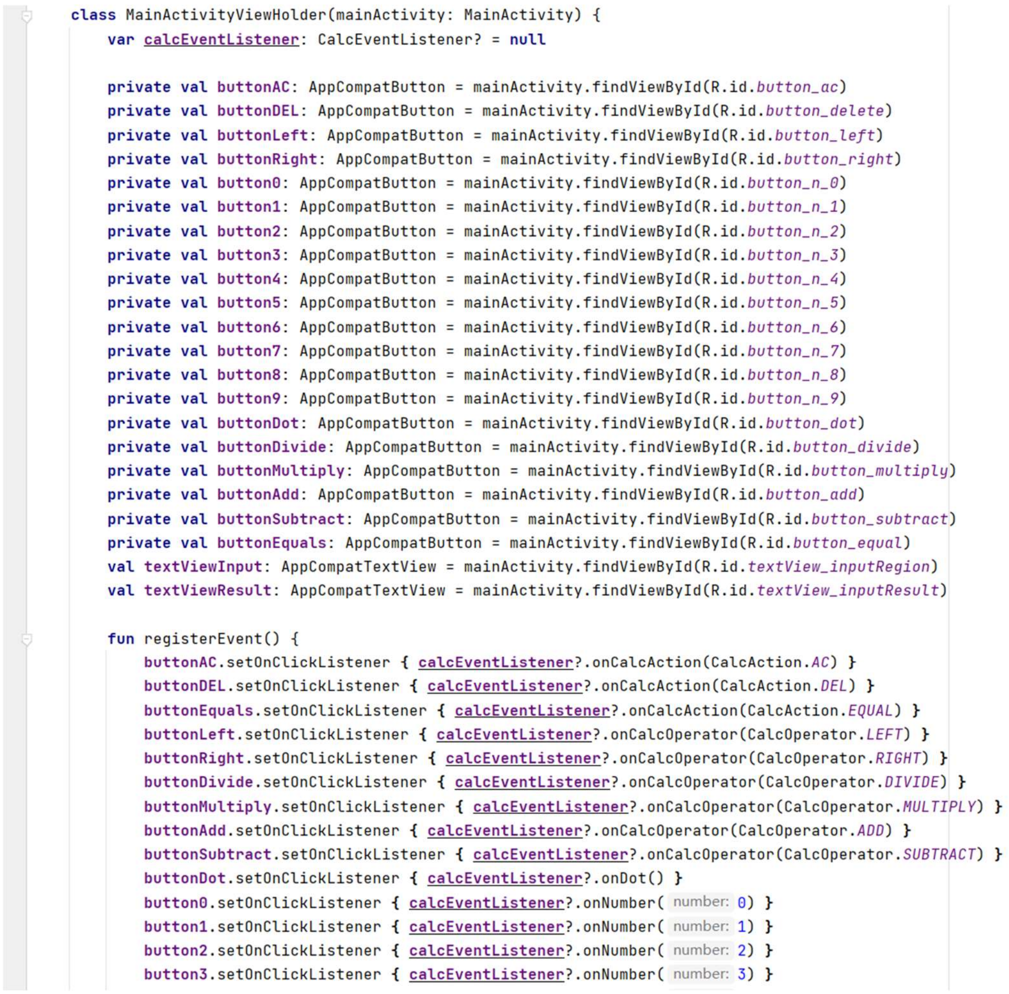


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >

    <androidx.constraintlayout.widget.Placeholder
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"/>

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/textView_inputRegion"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="32sp"
        android:paddingHorizontal="16dp"
        android:paddingVertical="8dp"
        android:layout_gravity="end"
        android:textAlignment="textEnd"
        android:text="0"/>

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/textView_inputResult"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:paddingHorizontal="16dp"
        android:layout_gravity="end"
        android:textAlignment="textEnd"
        android:text="=0"/>

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
```

其中上面分为输入显示栏和结果显示栏两个部分。下面为按钮部分，其中该计算器能够实现基本的四则运算以及带括号的运算。

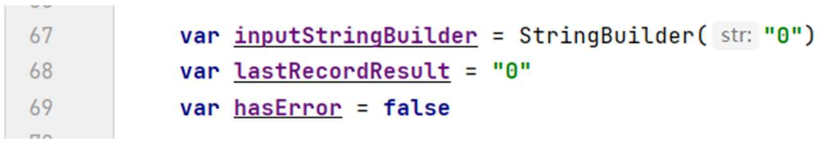
主要功能：实现复杂的四则运算，包括带括号的运算，使用一整个表达式求值而不仅仅限于单步求值。

代码说明:



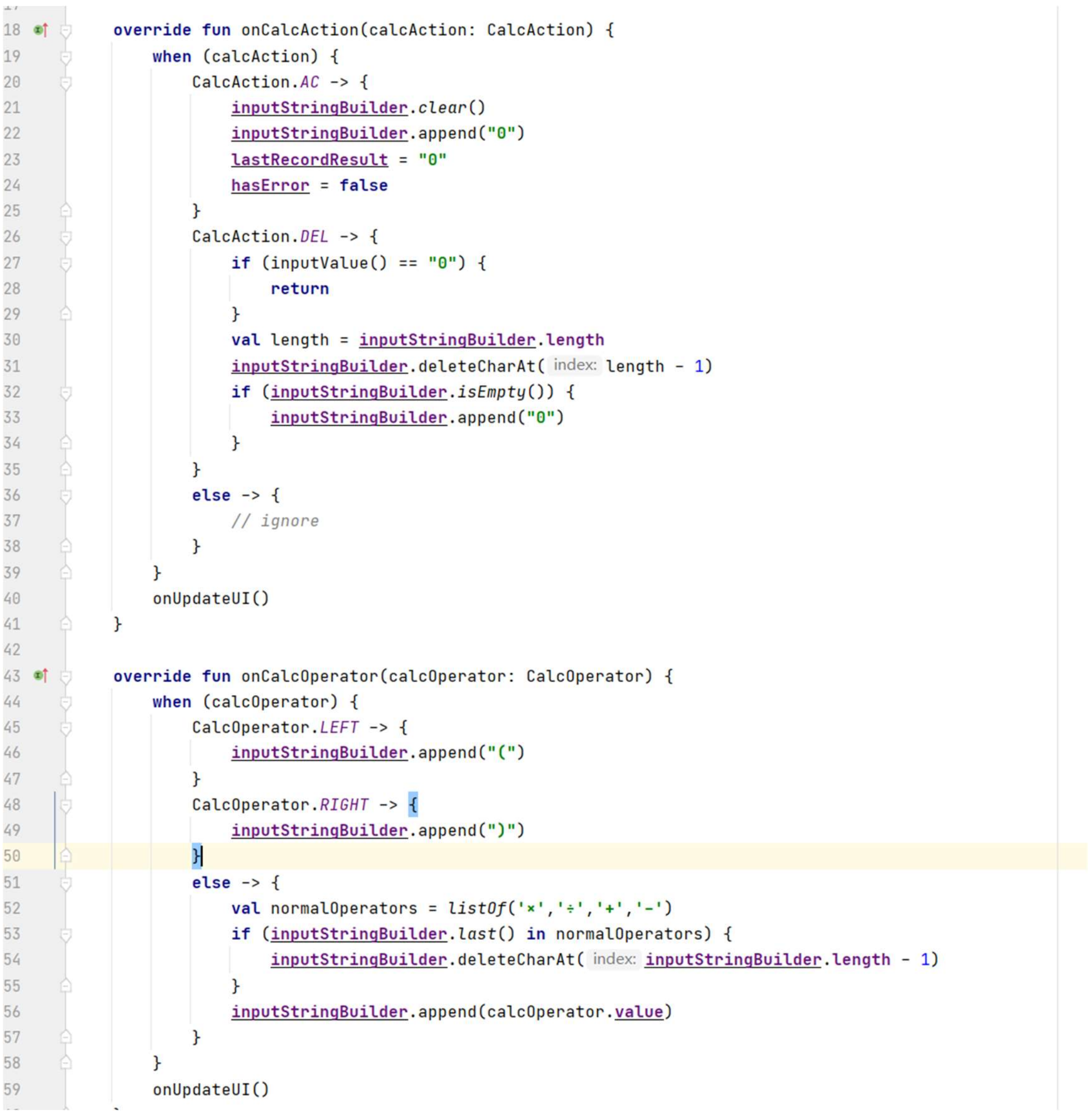
```kotlin
class MainActivityViewHolder(mainActivity: MainActivity) {
    var calcEventListener: CalcEventListener? = null

    private val buttonAC: AppCompatButton = mainActivity.findViewById(R.id.button_ac)
    private val buttonDEL: AppCompatButton = mainActivity.findViewById(R.id.button_delete)
    private val buttonLeft: AppCompatButton = mainActivity.findViewById(R.id.button_left)
    private val buttonRight: AppCompatButton = mainActivity.findViewById(R.id.button_right)
    private val button0: AppCompatButton = mainActivity.findViewById(R.id.button_n_0)
    private val button1: AppCompatButton = mainActivity.findViewById(R.id.button_n_1)
    private val button2: AppCompatButton = mainActivity.findViewById(R.id.button_n_2)
    private val button3: AppCompatButton = mainActivity.findViewById(R.id.button_n_3)
    private val button4: AppCompatButton = mainActivity.findViewById(R.id.button_n_4)
    private val button5: AppCompatButton = mainActivity.findViewById(R.id.button_n_5)
    private val button6: AppCompatButton = mainActivity.findViewById(R.id.button_n_6)
    private val button7: AppCompatButton = mainActivity.findViewById(R.id.button_n_7)
    private val button8: AppCompatButton = mainActivity.findViewById(R.id.button_n_8)
    private val button9: AppCompatButton = mainActivity.findViewById(R.id.button_n_9)
    private val buttonDot: AppCompatButton = mainActivity.findViewById(R.id.button_dot)
    private val buttonDivide: AppCompatButton = mainActivity.findViewById(R.id.button_divide)
    private val buttonMultiply: AppCompatButton = mainActivity.findViewById(R.id.button_multiply)
    private val buttonAdd: AppCompatButton = mainActivity.findViewById(R.id.button_add)
    private val buttonSubtract: AppCompatButton = mainActivity.findViewById(R.id.button_subtract)
    private val buttonEquals: AppCompatButton = mainActivity.findViewById(R.id.button_equal)
    val textViewInput: AppCompatTextView = mainActivity.findViewById(R.id.textView_inputRegion)
    val textViewResult: AppCompatTextView = mainActivity.findViewById(R.id.textView_inputResult)

    fun registerEvent() {
        buttonAC.setOnClickListener { calcEventListener?.onCalcAction(CalcAction.AC) }
        buttonDEL.setOnClickListener { calcEventListener?.onCalcAction(CalcAction.DEL) }
        buttonEquals.setOnClickListener { calcEventListener?.onCalcAction(CalcAction.EQUAL) }
        buttonLeft.setOnClickListener { calcEventListener?.onCalcOperator(CalcOperator.LEFT) }
        buttonRight.setOnClickListener { calcEventListener?.onCalcOperator(CalcOperator.RIGHT) }
        buttonDivide.setOnClickListener { calcEventListener?.onCalcOperator(CalcOperator.DIVIDE) }
        buttonMultiply.setOnClickListener { calcEventListener?.onCalcOperator(CalcOperator.MULTIPLY) }
        buttonAdd.setOnClickListener { calcEventListener?.onCalcOperator(CalcOperator.ADD) }
        buttonSubtract.setOnClickListener { calcEventListener?.onCalcOperator(CalcOperator.SUBTRACT) }
        buttonDot.setOnClickListener { calcEventListener?.onDot() }
        button0.setOnClickListener { calcEventListener?.onNumber( number: 0) }
        button1.setOnClickListener { calcEventListener?.onNumber( number: 1) }
        button2.setOnClickListener { calcEventListener?.onNumber( number: 2) }
        button3.setOnClickListener { calcEventListener?.onNumber( number: 3) }
```

首先设置了一个内部类 MainViewHolder 来获取对应的控件并将按钮的点击时间转化成自定义的时间 CalcEventListener，其具有四种方法，对应动作、操作符、点和数字四类，这样可以方便进行操作。



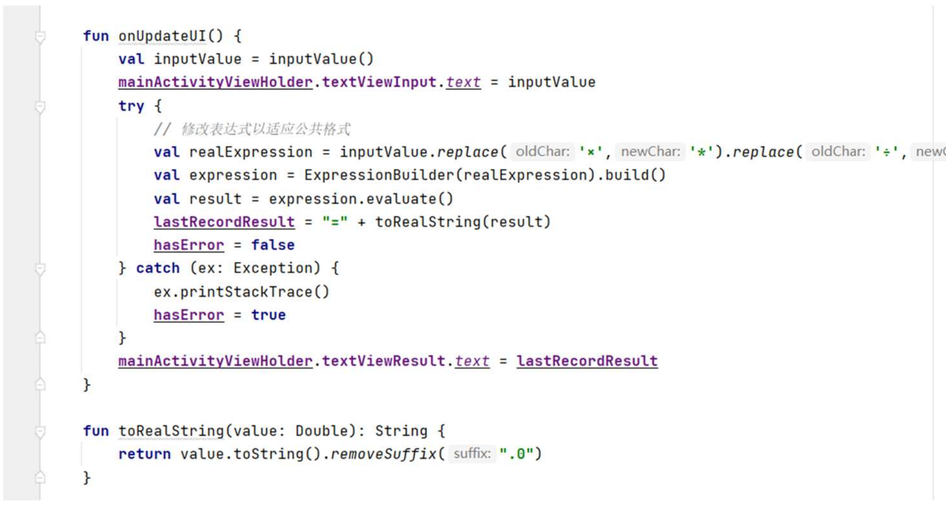
```kotlin
    var inputStringBuilder = StringBuilder( str: "0")
    var lastRecordResult = "0"
    var hasError = false
```

然后使用了三个字段，第一个字段用于记录输入的字符，第二个字段用于记录暂存的结果，第三个用于记录当前表达式是否出现了错误。



```kotlin
    override fun onCalcAction(calcAction: CalcAction) {
        when (calcAction) {
            CalcAction.AC -> {
                inputStringBuilder.clear()
                inputStringBuilder.append("0")
                lastRecordResult = "0"
                hasError = false
            }
            CalcAction.DEL -> {
                if (inputValue() == "0") {
                    return
                }
                val length = inputStringBuilder.length
                inputStringBuilder.deleteCharAt( index: length - 1)
                if (inputStringBuilder.isEmpty()) {
                    inputStringBuilder.append("0")
                }
            }
            else -> {
                // ignore
            }
        }
        onUpdateUI()
    }

    override fun onCalcOperator(calcOperator: CalcOperator) {
        when (calcOperator) {
            CalcOperator.LEFT -> {
                inputStringBuilder.append("(")
            }
            CalcOperator.RIGHT -> {
                inputStringBuilder.append(")")
            }
            else -> {
                val normalOperators = listOf('×','÷','+','-')
                if (inputStringBuilder.last() in normalOperators) {
                    inputStringBuilder.deleteCharAt( index: inputStringBuilder.length - 1)
                }
                inputStringBuilder.append(calcOperator.value)
            }
        }
        onUpdateUI()
```

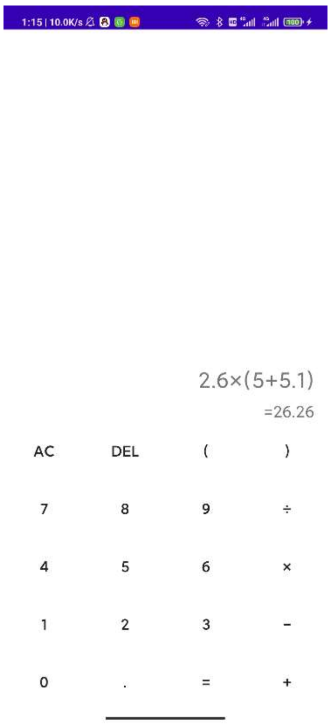
MainActivity 实现了自定义的时间方法，根据点击的按钮去动态修改 inputStringBuilder 的内容，以尽量达到和内置的计算器保持一致。



```kotlin
    fun onUpdateUI() {
        val inputValue = inputValue()
        mainActivityViewHolder.textViewInput.text = inputValue
        try {
            // 修改表达式以适应公共格式
            val realExpression = inputValue.replace( oldChar: '×', newChar: '*').replace( oldChar: '÷', newChar: '/')
            val expression = ExpressionBuilder(realExpression).build()
            val result = expression.evaluate()
            lastRecordResult = "=" + toRealString(result)
            hasError = false
        } catch (ex: Exception) {
            ex.printStackTrace()
            hasError = true
        }
        mainActivityViewHolder.textViewResult.text = lastRecordResult
    }

    fun toRealString(value: Double): String {
        return value.toString().removeSuffix( suffix: ".0")
    }
```

onUpdateUI()用于计算表达式的值并刷新 UI，为了简便起见，在本项目中使用了 exp4j 的开源数学公式库来辅助进行计算。

实现效果:


1:15 | 10.0K/s
2.6×(5+5.1)
=26.26
AC
DEL
(
)
7
8
9
÷
4
5
6
×
1
2
3
-
0
.
=
+