# 软件使用与设计说明书

## NEON 计算器

Fluent Design and NEON

HDU GEEK

# 一、 引言

## 1.1 编写目的

NEON 计算器详细设计的主要任务是在用户需求上，对设计中产生的功能模块进行过程描述，设计功能模块的内部细节，包括算法和详细数据结构，为源代码提供必要的说明。

总体设计解决了软件系统总体结构设计的问题，包括整个软件系统的结构、模块划分、模块功能和模块间的联系等。详细设计则要解决如何实现各个模块的内部功能，即模块设计内为已经写好的计算器各个模块的算法或者实现。

## 1.2 项目背景

根据任课老师的要求以及自己对计算器功能设计要求，利用 Java 语言编写一个具有科学计算功能的计算器，兼具界面美观，函数多样等功能，同时革新传统计算器的输入方式来大大提高使用效率。

## 1.3 定义

Java：顶层设计编程语言
JavaScript：解释型或即时编译型的编程语言，用于解析计算公式
JavaFX：RIA 开发工具，替代 Swing 的下一代 GUI 设计设计框架
FXML：以 xml 格式表示 JavaFX 的界面对象的文件，用于界面框架设计
CSS：层叠样式表，修饰框架各组件的样式属性
Stage：JavaFX 顶层容器，类似 Swing 中的 Frame
NEON：Win10 中 NEON 毛玻璃设计/高斯模糊
Material：Win10 流畅设计中的材质设计
Light：Win10 流畅设计中的光感设计

## 1.4 参考资料

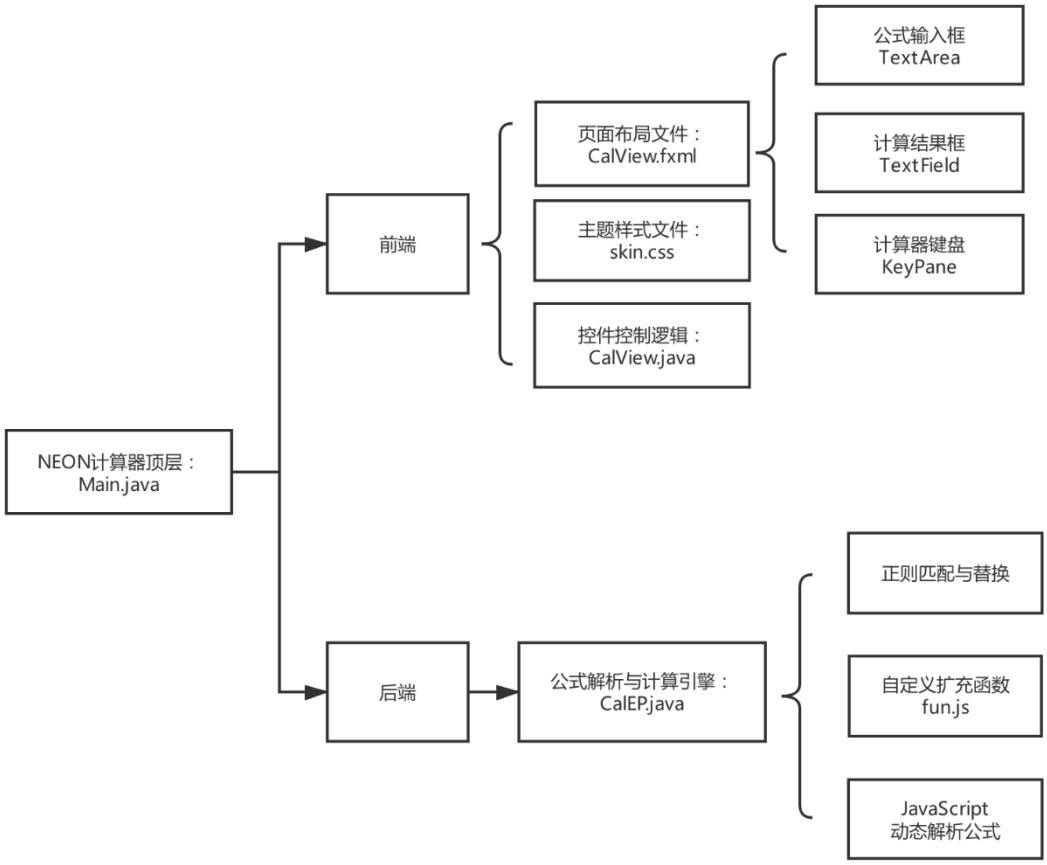
JavaFX2.2 开发 API https://docs.oracle.com/javafx/2/api/index.html

# 二、 总体设计

## 2.1 需求概述

按照题目要求，设计一个计算器，在拥有四则运算功能的基础上增加三角函数计算、开方运算、指数运算、幂运算、对数运算、阶乘、绝对值与取整等函数、e、π等常量。同时为满足个人对快速计算的需求，当按键输入长串的公式后能自动解析并计算出结果。在保证软件运行稳定、计算结果精确的情况下，为提升用户体验与界面美观程度，计算器的按键需要有交互反馈与流畅的动画以提升操作性，设计元素参考 Win10 扁平化流畅设计，界面设计参考 Win10 计算器 10.1910.0.0 版本（1910 版）。

## 2.2 软件结构




软件由前端与后端两部分，四大模块组成

# 三、 程序描述

## 3.1 顶层设计

顶层主文件 Main.java 功能：初始化 Stage，加载页面布局文件 CalView.fxml，加载主题样式文件 skin.css，具体代码如下：

```
public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root =
FXMLLoader.load(getClass().getResource("CalView.fxml"));
        primaryStage.setTitle("NEON 计算器");
        primaryStage.setMinHeight(700);
        primaryStage.setMinWidth(400);
        primaryStage.initStyle(StageStyle.TRANSPARENT);
        Scene scene = new Scene(root, 540, 800 , Color.TRANSPARENT);
        scene.getStylesheets().add("skin.css");
        scene.setFill(null);
        primaryStage.setScene(scene);
        // 拖动监听器
        DragUtil.addDragListener(primaryStage, root);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

初始化参数包含设置 Stage 为透明（StageStyle.*TRANSPARENT*），Scene540*800 大小、背景颜色为透明（Color.*TRANSPARENT*）。由于 Stage 为透明，使得 Win10 原生边框默认标题栏隐藏而无法拖动，这里增加拖动监听器类来监控鼠标拖动实现移动。

```
// 拖动监听器
class DragListener implements EventHandler<MouseEvent> {
    private double xOffset = 0;
    private double yOffset = 0;
    private final Stage stage;
    public DragListener(Stage stage) {
        this.stage = stage;
    }
    @Override
    public void handle(MouseEvent event) {
        event.consume();
        if (event.getEventType() == MouseEvent.MOUSE_PRESSED) {
            xOffset = event.getSceneX();
            yOffset = event.getSceneY();
        } else if (event.getEventType() == MouseEvent.MOUSE_DRAGGED) {
            stage.setX(event.getScreenX() - xOffset);
            if(event.getScreenY() - yOffset < 0) {
                stage.setY(0);
            }else {
                stage.setY(event.getScreenY() - yOffset);
            }
        }
    }
    public void enableDrag(Node node) {
        node.setOnMousePressed(this);
        node.setOnMouseDragged(this);
    }
}
class DragUtil {
    public static void addDragListener(Stage stage,Node root) {
        new DragListener(stage).enableDrag(root);
    }
}
```

以上内容即为顶层实现，具体的前端代码与业务逻辑将在后文阐述，其中本项目引入了以下原生类与方法：

Main.java 中：

```
import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.input.MouseEvent;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
```

页面布局文件 CalView.fxml 中：

```
<?import java.lang.String?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.MenuButton?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.effect.GaussianBlur?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.text.Font?>
```

控件控制逻辑 CalView.java 中：

```
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.*;
import javafx.scene.effect.GaussianBlur;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.GridPane;
import javax.script.ScriptEngineManager;
```

公式解析与计算引擎 CalEP.java 中：

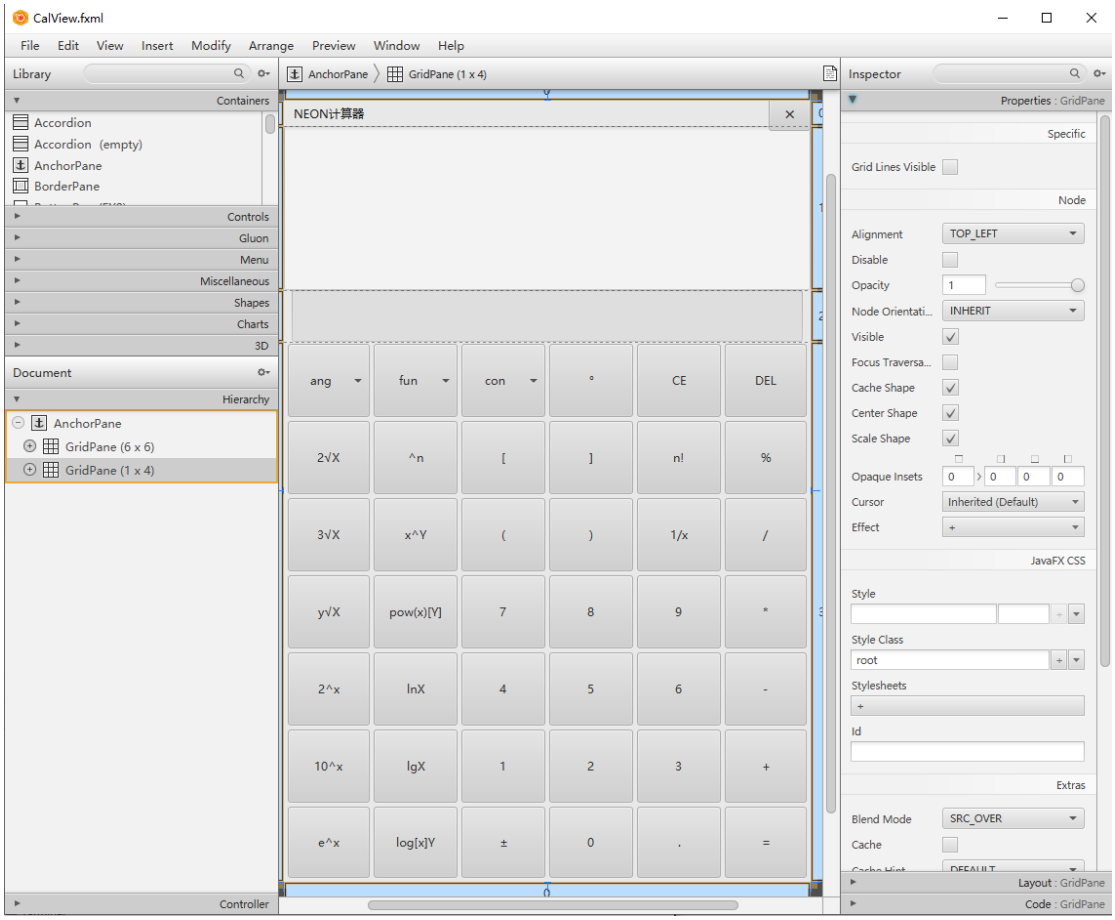
```
import javax.script.*;
import java.util.Stack;
```

## 3.2 前端-页面布局

页面布局使用 FXML 文件创建自定义界面，JavaFX FXML 基于 XML 语言并提供了将用户界面与程序逻辑代码分离的架构，可以使用 JavaFX Scene Builder 可视化布局环境迅速搭建界面框架。由于计算器按键与实现的函数特别多，使用代码布局界面将产生代码量庞大、很难调整控件位置、开发逻辑复杂等缺点；使用标记语言开发可以具有可以使用 CSS 样式实现自定制主题框架，前后端代码剥离，代码量大大减小等优点。

用户界面设计分为三部分：公式输入框、计算结果框、计算器键盘。公式输入框允许用户使用按键/键盘输入数学公式，使用“=”按钮可以计算结果并显示在计算结果框中。计算器键盘按下将触发控制逻辑代码，输入内容将在控制逻辑中提到。

以下为使用 Scene Builder 布局界面框架，可以看到样式均为默认样式，没有附加属性。框架中仅仅只是简单的设置了背景白色、透明度 0.9，按键的分布以及事件函数名以及各个控件的适合大小以及最大最小大小来适配拉伸等缩放场景，由于隐藏了系统自带标题栏样式，这里需要手动实现关闭按钮

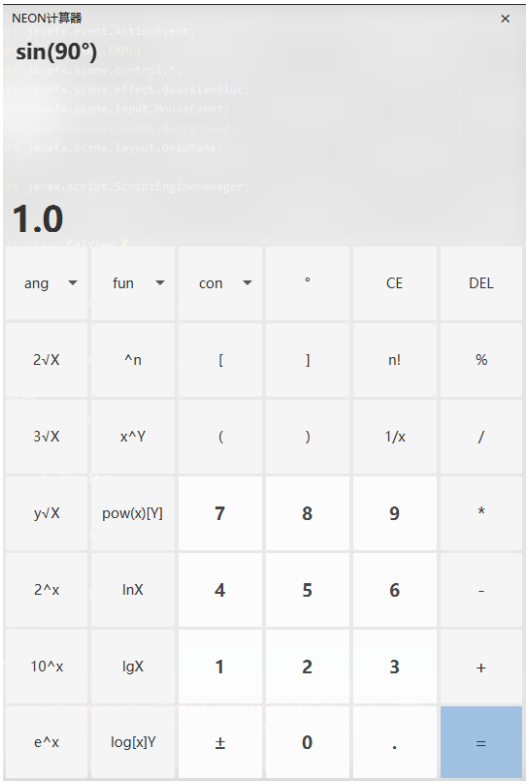


由于此部分标记语言代码较长，以下截取一部分来展示

```
<GridPane fx:id="key_7_6" onMouseEntered="#fun_gaussian_off"
onMouseMoved="#fun_gaussian_off" prefHeight="560" prefWidth="560"
GridPane.rowIndex="3">
        <columnConstraints>
            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
prefWidth="100.0" />
            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
prefWidth="100.0" />
            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
prefWidth="100.0" />
          <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
prefWidth="100.0" />
          <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
prefWidth="100.0" />
            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
prefWidth="100.0" />
        </columnConstraints>
        <rowConstraints>
            <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
          <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
          <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
          <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
        </rowConstraints>
         <children>
            <AnchorPane prefHeight="200.0" prefWidth="200.0">
               <children>
                  <MenuButton id="ang" fx:id="ang" alignment="CENTER"
layoutX="-18.0" layoutY="16.0" mnemonicParsing="false"
```

```
onMouseClicked="#fun_gaussian_on" prefHeight="59.697477077917625"
prefWidth="90.35293828009154" styleClass="btn-basic,btn-roman" text="ang"
textAlignment="CENTER" AnchorPane.bottomAnchor="2.0"
AnchorPane.leftAnchor="4.0" AnchorPane.rightAnchor="2.0"
AnchorPane.topAnchor="2.0">
                  <items>
                    <MenuItem fx:id="a_sin" mnemonicParsing="false"
onAction="#a_sin_fun" styleClass="btnin-basic" text="sin" />
                    <MenuItem fx:id="a_asin" mnemonicParsing="false"
onAction="#a_asin_fun" styleClass="btnin-basic" text="asin" />
                      <MenuItem fx:id="a_cos" mnemonicParsing="false"
onAction="#a_cos_fun" styleClass="btnin-basic" text="cos" />
                      <MenuItem fx:id="a_acos" mnemonicParsing="false"
onAction="#a_cos_fun" styleClass="btnin-basic" text="acos" />
                      <MenuItem fx:id="a_tan" mnemonicParsing="false"
onAction="#a_tan_fun" styleClass="btnin-basic" text="tan" />
                      <MenuItem fx:id="a_atan" mnemonicParsing="false"
onAction="#a_atan_fun" styleClass="btnin-basic" text="atan" />
                      <MenuItem fx:id="a_sec" mnemonicParsing="false"
onAction="#a_sec_fun" styleClass="btnin-basic" text="sec" />
                      <MenuItem fx:id="a_csc" mnemonicParsing="false"
onAction="#a_csc_fun" styleClass="btnin-basic" text="csc" />
                      <MenuItem fx:id="a_cot" mnemonicParsing="false"
onAction="#a_cot_fun" styleClass="btnin-basic" text="cot" />
                      <MenuItem fx:id="a_DEG" mnemonicParsing="false"
onAction="#a_DEG_fun" styleClass="btnin-basic" text="DEG" />
                      <MenuItem fx:id="a_RAD" mnemonicParsing="false"
onAction="#a_RAD_fun" styleClass="btnin-basic" text="RAD" />
                  </items>
                </MenuButton>
              </children>
            </AnchorPane>
```

以上代码规定了计算器键盘第一个 ang 按键的全部功能，按下 ang 角度函数按键，将会显示所有可用的三角函数，点击对应的函数可以在屏幕内输入对应函数，界面演示如下。


当按下按钮后计算器键盘全局将会应用毛玻璃/高斯模糊特效，显示用户正在互动的是附加子菜单，当鼠标移回计算器键盘将会取消该特效，建立了可视化的层次结构，提高了用户体验。

## 3.3 前端-主题样式文件

主题样式文件使用 CSS 样式表，样式表内规定了全局样式、通用按钮样式（包含鼠标经过、点击等样式）、特殊按钮样式（包含鼠标经过、点击等样式）、输入框样式等，提供基本的主题渲染，所有样式类包括但不限于如下：

全局样式.**root**

基本按钮样式.**btn-basic** 基本按钮鼠标浮动样式.**btn-equ:hover**

基本按钮鼠标点击样式.**btn-basic:pressed**

关闭按钮样式.**btn-close** 关闭按钮鼠标浮动样式.**btn-close:hover**

数字按钮样式.**btn-num**

等号按钮样式.**btn-equ** 等号按钮鼠标浮动样式.**btn-equ:hover**

等号按钮鼠标点击样式.**btn-equ:pressed**

加粗样式.**btn-blod**

大字体样式.**btn-bigfont**

下拉子菜单按钮样式.**btnin-basic** 下拉子菜单按钮鼠标浮动样式.**btnin-basic:hover**

鼠标聚焦样式.**btnin-basic:focused** 鼠标点击样式.**btnin-basic:pressed**

TextArea 文本输入框的一系列样式.**text-area** .**text-area.scroll-pane**

.**text-area.scroll-pane.viewpor** .**text-area.scroll-pane.content**

以下以.btn-basic 和.root 为例介绍主要的设计样式属性:

```
/*全局样式*/
.root{
    /*背景透明*/
    -fx-background-color: transparent;
    /*全局字体 14px*/
    -fx-font-size: 14px;
    /*全局字体微软雅黑*/
    -fx-font-family: "Microsoft YaHei UI";
}

/*按钮基础样式*/
.btn-basic {
    /*字体颜色黑色*/
    -fx-font-color: #010101;
    /*字体居中*/
    -fx-text-alignment: center;
    /*按键无边框*/
    -fx-border-style: none;
    /*鼠标样式点击*/
    -fx-cursor: pointer;
    /*边界圆角弧度*/
    -fx-border-radius: 1px;
    /*背景弧度*/
    -fx-background-radius: 1px;
    /*背景颜色白色*/
    -fx-background-color: #f5f5f5;
}
```

特殊的样式还有按键阴影，当鼠标移动到 按键上时，.btn-basic:hover 的样式将会应用-fx-effect: dropshadow(three-pass-box, #c7c7c7, 30,0, 0, 0);在该按钮地下渲染一层阴影，同时按钮变灰，这样随着鼠标移动，不同按键产生与周围不同的颜色以及阴影反馈，这样可以突出显示用户正在互动的控件状态，提高了用户体验与界面美观度。


以上设计特性均参考自 Win10 扁平化流畅设计，实现了包括以下效果：

**背景半透明**：当难以实现 Win10 毛玻璃背景的替代产物

按键与输入框全局**扁平化设计**

**Material 设计**：物理纹理上，数字键与其他按键透明度颜色等材质不同，使其拥有较高的对比度和易读性；空间深度上，ang、fun、con 等扩充功能子菜单展开后，全局按键将会使用高斯模糊以及阴影，增加了用户界面的视觉深度和维度，建立可视化层次结构

**Light 设计**：随着鼠标移动，不同按键产生与周围不同的颜色以及阴影反馈，这样可以突出显示用户正在互动的控件状态

## 3.4 前端-控件控制逻辑

控制逻辑使用 java 编写全部监听事件，事件包含每个按钮按下的输入功能，等于符号按下的计算功能，删除和清除的逻辑，鼠标按下 ang、fun、con 等扩充功能的按键高斯模糊函数实现。以下为所有扩充功能按键的输入内容，与用户输入格式要求，光标位置未填写说明光标在输入值后。

| 扩充功能按钮 | 下拉按钮值 | 实际输入值 | 光标位置 | 备注 |
|---|---|---|---|---|
| ang | sin | sin() | ()中 | |
| | asin | asin() | ()中 | 返回弧度 |
| | cos | cos () | ()中 | |
| | acos | acos() | ()中 | 返回弧度 |
| | tan | tan() | ()中 | |
| | atan | atan() | ()中 | 返回弧度 |
| | sec | sec() | ()中 | |
| | csc | csc() | ()中 | |
| | cot | cot() | ()中 | |
| | DEG | DEG() | ()中 | 角度转弧度 |
| | RAD | RAD() | ()中 | 弧度转角度 |

| | | | | |
|---|---|---|---|---|
| fun | \|x\| | abs() | ()中 | 求 x 绝对值 |
| | ceil(x) | ceil() | ()中 | x 向上取整 |
| | floor(x) | floor() | ()中 | x 向下取整 |
| | round(x) | round() | ()中 | x 四舍五入 |
| | random() | random() | | 随机数不需要参数 |
| | max(x,y) | max(.) | ,前 | x,y 最大值 |
| | min(x,y) | min(,) | ,前 | x,y 最小值 |
| con | e | e | | 欧拉常数，约等于 2.718 |
| | π | π | | 圆周率，约等于 3.14159 |
| | ln2 | LN2 | | 2 的自然对数，约等于 0.693 |
| | ln10 | LN10 | | 0 的自然对数，约等于 2.303 |
| | log[2]e | LOG2E | | 以 2 为底 E 的对数，约等于 1.443 |
| | log[10]e | LOG10E | | 以 10 为底 E 的对数，约等于 0.434 |
| | √2 | SQRT2 | | 2 的平方根,约等于 1.414 |
| | 1/√2 | SQRT1_2 | | 1/2 的平方根，约等于 0.707 |

以下为其他按钮的输入输入内容与格式:

| 按钮值 | 输入值 | 光标位置 | 备注 |
|---|---|---|---|
| ° | ° | | 角度符号 |
| CE | 输入框全部清除 | 初始位置 | 清除按钮 |
| DEL | 输入框退格 | 退格后位置 | 退格删除按钮 |
| n! | ! | | 先输入 n |
| 1/x | 1/ | | 后输入 x |
| 2√X | sqrt() | ()中 | 开 X 的平方 |
| 3√X | cbrt() | ()中 | 开 X 的三次方 |
| y√X | root[]() | []中 | 开(X)的[y]次方 |
| 2^x | 2^() | ()中 | 2 的 x 次方 |
| 10^x | 10^() | ()中 | 10 的 x 次方 |
| e^x | e^ | | e 的 x 次方 |
| ^n | ^() | ()中 | 先输入数值再算 n 次方 |
| x^Y | ()^() | 第一个()中 | x 的 Y 次方 |
| pow(x)[Y] | pow()[] | ()中 | x 的 Y 次方标准实现 |

| lnX | ln() | ()中 | 以 e 为底 X 的对数 |
|---|---|---|---|
| lgX | lg() | ()中 | 以 10 为底 X 的对数 |
| log[x]Y | log[]() | []中 | 以 x 为底 Y 的对数 |
| [ | [ | | 中括号 |
| ] | ] | | 中括号 |
| ( | ( | | 小括号 |
| ) | ) | | 小括号 |
| 1 | 1 | | |
| 2 | 2 | | |
| 3 | 3 | | |
| 4 | 4 | | |
| 5 | 5 | | |
| 6 | 6 | | |
| 7 | 7 | | |
| 8 | 8 | | |
| 9 | 9 | | |
| 0 | 0 | | |
| . | . | | 小数点 |
| ± | neg() | ()中 | 正负转换，等效- |
| % | % | | 取余 |
| / | / | | |
| * | * | | |
| - | - | | |
| + | + | | |
| = | = | | 等号计算 |

## 3.5 后端-公式解析与计算

后端的公式解析与计算引擎具有将输入的公式自动解析成 Java/ JavaScript 中 Math 对象方法与属性的功能并直接计算结果，使用 JavaScript 动态语言还可以随时扩充计算函数，实时反馈计算结果。

实现解析首先需要对于用户输入的公式去掉空格（方便后续解析式子，防止用户定义 JavaScript 的变量函数等破坏后端程序），由于输入是严格控制()[]的对称规则的，当用户输入的()[]不对称时，将会停止计算并提示用户修改输入。

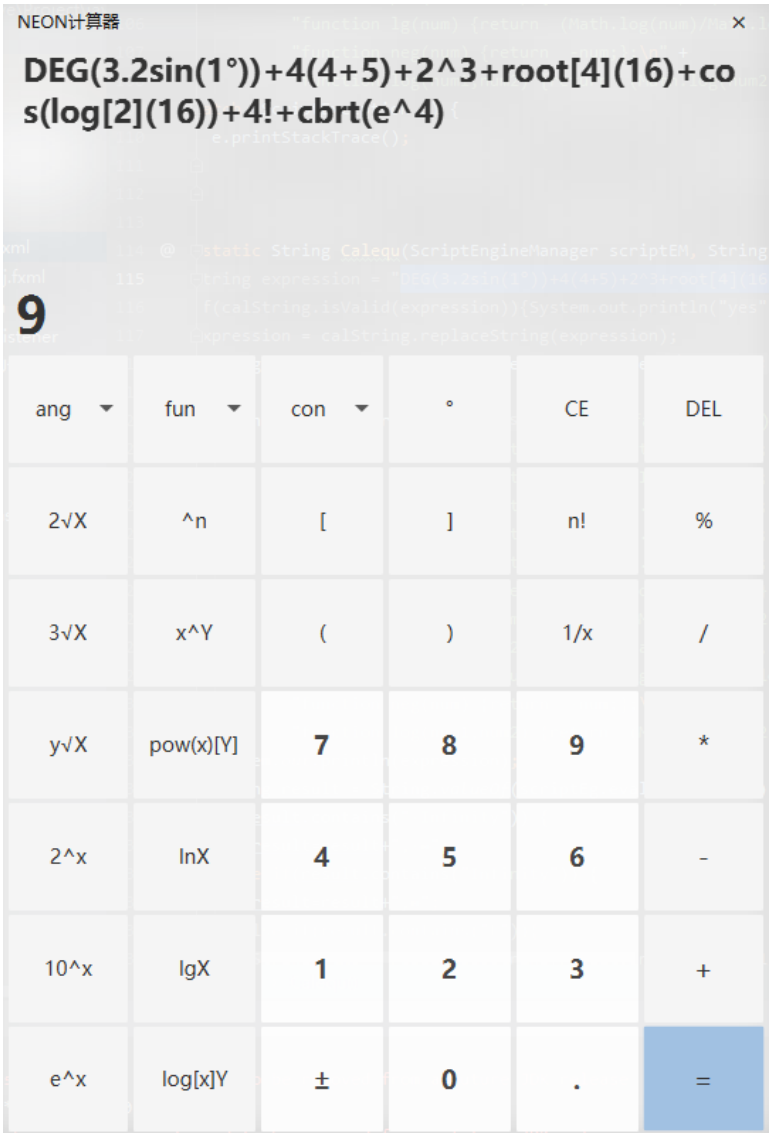
随后对输入公式使用一系列的正则表达式匹配并替换自然数学函数为 JavaScript 可以解析的式子，例如 e 替换为 Math.E，(\d+)\^\(([^\(]*)\)替换为 pow($1,$2)；同时还要处理数字位于()与函数前的乘法忽略，使用(?<![a-zA-z])(\d+)(?=[a-zA-Z])替换$1*可以实现 2sin 替换

为 2*sin，以下给出全部正则匹配替换公式:

```
public String replaceString(String input){
    //取余%与四则运算原生方法
    input = input.replace(" ","");
    //原生符号
    input = input.replaceAll("pi|PI|Pi|pI|π","Math.PI");
    input = input.replace("LN2","Math.LN2");
    input = input.replace("LN10","Math.LN10");
    input = input.replace("LOG2E","Math.LOG2E");
    input = input.replace("LOG10E","Math.LOG10E");
    input = input.replace("SQRT1_2","Math.SQRT1_2");
    input = input.replace("SQRT2","Math.SQRT2");
    //角度与弧度的实现
    input = input.replace("°","*(Math.PI/180)");
    //input = input.replaceAll("DEG(?=\\()","(180/Math.PI)*");
    //input = input.replaceAll("RAD(?=\\()","(Math.PI/180)*");
    //乘法补充
    input = input.replaceAll("\\)\\(",")*(");
    input = input.replaceAll("(?<![a-zA-z])(\\d+)(?=\\()","$1*");
    input = input.replaceAll("(?<![a-zA-z])(\\d+)(?=[a-zA-Z])","$1*");
    input = input.replace("LOG10*E","LOG10E");
    //原生三角函数
    input = input.replaceAll("(?<!a)sin(?=\\()","Math.sin");
    input = input.replaceAll("(?<!\\.)asin(?=\\()","Math.asin");
    input = input.replaceAll("(?<!a)cos(?=\\()","Math.cos");
    input = input.replaceAll("(?<!\\.)acos(?=\\()","Math.acos");
    input = input.replaceAll("(?<!a)tan(?=\\()","Math.tan");
    input = input.replaceAll("(?<!\\.)atan(?=\\()","Math.atan");
    input = input.replaceAll("(?<!\\.)atan2(?=\\()","Math.atan2");
    //扩充三角函数
    //input = input.replaceAll("(?<!\\.)sec\\(([^\\(]*)\\)","(1/Math.cos($1))");
    //input = input.replaceAll("(?<!\\.)csc\\(([^\\(]*)\\)","(1/Math.sin($1))");
    //input = input.replaceAll("(?<!\\.)cot\\(([^\\(]*)\\)","(1/Math.tan($1))");
    //原生取整
    input = input.replaceAll("(?<!\\.)ceil(?=\\()","Math.ceil");
    input = input.replaceAll("(?<!\\.)floor(?=\\()","Math.floor");
    input = input.replaceAll("(?<!\\.)round(?=\\()","Math.round");
    input = input.replaceAll("(?<!\\.)random(?=\\()","Math.random");
    //e 相关
```

```
    input =
input.replaceAll("(?<!\\.)e\\^((\\d+(\\.\\d+)?)|([+-]?(\\d+(\\.\\d+)?)))","
Math.exp($1)");
    input = input.replaceAll("(?<!\\.)e\\^(?=\\()","Math.exp");
    input = input.replaceAll("(?<!\\.|s|D|c|n)e(?!\\^)","Math.E");
    //任意指数实现
    input = input.replaceAll("(\\d+)\\^\\(([^\\(]*)\\)","pow($1,$2)");
    input = input.replaceAll("\\(([^\\(]*)(?=\\)\\^)","pow($1");
    input = input.replaceAll("\\)\\^\\(",",");
    input = input.replaceAll("\\)\\[",",");
    input = input.replaceAll("\\](?!\\()",")");
    //开方
    input = input.replaceAll("(?!\\.)sqrt(?=\\()","Math.sqrt");
    //input = input.replaceAll("(?!\\.)cbrt\\(([^\\(]*)\\)","Math.pow($1,1/3)");
    //input = input.replaceAll("(?!\\.)root\\[([^\\[]*)\\]\\(([^\\(]*)\\)","Math.pow($2,1/($1))");
    //对数实现
    input = input.replaceAll("(?!\\.)ln(?=\\()","Math.log");
    //input =
input.replaceAll("(?!\\.)log\\[([^\\[]*)\\]\\(([^\\(]*)\\)","(Math.log($2)/Math.log($1))");
    //比较与绝对值
    input = input.replaceAll("(?!\\.)max(?=\\()","Math.max");
    input = input.replaceAll("(?!\\.)min(?=\\()","Math.min");
    input = input.replaceAll("(?!\\.)abs(?=\\()","Math.abs");
    //！与（）！阶乘实现
    input = input.replaceAll("\\(([^\\(]*)\\)!","factorial($1)");
    input = input.replaceAll("(\\d+)!","factorial($1)");
    //对数与指数修改解析
    input = input.replaceAll("(?<!\\))\\[","(");
    input = input.replaceAll("\\]\\(",",");
    //返回解析后的
    return input;
}
```

替换完成后后只要加载扩充函数 JavaScript 文件，然后在 Java 中使用 scriptEg 运行 JavaScript 代码，最后把计算得到的结果放到输出结果框，例如输入：DEG(3.2sin(1°))+4(4+5)+2^3+root[4](16)+cos(log[2](16))+4!+cbrt(e^4)，控制台打印解析后的公式为结果为 DEG(3.2*Math.sin(1*(Math.PI/180)))+4*(4+5)+2^3+root(4,16)+Math.cos(log(2,16))+factorial(4)+cbrt(Math.exp(4))，同时计算器结果输出框显示 9，如下图所示




扩充 JavaScript 函数如下：

```
function factorial(num){if(num <= 1) {return 1;}else{return num *
factorial(num-1);}};
function DEG(num) {return  (180/Math.PI)*num;};
function RAD(num) {return  (Math.PI/180)*num;};
function sec(num) {return  (1/Math.cos(num));};
function csc(num) {return  (1/Math.sin(num));};
function cot(num) {return  (1/Math.tan(num));};
function cbrt(num) {return  Math.pow(num,1/3);};
function root(num1,num2) {return  Math.pow(num2,1/(num1));};
function Log(num1,num2) {return  (Math.log(num2)/Math.log(num1));};
```

计算错误信息测试举例:

**100/0：Infinity,∞**

**-100/0：-Infinity,-∞**

括号未对齐:

**(9+8：NaN / not aligned**

函数等使用不规范:

**9+8floor()：NaN**，此时抛出异常，大多数异常由此造成

什么都不输入：**null**

正确解析:

**DEG(3.2sin(1°))+4(4+5)+2^3+root[4](16)+cos(log[2](16))+4!+cbrt(e^4)：9**