**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 计算机一台、Android手机一部**

**系 别 计算机学院**

**专 业 计算机科学与技术**

**班级/学号**  计科1606 / 2016010311

**学生姓名**  耿瑞

**实验日期**  2018年12月7日

**成 绩** \_\_\_\_\_\_\_\_\_\_

**指导教师** \_\_\_\_ **郝保水**\_\_\_\_ \_\_

1 实验目的

（1）了解Android环境的搭建；

（2）理解并掌握layout;

（3）理解并掌握常见的控件使用方法；

（4）理解并掌握菜单；

（5）理解并掌握事件处理机制；

（6）理解并掌握Intent使用方法；

（7）初步了解ActionBar和Fragment使用。

2 实验要求

（1）课前预习实验内容，并查找相关资料。

（2）按照实验步骤完成各个相关内容。

（3）撰写实验报告。

A） 实验报告格式必须符合学校要求（例如必须采用学校规定的实验封面）；

B） 写出实验详细步骤，包括主要采用的技术方案、相关分析和核心代码。注意：不要简单地近包括截屏和代码，完整代码可以作为附录放在实验报告结尾；

C） 总结实验中遇到的问题、分析和解决方法。

D） 写出心得体会与收获等。

3 实验步骤

主要分为几个大步骤进行实现。

（1）需求分析，调研计算器具有哪些功能，确定界面。

（2）系统设计，对计算器进行模块划分，确定技术方案。

（3）编码，编写代码实现各项功能。

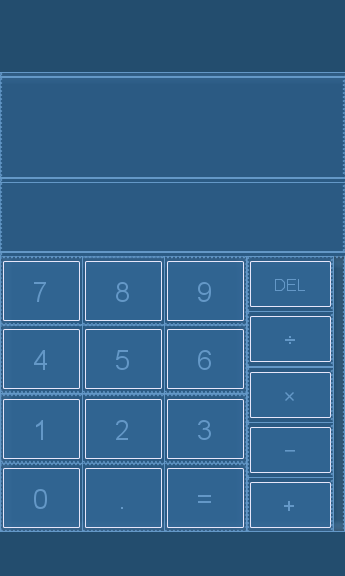
（4）测试。

3.1 需求分析

（1）主要功能

* 支持整数和小数运算（内部采用BigDecimal）
* 支持加减乘除、括号等运算
* 支持连续加减等复杂运算
* 支持三角函数、幂函数、对数等多种函数计算
* 支持输入表达式进行运算

（2）界面设计



主界面

3.2 系统设计

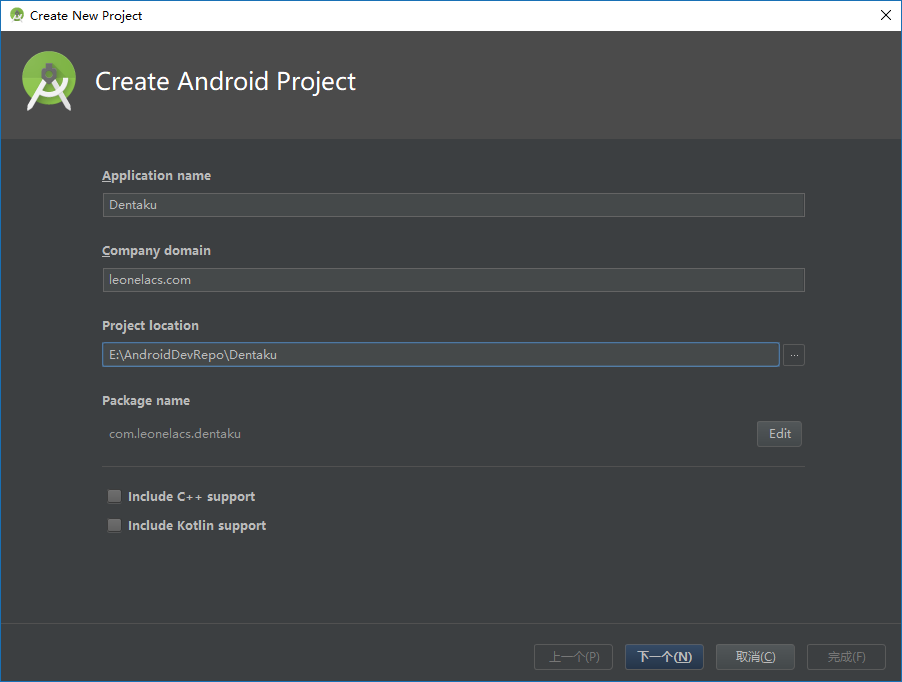
计算器应用包括以下模块：

* 主界面模块
* 侧滑函数符号菜单模块
* 表达式处理与计算模块

3.3 项目实施

主要步骤：

（1）创建工程



在Android Studio中创建Android项目。

应用名称：Dentaku

域名：leonelacs.com

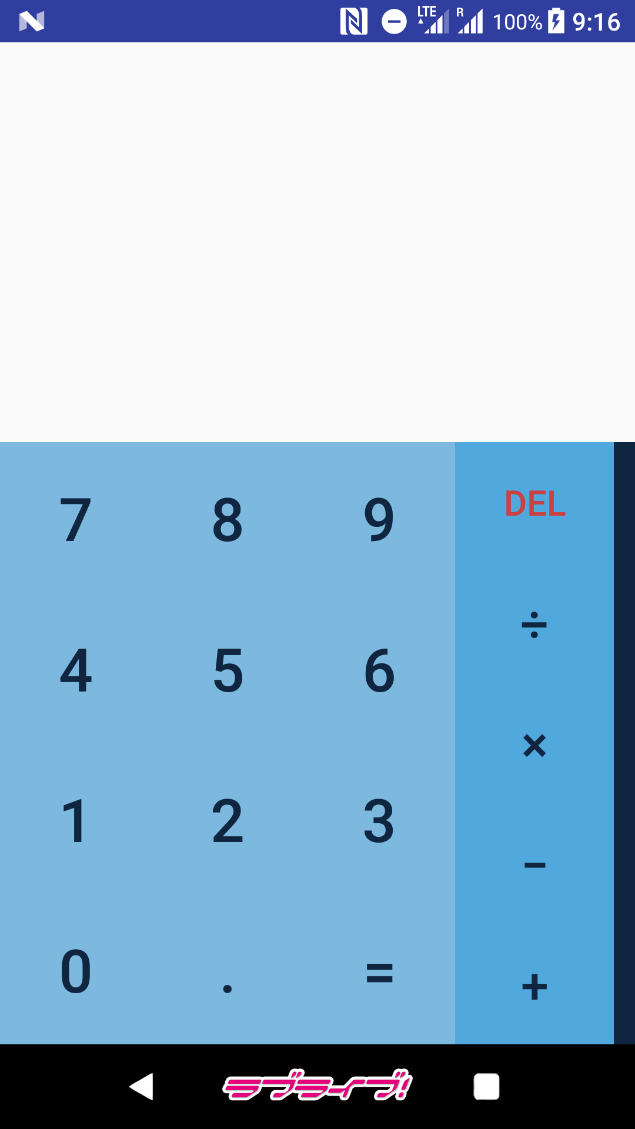
包名：com.leonelacs.dentaku

最低SDK版本API 24: Android 7.0

（2）编写各界面布局、菜单等资源文件

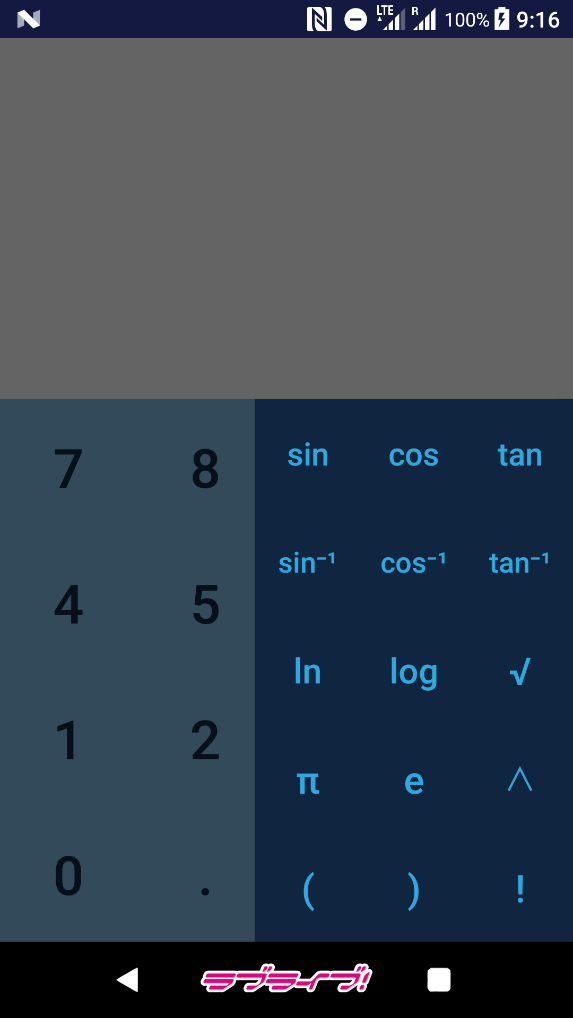
计算器主界面，使用LinearLayout实现，包含：

* 一个用于显示输入表达式的TextView：**InputDisplay**；
* 一个用于显示运算结果的TextView：**ResultDisplay**；
* 数字、四则运算、小数点、等于、删除按钮。



侧滑函数符号菜单，使用DrawerLayout实现，包含：

* 三角函数与反三角函数按钮；
* 对数、平方根、乘方、阶乘按钮；
* 圆周率、自然常数、左右括号按钮。



表达式处理与计算采用自己编写的库calcium。该库原为C#语言实现，采用正则表达式处理以字符串形式读入的算术表达式，使用操作数栈与操作符栈完成表达式的计算。

定义的运算符及其优先级：

符号 左优先级 右优先级

+ 加号 3 2

- 减号 3 2

\* 乘号 5 4

/ 除号 5 4

^ 乘方 7 6

( 左括号 1 8

S 反正弦函数和左括号 1 8

C 反余弦函数和左括号 1 8

T 反正切函数和左括号 1 8

s 正弦函数和左括号 1 8

c 余弦函数和左括号 1 8

t 正切函数和左括号 1 8

n 自然对数和左括号 1 8

g 以10为底的对数及左括号 1 8

q 平方根和左括号 1 8

) 右括号 8 1

# 表达式结束及栈底标记 0 0

com.leonelacs.calcium中包含数学符号类**MathOperator**，成员变量为：

符号String **Content**；

左优先级int **LPriority**；

右优先级int **RPriority**。

com.leonelacs.calcium中包含计算类**Calc**，主要包含以下方法：

符号优先级对比

public static int **OperatorPrioritiesCompare**(char op1, char op2)

格式化表达式

public String **ExpressionFormat**(String exp)

括号补全

public String **BracketComplete**(String exp)

括号匹配校验（栈实现）

public boolean **BracketVaildate**(String exp)

乘号补全

public String **MultiplyComplete**(String exp)

分离格式化表达式操作数

public List<BigDecimal> **NumbersSeparate**(String expf)

大数乘方（BigInteger实现）

public BigInteger **BigFactorial**(BigInteger bas)

计算格式化表达式

public DecimalAnswer **FormatedExpressionCalculate**(String expf)

计算表达式字符串

public DecimalAnswer **Compute**(String expression)

计算表达式字符串（字符串输出）

public String **StrCompute**(String exp)

详细代码见附录。

3.4 测试

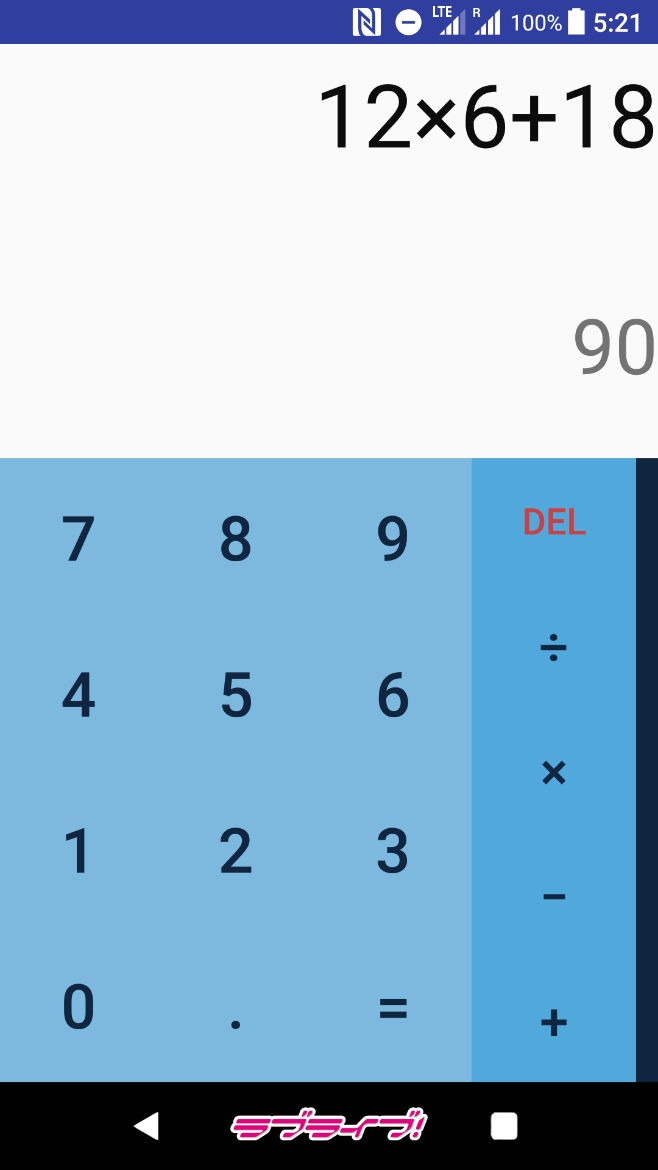
（1）主界面

点击数字、四则运算、小数点按钮可以输入算术表达式，显示在InputDisplay中；

点击删除按钮可以逐位从最后删除表达式，长按删除按钮可以清除输入的表达式。

点击等于按钮，在ResultDisplay中显示表达式的结果。

例：输入12×6+18，显示计算结果为90

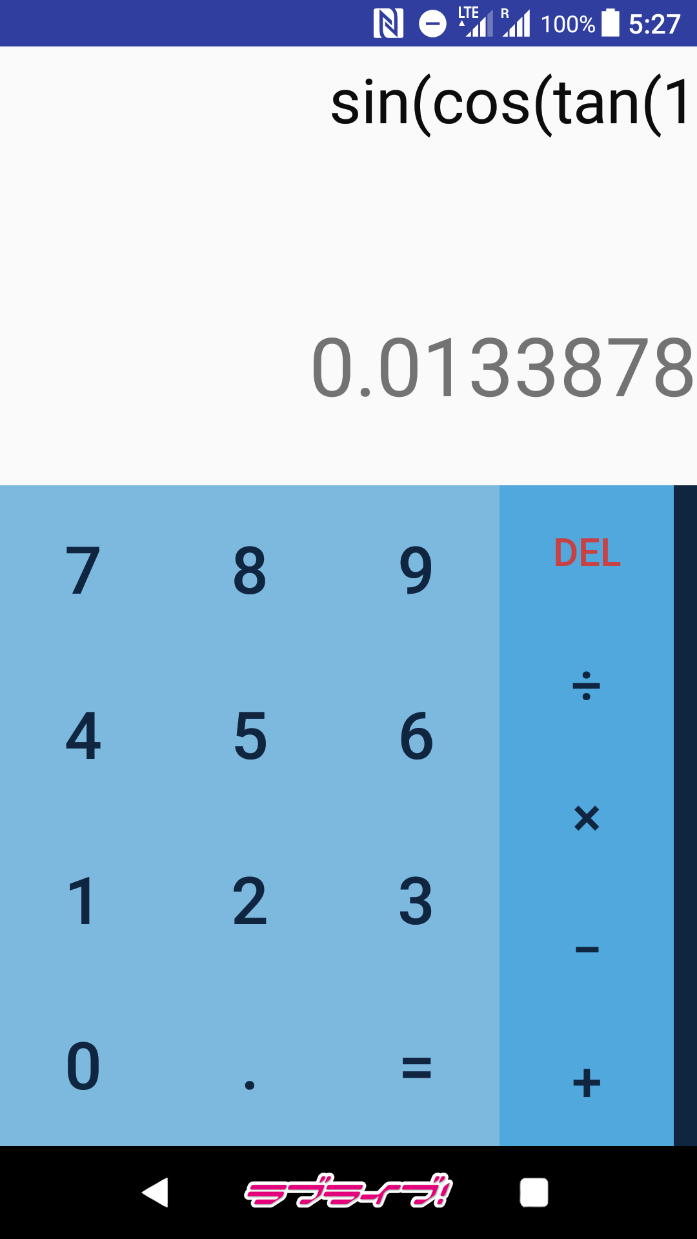


（2）侧滑函数符号菜单

从屏幕右侧向左滑动可滑出函数符号菜单；

点击菜单中的按钮可以输入复杂的算术表达式。

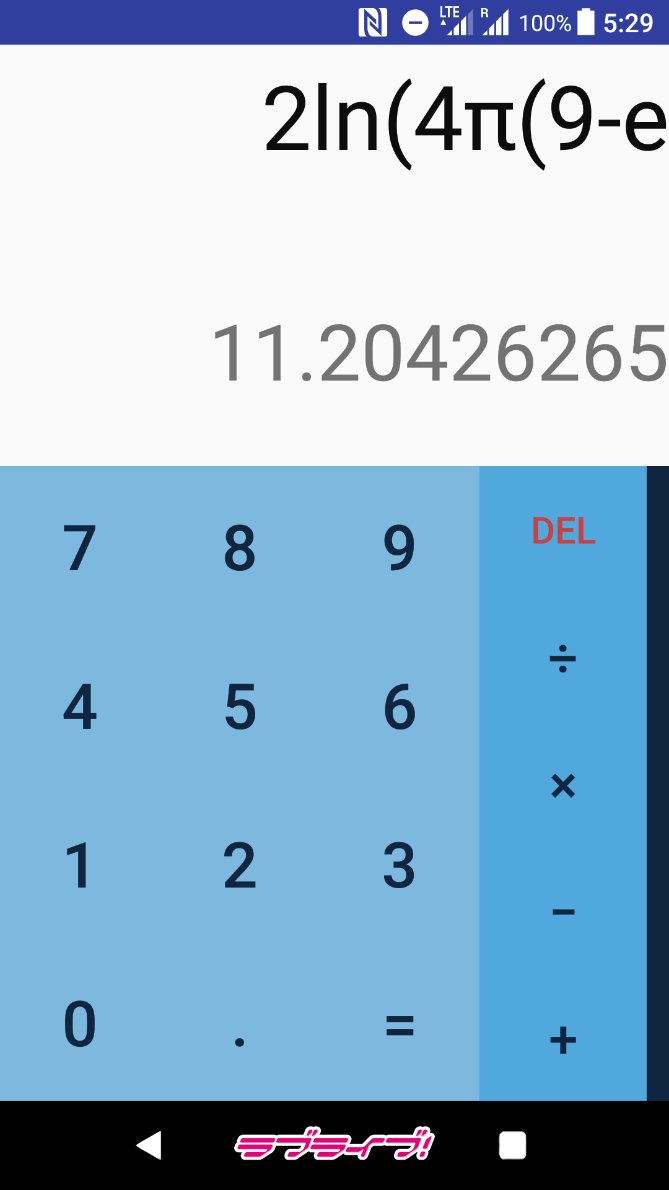
例：输入sin(cos(tan(1，显示计算结果为0.0133878



（3）表达式处理与计算

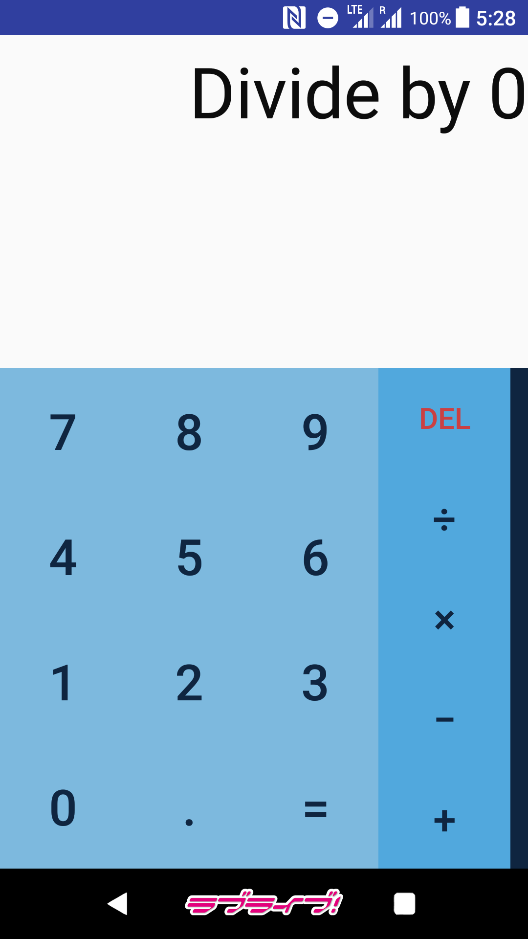
StrCompute方法得到表达式字符串，先由格式化方法使用正则表达式对表达式字符串进行括号补全、乘号补全和符号替换，在对格式化表达式合法性进行判断后，分离出表达式中的操作数装入操作数列表中，再使用操作数栈与操作符栈对表达式进行计算，计算结束后对结果进行格式化输出。

例：输入2ln(4π(9-e，显示计算结果为11.20426265

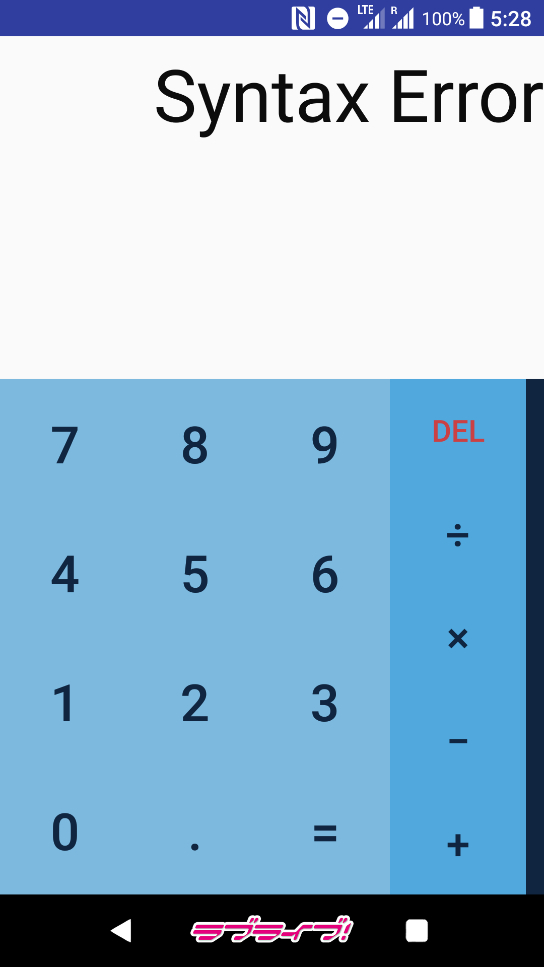


若表达式不合法（例如括号不匹配）或者在计算过程中产生错误（例如除以零），都会终止计算并输出错误类型。

例：输入5/(3-2-1)，显示错误信息Divide by zero



例：输入sin()，显示错误信息Syntax Error



4 实验中遇到的问题、分析和解决方法

（1）C#中的Decimal与Java中的BigDecimal有所不同

重写时查阅Java资料解决问题。

（2）BigDecimal不适用于Math库方法

将BigDecimal对象转换为Double再进行操作，或是自己实现所需方法。

（3）主界面输入数字无法显示

后发现是TextView被顶部标题栏遮挡，隐藏标题栏解决了问题。

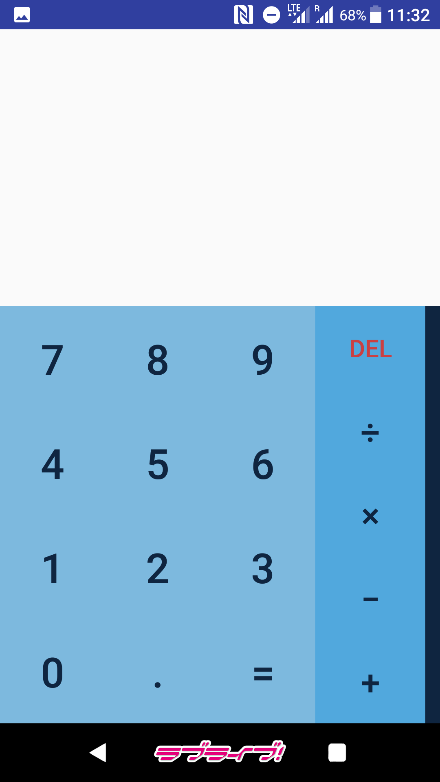
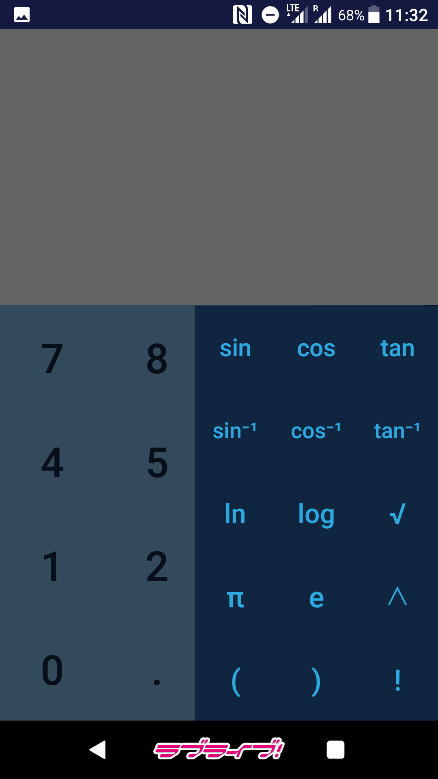
5 心得体会与收获

通过这次实验，我完成了自己的第一个Android应用的开发。了解了Android环境的搭建；理解并掌握了Layout、常见的控件使用方法、菜单、事件处理机制、Intent使用方法；初步了解了ActionBar和Fragment使用。

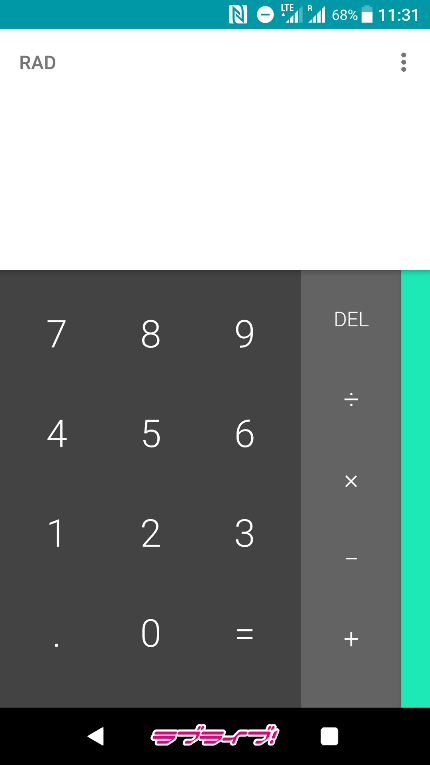
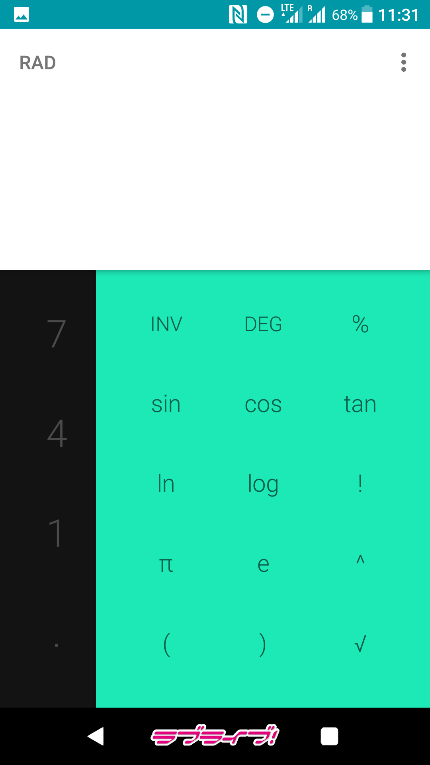
这次试验的计算器程序的外观与功能是以原生Android自带的计算器为范本设计的。除此之外，我还运用了一些以前数据结构课程的程序，将其用Java重写后供Android应用使用。

与原生Android自带的计算器的外观对比

自己编写的计算器外观：

原生Android自带计算器外观：

6 附录

代码仓库：<https://github.com/leonelacs/Dentaku>

程序完整代码：

**com.leonelacs.calcium**

MathOperator.java

package com.leonelacs.calcium;  
  
public class MathOperator {  
 public char Content;  
 public int LPriority;  
 public int RPriority;  
  
 public MathOperator(char content, int lp, int rp) {  
 this.Content = content;  
 this.LPriority = lp;  
 this.RPriority = rp;  
 }  
}

Calc.java

package com.leonelacs.calcium;  
  
import java.math.BigDecimal;  
import java.math.BigInteger;  
import java.math.RoundingMode;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Stack;  
  
public class Calc {  
 public static List<MathOperator> *mathOperators* = new ArrayList<MathOperator>();  
  
 public Calc() {  
 *mathOperators*.add(new MathOperator('+', 3, 2));  
 *mathOperators*.add(new MathOperator('-', 3, 2));  
 *mathOperators*.add(new MathOperator('\*', 5, 4));  
 *mathOperators*.add(new MathOperator('/', 5, 4));  
 *mathOperators*.add(new MathOperator('^', 7, 6));  
 *mathOperators*.add(new MathOperator('(', 1, 8));  
 *mathOperators*.add(new MathOperator('S', 1, 8));  
 *mathOperators*.add(new MathOperator('C', 1, 8));  
 *mathOperators*.add(new MathOperator('T', 1, 8));  
 *mathOperators*.add(new MathOperator('s', 1, 8));  
 *mathOperators*.add(new MathOperator('c', 1, 8));  
 *mathOperators*.add(new MathOperator('t', 1, 8));  
 *mathOperators*.add(new MathOperator('n', 1, 8));  
 *mathOperators*.add(new MathOperator('g', 1, 8));  
 *mathOperators*.add(new MathOperator('q', 1, 8));  
 *mathOperators*.add(new MathOperator(')', 8, 1));  
 *mathOperators*.add(new MathOperator('#', 0, 0));  
 }  
  
 public class DecimalAnswer {  
 BigDecimal result;  
 String errInfo;  
  
 public DecimalAnswer(BigDecimal result, String errInfo) {  
 this.result = result;  
 this.errInfo = errInfo;  
 }  
 }  
  
 public static int OperatorPrioritiesCompare(char op1, char op2) {  
 int op1Prior = -1;  
 int op2Prior = -1;  
 for (MathOperator mathOp: *mathOperators*) {  
 if (mathOp.Content == op1) {  
 op1Prior = mathOp.LPriority;  
 }  
 if (mathOp.Content == op2) {  
 op2Prior = mathOp.RPriority;  
 }  
 }  
  
 if (op1Prior < op2Prior) {  
 return 1;  
 }  
 else if (op1Prior > op2Prior) {  
 return -1;  
 }  
 else {  
 return 0;  
 }  
 }  
  
 public String ExpressionFormat(String exp) {  
 exp = exp.trim();  
 exp = "{" + exp + "}";  
 exp = BracketComplete(exp);  
 boolean bracketgood = BracketVaildate(exp);  
 if (!bracketgood) {  
 return "b";  
 }  
 exp = exp.replaceAll("×", "\*");  
 exp = exp.replaceAll("÷", "/");  
 exp = exp.replaceAll("\\{-", "{~");  
 exp = exp.replaceAll("\\(-", "(~");  
 exp = exp.replaceAll("\\{", "");  
 exp = exp.replaceAll("\\}", "");  
 exp = exp.replaceAll("arcsin\\(", "S");  
 exp = exp.replaceAll("arccos\\(", "C");  
 exp = exp.replaceAll("arctan\\(", "T");  
 exp = exp.replaceAll("sin\\(", "s");  
 exp = exp.replaceAll("cos\\(", "c");  
 exp = exp.replaceAll("tan\\(", "t");  
 exp = exp.replaceAll("ln\\(", "n");  
 exp = exp.replaceAll("log\\(", "g");  
 exp = exp.replaceAll("√\\(", "q");  
 exp = exp.replaceAll("π", "3.14159");  
 exp = exp.replaceAll("e", "2.71828");  
 exp = MultiplyComplete(exp);  
 return exp;  
 }  
  
 public String BracketComplete(String exp) {  
 char[] expArray = exp.toCharArray();  
 int left = 0, right = 0;  
 for (char c: expArray) {  
 if (c == '(') {  
 left++;  
 }  
 if (c == ')') {  
 right++;  
 }  
 }  
 if (right < left) {  
 int diff = left - right;  
 for (int i = 0; i < diff; i++) {  
 exp = exp + ")";  
 }  
 }  
 return exp;  
 }  
  
 public boolean BracketVaildate(String exp) {  
 if (exp.contains("()")) {  
 return false;  
 }  
 char[] expArray = exp.toCharArray();  
 Stack<Integer> veriStack = new Stack<Integer>();  
 for (char c: expArray) {  
 if (c == '(') {  
 veriStack.push(1);  
 }  
 else if (c == ')') {  
 if (veriStack.isEmpty()) {  
 return false;  
 }  
 else {  
 veriStack.pop();  
 }  
 }  
 }  
 return true;  
 }  
  
 public String MultiplyComplete(String exp) {  
 String patReplace = "([\\d\\)])([SCTstcngq\\(])";  
 exp = exp.replaceAll(patReplace, "$1" + "\*" + "$2");  
 return exp;  
 }  
  
 public List<BigDecimal> NumbersSeparate(String expf) {  
 String patSplit = "[\\+\\-\\\*/\\^\\(SCTsctngq\\)!]";  
 String[] numStrs = expf.split(patSplit);  
 List<BigDecimal> numbers = new ArrayList<BigDecimal>();  
 String patMatch = "(~)?\\d+(\\.\\d+)?";  
 for (String s: numStrs) {  
 if (s.matches(patMatch)) {  
 s = s.replace('~', '-');  
 BigDecimal decTemp = new BigDecimal(s);  
 numbers.add(decTemp);  
 }  
 }  
 return numbers;  
 }  
  
 public DecimalAnswer FormatedExpressionCalculate(String expf) {  
 if (expf.equals("b")) {  
 DecimalAnswer ans = new DecimalAnswer(null, "SYNEb");  
 return ans;  
 }  
 try {  
 Stack<Character> operators = new Stack<Character>();  
 Stack<BigDecimal> numbers = new Stack<BigDecimal>();  
 List<BigDecimal> numSet = NumbersSeparate(expf);  
 String patReplace = "(~)?\\d+(\\.\\d+)?";  
 String expm = expf.replaceAll(patReplace, "@");  
 expm = expm + "#";  
 operators.push('#');  
 if (!numSet.isEmpty()) {  
 for (char c : expm.toCharArray()) {  
 if (c == '@') {  
 numbers.push(numSet.get(0));  
 numSet.remove(0);  
 } else {  
 boolean flag = true;  
 while (flag) {  
 if (c == '#' && operators.peek() == '#') {  
 DecimalAnswer ans = new DecimalAnswer(numbers.peek(), "");  
 return ans;  
 }  
 if (c == '!') {  
 BigDecimal opnd = numbers.pop();  
 BigInteger ndint;  
 try {  
 ndint = opnd.toBigIntegerExact();  
 } catch (Exception e) {  
 DecimalAnswer ans = new DecimalAnswer(null, "DOME");  
 return ans;  
 }  
 BigInteger bigres = BigFactorial(ndint);  
 numbers.push(new BigDecimal(bigres));  
 break;  
 }  
 int sign = *OperatorPrioritiesCompare*(operators.peek(), c);  
 if (sign == 1) {  
 operators.push(c);  
 flag = false;  
 } else if (sign == 0) {  
 char curOp = operators.pop();  
 flag = false;  
 if (curOp == '(') {  
 continue;  
 } else {  
 BigDecimal opnd = numbers.pop();  
 double nddou = opnd.doubleValue();  
 Double res = 0.;  
 if (curOp == 'S') res = Math.*asin*(nddou);  
 else if (curOp == 'C') res = Math.*acos*(nddou);  
 else if (curOp == 'T') res = Math.*atan*(nddou);  
 else if (curOp == 's') res = Math.*sin*(nddou);  
 else if (curOp == 'c') res = Math.*cos*(nddou);  
 else if (curOp == 't') res = Math.*tan*(nddou);  
 else if (curOp == 'n') res = Math.*log*(nddou);  
 else if (curOp == 'g') res = Math.*log10*(nddou);  
 else if (curOp == 'q') res = Math.*sqrt*(nddou);  
 String restr = res.toString();  
 BigDecimal resbd = new BigDecimal(restr);  
 numbers.push(resbd);  
 }  
 } else if (sign == -1) {  
 char op = operators.pop();  
 BigDecimal b = numbers.pop();  
 BigDecimal a = numbers.pop();  
 BigDecimal res;  
 if (op == '+') res = a.add(b);  
 else if (op == '-') res = a.subtract(b);  
 else if (op == '\*') res = a.multiply(b);  
 else if (op == '/') {  
 try {  
 BigInteger bint = b.toBigIntegerExact();  
 if (bint.equals(new BigInteger("0"))) {  
 DecimalAnswer ans = new DecimalAnswer(null, "DIV0");  
 return ans;  
 }  
 } catch (Exception e) {}  
 res = a.divide(b, 8, RoundingMode.*DOWN*);  
 } else if (op == '^') {  
 Double adou = a.doubleValue();  
 Double bdou = b.doubleValue();  
 Double resdou = Math.*pow*(adou, bdou);  
 String resstr = resdou.toString();  
 res = new BigDecimal(resstr);  
 } else {  
 res = new BigDecimal("0");  
 DecimalAnswer ans = new DecimalAnswer(null, "SYNEp");  
 }  
 numbers.push(res);  
 }  
 }  
 }  
 }  
 }  
 DecimalAnswer ans = new DecimalAnswer(null, "UNKE");  
 return ans;  
 }  
 catch (Exception e) {  
 DecimalAnswer ans = new DecimalAnswer(null, "SYNEs");  
 return ans;  
 }  
 }  
  
 public DecimalAnswer Compute(String expression) {  
 String expf = ExpressionFormat(expression);  
 DecimalAnswer answer = FormatedExpressionCalculate(expf);  
 return answer;  
 }  
  
 public BigInteger BigFactorial(BigInteger bas) {  
 BigInteger zero = new BigInteger("0");  
 BigInteger one = new BigInteger("1");  
 //BigInteger two = new BigInteger("2");  
 BigInteger absRes = new BigInteger("1");  
 BigInteger symb = new BigInteger("1");  
 BigInteger i = bas;  
 if (bas.compareTo(zero) == -1) {  
 symb = new BigInteger("-1");  
 }  
 i = i.multiply(symb);  
 while (i.compareTo(one) == 1) {  
 absRes = absRes.multiply(i);  
 i = i.subtract(one);  
 }  
 absRes = absRes.multiply(symb);  
 return absRes;  
 }  
  
 public String StrCompute(String exp) {  
 DecimalAnswer deca = Compute(exp);  
 String stra;  
 if (deca.result != null) {  
 BigDecimal biggestView = new BigDecimal("999999999999");  
 BigDecimal smallestView = new BigDecimal("-99999999999");  
 BigDecimal toPosi0View = new BigDecimal("0.0000001");  
 BigDecimal toNega0View = new BigDecimal("-0.0000001");  
  
 if (deca.result.compareTo(biggestView) == 1 || deca.result.compareTo(smallestView) == -1 || (deca.result.compareTo(toPosi0View) == -1 && deca.result.compareTo(toNega0View) == 1)) {  
 stra = deca.result.stripTrailingZeros().toString();  
 String patScale = "(\\d+)(.?)(\\d{0,8})(\\d\*)(E)([+-])(\\d+)";  
 stra = stra.replaceAll(patScale, "$1" + "." + "$3" + "E" + "$6" + "$7");  
 //stra = "1.123456789E"  
 }  
 else {  
 stra = deca.result.setScale(8, RoundingMode.*DOWN*).stripTrailingZeros().toPlainString();  
 }  
  
 //stra = deca.result.stripTrailingZeros().toString();  
 }  
 else {  
 stra = deca.errInfo;  
 if (stra.equals("UNKE")) stra = "Unknown Error";  
 else if (stra.equals("DIV0")) stra = "Divide by 0";  
 else if (stra.equals("DOME")) stra = "Domain Error";  
 else stra = "Syntax Error";  
 }  
 return stra;  
 }  
  
  
}

**com.leonelacs.dentaku**

GlanceActivity.java

package com.leonelacs.dentaku;  
  
import android.os.Bundle;  
import android.view.View;  
import android.support.design.widget.NavigationView;  
import android.support.v4.view.GravityCompat;  
import android.support.v4.widget.DrawerLayout;  
import android.support.v7.app.ActionBarDrawerToggle;  
import android.support.v7.app.AppCompatActivity;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.widget.Button;  
import android.widget.TextView;  
import com.leonelacs.calcium.\*;  
  
public class GlanceActivity extends AppCompatActivity  
 implements NavigationView.OnNavigationItemSelectedListener, View.OnClickListener {  
  
 protected String newExpDisp = "", newResDisp = "";  
 String div0 = "Divide by 0";  
 String unke = "Unknown Error";  
 String syne = "Syntax Error";  
 String dome = "Domain Error";  
  
 protected void RefreshResult() {  
 TextView tvExpression = (TextView)findViewById(R.id.*InputDisplay*);  
 TextView tvResult = (TextView)findViewById(R.id.*ResultDisplay*);  
 String expression = tvExpression.getText().toString().trim();  
  
 if (!expression.equals("")) {  
 Calc calc = new Calc();  
 String result = calc.StrCompute(expression);  
 //tvExpression.setText(result);  
 if (result.equals(div0) || result.equals(unke) || result.equals(syne) || result.equals(dome)) {  
  
 }  
 else {  
 tvResult.setText(result);  
 }  
 }  
 }  
  
 protected void RefreshTextSize() {  
 TextView tvExpression = (TextView)findViewById(R.id.*InputDisplay*);  
 TextView tvResult = (TextView)findViewById(R.id.*ResultDisplay*);  
 String exps = tvExpression.getText().toString();  
 String ress = tvResult.getText().toString();  
 int expcnt = exps.length();  
 int rescnt = ress.length();  
 if (expcnt > 12) tvExpression.setTextSize(48 \* 2 / 3);  
 else tvExpression.setTextSize(48);  
 if (rescnt > 14) tvResult.setTextSize(42 \* 2 / 3);  
 else tvResult.setTextSize(42);  
 }  
  
 @Override  
  
 public void onClick(View view) {  
 TextView tvExpression = (TextView)findViewById(R.id.*InputDisplay*);  
 TextView tvResult = (TextView)findViewById(R.id.*ResultDisplay*);  
  
 switch ((int)view.getTag()) {  
 case 0:  
 newExpDisp = tvExpression.getText()+"0";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 1:  
 newExpDisp = tvExpression.getText()+"1";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 2:  
 newExpDisp = tvExpression.getText()+"2";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 3:  
 newExpDisp = tvExpression.getText()+"3";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 4:  
 newExpDisp = tvExpression.getText()+"4";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 5:  
 newExpDisp = tvExpression.getText()+"5";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 6:  
 newExpDisp = tvExpression.getText()+"6";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 7:  
 newExpDisp = tvExpression.getText()+"7";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 8:  
 newExpDisp = tvExpression.getText()+"8";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 9:  
 newExpDisp = tvExpression.getText()+"9";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 10:  
 newExpDisp = tvExpression.getText()+".";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 11:  
 String expression = tvExpression.getText().toString().trim();  
 if (!expression.equals("")) {  
 Calc calc = new Calc();  
 String result = calc.StrCompute(expression);  
 tvResult.setText("");  
 tvExpression.setText(result);  
 }  
 RefreshTextSize();  
 break;  
 case 12:  
 newExpDisp = tvExpression.getText()+"+";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 13:  
 newExpDisp = tvExpression.getText()+"-";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 14:  
 //tvExpression.setText(tvExpression.getText()+"×");  
 newExpDisp = tvExpression.getText()+"×";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 15:  
 //tvExpression.setText(tvExpression.getText()+"÷");  
 newExpDisp = tvExpression.getText()+"÷";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 16: //del  
 String newExp = tvExpression.getText().toString();  
 newExp = newExp.trim();  
 if (!newExp.equals("")) {  
 if (newExp.equals(div0) || newExp.equals(unke) || newExp.equals(syne) || newExp.equals(dome)) {  
 newExp = "";  
 tvExpression.setText(newExp);  
 }  
 else {  
 newExp = newExp.substring(0, newExp.length() - 1);  
 tvExpression.setText(newExp);  
 }  
 if (newExp.equals("")) {  
 tvResult.setText("");  
 }  
 }  
 else {  
 tvResult.setText("");  
 }  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 17:  
 newExpDisp = tvExpression.getText()+"sin(";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 18:  
 newExpDisp = tvExpression.getText()+"cos(";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 19:  
 newExpDisp = tvExpression.getText()+"tan(";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 20:  
 newExpDisp = tvExpression.getText()+"arcsin(";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 21:  
 newExpDisp = tvExpression.getText()+"arccos(";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 22:  
 newExpDisp = tvExpression.getText()+"arctan(";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 23:  
 newExpDisp = tvExpression.getText()+"ln(";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 24:  
 newExpDisp = tvExpression.getText()+"log(";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 25:  
 newExpDisp = tvExpression.getText()+"√(";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 26:  
 newExpDisp = tvExpression.getText()+"π";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 27:  
 newExpDisp = tvExpression.getText()+"e";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 case 28:  
 newExpDisp = tvExpression.getText()+"^";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 29:  
 newExpDisp = tvExpression.getText()+"(";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 30:  
 newExpDisp = tvExpression.getText()+")";  
 tvExpression.setText(newExpDisp);  
 RefreshTextSize();  
 break;  
 case 31:  
 newExpDisp = tvExpression.getText()+"!";  
 tvExpression.setText(newExpDisp);  
 RefreshResult();  
 RefreshTextSize();  
 break;  
 }  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_glance*);  
  
 Button bt0 = (Button)findViewById(R.id.*B0*);  
 Button bt1 = (Button)findViewById(R.id.*B1*);  
 Button bt2 = (Button)findViewById(R.id.*B2*);  
 Button bt3 = (Button)findViewById(R.id.*B3*);  
 Button bt4 = (Button)findViewById(R.id.*B4*);  
 Button bt5 = (Button)findViewById(R.id.*B5*);  
 Button bt6 = (Button)findViewById(R.id.*B6*);  
 Button bt7 = (Button)findViewById(R.id.*B7*);  
 Button bt8 = (Button)findViewById(R.id.*B8*);  
 Button bt9 = (Button)findViewById(R.id.*B9*);  
  
 Button btDot = (Button)findViewById(R.id.*BDot*);  
 Button btEqual = (Button)findViewById(R.id.*BEqual*);  
 Button btAdd = (Button)findViewById(R.id.*BAdd*);  
 Button btSubtract = (Button)findViewById(R.id.*BSubtract*);  
 Button btMultiply = (Button)findViewById(R.id.*BMultiply*);  
 Button btDivide = (Button)findViewById(R.id.*BDivide*);  
 Button btDelete = (Button)findViewById(R.id.*BDelete*);  
  
 Button btSin = (Button)findViewById(R.id.*BSin*);  
 Button btCos = (Button)findViewById(R.id.*BCos*);  
 Button btTan = (Button)findViewById(R.id.*BTan*);  
 Button btArcSin = (Button)findViewById(R.id.*BAsin*);  
 Button btArcCos = (Button)findViewById(R.id.*BAcos*);  
 Button btArcTan = (Button)findViewById(R.id.*BAtan*);  
 Button btLog = (Button)findViewById(R.id.*BLog*);  
 Button btLn = (Button)findViewById(R.id.*BLn*);  
 Button btSqrt = (Button)findViewById(R.id.*BSqrt*);  
 Button btPow = (Button)findViewById(R.id.*BPow*);  
 Button btPI = (Button)findViewById(R.id.*BPi*);  
 Button btE = (Button)findViewById(R.id.*Be*);  
 Button btFact = (Button)findViewById(R.id.*BFact*);  
 Button btLeftBracket = (Button)findViewById(R.id.*BLeft*);  
 Button btRightBracket = (Button)findViewById(R.id.*BRight*);  
  
 bt0.setTag(0);  
 bt1.setTag(1);  
 bt2.setTag(2);  
 bt3.setTag(3);  
 bt4.setTag(4);  
 bt5.setTag(5);  
 bt6.setTag(6);  
 bt7.setTag(7);  
 bt8.setTag(8);  
 bt9.setTag(9);  
  
 btDot.setTag(10);  
 btEqual.setTag(11);  
 btAdd.setTag(12);  
 btSubtract.setTag(13);  
 btMultiply.setTag(14);  
 btDivide.setTag(15);  
 btDelete.setTag(16);  
  
 btSin.setTag(17);  
 btCos.setTag(18);  
 btTan.setTag(19);  
 btArcSin.setTag(20);  
 btArcCos.setTag(21);  
 btArcTan.setTag(22);  
 btLn.setTag(23);  
 btLog.setTag(24);  
 btSqrt.setTag(25);  
 btPI.setTag(26);  
 btE.setTag(27);  
 btPow.setTag(28);  
 btLeftBracket.setTag(29);  
 btRightBracket.setTag(30);  
 btFact.setTag(31);  
  
 bt0.setOnClickListener(this);  
 bt1.setOnClickListener(this);  
 bt2.setOnClickListener(this);  
 bt3.setOnClickListener(this);  
 bt4.setOnClickListener(this);  
 bt5.setOnClickListener(this);  
 bt6.setOnClickListener(this);  
 bt7.setOnClickListener(this);  
 bt8.setOnClickListener(this);  
 bt9.setOnClickListener(this);  
  
 btDot.setOnClickListener(this);  
 btEqual.setOnClickListener(this);  
 btAdd.setOnClickListener(this);  
 btSubtract.setOnClickListener(this);  
 btMultiply.setOnClickListener(this);  
 btDivide.setOnClickListener(this);  
 btDelete.setOnClickListener(this);  
  
 final TextView tvExpression = (TextView)findViewById(R.id.*InputDisplay*);  
 final TextView tvResult = (TextView)findViewById(R.id.*ResultDisplay*);  
  
 btDelete.setOnLongClickListener(new View.OnLongClickListener() {  
 @Override  
 public boolean onLongClick(View view) {  
 tvExpression.setText("");  
 tvResult.setText("");  
 return false;  
 }  
 });  
  
 btSin.setOnClickListener(this);  
 btCos.setOnClickListener(this);  
 btTan.setOnClickListener(this);  
 btArcSin.setOnClickListener(this);  
 btArcCos.setOnClickListener(this);  
 btArcTan.setOnClickListener(this);  
 btLn.setOnClickListener(this);  
 btLog.setOnClickListener(this);  
 btSqrt.setOnClickListener(this);  
 btPI.setOnClickListener(this);  
 btE.setOnClickListener(this);  
 btPow.setOnClickListener(this);  
 btLeftBracket.setOnClickListener(this);  
 btRightBracket.setOnClickListener(this);  
 btFact.setOnClickListener(this);  
  
  
 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);  
 ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(  
// this, drawer, toolbar, R.string.navigation\_drawer\_open, R.string.navigation\_drawer\_close);  
 this, drawer, R.string.*navigation\_drawer\_open*, R.string.*navigation\_drawer\_close*);  
 drawer.addDrawerListener(toggle);  
 toggle.syncState();  
 }  
  
 @Override  
 public void onBackPressed() {  
 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);  
 if (drawer.isDrawerOpen(GravityCompat.*START*)) {  
 drawer.closeDrawer(GravityCompat.*START*);  
 } else {  
 super.onBackPressed();  
 }  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 // Inflate the menu; this adds items to the action bar if it is present.  
 getMenuInflater().inflate(R.menu.*glance*, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 // Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.  
 int id = item.getItemId();  
  
 //noinspection SimplifiableIfStatement  
 if (id == R.id.*action\_settings*) {  
 return true;  
 }  
  
 return super.onOptionsItemSelected(item);  
 }  
  
 @SuppressWarnings("StatementWithEmptyBody")  
 @Override  
 public boolean onNavigationItemSelected(MenuItem item) {  
 // Handle navigation view item clicks here.  
 int id = item.getItemId();  
  
 if (id == R.id.*nav\_camera*) {  
 // Handle the camera action  
 } else if (id == R.id.*nav\_gallery*) {  
  
 } else if (id == R.id.*nav\_slideshow*) {  
  
 } else if (id == R.id.*nav\_manage*) {  
  
 } else if (id == R.id.*nav\_share*) {  
  
 } else if (id == R.id.*nav\_send*) {  
  
 }  
  
 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);  
 drawer.closeDrawer(GravityCompat.*START*);  
 return true;  
 }  
}