计算器

1.计算器上的键的显示名字

1.0 继承 JFrame 类

```
public class Calculate extends JFrame {

}
```

1.1 定义常量

```
    /** 计算器上的键的显示名字 */
    public final String[] KEYS = { "7", "8", "9", "/", "sqrt", "4",
"5", "6", "*", "%", "1", "2", "3", "-", "1/x", "0", "+/-", ".", "+",
"=" };
    /** 计算器上的功能键的显示名字 */
    public final String[] COMMAND = { "Backspace", "CE", "C" };

    /** 计算器左边的M的显示名字 */
    private final String[] M = { " ", "MC", "MR", "MS", "M+" };
```

1.2 为对应的按键开辟空间

```
    /** 计算器上的功能键的按钮 */
    private JButton commands[] = new JButton[COMMAND.length];

    /** 计算器左边的M的按钮 */
    private JButton m[] = new JButton[M.length];

    /** 计算结果文本框 */
    private JTextField resultText = new JTextField("0");
```

1.3 最终效果图：

2 内部布局代码

2.1 按键按钮代码

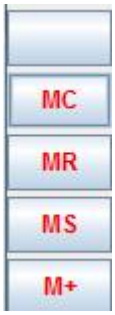
```
// 初始化计算器上键的按钮，将键放在一个画板内
        JPanel keyPanel = new JPanel();
        // 用网格布局器，4行，5列的网格，网格之间的水平方向间隔为3个象素，垂直方向间隔为3个象素
        keyPanel.setLayout(new GridLayout(4, 5, 3, 3));
        for (int i = 0; i < KEYS.length; i++) {
            // 数字放入到按键中
            keys[i] = new JButton(KEYS[i]);
            // 将按键添加到面板
            keyPanel.add(keys[i]);
            // 设置颜色为淡蓝色
            keys[i].setForeground(Color.blue);
        }
        // 运算符键用红色标示，其他键用蓝色表示
        keys[3].setForeground(Color.red);
        keys[8].setForeground(Color.red);
        keys[13].setForeground(Color.red);
        keys[18].setForeground(Color.red);
        keys[19].setForeground(Color.red);
```

2.2 初始化功能键代码

```
// 初始化功能键，都用红色标示。将功能键放在一个画板内
        JPanel commandsPanel = new JPanel();
        // 用网格布局器，1行，3列的网格，网格之间的水平方向间隔为3个象素，垂直方向间隔为3个象素
        commandsPanel.setLayout(new GridLayout(1,3,3,3));
        for(int i = 0; i < COMMAND.length; i++){
            commands[i] = new JButton(COMMAND[i]);
            commandsPanel.add(commands[i]);
            commands[i].setForeground(Color.red);
        }
```

2.3 初始化 M 键代码

```
        // 初始化M键，用红色标示，将M键放在一个画板内
        JPanel calmsPanel = new JPanel();
        calmsPanel.setLayout(new GridLayout(5,1,3,3));
        for(int i = 0 ; i < M.length; i++){
            m[i] = new JButton(M[i]);
            calmsPanel.add(m[i]);
            m[i].setForeground(Color.red);
        }
```



2.3 设置文本框代码

```
// 文本框中的内容采用右对齐方式
        resultText.setHorizontalAlignment(JTextField.RIGHT);
        // 不允许修改结果文本框
        resultText.setEditable(false);
        // 设置文本框背景颜色为白色
        resultText.setBackground(Color.WHITE);
```

2.4 整体布局代码

```
        JPanel panel1 = new JPanel();
// 画板采用边界布局管理器，画板里组件之间的水平和垂直 间隔都为3象素
        panel1.setLayout(new BorderLayout(3, 3));
        panel1.add("Center", keyPanel);
        panel1.add("North",commandsPanel);
```


```
        // 建立一个画板放文本框
        JPanel top = new JPanel();
        top.setLayout(new BorderLayout());
        top.add("Center", resultText);
```


```
// 整体布局
//画板里组件之间的水平间隔都为3象素,垂直方向上间隔都为5象素
getContentPane().setLayout(new BorderLayout(3, 5));
getContentPane().add("North", top);
getContentPane().add("Center", panel1);
getContentPane().add("West", calmsPanel);
```

2.5 添加事件监听

实现 ActionListener 类，复写 actionPerformed 方法

```
public class Test1 extends JFrame implements ActionListener
{                    ...
    //为各按钮添加事件侦听器
    //都使用同一个事件侦听器，即本对象。本类的声明中有implements
ActionListener
        for(int i = 0; i < KEYS.length; i++){
            keys[i].addActionListener(this);
        }
        for(int i = 0; i < COMMAND.length; i++){
            commands[i].addActionListener(this);
        }
        for(int i = 0; i < M.length; i++){
            m[i].addActionListener(this);
        }
}
```

2.6 处理回退事件

```
/** 处理Backspace键被按下的事件 */
    private void handleBackspace() {
        //获取文本框的内容
        String text = resultText.getText();
        int len  = text.length();
        if( len > 0){
            //表示有数字写入
            text = text.substring(0, len - 1);
            if(text.length() == 0){
                // 如果文本没有了内容，则初始化计算器的各种值
                resultText.setText("0");
            }else{
                resultText.setText(text);
            }
        }
```

2.7 处理 CE 按键

```
// 用户按了"CE"键
            resultText.setText("0");
```

2.8 处理数字

定义变量：

```
// 标志用户按的是否是整个表达式的第一个数字,或者是运算符后的第一个数字
    private boolean firstDigit = true;
    /**
     * 处理数字键被按下的事件
     */
    private void handleNumber(String label) {
        //firstDigit 默认是true
        if(firstDigit){
```

```
        //用户按的是第一个是数字
            resultText.setText(label);
        }else if((label.equals(".")) &&
(resultText.getText().indexOf(".") < 0) ){
        // 输入的是小数点，并且之前没有小数点，则将小数点附在结果文本框
的后面
            resultText.setText(resultText.getText() + ".");
        }else if(! label.equals(".")){
            // 如果输入的不是小数点，则将数字附在结果文本框的后面
            resultText.setText(resultText.getText() + label);
        }
        // 以后输入的肯定不是第一个数字了
        firstDigit = false;
    }
```

2.9 处理运算符键被按下的事件

0. 设置操作是否正常标志位 operateValidFlag = true

   设置中间变量 resultNum = 0.0

   设置操作符变量 operator = "" 主要数获取当前页显示的数字

1. "=" 主要是用来显示数字，可以是当前输入的数字，

   也可以是用算后赋值给界面的数字

2. 赋值当前的操作符号给变量 operator

3. 设置 firstDigt = true 用于下次界面的显示

```
    //当前运算的运算符
    private String operator = "=";
    //设置一个标志位，判断是否合法
    private boolean operateValidFlag = true;
    // 计算的中间结果。
    private double resultNum = 0.0;
```

```
     * 处理运算符键被按下的事件
     * @param key
     */
    private void handleOperator(String label) {
        // 除法运算
        if (operator.equals("/")) {
            if (getNumberFromText() == 0.0) {
                // 操作不合法
                operateValidFlag = false;
                resultText.setText("除数不能为零");
            } else {
                resultNum /= getNumberFromText();
            }
            // 倒数运算
        } else if (operator.equals("1/x")) {
            if (resultNum == 0.0) {
                operateValidFlag = false;
                resultText.setText("除数不能为零");
            } else {
                resultNum = 1 / resultNum;
            }
        } else if (operator.equals("+")) {
            // 加法运算
            resultNum += getNumberFromText();
        } else if (operator.equals("-")) {
            // 减法运算
            resultNum -= getNumberFromText();
        } else if (operator.equals("*")) {
            // 乘法运算
            resultNum *= getNumberFromText();
        } else if (operator.equals("sqrt")) {
            // 开方运算
            resultNum = Math.sqrt(resultNum);
        } else if (operator.equals("+/-")) {
            // 正数负数运算
            resultNum = resultNum * (-1);
        } else if (operator.equals("=")) {
            // 赋值运算
            resultNum = getNumberFromText();
        }
        if(operateValidFlag){
```

```
            // 双精度浮点数的运算
            long t1 ;
            double t2;
            t1 = (long) resultNum;
            t2 = resultNum - t1;
            if(t2 == 0){
                resultText.setText(String.valueOf(t1));

    System.out.println("resultText.setText(String.valueOf(t1))" +
String.valueOf(t1));
            }else{
                resultText.setText(String.valueOf(resultNum));

    System.out.println("resultText.setText(String.valueOf(resultNum))
" + String.valueOf(resultNum));
            }
        }
            // 运算符等于用户按的按钮
            operator = label;
            firstDigit = true;
            operateValidFlag = true;
    }




    /**
     * 从结果文本框中获取数字
     */
    private double getNumberFromText() {
        double result = 0;
        result = Double.valueOf(resultText.getText());
        return result;
    }
```

2.10 处理 C 按键

```
    /**
```

```
     * 处理C键被按下的事件
     */
    private void handleC() {
        // 初始化计算器的各种值
        resultText.setText("0");
        this.firstDigit = true;
    }
```

3 附完整代码：

```
package test;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 * 一个计算器，与Windows附件自带计算器的标准版功能、界面相仿。 但还不支持键盘操作。
 */
public class T extends JFrame implements ActionListener {
    /** 计算器上的键的显示名字 */
    private final String[] KEYS = { "7", "8", "9", "/", "sqrt", "4", "5", "6",
            "*", "%", "1", "2", "3", "-", "1/x", "0", "+/-", ".", "+", "=" };
    /** 计算器上的功能键的显示名字 */
    private final String[] COMMAND = { "Backspace", "CE", "C" };
    /** 计算器左边的M的显示名字 */
    private final String[] M = { " ", "MC", "MR", "MS", "M+" };
    /** 计算器上键的按钮 */
```

```
    private JButton keys[] = new JButton[KEYS.length];
    /** 计算器上的功能键的按钮 */
    private JButton commands[] = new JButton[COMMAND.length];
    /** 计算器左边的M的按钮 */
    private JButton m[] = new JButton[M.length];
    /** 计算结果文本框 */
    private JTextField resultText = new JTextField("0");

    // 标志用户按的是否是整个表达式的第一个数字,或者是运算符后的第一个数字
    private boolean firstDigit = true;
    // 计算的中间结果。
    private double resultNum = 0.0;
    // 当前运算的运算符
    private String operator = "=";
    // 操作是否合法
    private boolean operateValidFlag = true;

    /**
     * 构造函数
     */
    public T() {
        super();
        // 初始化计算器
        init();
        // 设置计算器的背景颜色
        this.setBackground(Color.LIGHT_GRAY);
        this.setTitle("计算器");
        // 在屏幕(500, 300)坐标处显示计算器
        this.setLocation(500, 300);
        // 不许修改计算器的大小
        this.setResizable(false);
        // 使计算器中各组件大小合适
        this.pack();
    }

    /**
     * 初始化计算器
     */
    private void init() {
        // 文本框中的内容采用右对齐方式
```

```
        resultText.setHorizontalAlignment(JTextField.RIGHT);
        // 不允许修改结果文本框
        resultText.setEditable(false);
        // 设置文本框背景颜色为白色
        resultText.setBackground(Color.WHITE);

        // 初始化计算器上键的按钮，将键放在一个画板内
        JPanel calckeysPanel = new JPanel();
        // 用网格布局器，4行，5列的网格，网格之间的水平方向间隔为3个象素，垂直方向间隔为3个象素
        calckeysPanel.setLayout(new GridLayout(4, 5, 3, 3));
        for (int i = 0; i < KEYS.length; i++) {
            keys[i] = new JButton(KEYS[i]);
            calckeysPanel.add(keys[i]);
            keys[i].setForeground(Color.blue);
        }
        // 运算符键用红色标示，其他键用蓝色表示
        keys[3].setForeground(Color.red);
        keys[8].setForeground(Color.red);
        keys[13].setForeground(Color.red);
        keys[18].setForeground(Color.red);
        keys[19].setForeground(Color.red);

        // 初始化功能键，都用红色标示。将功能键放在一个画板内
        JPanel commandsPanel = new JPanel();
        // 用网格布局器，1行，3列的网格，网格之间的水平方向间隔为3个象素，垂直方向间隔为3个象素
        commandsPanel.setLayout(new GridLayout(1, 3, 3, 3));
        for (int i = 0; i < COMMAND.length; i++) {
            commands[i] = new JButton(COMMAND[i]);
            commandsPanel.add(commands[i]);
            commands[i].setForeground(Color.red);
        }

        // 初始化M键，用红色标示，将M键放在一个画板内
        JPanel calmsPanel = new JPanel();
        // 用网格布局管理器，5行，1列的网格，网格之间的水平方向间隔为3个象素，垂直方向间隔为3个象素
        calmsPanel.setLayout(new GridLayout(5, 1, 3, 3));
        for (int i = 0; i < M.length; i++) {
            m[i] = new JButton(M[i]);
```

```
            calmsPanel.add(m[i]);
            m[i].setForeground(Color.red);
        }

        // 下面进行计算器的整体布局，将calckeys和command画板放在计算器的中部，
        // 将文本框放在北部，将calms画板放在计算器的西部。

        // 新建一个大的画板，将上面建立的command和calckeys画板放在该画板内
        JPanel panel1 = new JPanel();
        // 画板采用边界布局管理器，画板里组件之间的水平和垂直方向上间隔都为3象素
        panel1.setLayout(new BorderLayout(3, 3));
        panel1.add("North", commandsPanel);
        panel1.add("Center", calckeysPanel);

        // 建立一个画板放文本框
        JPanel top = new JPanel();
        top.setLayout(new BorderLayout());
        top.add("Center", resultText);

        // 整体布局
        getContentPane().setLayout(new BorderLayout(3, 5));
        getContentPane().add("North", top);
        getContentPane().add("Center", panel1);
        getContentPane().add("West", calmsPanel);
        // 为各按钮添加事件侦听器
        // 都使用同一个事件侦听器，即本对象。本类的声明中有implements ActionListener
        for (int i = 0; i < KEYS.length; i++) {
            keys[i].addActionListener(this);
        }
        for (int i = 0; i < COMMAND.length; i++) {
            commands[i].addActionListener(this);
        }
        for (int i = 0; i < M.length; i++) {
            m[i].addActionListener(this);
        }
    }
```

```
/**
 * 处理事件
 */
public void actionPerformed(ActionEvent e) {
    // 获取事件源的标签
    String label = e.getActionCommand();
    if (label.equals(COMMAND[0])) {
        // 用户按了"Backspace"键
        handleBackspace();
    } else if (label.equals(COMMAND[1])) {
        // 用户按了"CE"键
        resultText.setText("0");
    } else if (label.equals(COMMAND[2])) {
        // 用户按了"C"键
        handleC();
    } else if ("0123456789.".indexOf(label) >= 0) {
        // 用户按了数字键或者小数点键
        handleNumber(label);
        // handlezero(zero);
    } else {
        // 用户按了运算符键
        handleOperator(label);
    }
}

/**
 * 处理Backspace键被按下的事件
 */
private void handleBackspace() {
    String text = resultText.getText();
    int i = text.length();
    if (i > 0) {
        // 退格，将文本最后一个字符去掉
        text = text.substring(0, i - 1);
        if (text.length() == 0) {
            // 如果文本没有了内容，则初始化计算器的各种值
            resultText.setText("0");
            firstDigit = true;
            operator = "=";
        } else {
            // 显示新的文本
```

```
                resultText.setText(text);
            }
        }
    }

    /**
     * 处理数字键被按下的事件
     *
     * @param key
     */
    private void handleNumber(String key) {
        if (firstDigit) {
            // 输入的第一个数字
            resultText.setText(key);
        } else if ((key.equals(".")) &&
(resultText.getText().indexOf(".") < 0)) {
            // 输入的是小数点，并且之前没有小数点，则将小数点附在结果文
本框的后面
            resultText.setText(resultText.getText() + ".");
        } else if (!key.equals(".")) {
            // 如果输入的不是小数点，则将数字附在结果文本框的后面
            System.out.println("resultText.getText()" +
resultText.getText());
            resultText.setText(resultText.getText() + key);
            System.out.println("resultText.getText()" +
resultText.getText() + key);
        }
        // 以后输入的肯定不是第一个数字了
        firstDigit = false;
    }

    /**
     * 处理C键被按下的事件
     */
    private void handleC() {
        // 初始化计算器的各种值
        resultText.setText("0");
        firstDigit = true;
        operator = "=";
    }
```

```
    /**
     * 处理运算符键被按下的事件
     */
    private void handleOperator(String key) {
        System.out.println(operator.equals("/"));
        if (operator.equals("/")) {
            System.out.println("进入到除法里面");
            // 除法运算
            // 如果当前结果文本框中的值等于0
            if (getNumberFromText() == 0.0) {
                // 操作不合法
                operateValidFlag = false;
                resultText.setText("除数不能为零");
            } else {
                System.out.println("resultNum:" + resultNum);
                resultNum /= getNumberFromText();
                System.out.println("resultNum /=
getNumberFromText():" + resultNum);

            }
        } else if (operator.equals("1/x")) {
            // 倒数运算
            if (resultNum == 0.0) {
                // 操作不合法
                operateValidFlag = false;
                resultText.setText("零没有倒数");
            } else {
                resultNum = 1 / resultNum;
            }
        } else if (operator.equals("+")) {
            // 加法运算
            resultNum += getNumberFromText();
        } else if (operator.equals("-")) {
            // 减法运算
            resultNum -= getNumberFromText();
        } else if (operator.equals("*")) {
            // 乘法运算
            resultNum *= getNumberFromText();
        } else if (operator.equals("sqrt")) {
            // 平方根运算
            resultNum = Math.sqrt(resultNum);
```

```
        } else if (operator.equals("%")) {
            // 百分号运算，除以100
            resultNum = resultNum / 100;
        } else if (operator.equals("+/-")) {
            // 正数负数运算
            resultNum = resultNum * (-1);
        } else if (operator.equals("=")) {
            // 赋值运算
            resultNum = getNumberFromText();
        }
        if (operateValidFlag) {
            // 双精度浮点数的运算
            long t1;
            double t2;
            t1 = (long) resultNum;
            System.out.println(t1);
            t2 = resultNum - t1;
            System.out.println(t2);
            if (t2 == 0) {
                resultText.setText(String.valueOf(t1));
            } else {
                resultText.setText(String.valueOf(resultNum));

    System.out.println("resultText.setText(String.valueOf(resul
tNum))" + String.valueOf(resultNum));
            }
        }
        // 运算符等于用户按的按钮
        operator = key;
        firstDigit = true;
        operateValidFlag = true;
    }

    /**
     * 从结果文本框中获取数字
     *
     * @return
     */
    private double getNumberFromText() {
        double result = 0;
        try {
```

```
            result =
Double.valueOf(resultText.getText()).doubleValue();
        } catch (NumberFormatException e) {
        }
        return result;
    }

    public static void main(String args[]) {
        T calculator1 = new T();
        calculator1.setVisible(true);

    calculator1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```