**实 验 报 告**

**课程名称** 移动应用系统

**实验项目**  基本移动程序界面设计

**实验仪器** 台式电脑

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验报告评分 | | | | | |
| 一（15分） | 二（20分） | 三（30分） | 四（25分） | 参考文献（10分） | 总分 |
|  |  |  |  |  |  |

**专 业** 计算机科学与技术

**班级/学号** **计科1905 /2019010132**

**学生姓名** 陈浩

**实验日期** 2020-10-22

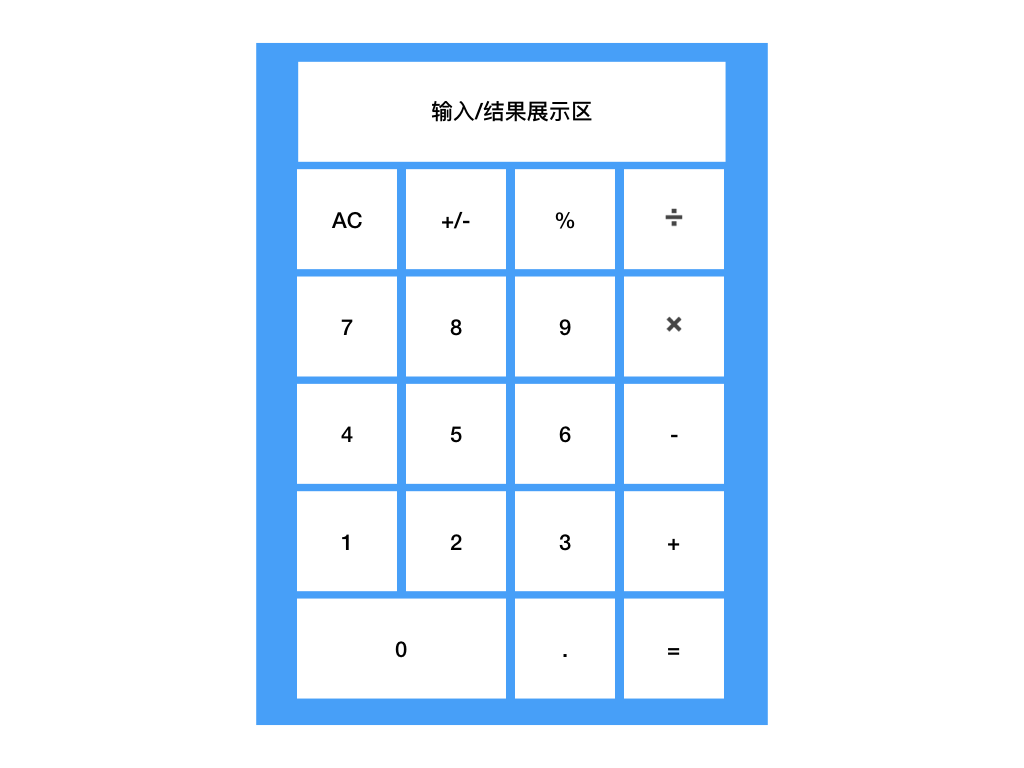
**成 绩**

**指导教师** 陈雷

实验**一 基本移动程序界面设计**

1. **需求分析（15分）**

1、设计并实现一个计算器程序，包含类似于下图的用户界面(下图为示意图，实际布局可以自行决定)：



具体要求：

1. 支持十进制正整数的加减乘除运算，可以不考虑小数点。
2. 必须考虑输入数据的校验，若用户输入非法数据，需要做适当的处理。例如，在除法中，若除数为0，需要给出提示。可以选择在“输入/结果展示区”进行适当的提示，也可以选择使用Toast给出提示

2、在基本要求的基础上，添加下列功能：

1）考虑正负数的情况（提供一个按钮切换正负数）

2）考虑带小数点的浮点数的加减乘除运算

3、在基本要求和扩展要求一基础上，添加下列功能：

1）实现开平方根运算,对于负数开平方根,以Toast的形式给出错误提示

注：除开平方根外，也可以实现其他的数学运算，如求三角函数（sin/cos），阶乘等等

1. **软件设计（20分）**

主要采用在 activity\_main.xml 中设计页面和布局，在MainActivity.Java中监听调用布局页面所定义的Button来实现点击获取输入并显示在EditTest上，并且在错误输入地方采用Toast输出报错语句。

本次设计采用了TableLayout将页面划分为显示内容区和输入按钮区，并额外添加了Toolbar修改了程序上方的标题栏，改进了美观。

1. **核心程序代码及运行结果（30分）**

*@Override  
 public void* onClick(View v) {  
  
 *switch* (v.getId()){  
  
 *// 操作数 (进行等号点击判断)  
 case* R.id.num\_zero:  
 existedText = isOver(existedText,"0");  
 *break*;  
 *case* R.id.num\_one:  
 existedText = isOver(existedText,"1");  
 *break*;  
 *case* R.id.num\_two:  
 existedText = isOver(existedText,"2");  
 *break*;  
 *case* R.id.num\_three:  
 existedText = isOver(existedText,"3");  
 *break*;  
 *case* R.id.num\_four:  
 existedText = isOver(existedText,"4");  
 *break*;  
 *case* R.id.num\_five:  
 existedText = isOver(existedText,"5");  
 *break*;  
 *case* R.id.num\_six:  
 existedText = isOver(existedText,"6");  
 *break*;  
 *case* R.id.num\_seven:  
 existedText = isOver(existedText,"7");  
 *break*;  
 *case* R.id.num\_eight:  
 existedText = isOver(existedText,"8");  
 *break*;  
 *case* R.id.num\_nine:  
 existedText = isOver(existedText,"9");  
 *break*;  
  
 *// 加法运算:  
 case* R.id.plus\_btn:  
  
 *// 错误语句判断  
 // 若输出了错误语句 则所有操作符(除删除操作符)均失效  
 if*(mistakeJudge()){}  
  
 *else* {  
 *// 判断表达式是否可以进行计算  
 // 若先前有符号位和两个数值 则先计算再添加符号  
 // 若先前只有一个数值 则先添加符号  
  
 // 符合运算操作时 (即两个操作数时)  
 if* (judgeExpression()) {  
 existedText = getResult();  
 existedText += "+";  
 }  
  
 *else* {  
 *// 计算后自动设等号为未点击  
 if* (isCounted) {  
 isCounted = *false*;  
 }  
  
 *// 只有单个数时  
 // 确保连续按操作数时不会出现多个操作数  
 if* ((existedText.substring(existedText.length() - 1)).equals("-")) {  
 existedText = existedText.replace("-", "+");  
 } *else if* ((existedText.substring(existedText.length() - 1)).equals("×")) {  
 existedText = existedText.replace("×", "+");  
 } *else if* ((existedText.substring(existedText.length() - 1)).equals("÷")) {  
 existedText = existedText.replace("÷", "+");  
 } *else if* (!(existedText.substring(existedText.length() - 1)).equals("+")) {  
 existedText += "+";  
 }  
  
 }  
 }  
 *break*;  
  
 *// 减法运算:  
 case* R.id.subtract\_btn:  
  
 *if*(mistakeJudge()){}  
  
 *else* {  
 *if* (judgeExpression()) {  
 existedText = getResult();  
 existedText += "-";  
 }  
 *else* {  
  
 *if* (isCounted) {  
 isCounted = *false*;  
 }  
  
 *if* ((existedText.substring(existedText.length() - 1)).equals("+")) {  
 existedText = existedText.replace("+", "-");  
 } *else if* ((existedText.substring(existedText.length() - 1)).equals("×")) {  
 existedText = existedText.replace("×", "-");  
 } *else if* ((existedText.substring(existedText.length() - 1)).equals("÷")) {  
 existedText = existedText.replace("÷", "-");  
 } *else if* (!(existedText.substring(existedText.length() - 1)).equals("-")) {  
 existedText += "-";  
 }  
 }  
 }  
 *break*;  
  
 *// 乘法运算:  
 case* R.id.multiply\_btn:  
  
 *if*(mistakeJudge()){}  
  
 *else* {  
 *if* (judgeExpression()) {  
 existedText = getResult();  
 existedText += "×";  
 }  
 *else* {  
  
 *if* (isCounted) {  
 isCounted = *false*;  
 }  
  
 *if* ((existedText.substring(existedText.length() - 1)).equals("+")) {  
 existedText = existedText.replace("+", "×");  
 } *else if* ((existedText.substring(existedText.length() - 1)).equals("-")) {  
 existedText = existedText.replace("-", "×");  
 } *else if* ((existedText.substring(existedText.length() - 1)).equals("÷")) {  
 existedText = existedText.replace("÷", "×");  
 } *else if* (!(existedText.substring(existedText.length() - 1)).equals("×")) {  
 existedText += "×";  
 }  
 }  
 }  
 *break*;  
  
 *// 除法运算:  
 case* R.id.divide\_btn:  
  
 *if*(mistakeJudge()){}  
  
 *else*{  
 *if* (judgeExpression()) {  
 existedText = getResult();  
 existedText += "÷";  
 }  
 *else* {  
  
 *if* (isCounted) {  
 isCounted = *false*;  
 }  
  
 *if* ((existedText.substring(existedText.length() - 1)).equals("+")) {  
 existedText = existedText.replace("+", "÷");  
 } *else if* ((existedText.substring(existedText.length() - 1)).equals("-")) {  
 existedText = existedText.replace("-", "÷");  
 } *else if* ((existedText.substring(existedText.length() - 1)).equals("×")) {  
 existedText = existedText.replace("×", "÷");  
 } *else if* (!(existedText.substring(existedText.length() - 1)).equals("÷")) {  
 existedText += "÷";  
 }  
 }  
 }  
 *break*;  
  
 *// 点击等号时出结果:  
 // 并设置为已点击等号(isCounted)  
 case* R.id.equal\_btn:  
 existedText = getResult();  
 isCounted = *true*;  
 *break*;  
  
 *// 小数点:  
 case* R.id.pot\_btn:  
  
 *// 所添加的错误判断：  
 if*(mistakeJudge()) {}  
  
 *else* {  
 *// 未运算过(未点击等号):  
 if* (!isCounted){  
  
 *// 拥有运算式时  
 if* (existedText.contains("+") || existedText.contains("-") ||  
 existedText.contains("×") || existedText.contains("÷") ){  
  
 String param1 = *null*;  
 String param2 = *null*;  
  
 *if* (existedText.contains("+")) {  
 param1 = existedText.substring(0, existedText.indexOf("+"));  
 param2 = existedText.substring(existedText.indexOf("+") + 1);  
 }  
 *else if* (existedText.contains("-")) {  
 param1 = existedText.substring(0, existedText.indexOf("-"));  
 param2 = existedText.substring(existedText.indexOf("-") + 1);  
 }  
 *else if* (existedText.contains("×")) {  
 param1 = existedText.substring(0, existedText.indexOf("×"));  
 param2 = existedText.substring(existedText.indexOf("×") + 1);  
 }  
 *else if* (existedText.contains("÷")) {  
 param1 = existedText.substring(0, existedText.indexOf("÷"));  
 param2 = existedText.substring(existedText.indexOf("÷") + 1);  
 }  
  
 *// 判断后操作数是否为小数  
 assert* param2 != *null*;  
 *boolean* isContainedPoint = param2.contains(".");  
  
 *// 确保不会出现多个小数点的情况  
 // 没有小数点时  
 if* (!isContainedPoint){  
 *// 若第二操作数为空  
 if* (param2.equals("")){  
 existedText = "无法添加小数点!";  
 setToast(MainActivity.*this*, existedText);  
 *break*;  
 }  
 *// 若第二操作数不为空  
 // 则最多只能添加1个小数点  
 else* {  
 existedText += ".";  
 }  
 }  
 *// 若已包含小数点则不做操作* }  
  
 *// 仅有单个数时  
 else* {  
  
 *boolean* isContainedPoint = existedText.contains(".");  
  
 *// 不包含小数点 (即可添加小数点)  
 if* (!isContainedPoint) {  
 *if*(existedText.equals("无法添加小数点!"))  
 *break*;  
 existedText += ".";  
 }  
 *// 若已包含小数点则不做操作* }  
 isCounted = *false*;  
  
 }  
  
 *else* {  
 *// 已经出结果  
 boolean* isContainedPoint = existedText.contains(".");  
  
 *// 不含小数点时 最多添加1个小数点  
 if*(!isContainedPoint)  
 existedText += ".";  
 *// 若已包含小数点则不做操作* isCounted = *false*;  
 }  
 }  
 *break*;  
  
 *// 取余操作:  
 case* R.id.percent\_btn:  
  
 *// 若为错误操作 则均不做操作  
 if*(mistakeJudge()){}  
  
 *else* {  
  
 *// 将算式带入三个算术操作式中* getCondition();  
  
 *// 判断是否为符合操作的运算 (非单个数)  
 if* (startWithSubtract || bothWithSubtract || noStartWithSubtract) {  
 *// 若符合运算规则 则需要得出结果才能够取余 (即按等号)  
 if*(existedText.substring(existedText.length()-1).equals("+") ||  
 existedText.substring(existedText.length()-1).equals("-") ||  
 existedText.substring(existedText.length()-1).equals("×") ||  
 existedText.substring(existedText.length()-1).equals("÷")) {  
 *// 非法取余操作* existedText = "无法取余!";  
 setToast(MainActivity.*this*, existedText);  
 }  
 }  
  
 *// 仅含有前操作数  
 else* {  
 *// 若为0 则无法取余  
 if* (existedText.equals("0")) {  
 *return*;  
 }  
  
 *// 若不为0 则 ( / 100) 取余  
 else* {  
 *double* temp = Double.parseDouble(existedText);  
 existedText = String.valueOf(temp / 100);  
 isCounted = *true*;  
 }  
 }  
 }  
 *break*;  
  
 *// 取sin：  
 case* R.id.sin\_btn:  
  
 *if*(mistakeJudge()){}  
  
 *else* {  
 *// 判断是否满足运算式* getCondition();  
  
 *// 判断是否为符合操作的运算 (非单个数)  
 if* (startWithSubtract || bothWithSubtract || noStartWithSubtract) {  
 *// 若符合运算规则 则需要得出结果才能够取余 (即按等号)  
 if*(existedText.substring(existedText.length()-1).equals("+") ||  
 existedText.substring(existedText.length()-1).equals("-") ||  
 existedText.substring(existedText.length()-1).equals("×") ||  
 existedText.substring(existedText.length()-1).equals("÷")) {  
 *// 非法取余操作* existedText = "无法取sin!";  
 setToast(MainActivity.*this*, existedText);  
 isCounted = *true*;  
 }  
 }  
  
 *// 仅只有单个操作数时  
 else* {  
 *double* temp = Double.parseDouble(existedText);  
 *// 弧度转度数* temp = Math.toRadians(temp);  
 existedText = String.valueOf(Math.sin(temp));  
 isCounted = *true*;  
 }  
 }  
 *break*;  
  
 *// 取cos：  
 case* R.id.cos\_btn:  
  
 *if*(mistakeJudge()){}  
  
 *else*{  
 *// 判断是否满足运算式* getCondition();  
  
 *// 判断是否为符合操作的运算 (非单个数)  
 if* (startWithSubtract || bothWithSubtract || noStartWithSubtract) {  
 *// 若符合运算规则 则需要得出结果才能够取余 (即按等号)  
 if*(existedText.substring(existedText.length()-1).equals("+") ||  
 existedText.substring(existedText.length()-1).equals("-") ||  
 existedText.substring(existedText.length()-1).equals("×") ||  
 existedText.substring(existedText.length()-1).equals("÷")) {  
 *// 非法取余操作* existedText = "无法取cos!";  
 setToast(MainActivity.*this*, existedText);  
 isCounted = *true*;  
 }  
 }  
  
 *// 仅只有单个操作数时  
 else* {  
 *double* temp = Double.parseDouble(existedText);  
 *// 弧度转度数* temp = Math.toRadians(temp);  
 existedText = String.valueOf(Math.cos(temp));  
  
 isCounted = *true*;  
 }  
 }  
 *break*;  
  
 *// 取阶乘：  
 case* R.id.fac\_btn:  
  
 *if*(mistakeJudge()){}  
  
 *else* {  
 *// 将算式带入三个算术操作式中* getCondition();  
  
 *// 判断是否为符合操作的运算 (非单个数)  
 if* (startWithSubtract || bothWithSubtract || noStartWithSubtract) {  
 *// 若符合运算规则 则需要得出结果才能够取余 (即按等号)  
 if*(existedText.substring(existedText.length()-1).equals("+") ||  
 existedText.substring(existedText.length()-1).equals("-") ||  
 existedText.substring(existedText.length()-1).equals("×") ||  
 existedText.substring(existedText.length()-1).equals("÷")) {  
 *// 非法取余操作* existedText = "无法计算阶乘!";  
 setToast(MainActivity.*this*, existedText);  
 isCounted = *true*;  
 }  
 }  
 *else* {  
 *// 若不为负数  
 if*(!existedText.contains("-")) {  
 *int* temp = Integer.parseInt(existedText);  
 *int* fac = 1;  
  
 *for* (*int* i = 1; i <= temp; i++) {  
 fac \*= i;  
 }  
  
 existedText = String.valueOf(fac);  
 isCounted = *true*;  
  
 }  
 *// 若为负数 (无法计算阶乘)  
 else* {  
 existedText = "阶乘无法计算负数!";  
 setToast(MainActivity.*this*, existedText);  
 isCounted = *true*;  
 }  
 }  
 }  
 *break*;  
  
 *// 正负号交换：  
 case* R.id.switch\_btn:  
  
 *if*(mistakeJudge()){}  
  
 *else* {  
  
 *if*(existedText.equals("0")) {}  
  
 *else* {  
 *// 带有负号时 去除负号  
 if*(existedText.contains("-")) {  
 existedText = existedText.replace("-", "");  
 }  
 *// 没有负号时 加上负号  
 else* {  
 existedText = existedText.replaceFirst("","-");  
 }  
 }  
 }  
 *break*;  
  
  
 *// 单值删除:  
 case* R.id.delete\_btn:  
  
 *// 若为error 则直接为0  
 if* (existedText.equals("error")){  
 existedText = "0";  
 }  
 *// 长度大于0  
 else if* (existedText.length() > 0){  
 *// 若仅为一个数 则直接归0  
 if* (existedText.length() == 1) {  
 existedText = "0";  
 }  
 *// 否则向前位移一位  
 else* {  
 existedText = existedText.substring(0,existedText.length()-1);  
 }  
 }  
 *break*;  
  
 *// 整体清除:  
 case* R.id.ac\_btn:  
 existedText = "0";  
 *break*;  
 }  
  
 *// 结果显示于EditText窗口* etResultText.setText(existedText);  
 }  
  
 *// 获取结果方法  
 @SuppressLint*("DefaultLocale")  
 *private* String getResult() {  
  
 String tempResult = *null*;  
  
 String param1, param2;  
  
 *double* arg1, arg2, result;  
  
 *// 判断条件* getCondition();  
  
 *// 若已存在符合条件的运算式 则可以直接计算  
 if* ( startWithSubtract || noStartWithSubtract || bothWithSubtract) {  
  
 *if* (existedText.contains("+")) {  
  
 *// 获取操作数* param1 = existedText.substring(0, existedText.indexOf("+"));  
 param2 = existedText.substring(existedText.indexOf("+") + 1);  
  
 *// 若第二个操作数为空 则直接显示为第一个操作数  
 if* (param2.equals("")) {  
 tempResult = existedText.substring(0, existedText.length() - 1);  
 } *else* {  
 *// 强转计算 并格式化输出* arg1 = Double.parseDouble(param1);  
 arg2 = Double.parseDouble(param2);  
 result = arg1 + arg2;  
 tempResult = String.format("%f", result);  
 *// 采用正则表达式 消除多余0和非法输出* tempResult = subZeroAndDot(tempResult);  
 }  
 }  
  
 *else if* (existedText.contains("×")) {  
  
 param1 = existedText.substring(0, existedText.indexOf("×"));  
 param2 = existedText.substring(existedText.indexOf("×") + 1);  
  
 *if* (param2.equals("")) {  
 tempResult = existedText.substring(0, existedText.length() - 1);  
 } *else* {  
 arg1 = Double.parseDouble(param1);  
 arg2 = Double.parseDouble(param2);  
 result = arg1 \* arg2;  
 tempResult = String.format("%f", result);  
 tempResult = subZeroAndDot(tempResult);  
 }  
 }  
  
 *else if* (existedText.contains("÷")) {  
  
 param1 = existedText.substring(0, existedText.indexOf("÷"));  
 param2 = existedText.substring(existedText.indexOf("÷") + 1);  
  
 *// 除数为0 非法操作  
 if* (param2.equals("0")){  
 tempResult = "除数不能为0!";  
 setToast(MainActivity.*this*, tempResult);  
 } *else if* (param2.equals("")){  
 tempResult = existedText.substring(0, existedText.length() - 1);  
 } *else* {  
 arg1 = Double.parseDouble(param1);  
 arg2 = Double.parseDouble(param2);  
 result = arg1 / arg2;  
 tempResult = String.format("%f", result);  
 tempResult = subZeroAndDot(tempResult);  
 }  
 }  
  
 *else if* (existedText.contains("-")) {  
  
 *// 负数开头 避免不同的 '-' 被误用* param1 = existedText.substring(0, existedText.lastIndexOf("-"));  
 param2 = existedText.substring(existedText.lastIndexOf("-") + 1);  
  
 *if* (param2.equals("")){  
 tempResult = existedText.substring(0, existedText.length() - 1);  
 } *else* {  
 arg1 = Double.parseDouble(param1);  
 arg2 = Double.parseDouble(param2);  
 result = arg1 - arg2;  
 tempResult = String.format("%f", result);  
 tempResult = subZeroAndDot(tempResult);  
 }  
 }  
 }  
  
 *// 若不是标准形式(为单个数 或 直接点击符号) 则直接显示为第一个操作数  
 else* {  
 tempResult = existedText;  
 }  
  
 *return* tempResult;  
 }  
  
 *// 判断是否点击过特殊运算符  
 // 包括：等号 取余 正负号转换 sin cos 阶乘  
 private* String isOver(String existedText, String s) {  
  
 *// 若没按过 则等待下一个数的输入  
 if* (!isCounted){  
  
 *// 清空开头的0  
 if* (existedText.equals("0")){  
 existedText = "";  
 }  
  
 existedText += s;  
 }  
  
 *// 若已经按过  
 // 则自动变为下一个数进行显示  
 else* {  
 existedText = s;  
 isCounted = *false*;  
 }  
  
 *return* existedText;  
 }  
  
 *// 使用正则表达式  
 // 若运算式中包含小数点  
 // 则自动消除多余0和非法结尾小数点  
 public static* String subZeroAndDot(String s){  
 *if*(s.indexOf(".") > 0){  
 s = s.replaceAll("0+?$", ""); *//去掉多余的0* s = s.replaceAll("[.]$", ""); *//如最后一位是.则去掉* }  
  
 *return* s;  
 }  
  
 *// 判断表达式类型 (是 单个操作数 还是 单个操作数 + 操作符 还是 两个操作数)  
 private boolean* judgeExpression() {  
  
 getCondition();  
  
 String tempParam2;  
  
 *if* ( startWithSubtract || noStartWithSubtract || bothWithSubtract ) {  
  
 *if* (existedText.contains("+")) {  
  
 *// 获取第二操作数* tempParam2 = existedText.substring(existedText.indexOf("+") + 1);  
  
 *// 若第二操作数为空 则正常操作不成立  
 if* (tempParam2.equals("")) {  
 *return false*;  
 }  
 *else* {  
 *return true*;  
 }  
 }  
  
 *else if* (existedText.contains("×")) {  
  
 tempParam2 = existedText.substring(existedText.indexOf("×") + 1);  
  
 *if* (tempParam2.equals("")) {  
 *return false*;  
 }  
 *else* {  
 *return true*;  
 }  
 }  
  
 *else if* (existedText.contains("÷")) {  
  
 tempParam2 = existedText.substring(existedText.indexOf("÷") + 1);  
  
 *if* (tempParam2.equals("")) {  
 *return false*;  
 }  
 *else* {  
 *return true*;  
 }  
 }  
  
 *// 双负号时需要注意: 要保留前操作数的符号, 则需要在最后一个 '-' 处取第二操作数  
 else if* (existedText.contains("-")) {  
  
 tempParam2 = existedText.substring(existedText.lastIndexOf("-") + 1);  
  
 *if* (tempParam2.equals("")) {  
 *return false*;  
 }  
 *else* {  
 *return true*;  
 }  
 }  
 }  
 *return false*;  
 }  
  
 *// 三种合法执行条件：  
 private void* getCondition() {  
 *// 检测是否以指定符号开头  
  
 // 1.以负号开头, 且运算符不是减号* startWithSubtract = existedText.startsWith("-") && ( existedText.contains("+") ||  
 existedText.contains("×") || existedText.contains("÷") );  
  
 *// 2.以负号开头, 且运算符是减号* bothWithSubtract = existedText.startsWith("-") && ( existedText.lastIndexOf("-") != 0 );  
  
 *// 3.非负号开头, 且含有运算符* noStartWithSubtract = !existedText.startsWith("-") && ( existedText.contains("+") ||  
 existedText.contains("-") || existedText.contains("×") || existedText.contains("÷"));  
 }  
  
 *public void* setToast(Context context, String text) {  
 Toast toast = Toast.makeText(context, text, Toast.LENGTH\_SHORT);  
 toast.show();  
 }  
  
 *private boolean* mistakeJudge() {  
 *// 若为错误操作 则均不做操作  
 if* (existedText.equals("无法添加小数点!") ||  
 existedText.equals("除数不能为0!") ||  
 existedText.equals("无法取余!") ||  
 existedText.equals("无法取sin!") ||  
 existedText.equals("无法取cos!") ||  
 existedText.equals("无法计算阶乘!") ||  
 existedText.equals("阶乘无法计算负数!")) {  
 *return true*;  
 }  
 *else  
 return false*;  
 }

实验截图：



本次实验的Github项目地址：

<https://github.com/Minakami-Yuuki/Assignments.git>

1. **实验中遇到的问题和解决方法（25分）**

通过本次安卓计算器的开发，我初步的对安卓开发有了一点认知。

首先对于安卓的开发需要具备一定的Java知识；其次需要掌握Android的部分组件，如：Activity，基本页面布局等；还需要了解部分Listener监听器的使用，即能够调用Button按钮进行操作或在程序出错时返回Toast等。

本次安卓计算机开发，我分别采用了LinearLayout线性布局和TableLaout表格布局进行排版，点击按钮Button通过Oncreat()在主活动中进行创建并通过监听器来返回结果，进一步在EditText中显示。若发生算术运算错误，能够及时调用Toast输出错误反馈，而不会发生程序崩溃闪退。

在这次实验中，我主要遇到了两个问题。一是如何将计算器的排版进行一个有效而美观的布局；二是相关计算器的逻辑该如何编写。通过查询网上资料，我发现采用LinearLaout和TableLaout能够有效的将文本显示框和按钮框相结合，并形成一个等比的排版模式。而有关计算器的逻辑处理，即：加减乘除、sin、cos、阶乘、百分比等，都在对数据输入输出处理完毕后成功解决。

通过本次实验，让我了解到了许多关于Android开发的知识，希望下一次实验能够掌握更多的相关知识。

**参考文献（10分）**

[1]中华人民共和国国家质量监督检验检疫总局、中国国家标准化管理委员会.信息与文献 参考文献著录规则:GB/T 7714-2015[S],2015.