2020-6-16

Android移动应用开发

——高级计算器



系(院)名 称：计算机与信息工程学院

专 业 名 称：软件工程

学 生 班 级：计18 软件班

学 生 姓 名：陈 天 莹

学 生 学 号：1815080127

指 导 教 师：邓 育 林 讲师

目录

[一、问题描述与需求分析 2](#_Toc44155819)

[1、问题描述 2](#_Toc44155820)

[2、功能需求分析 2](#_Toc44155821)

[二、概要设计 3](#_Toc44155822)

[1、程序的总体设计思路、方法 3](#_Toc44155823)

[2、界面控件设计 4](#_Toc44155824)

[3、相对于基本需求的新增功能 5](#_Toc44155825)

[三、主要类源代码 6](#_Toc44155826)

[1、MainActivity类 6](#_Toc44155827)

[2、ScientificCalculatorActivity类 6](#_Toc44155828)

[3、StandardCalculator类 7](#_Toc44155829)

[4、ScientificCalculator类 16](#_Toc44155830)

[5、InputItem类 22](#_Toc44155831)

[6、Calculate类 24](#_Toc44155832)

[四、功能运行截图 29](#_Toc44155833)

[1、基础运算界面的正数加减乘除 29](#_Toc44155834)

[2、基础运算界面的负数加减乘除 30](#_Toc44155835)

[3、高级运算界面的运算 31](#_Toc44155836)

[4、括号的优先运算 31](#_Toc44155837)

[5、π的运算 32](#_Toc44155838)

[6、e的运算 32](#_Toc44155839)

[7、三角函数的运算 33](#_Toc44155840)

[8、对数函数的运算 33](#_Toc44155841)

[9、阶乘的运算 34](#_Toc44155842)

[10、百分号的运算 34](#_Toc44155843)

[11、平方运算、开平方运算与幂运算 35](#_Toc44155844)

[12、在前一步的运算结果上继续运算 35](#_Toc44155845)

[五、参考文献 36](#_Toc44155846)

[六、设计中的难点与解决方法 36](#_Toc44155847)

[1、UI布局不符合预期 36](#_Toc44155848)

[2、按钮透明效果 36](#_Toc44155849)

[3、计算器功能设计 37](#_Toc44155850)

一、问题描述与需求分析

1、问题描述

实现一个计算器app，要有输入框清空和删除前一位输入功能，可以实现在前一步计算结果上继续作加、减、乘、除，即连加、减、乘、除操作。除了0~9这10个数字按键以及+、-、\*、/等还应支持（）的优先运算和平方、开平方、幂运算等功能。

2、功能需求分析

1) 有良好的界面UI设计；

2) 设计的app有一个一目了然的中文名称；

3) 为app更换默认的系统图标；

4) 该计算器应是一个简单的科学计算器；

5) 应有0-9这10个数字按键；

6) 本计算器应支持输入框清空和删除前一位输入的功能，分别用AC和DEL按钮控制；

7) 可以进行加、减、乘、除的操作，且可以在前一步的结果上继续进行运算；

8) 可支持括号的优先运算和平方、开平方、幂运算功能；

9) 简单地模仿手机计算器的其他功能。

二、概要设计

1、程序的总体设计思路、方法

本次课程设计制作的高级计算器简单地模仿了手机上的计算器的功能。通过点击按钮可以切换基础计算界面和高级计算界面。

在基础计算界面中，实现了0-9这10个数字按键和加、减、乘、除功能，以及输入框清空和删除前一位输入的功能，另外，本计算器不但能进行正数运算，还能进行小数运算和负数运算。

高级计算界面中，在实现基础界面的功能的基础上，实现了括号的优先运算和平方、开平方、幂运算功能，增加了三角函数、对数函数、百分号、π、e、阶乘的运算，其中，三角函数的运算分别为sin()、cos()、tan()，对数函数的运算分别为ln()、log()。

MainActivity和ScientificCalculatorActivity两个Activity分别被定义来用于加载基础运算界面和高级运算界面的布局文件。

StandardCalculator类和ScientificCalculator类中的代码分别关于基础运算和高级运算界面，其被用于布局文件中的控件的加载、控件之间的值传递和关系的控制，以及各个按钮的响应事件。

InputItem类用于提高用户输入容错，如一个数字不能输入两个小数点、不能连续输入两个运算符等等。并划分本系统运算表达式字符类型为：数字型、操作型、类操作符的数字型、类数字型的操作符。

Calculate类用于储存各个运算功能的实现代码和数值传递的控制代码。为了实现加、减、乘、除功能，在设计本计算器时调用了java.math.BigDecimal.multiply()方法，返回一个BigDecimal，其值由toString()获取，而百分号则通过调用java.math.BigDecimal.divide()方法来实现，同样返回一个BigDecimal，由toString()获取其值，三角函数和对数函数，以及π、e、阶乘的运算同样通过为实现以上功能而导入的包中相应的方法来实现。

2、界面控件设计

基础计算界面：

* **文本框 TextView:** show\_result\_tv和show\_input\_tv分别用于显示运算过程结果和当前输入内容；
* 功能Button：c\_btn，del\_btn，res\_btn分别对应功能清除文本框内容、删除前一次输入、上一次运算结果；
* 运算符Button：+、-、\*、/、= ；
* 数字Button：0、1、2、3、4、5、6、7、8、9、. ；

高级计算界面：

* **文本框 TextView:** show\_result\_tv和show\_input\_tv分别用于显示运算过程结果和当前输入内容；
* 功能Button：c\_btn，del\_btn，res\_btn分别对应功能清除文本框内容、删除前一次输入、上一次运算结果；
* 运算符Button：+、-、\*、/、= 、()、%、π、e等；
* 数字Button：0、1、2、3、4、5、6、7、8、9、. ；
* 函数运算Button：sin、cos、tan、ln、lg。

3、相对于基本需求的新增功能

1）三个三角函数的运算：sin()、cos()、tan()；

2）两个对数函数的运算：ln()、lg()；

3）百分号计算百分数的运算，改善vivo、华为等手机的计算器无法准确计算10%+10%的问题；

4）实现一个数的阶乘运算；

5）增加π的计算；

6）增加e的运算；

7）按键M可获取上一次运算的结果；

8）按键tra可以使界面在基础运算和高级运算增加转换。

三、主要类源代码

1、MainActivity类

**package** com.example.calculator4.calculator;  
**import** android.os.Bundle;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** com.example.calculator4.R;  
**import** com.example.calculator4.service.StandardCalculator;  
**public class** MainActivity **extends** AppCompatActivity {  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 *//加载基础运算界面的布局文件* setContentView(R.layout.***calculator\_standard\_layout***);  
 *//调用StandardCalculator类* StandardCalculator standard = **new** StandardCalculator(**this**);  
 }  
}

2、ScientificCalculatorActivity类

**package** com.example.calculator4.calculator;  
**import** android.os.Bundle;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** com.example.calculator4.R;  
**import** com.example.calculator4.service.ScientificCalculator;

**public class** ScientificCalculatorActivity **extends** AppCompatActivity {  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 *//加载高级运算界面的布局文件* setContentView(R.layout.***calculator\_scientific\_layout***);  
 *//调用ScientificCalculator类* ScientificCalculator scientificCalculator = **new** ScientificCalculator(**this**);  
 }  
}

3、StandardCalculator类

**package** com.example.calculator4.service;  
**import** android.app.Activity;  
**import** android.content.Intent;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
**import** com.example.calculator4.R;  
**import** com.example.calculator4.Utils.Calculate;  
**import** com.example.calculator4.calculator.ScientificCalculatorActivity;  
**import** com.example.calculator4.common.InputItem;  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** java.util.List;  
**import** java.util.Map;

**public class** StandardCalculator {  
 **protected** Activity **activity**;  
 **protected** TextView **showResultTv**;  
 **protected** TextView **showInputTv**;  
 **protected** Button **cBtn**;  
 **protected** Button **delBtn**;  
 **protected** Button **resBtn**;  
 **protected** Button **divBtn**;  
 **protected** Button **sevenBtn**;  
 **protected** Button **eightBtn**;  
 **protected** Button **nineBtn**;  
 **protected** Button **mulBtn**;  
 **protected** Button **fourBtn**;  
 **protected** Button **fiveBtn**;  
 **protected** Button **sixBtn**;  
 **protected** Button **subBtn**;  
 **protected** Button **oneBtn**;  
 **protected** Button **twoBtn**;  
 **protected** Button **threeBtn**;  
 **protected** Button **addBtn**;  
 **protected** Button **transformBtn**;  
 **protected** Button **zeroBtn**;  
 **protected** Button **pointBtn**;  
 **protected** Button **equBtn**;  
 **protected** Map<View, String> **map**; *//将View和String映射起来* **protected** List<InputItem> **inputList**; *//定义记录每次输入的数* **protected** String **lastResult**;  
 **protected** String **expressionHistory**;  
 **protected** CurrentStatus **currentStatus** = CurrentStatus.***INIT***;  
  
 **public static enum** CurrentStatus {  
 ***INIT***, ***CALCU***,***END***, ***ERROR*** }  
  
 **public** StandardCalculator() {  
 }  
  
 **public** StandardCalculator(Activity activity) {  
 **this**.**activity** = activity;  
 init();  
 setOnClickListener();  
 }  
  
 **protected void** init() {  
 **showResultTv** = (TextView) **activity**.findViewById(R.id.***show\_result\_tv***);  
 **showInputTv** = (TextView) **activity**.findViewById(R.id.***show\_input\_tv***);  
 **cBtn** = (Button) **activity**.findViewById(R.id.***c\_btn***);  
 **delBtn** = (Button) **activity**.findViewById(R.id.***del\_btn***);  
 **resBtn** = (Button) **activity**.findViewById(R.id.***res\_btn***);  
 **divBtn** = (Button) **activity**.findViewById(R.id.***divide\_btn***);  
 **sevenBtn** = (Button) **activity**.findViewById(R.id.***seven\_btn***);  
 **eightBtn** = (Button) **activity**.findViewById(R.id.***eight\_btn***);  
 **nineBtn** = (Button) **activity**.findViewById(R.id.***nine\_btn***);  
 **mulBtn** = (Button) **activity**.findViewById(R.id.***multiply\_btn***);  
 **fourBtn** = (Button) **activity**.findViewById(R.id.***four\_btn***);  
 **fiveBtn** = (Button) **activity**.findViewById(R.id.***five\_btn***);  
 **sixBtn** = (Button) **activity**.findViewById(R.id.***six\_btn***);  
 **subBtn** = (Button) **activity**.findViewById(R.id.***sub\_btn***);  
 **oneBtn** = (Button) **activity**.findViewById(R.id.***one\_btn***);  
 **twoBtn** = (Button) **activity**.findViewById(R.id.***two\_btn***);  
 **threeBtn** = (Button) **activity**.findViewById(R.id.***three\_btn***);  
 **addBtn** = (Button) **activity**.findViewById(R.id.***add\_btn***);  
 **transformBtn** = (Button) **activity**.findViewById(R.id.***transform\_btn***);  
 **zeroBtn** = (Button) **activity**.findViewById(R.id.***zero\_btn***);  
 **pointBtn** = (Button) **activity**.findViewById(R.id.***point\_btn***);  
 **equBtn** = (Button) **activity**.findViewById(R.id.***equal\_btn***);  
 **map** = **new** HashMap<>();  
 **map**.put(**zeroBtn**, **"0"**);  
 **map**.put(**oneBtn**, **"1"**);  
 **map**.put(**twoBtn**, **"2"**);  
 **map**.put(**threeBtn**, **"3"**);  
 **map**.put(**fourBtn**, **"4"**);  
 **map**.put(**fiveBtn**, **"5"**);  
 **map**.put(**sixBtn**, **"6"**);  
 **map**.put(**sevenBtn**, **"7"**);  
 **map**.put(**eightBtn**, **"8"**);  
 **map**.put(**nineBtn**, **"9"**);  
 **map**.put(**pointBtn**, **activity**.getString(R.string.***point***));  
 **map**.put(**addBtn**, **"+"**);  
 **map**.put(**subBtn**, **"-"**);  
 **map**.put(**mulBtn**, **activity**.getString(R.string.***multiply***) );  
 **map**.put(**divBtn**, **activity**.getString(R.string.***divide***));  
 **map**.put(**equBtn**, **"="**);  
 **inputList** = **new** ArrayList<>();  
 **lastResult** = **""**;  
 initInput();  
 }  
  
 **protected void** setOnClickListener() {  
 **cBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 clear();  
 }  
 });  
 **delBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 delete();  
 }  
 });  
 **resBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputLastRes();  
 }  
 });  
 **divBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputOpe(v);  
 }  
 });  
 **mulBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputOpe(v);  
 }  
 });  
 **addBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputOpe(v);  
 }  
 });  
 **subBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputOpe(v);  
 }  
 });  
 **sevenBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNum(v);  
 }  
 });  
 **eightBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNum(v);  
 }  
 });  
 **nineBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNum(v);  
 }  
 });  
 **fourBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNum(v);  
 }  
 });  
 **fiveBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNum(v);  
 }  
 });  
 **sixBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNum(v);  
 }  
 });  
 **oneBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNum(v);  
 }  
 });  
 **twoBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNum(v);  
 }  
 });  
 **threeBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNum(v);  
 }  
 });  
 **zeroBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNum(v);  
 }  
 });  
 **pointBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNum(v);  
 }  
 });  
 **transformBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 transform();  
 }  
 });  
 **equBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 getResult(v);  
 }  
 });  
 }  
  
 **protected void** transform() {  
 Intent intent = **new** Intent(**activity**, ScientificCalculatorActivity.**class**);  
 **activity**.startActivity(intent);  
 }  
  
  
 **protected void** clear() {  
 **if** (!isInitInputList()) {  
 initInput();  
 Log.*i*(**activity**.getClass().getSimpleName(),**"cleaned inputTV \n inputTV: "**+**showInputTv**.getText()  
 +**"\n inputList:"**+getInputListValues());  
 } **else** {  
 **showResultTv**.setText(**""**);  
 Log.*i*(**activity**.getClass().getSimpleName(),**"cleaned resultTV"**);  
 }  
 }  
  
 **protected void** delete() {  
 **if** (!isInitInputList()) {  
 **if** (**inputList**.size() == 1) {  
 initInput();  
 } **else** {  
 Log.*i*(**activity**.getClass().getSimpleName(),**"delete input: "**+ subInputListAndInputTV().getValue());  
 }  
 }  
 }  
  
 **protected boolean** isInitInputList() { *//判断是否是初始状态* **if**(**inputList**.size()==1 && **inputList**.get(0).getValue().equals(**map**.get(**zeroBtn**))){  
 **return true**;  
 }  
 **return false**;  
 }  
  
 **protected void** initInput() {  
 **inputList**.clear();  
 **inputList**.add(**new** InputItem(**map**.get(**zeroBtn**), InputItem.TYPE.***NUM***));  
 **showInputTv**.setText(**map**.get(**zeroBtn**));  
 **currentStatus** = CurrentStatus.***INIT***;  
 }  
  
 **protected void** addTV(View view) {  
 **showInputTv**.setText(**showInputTv**.getText() + **map**.get(view));  
 }  
  
 **protected** String updateInputTV(){  
 String inputString = **""**;  
 **for**(InputItem inputItem: **inputList**){  
 inputString += inputItem.getValue();  
 }  
 **showInputTv**.setText(inputString);  
 **return** inputString;  
 }  
  
 **protected** InputItem subInputListAndInputTV(){  
 **if**(**inputList**.size()>0){  
 InputItem inputItem = **inputList**.remove(**inputList**.size()-1);  
 updateInputTV();  
 **return** inputItem;  
 }  
 **return null**;  
 }  
  
 **protected** InputItem getLastInputItem(){  
 **if**(**inputList**.size()>0){  
 **return inputList**.get(**inputList**.size()-1);  
 }  
 **return null**;  
 }  
  
 **protected void** inputLastRes(){  
 **if**(!**lastResult**.isEmpty() && (getLastInputItem().getType()!= InputItem.TYPE.***NUM*** || **currentStatus**==CurrentStatus.***INIT***)){  
 **for**(**int** i = 0;i<**lastResult**.length();i++){  
 inputNum(String.*valueOf*(**lastResult**.charAt(i)));  
 }  
 updateInputTV();  
 }  
 }  
  
 **protected void** inputOpe(View view) {  
 **if**(getLastInputItem().getType()== InputItem.TYPE.***OPE***){ *//连续输入两个OPR型操作符，删除第一个* Log.*i*(**activity**.getClass().getSimpleName(),**"delete OPT: "**+ subInputListAndInputTV().getValue());  
 }  
 **inputList**.add(**new** InputItem(**map**.get(view), InputItem.TYPE.***OPE***));  
 addTV(view);  
 Log.*i*(**activity**.getClass().getSimpleName(),**"Add OPT; "**+**map**.get(view)+  
 **"\ninputList: "**+getInputListValues());  
 }  
  
 **protected void** inputNum(View view){  
 inputNum(**map**.get(view));  
 }  
  
 **protected void** inputNum(String num) {  
 initEndStatus(); *//处理上次运算结束的情况* **if**(getLastInputItem().getType()== InputItem.TYPE.***OPE\_NUM***){  
 inputOpe(**mulBtn**);  
 }  
 **if** (isInitInputList() && (!num.equals(**"."**))) { *//初始状态下输入数字替换原始0，小数点不替换* Log.*i*(**activity**.getClass().getSimpleName(),**"delete inputTV: "**+subInputListAndInputTV().getValue());  
 }  
 **if**(num.equals(**"."**)){ *//限制一个数中最多只有一个小数点* **for** (**int** i = **inputList**.size() - 1; i > 0; i--) {  
 **if** (**inputList**.get(i).getType() != InputItem.TYPE.***NUM***) {  
 **break**;  
 } **else if** (**inputList**.get(i).getValue() == **map**.get(**pointBtn**)) {  
 **return**;  
 }  
 }  
 }  
 **inputList**.add(**new** InputItem(num, InputItem.TYPE.***NUM***));  
 **showInputTv**.setText(**showInputTv**.getText() + num);  
 Log.*i*(**activity**.getClass().getSimpleName(), **"add: "** + num+  
 **"\ninputList: "** + getInputListValues());  
 }  
  
 **protected** String getInputListValues(){ *//给日志打印inputList信息* String s = **""**;  
 **for**(InputItem inputItem: **inputList**){  
 s += inputItem.getValue()+**" "**;  
 }  
 **return** s;  
 }  
  
 **protected void** initEndStatus(){ *//上次计算结束后，初始化input* **if**(**currentStatus** == CurrentStatus.***END*** && getLastInputItem().getType()== InputItem.TYPE.***NUM***){  
 **showResultTv**.setText(String.*valueOf*(**showResultTv**.getText())+**showInputTv**.getText());  
 initInput();  
 **currentStatus** = CurrentStatus.***CALCU***;  
 }  
 }  
  
 **protected void** getResult(View view) {  
 **if** (**showInputTv**.getText().charAt(**showInputTv**.length() - 1) != **'='**) {  
 addTV(view);  
 }  
 String res = **""**;  
 **try**{  
 res = Calculate.*getResult*(**inputList**,**activity**); *//取得计算结果* resultManage(res); *//处理界面显示信息* }**catch** (ArithmeticException e){  
 res = **"0不能做除数！"**;  
 Toast.*makeText*(**activity**,res,Toast.***LENGTH\_SHORT***).show();  
 }**catch** (Exception e){  
 res = **"算术式不正确！"**;  
 Toast.*makeText*(**activity**,res,Toast.***LENGTH\_SHORT***).show();  
 }  
 Log.*i*(**activity**.getClass().getSimpleName(),**"got result: "**+res+  
 **"\ninputList:"**+getInputListValues());  
 }  
  
 **protected void** resultManage(String result){  
 **showResultTv**.setText(**showInputTv**.getText());  
 **showInputTv**.setText(result);  
 **inputList**.clear();  
 **lastResult** = result;  
 **for**(**char** c : result.toCharArray()){  
 **inputList**.add(**new** InputItem(String.*valueOf*(c), InputItem.TYPE.***NUM***));  
 }  
 **currentStatus** = CurrentStatus.***END***;  
 }  
}

4、ScientificCalculator类

**package** com.example.calculator4.service;  
**import** android.app.Activity;  
**import** android.content.Intent;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** com.example.calculator4.R;  
**import** com.example.calculator4.calculator.MainActivity;  
**import** com.example.calculator4.common.InputItem;  
  
**public class** ScientificCalculator **extends** StandardCalculator {  
 **protected** Button **antiBtn**;  
 **protected** Button **perCent**;  
 **protected** Button **sinBtn**;  
 **protected** Button **cosBtn**;  
 **protected** Button **tanBtn**;  
 **protected** Button **powerBtn**;  
 **protected** Button **lgBtn**;  
 **protected** Button **lnBtn**;  
 **protected** Button **leftBraBtn**;  
 **protected** Button **rightBraBtn**;  
 **protected** Button **squareRootBtn**;  
 **protected** Button **factorialBtn**;  
 **protected** Button **reciprocalBtn**;  
 **protected** Button **pIBtn**;  
 **protected** Button **eBtn**;  
  
 **public** ScientificCalculator(Activity activity){  
 **super**(activity);  
 }  
  
 @Override  
 **protected void** init() {  
 **super**.init();  
 **antiBtn** = (Button) **activity**.findViewById(R.id.***anti\_btn***);  
 **sinBtn** = (Button) **activity**.findViewById(R.id.***sin\_btn***);  
 **cosBtn** = (Button) **activity**.findViewById(R.id.***cos\_btn***);  
 **tanBtn** = (Button) **activity**.findViewById(R.id.***tan\_btn***);  
 **powerBtn** = (Button) **activity**.findViewById(R.id.***power\_btn***);  
 **lgBtn** = (Button) **activity**.findViewById(R.id.***log\_btn***);  
 **lnBtn** = (Button) **activity**.findViewById(R.id.***ln\_btn***);  
 **leftBraBtn** = (Button) **activity**.findViewById(R.id.***leftBracket\_btn***);  
 **rightBraBtn** = (Button) **activity**.findViewById(R.id.***rightBracket\_btn***);  
 **squareRootBtn** = (Button) **activity**.findViewById(R.id.***squareRoot\_btn***);  
 **factorialBtn** = (Button) **activity**.findViewById(R.id.***factorial\_btn***);  
 **reciprocalBtn** = (Button) **activity**.findViewById(R.id.***reciprocal\_btn***);  
 **pIBtn** = (Button) **activity**.findViewById(R.id.***PI\_btn***);  
 **eBtn** = (Button) **activity**.findViewById(R.id.***e\_btn***);  
 **perCent** = (Button) **activity**.findViewById(R.id.***perCent\_btn***);  
 **map**.put(**sinBtn**,**activity**.getString(R.string.***sin***));  
 **map**.put(**cosBtn**,**activity**.getString(R.string.***cos***));  
 **map**.put(**tanBtn**,**activity**.getString(R.string.***tan***));  
 **map**.put(**powerBtn**,**activity**.getString(R.string.***power***));  
 **map**.put(**perCent**, **"%"**);  
 **map**.put(**lgBtn**,**activity**.getString(R.string.***log***));  
 **map**.put(**lnBtn**,**activity**.getString(R.string.***ln***));  
 **map**.put(**leftBraBtn**,**activity**.getString(R.string.***leftBra***));  
 **map**.put(**rightBraBtn**,**activity**.getString(R.string.***rightBra***));  
 **map**.put(**squareRootBtn**,**activity**.getString(R.string.***squareRoot***));  
 **map**.put(**factorialBtn**,**activity**.getString(R.string.***factorial***));  
 **map**.put(**reciprocalBtn**,**activity**.getString(R.string.***reciprocal***));  
 **map**.put(**pIBtn**,**activity**.getString(R.string.***PI***));  
 **map**.put(**eBtn**,**"e"**);  
 }  
  
 @Override  
 **protected void** setOnClickListener() {  
 **super**.setOnClickListener();  
 **powerBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputOpe(v);  
 }  
 });  
 **perCent**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputOpeOpe(v);  
 }  
 });  
 **antiBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 transTrigon();  
 }  
 });  
 **sinBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNumOpe(v);  
 }  
 });  
 **cosBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNumOpe(v);  
 }  
 });  
 **tanBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNumOpe(v);  
 }  
 });  
 **lgBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNumOpe(v);  
 }  
 });  
 **lnBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNumOpe(v);  
 }  
 });  
 **squareRootBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputNumOpe(v);  
 }  
 });  
 **leftBraBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputLeftBra();  
 }  
 });  
 **rightBraBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputRightBra();  
 }  
 });  
 **factorialBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputOpeOpe(v);  
 }  
 });  
 **reciprocalBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputOpeOpe(v);  
 }  
 });  
 **pIBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputOpeNum(v);  
 }  
 });  
 **eBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 inputOpeNum(v);  
 }  
 });  
 }  
  
 @Override  
 **protected void** transform() {  
 Intent intent = **new** Intent(**activity**, MainActivity.**class**);  
 **activity**.startActivity(intent);  
 }  
  
 **protected void** transTrigon(){  
 **if**(**antiBtn**.getText().equals(**activity**.getString(R.string.***radian***))){  
 **antiBtn**.setText(**activity**.getString(R.string.***deg***));  
 }**else if**(**antiBtn**.getText().equals(**activity**.getString(R.string.***deg***))){  
 **antiBtn**.setText(**activity**.getString(R.string.***radian***));  
 }  
 }  
  
 **protected void** inputTrigonometricFunction(View view){  
 }  
  
 **protected void** inputNumOpe(View view){  
 **if**( **currentStatus** == CurrentStatus.***END***){  
 initEndStatus();  
 }  
 **if**(isInitInputList()){  
 subInputListAndInputTV();  
 } **else if**(getLastInputItem().getType()== InputItem.TYPE.***NUM*** ||  
 getLastInputItem().getType()== InputItem.TYPE.***RIGHT\_BRACKET***){  
 inputOpe(**mulBtn**);  
 }  
 **inputList**.add(**new** InputItem(**map**.get(view), InputItem.TYPE.***NUM\_OPE***));  
 addTV(view);  
 inputLeftBra();  
 Log.*i*(**activity**.getClass().getSimpleName(),**"add ope: "**+**map**.get(view)+  
 **"\ninputList: "**+getInputListValues());  
 }  
  
 **protected void** inputOpeOpe(View view) {  
 **if** (getLastInputItem().getType()!= InputItem.TYPE.***NUM*** *//OPT\_OPT前必须是数字或右括号* && getLastInputItem().getType()!= InputItem.TYPE.***RIGHT\_BRACKET***) {  
 Log.*i*(**activity**.getClass().getSimpleName(),**"Last input is "**+  
 getLastInputItem().getValue()+**", can not add "**+**map**.get(view));  
 **return**;  
 }  
 **inputList**.add(**new** InputItem(**map**.get(view), InputItem.TYPE.***OPE\_OPE***));  
 addTV(view);  
 Log.*i*(**activity**.getClass().getSimpleName(),**"inputList: "**+getInputListValues());  
 }  
  
 **protected void** inputOpeNum(View view){  
 **if**( **currentStatus** == CurrentStatus.***END***){  
 initEndStatus();  
 }  
 **if**(isInitInputList()){  
 subInputListAndInputTV();  
 }**else if**(getLastInputItem().getType()== InputItem.TYPE.***NUM*** ||  
 getLastInputItem().getType()== InputItem.TYPE.***RIGHT\_BRACKET***){  
 inputOpe(**mulBtn**);  
 }  
 **inputList**.add(**new** InputItem(**map**.get(view), InputItem.TYPE.***OPE\_NUM***));  
 addTV(view);  
 Log.*i*(**activity**.getClass().getSimpleName(),**"add ope: "**+**map**.get(view)+  
 **"\ninputList: "**+getInputListValues());  
 }  
  
 **protected void** inputLeftBra(){  
 **if**( **currentStatus** == CurrentStatus.***END***){  
 initEndStatus();  
 }  
 **if**(isInitInputList()){  
 subInputListAndInputTV();  
 }**else if**(getLastInputItem().getType()!= InputItem.TYPE.***OPE*** && getLastInputItem().getType()!= InputItem.TYPE.***NUM\_OPE*** && getLastInputItem().getType()!= InputItem.TYPE.***LEFT\_BRACKET***){  
 inputOpe(**mulBtn**);  
 }  
 **inputList**.add(**new** InputItem(**map**.get(**leftBraBtn**), InputItem.TYPE.***LEFT\_BRACKET***));  
 addTV(**leftBraBtn**);  
 Log.*i*(**activity**.getClass().getSimpleName(),**"add ope: "**+**map**.get(**leftBraBtn**)+  
 **"\ninputList: "**+getInputListValues());  
 }  
  
 **protected void** inputRightBra(){  
 **if**(getLastInputItem().getType()== InputItem.TYPE.***OPE***){  
 subInputListAndInputTV();  
 }  
 **inputList**.add(**new** InputItem(**map**.get(**rightBraBtn**), InputItem.TYPE.***RIGHT\_BRACKET***));  
 addTV(**rightBraBtn**);  
 Log.*i*(**activity**.getClass().getSimpleName(),**"add ope: "**+**map**.get(**rightBraBtn**)+  
 **"\ninputList: "**+getInputListValues());  
 }  
}

5、InputItem类

**package** com.example.calculator4.common;  
**import** java.util.List;  
**import** java.util.Map;  
  
**public class** InputItem {  
 **private** String **value**;  
 **private** TYPE **type**;  
 **private** List<TYPE> **frontType**;

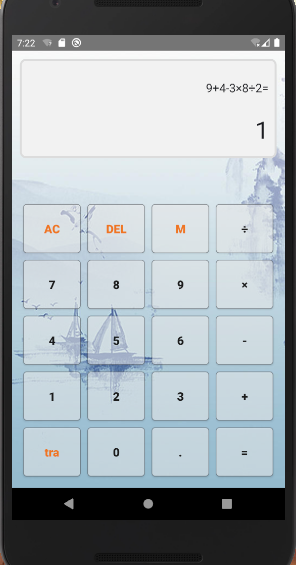
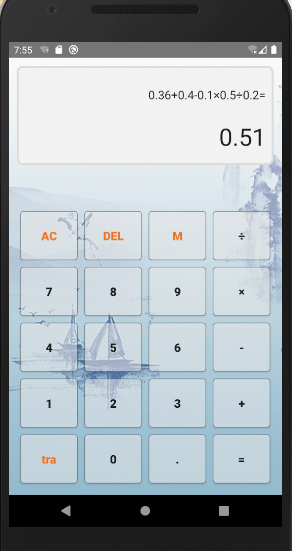
*/\*  
 \* 为了提高用户输入容错（如一个数字不能输入两个小数点、不能连续输入两个运算符等等），  
 \* 本系统将运算表达式字符分为：  
 \* NUM(数字型)，包括小数点和0-9的数字，可以连续输入数字型；  
 \* OPE(操作符)，包括加减乘除乘方，不能连续输入，它们前面和后面都必须加数字型或其他类型运算符  
 \* OPE\_NUM(类操作符的数字型)，包括PI和e，叫OPE\_NUM是因为它们像操作符一样不能直接加数字（如不能输入PI9，e8等），但又像数字一样可以加运算符  
 \* NUM\_OPE(类数字型的操作符)，包括三角函数运算符,对数运算符，指数运算符，根号，叫NUM\_OPE是因为它们前面像数字一样加操作符，后面像操作符一样加数字  
 \* OPE\_OPE，包括倒数运算，阶乘运算和百分号运算，前面加数字，后面加操作符  
 \* LEFT\_BRACKET(左括号)  
 \* RIGHT\_BRACKET(右括号)\* \*/* **public static enum** TYPE{  
 ***NUM***, *//'.'and'0-9'* ***OPE***, *//加减乘除 乘方* ***OPE\_NUM***, *//PI,e* ***NUM\_OPE***, *//sin,lg,ln,root,* ***OPE\_OPE***, *//倒数，阶乘，,%* ***LEFT\_BRACKET***,  
 ***RIGHT\_BRACKET***,  
 }  
  
 **public** InputItem(){}  
  
 **public** InputItem(String value, TYPE type){  
 **this**.**value** = value;  
 **this**.**type** = type;  
 }  
  
 **public** String getValue() {  
 **return value**;  
 }  
  
 **public void** setValue(String value) {  
 **this**.**value** = value;  
 }  
  
 **public** TYPE getType() {  
 **return type**;  
 }  
  
 **public void** setType(TYPE type) {  
 **this**.**type** = type;  
 }  
}

6、Calculate类

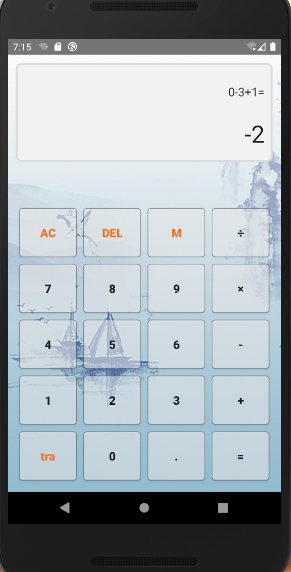
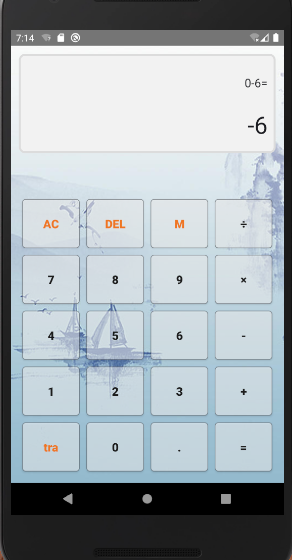
**package** com.example.calculator4.Utils;  
**import** android.content.Context;  
**import** android.util.Log;  
**import** com.example.calculator4.R;  
**import** com.example.calculator4.common.InputItem;  
**import** java.math.BigDecimal;  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** java.util.Stack;  
  
**public class** Calculate {  
  
 */\*此求值算法的核心是常用的双栈处理方式；  
 对于操作数直接压栈，对于操作符，先查看栈顶操作数的优先级，如果栈顶元素优先级低于当前操作数优先级，  
 当前操作符压栈，如果不低于，栈顶操作符出栈计算；\*/* **public static** String getResult(List<InputItem> inputItemList, Context context)  
 **throws** ArithmeticException {  
 Stack<String> numberStack = **new** Stack<>();  
 Stack<String> operatorStack = **new** Stack<>();  
 List<InputItem> expressionList = *preconditioning*(inputItemList);  
 **for** (InputItem item : expressionList) {  
 Log.*d*(**"Calculate"**, **"InputItem: "** + item.getValue()+**"\n numberStack:"** +numberStack.toString()+**"\n operatorStack: "**+operatorStack.toString());  
 **if** (item.getType() == InputItem.TYPE.***NUM***) {  
 numberStack.push(item.getValue());  
 } **else if** (item.getType() == InputItem.TYPE.***OPE\_NUM***) {  
 **if** (item.getValue().equals(context.getString(R.string.***PI***))) {  
 numberStack.push(String.*valueOf*(Math.***PI***));  
 } **else if** (item.getValue().equals(context.getString(R.string.***e***))) {  
 numberStack.push(String.*valueOf*(Math.***E***));  
 }  
 } **else** {  
 **if** (item.getType() == InputItem.TYPE.***LEFT\_BRACKET***) {  
 operatorStack.push(item.getValue());  
 } **else if** (item.getType() == InputItem.TYPE.***RIGHT\_BRACKET***) {  
 **while** (!operatorStack.empty() && !operatorStack.peek().equals(context.getString(R.string.***leftBra***))) {  
 *processAnOperator*(numberStack, operatorStack, context);  
 }  
 **if**(!operatorStack.empty()){  
 operatorStack.pop();  
 }  
 } **else if** (item.getType() != InputItem.TYPE.***OPE*** || item.getValue().equals(context.getString(R.string.***power***))) {  
 **while** (!operatorStack.empty() && !operatorStack.peek().equals(**"+"**) &&  
 !operatorStack.peek().equals(**"-"**) && !operatorStack.peek().equals(  
 context.getString(R.string.***multiply***)) && !operatorStack.peek().equals(**"+"**) && !operatorStack.peek().equals(context.getString(R.string.***divide***))  
 && !operatorStack.peek().equals(context.getString(R.string.***leftBra***))) {  
 *processAnOperator*(numberStack, operatorStack, context);  
 }  
 operatorStack.push(item.getValue());  
 } **else if** (item.getValue().equals(context.getString(R.string.***multiply***)) || item.getValue().equals(context.getString(R.string.***divide***))) {  
 **while** (!operatorStack.empty() && !operatorStack.peek().equals(**"+"**) &&  
 !operatorStack.peek().equals(**"-"**) && !operatorStack.peek().equals(context.getString(R.string.***leftBra***))) {  
 *processAnOperator*(numberStack, operatorStack, context);  
 }  
 operatorStack.push(item.getValue());  
 } **else if** (item.getValue().equals(**"+"**) || item.getValue().equals(**"-"**)) {  
 **while** (!operatorStack.empty() && !operatorStack.peek().equals(context.getString(R.string.***leftBra***))) {  
 *processAnOperator*(numberStack, operatorStack, context);  
 }  
 operatorStack.push(item.getValue());  
 }  
 }  
 }  
 **while** (!operatorStack.isEmpty()) {  
 *processAnOperator*(numberStack, operatorStack, context);  
 }  
 **return** *dealWithResult*(numberStack.pop());  
 }  
  
 **protected static** String dealWithResult(String result){ *//修正浮点计算带来的误差* String res = String.*format*(**"%.12f"**,Double.*parseDouble*(result));  
 **for**(**int** i = res.length()-1;i>0;i--){  
 **if**(res.charAt(i)==**'.'**){  
 **return** res.substring(0,i);  
 }**else if**(res.charAt(i)!=**'0'**){  
 **break**;  
 } **else** {  
 res = res.substring(0,i);  
 }  
 }  
 **if**(res.equals(**"-0"**)){  
 **return "0"**;  
 }  
 **return** res;  
 }  
  
 **protected static void** processAnOperator(Stack<String> numberStack, Stack<String> operatorStack, *//求值* Context context) **throws** ArithmeticException {  
 String ope = operatorStack.pop();  
 Log.*i*(**"Calculate"**, **"operator: "** + numberStack.peek() + ope);  
 **if** (ope.equals(**"+"**)) {  
 numberStack.push(**new** BigDecimal(numberStack.pop()).add(**new** BigDecimal(numberStack.pop())).toString());  
 } **else if** (ope.equals(**"-"**)) {  
 BigDecimal bigDecimal = **new** BigDecimal(numberStack.pop());  
 numberStack.push(**new** BigDecimal(numberStack.pop()).subtract(bigDecimal).toString());  
 } **else if** (ope.equals(context.getString(R.string.***multiply***))) {  
 numberStack.push(**new** BigDecimal(numberStack.pop()).multiply(**new** BigDecimal(numberStack.pop())).toString());  
 } **else if** (ope.equals(context.getString(R.string.***divide***))) {  
 BigDecimal bigDecimal1 = **new** BigDecimal(numberStack.pop());  
 BigDecimal bigDecimal2 = **new** BigDecimal(numberStack.pop());  
 **try** {  
 numberStack.push(bigDecimal2.divide(bigDecimal1).toString());  
 } **catch** (ArithmeticException e) {  
 Log.*e*(**"Calculate"**, **"divide operation : divide "** + bigDecimal1 + **"\n"** + e);  
 numberStack.push(bigDecimal2.divide(bigDecimal1, 12, BigDecimal.***ROUND\_HALF\_UP***).toString());  
 }  
 } **else if** (ope.equals(context.getString(R.string.***perCent***))) {  
 numberStack.push(**new** BigDecimal(numberStack.pop()).divide(**new** BigDecimal(100)).toString());  
 } **else if** (ope.equals(context.getString(R.string.***sin***))) {  
 numberStack.push(String.*valueOf*(Math.*sin*(Double.*parseDouble*(numberStack.pop()))));  
 } **else if** (ope.equals(context.getString(R.string.***cos***))) {  
 numberStack.push(String.*valueOf*(Math.*cos*(Double.*parseDouble*(numberStack.pop()))));  
 } **else if** (ope.equals(context.getString(R.string.***tan***))) {  
 numberStack.push(String.*valueOf*(Math.*tan*(Double.*parseDouble*(numberStack.pop()))));  
 } **else if** (ope.equals(context.getString(R.string.***power***))) {  
 Double d = Double.*parseDouble*(numberStack.pop());  
 numberStack.push(String.*valueOf*(Math.*pow*(Double.*parseDouble*(numberStack.pop()), d)));  
 } **else if** (ope.equals(context.getString(R.string.***log***))) {  
 numberStack.push(String.*valueOf*(Math.*log10*(Double.*parseDouble*(numberStack.pop()))));  
 } **else if** (ope.equals(context.getString(R.string.***ln***))) {  
 numberStack.push(String.*valueOf*(Math.*log*(Double.*parseDouble*(numberStack.pop()))));  
 } **else if** (ope.equals(context.getString(R.string.***squareRoot***))) {  
 numberStack.push(String.*valueOf*(Math.*sqrt*(Double.*parseDouble*(numberStack.pop()))));  
 } **else if** (ope.equals(context.getString(R.string.***factorial***))) {  
 **int** a = 1;  
 Double n = Math.*floor*(Double.*parseDouble*(numberStack.pop()));  
 **for** (**int** i = 1; i <= n; i++) {  
 a \*= i;  
 }  
 **if** (n == 0.0) {  
 a = 0;  
 }  
 numberStack.push(String.*valueOf*(a));  
 } **else if** (ope.equals(context.getString(R.string.***reciprocal***))) {  
 numberStack.push(String.*valueOf*(Math.*pow*(Double.*parseDouble*(numberStack.pop()), 2)));  
 }  
 Log.*i*(**"Calculate"**, **"processAnOperator result: "** + numberStack.peek());  
 }  
  
 **protected static** List<InputItem> preconditioning(List<InputItem> inputItemList) { *//将用户输入结果转换为表达式，即数字字符合并为数字* List<InputItem> list = **new** ArrayList<>();  
 String numberString = **""**;  
 **for** (InputItem inputItem : inputItemList) {  
 **if** (inputItem.getType().equals(InputItem.TYPE.***NUM***)) {  
 numberString += inputItem.getValue();  
 } **else** {  
 **if** (numberString != **""**) {  
 list.add(**new** InputItem(numberString, InputItem.TYPE.***NUM***));  
 numberString = **""**;  
 }  
 list.add(inputItem);  
 }  
 }  
 **if** (numberString != **""**) {  
 list.add(**new** InputItem(numberString, InputItem.TYPE.***NUM***));  
 }  
 String expression = **""**;  
 **for** (InputItem inputItem : list) {  
 expression += inputItem.getValue();  
 }  
 Log.*i*(**"Calculate"**, **"expression: "** + expression);  
 **return** list;  
 }  
}

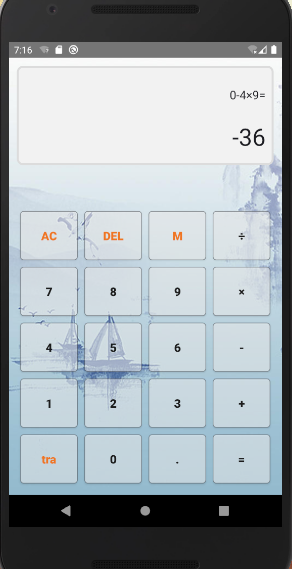
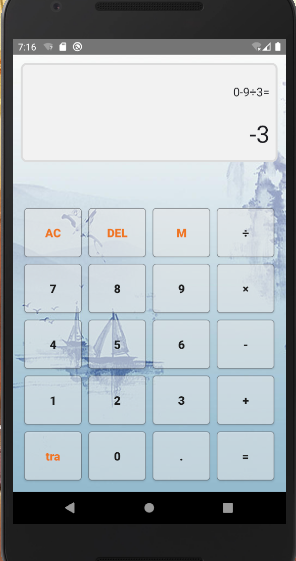
四、功能运行截图

1、基础运算界面的正数加减乘除

2、基础运算界面的负数加减乘除

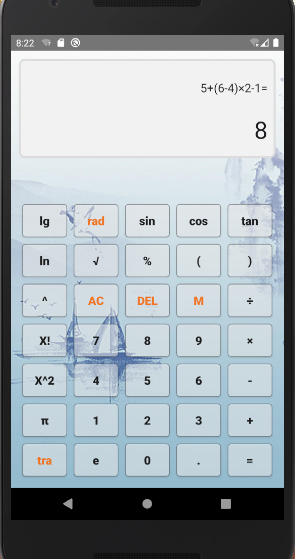
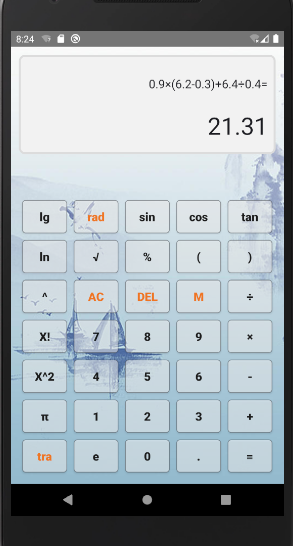
 

3、高级运算界面的运算

4、括号的优先运算

5、π的运算

6、e的运算

7、三角函数的运算

8、对数函数的运算

9、阶乘的运算

10、百分号的运算

11、平方运算、开平方运算与幂运算

12、在前一步的运算结果上继续运算

五、参考文献

1、《基于Android Studio的应用程序开发教程》

李宁宁，电子工业出版社，2016年07月

2、《第一行代码：Android（第2版）》

郭霖，人民邮电出版社，2016年11月

3、《Android Studio项目开发实战——从基础入门到趣味开发》

马静,邝楚文,肖国金，电子工业出版社，2020年3月

六、设计中的难点与解决方法

1、UI布局不符合预期

问题：书本建议使用的TableLayout布局设计出的UI界面不满足预期，总是偏左或偏右，控制控件居中的代码不起作用，严重影响背景图片的设置。

解决：改用多个水平和垂直的LinearLayout互相嵌套的方式设计，背景图片设置在第一个垂直的LinearLayout里即可。

2、按钮透明效果

问题：设计效果为按钮半透明，使背景图片能透过图片，但是网络上推荐的半透明代码调用起来是黑色的，全透明则是白色，皆不符合设计要求，使用alpha代码来控制透明度的方法也失败了。

解决：新建样式文件，分别在两个样式文件中定义未选中时的状态和选中时的状态，控制未选中时状态的样式文件中定义透明色，并进行调色，接下来只要在布局文件中引用该样式文件即可。

3、计算器功能设计

问题：算法的逻辑算法设计过于复杂。

解决：import java.math.BigDecimal包，直接调用相应方法即可。