1.倒计时器

设计流程为：了解“倒计时器”功能→进行窗口布局→美化界面→实现窗口中各个组件的预设功能→调试→完成选题。现在就设计结果说明如下。

（1）程序初始运行界面图 1。其中标题为“倒计时器”，界面采用网格布局，功能分区合理；并设置了字体的样式及颜色，按钮也用了彩色，显得醒目；第一行是“时、分、秒”三个标签，显示时间的单位；第二行是三个文本区，用于输入时间并在倒计时开始后显示时间；第三行是两个按钮，第一个具有“开始”与“暂停”双重功能，第二个用于复位。

图 1 初始界面

（2）先输入需要倒计的时间。

图 2 输入倒计时间

（3）点击“开始”，文本区变灰（此时不可编辑，只用于显示剩余时间），同时“开始”按钮变为“暂停”按钮（以便暂停倒计时），倒计时开始。

图 3 点击“开始”按钮后启动倒计时

（3）随时按下“复位”按钮，则界面变回到图 1 的初始状态。

（4）如果按点击“暂停”，则倒计时暂停，等待重新开始或复位。

图 4　点击“暂停”按钮后倒计时暂停

（5）倒计时结束后弹出提示窗口，并有声音提示。

图 5 倒计时结束

（6）异常处理。鉴于显示生活中倒计时器的时间一般不会太长，故本倒计时器的时间设定范围为 1 秒~23 小时 59 分 59 秒。若输入非数字字符、小数、负数、不在设定时间范围内的正整数，则弹出提示窗口。

图 6　输入非数字字符

图 7　输入小数

图 8　输入负数

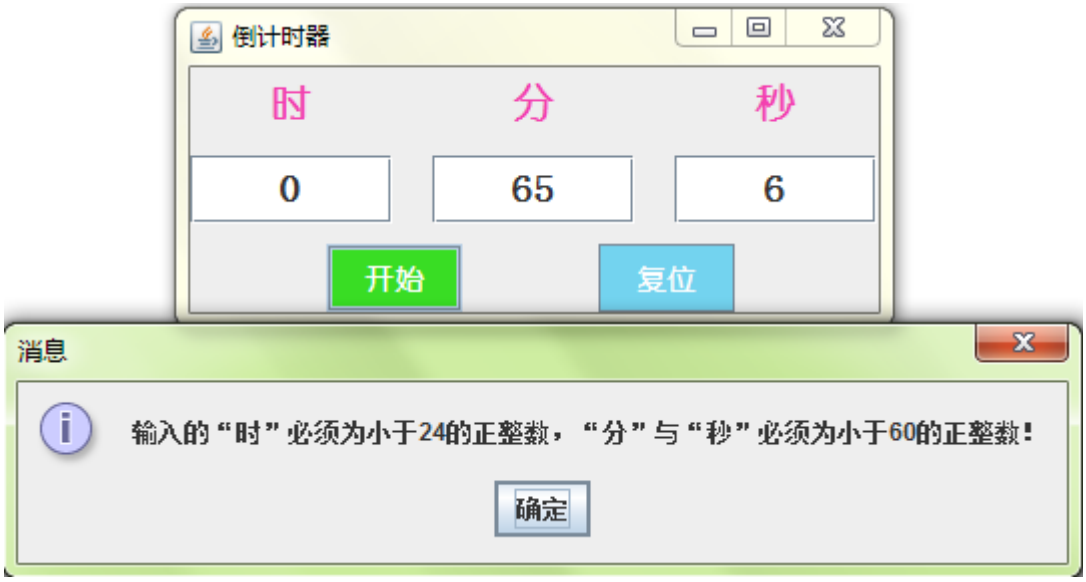


图 9　输入不在设定范围内的时间

图 10　输入时间为 0

（7）源代码及注释

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

publicclassCountdownTimerextends JFrame
implements ActionListener,Runnable
{
```

```
    private Font f1,f2;
    private Color c1,c2,c3;
    private JLabel label_hour,label_minute,label_second;
    private JTextField text_hour,text_minute,text_second;
    private JButton button_start,button_reset;
    private Thread thread=new Thread(this);//线程对象
    privateint hour,minute,second;          //设置的倒计时时间
    privateint num_control=0;                //控制“开始”按钮
     //与“暂停”按钮的相互转换

    public CountdownTimer()
    {
        super("倒计时器");                          //框架标题
        this.setSize(360,160);                  //框架大小，单位是像素
        this.setLayout(new GridLayout(3,1,20,10));
                                                //框架布局，3 行 1 列，
                                                //组件水平间距 20 像素，
                                                //垂直间距 10 像素
        this.setLocationRelativeTo(null);   //框架居中
        this.setDefaultCloseOperation(EXIT_ON_CLOSE); //关闭框架
        f1=new Font("幼圆",Font.BOLD,20);              //设置新字体
        f2=new Font("幼圆",Font.BOLD,14);
        c1=new Color(244,65,175);           //洋红色
        c2=new Color(57,221,36);            //浅绿色
        c3=new Color(115,211,239);          //浅蓝色

        //显示“时、分、秒”三个汉字的面板
        JPanel panel1=newJPanel(new GridLayout(1,3,20,10));
        label_hour=new JLabel("时",JLabel.CENTER);
label_hour.setFont(f1);                 //设置字体格式
        label_hour.setForeground(c1);       //设置字体颜色

        label_minute=new JLabel("分",JLabel.CENTER);
        label_minute.setFont(f1);
        label_minute.setForeground(c1);

        label_second=new JLabel("秒",JLabel.CENTER);
        label_second.setFont(f1);
        label_second.setForeground(c1);

        panel1.add(label_hour);
        panel1.add(label_minute);
        panel1.add(label_second);
```

```
//设置时间并显示剩余时间的面板
JPanel panel2=newJPanel(new GridLayout(1,3,20,10));
text_hour=newJTextField("0");
text_hour.setFont(f1);                    //设置字体格式
text_hour.setHorizontalAlignment(JTextField.CENTER);
                                          //设置文本对齐方式，居中

text_minute=newJTextField("0");
text_minute.setFont(f1);
text_minute.setHorizontalAlignment(JTextField.CENTER);

text_second=newJTextField("0");
text_second.setFont(f1);
text_second.setHorizontalAlignment(JTextField.CENTER);

panel2.add(text_hour);
panel2.add(text_minute);
panel2.add(text_second);

//按钮面板
JPanel panel3=newJPanel(new GridLayout(1,5));
                                          //面板布局，1 行 5 列
button_start=new JButton("开始");
button_start.setFont(f2);
button_start.setForeground(Color.white);
                                          //按钮字体颜色为白色
button_start.setBackground(c2);
button_start.setOpaque(true);
                //setOpaqueIture)方法的目的是让组件变成不透明，
//这样在 button_next 上所设置的颜色才能显示出来。
button_start.addActionListener(this);
                //为“开始”按钮注册动作事件监听器

button_reset=new JButton("复位");
button_reset.setFont(f2);
button_reset.setForeground(Color.white);

button_reset.setBackground(c3);
button_reset.setOpaque(true);
button_reset.addActionListener(this);

panel3.add(new JLabel(""));
panel3.add(button_start);
panel3.add(new JLabel(""));
```

```
        panel3.add(button_reset);
        panel3.add(new JLabel(""));

        this.add(panel1);
        this.add(panel2);
        this.add(panel3);

        this.setVisible(true);          //设置框架可视
}

//判断字符串是否为非负整数
publicstaticboolean isNonNagativeInteger(String value)
 {
      try
       {
            int temp=Integer.parseInt(value);
            if(temp>=0)
                  returntrue;
            else
                  returnfalse;
         }
      catch (NumberFormatException e)
       {
      returnfalse;
       }
   }

//实现线程体
publicvoid run()
{
      while(true)
            try
            {
                  //时间显示设定
                  if(second==-1)
                  {
                        second=59;
                        minute--;
                  }
                  if(minute==-1)
                  {
                        minute=59;
                        hour--;
                  }
```

```
                //文本行显示剩余时间

                text_hour.setText(String.valueOf(hour));
                text_minute.setText(String.valueOf(minute));
                text_second.setText(String.valueOf(second));
                Thread.sleep(1000);              //线程睡眠 1 秒
                second--;

                //倒计时完成时，倒计时复位，有提示音并弹出对话框
                if(hour==0&&minute==0&&second==-1)
                {
                    button_start.setText("开始");
                    hour=0;
                    minute=0;
                    second=0;
                    text_hour.setEditable(true);    //设置可编辑
                    text_minute.setEditable(true);
                    text_second.setEditable(true);
                    text_hour.setText(""+hour);
                    text_minute.setText(""+minute);
                    text_second.setText(""+second);
                    num_control=0;

                    java.awt.Toolkit.getDefaultToolkit().beep();
                    JOptionPane.showMessageDialog(null,
                                                  "倒计时结束！ ");
                    break;
                }
            }

        catch(InterruptedException ex)
            {
            break;
            }
}

//实现按钮的动作事件监听器接口
publicvoid actionPerformed(ActionEvent ev)
{
    //按下“复位”按钮时
    if(ev.getSource()==button_reset)
    {
        button_start.setText("开始");
```

```
        hour=0;
        minute=0;
        second=0;
        text_hour.setEditable(true);          //设置可编辑
        text_minute.setEditable(true);
        text_second.setEditable(true);
        text_hour.setText(""+hour);
        text_minute.setText(""+minute);
        text_second.setText(""+second);
        num_control=0;
        thread.interrupt();
}

//按下“开始”或“暂停”按钮时
elseif(ev.getSource()==button_start)
{
        String str_hour,str_minute,str_second;
        str_hour=text_hour.getText();
        str_minute=text_minute.getText();
        str_second=text_second.getText();

//只有输入非负整数时间时，才执行相关步骤，否则弹出提示对话框
if(isNonNagativeInteger(str_hour)
&&isNonNagativeInteger(str_minute)
&&isNonNagativeInteger(str_second))
        {
        hour=Integer.parseInt(str_hour);
        minute=Integer.parseInt(str_minute);
        second=Integer.parseInt(str_second);

        if(hour==0&&minute==0&&second==0)
                JOptionPane.showMessageDialog(this,
                                "输入时间不能为 0！");

                //输入的“时”必须为小于 24 的正整数，
                //“分”与“秒”必须为小于 60 的正整数！
        elseif(hour<24&&minute<60&&second<60)
                {
                if(num_control==0)
                {
                        text_hour.setEditable(false);
                                        //设置不可编辑，只用于显示
                                text_minute.setEditable(false);
                                text_second.setEditable(false);
```

```
                    button_start.setText("暂停");
                thread=new Thread(this);
                thread.start();
                    num_control=1;
                }
                elseif(num_control==1)
                {
                    button_start.setText("开始");
                    num_control=0;
                    thread.interrupt();
                }

                }
            else
                JOptionPane.showMessageDialog(this,
                                "输入的" +
                                ""时"必须为小于 24 的正整数，" +
                                ""分"与"秒"必须为小于 60 的正整数！");
            }

        else
            JOptionPane.showMessageDialog(this,
                    "请输入正整数时间！");
        }
    }

    publicstaticvoid main(String arg[])
    {
        new CountdownTimer();     //新建框架对象
    }
}
```

（8）设计总结

倒计时器设计的重点与难点在于对线程的操作。需要通过线程语句 sleep() 来控制线程的暂停，即每过一秒，进行一次输出显示，从而达到倒计时的效果。

2. 算术运算测试程序

设计流程为：了解“四则运算出题器”功能→进行窗口布局→美化界面→实现窗口中各个组件的预设功能→调试→完成选题。现在就设计结果说明如下。

（1）程序初始运行界面图 11。其中标题为“四则运算出题器”，界面采用网格布局。第一、二行是提示标签，说明出题器的基本功能；第三行用于选择需要做题的数量；第四行显示当前剩余的题数；第五行是出题部位，需要输入答案；第六行的“下一题”按钮用于继续答题。

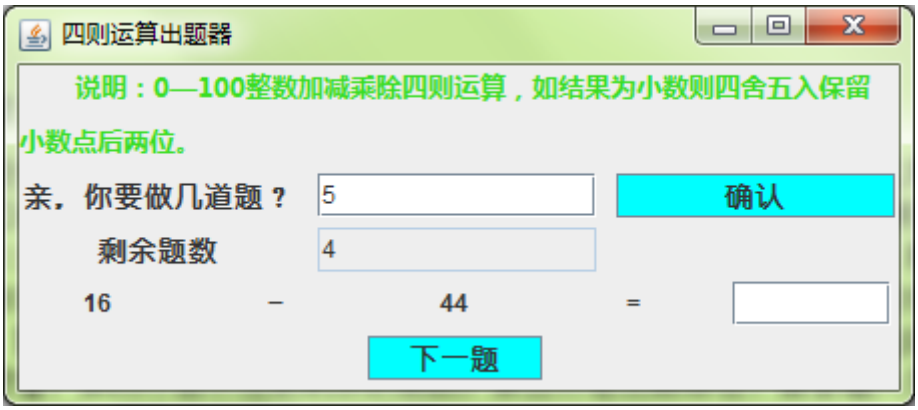


图 11 初始界面

（2）输入需要做的题数，如 6 道，点击“确定”按钮。

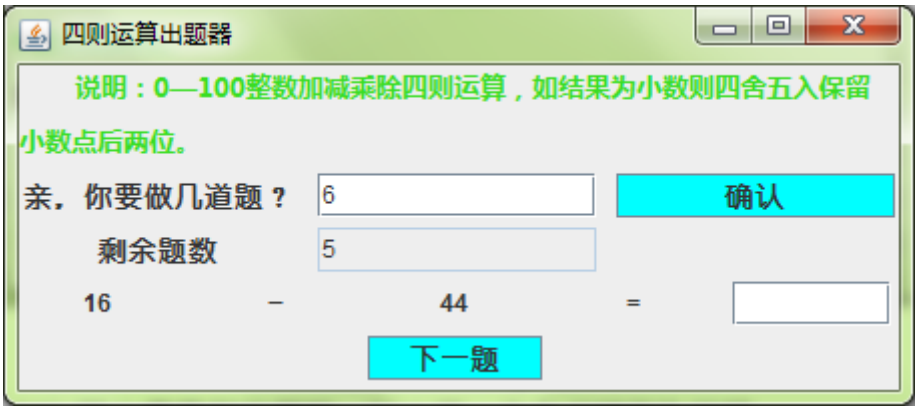
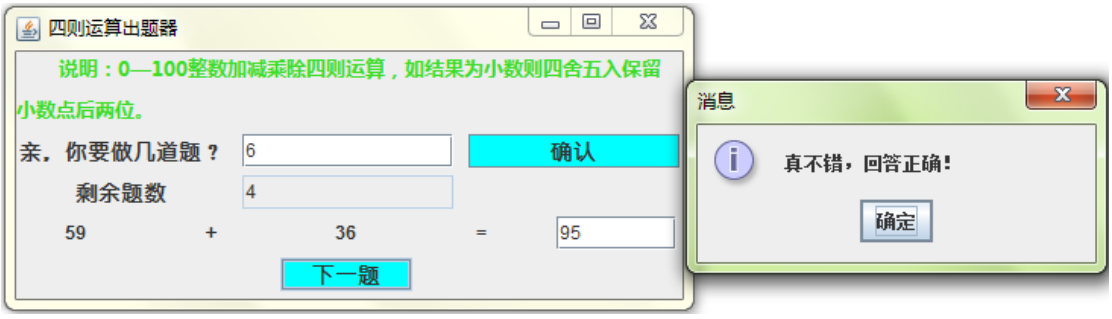


图 12 输入题数并确定

（3）开始答题。输入答案后点击“下一题”继续答题，直到所有题都答完。



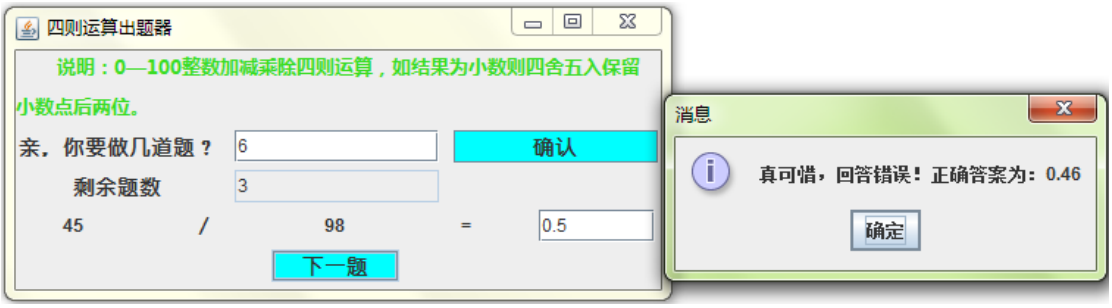
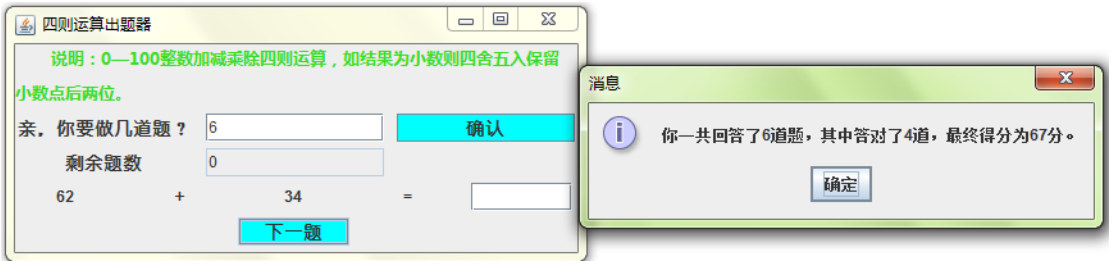


图 12　开始答题



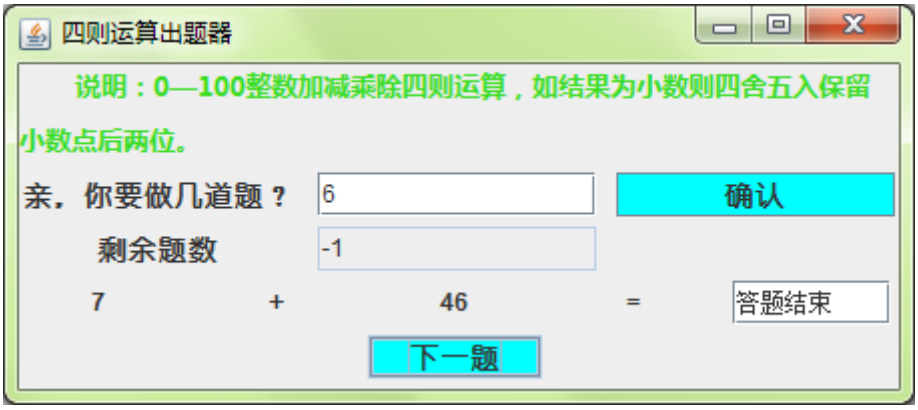
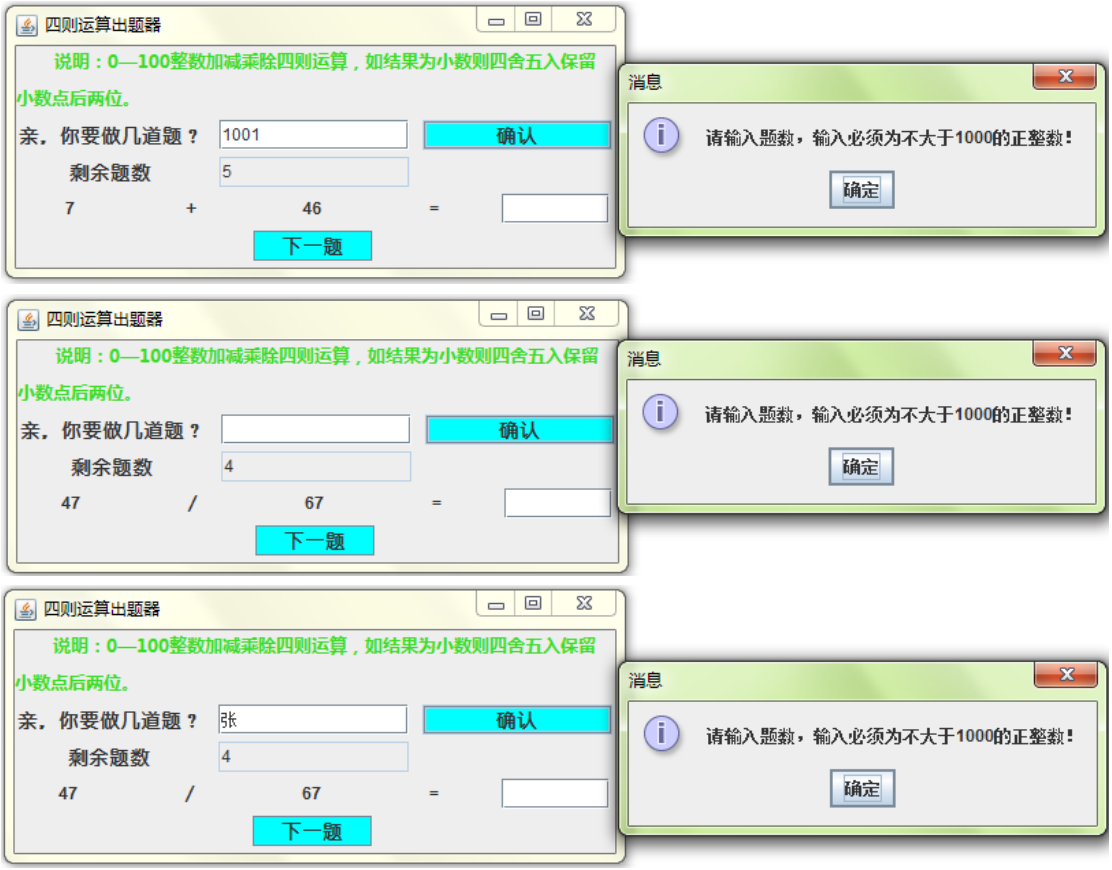


图 13　答题结束

(4)异常处理



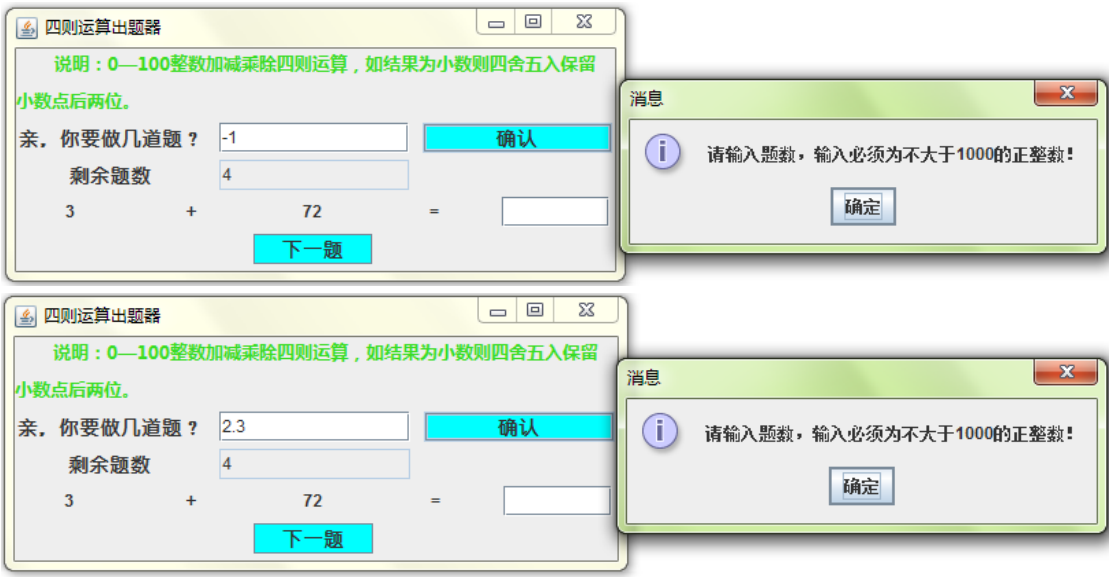


图 14　输入题数不符合要求

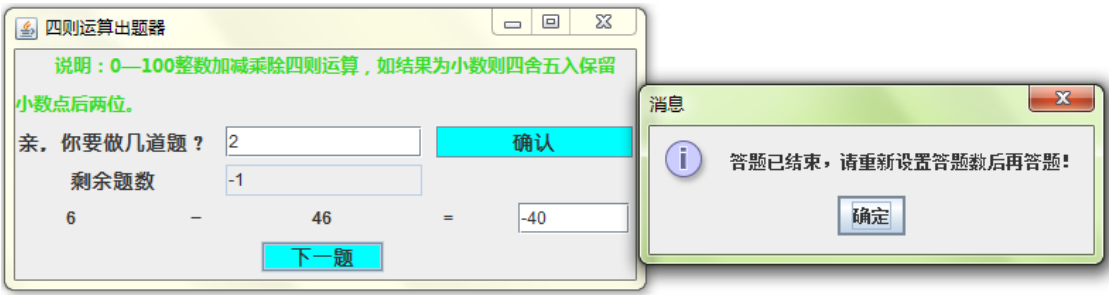


图 15　答题结束后仍试图答题

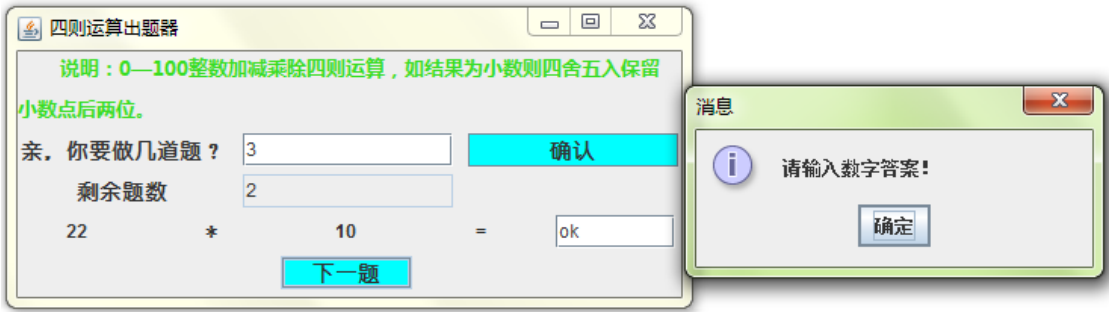


图 16　输入非数字答案

（5）源代码及注释

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
publicclassSizeyunsuanextends JFrame implements ActionListener
{
    private Font f1,f2;
```

```
private Color c1;
private JLabel label_tishi1,label_tishi2;
private JLabel label_choose;
private JTextField text_choose;              //选择做题的数量
private JLabel label_rest;
private JTextField text_rest;                //剩余题数
private JLabel label_num1,label_operator,label_num2;
private JTextField text_result;
private JButton button_confirm,button_next;
privateint r;                                //用于随机产生运算符
privateint num_choose;                       //用于记录想要做的题目数
privateint num_rest;                         //用于记录剩余题数
privateint count=0;                          //统计答对的题数

public Sizeyunsuan()                         //框架的构造方法
{
    super("四则运算出题器");                    //框架标题
    this.setSize(455,200);                   //框架大小，单位是像素
    this.setLayout(new GridLayout(6,1,10,5));
                                 //框架布局，6 行 1 列，
                                 //组件水平间距 10 像素，垂直间距 5 像素
    this.setLocationRelativeTo(null);  //框架居中
    this.setDefaultCloseOperation(EXIT_ON_CLOSE); //关闭框架
    f1=new Font("微软雅黑",Font.BOLD,13);  //设置新字体
    f2=new Font("幼圆",Font.BOLD,14);
    c1=new Color(57,221,36);                 //设置新颜色

    label_tishi1=new JLabel("          说明：0—100 整数加减乘除" +
                            "四则运算，如结果为小数则四舍五入保留");
    this.add(label_tishi1);        //框架第 1 行，显示相关提示消息
    label_tishi1.setFont(f1);                //设置字体格式
    label_tishi1.setForeground(c1);        //设置字体颜色
    label_tishi2=new JLabel("小数点后两位。");
    this.add(label_tishi2);        //框架第 2 行，显示相关提示消息
    label_tishi2.setFont(f1);                //设置字体格式
    label_tishi2.setForeground(c1);        //设置字体颜色

    //选择出题数的面板
    JPanel panel1=newJPanel(new GridLayout(1,3,10,5));
    label_choose=new JLabel("亲，你要做几道题？",JLabel.CENTER);
    panel1.add(label_choose);
    label_choose.setFont(f2);
    text_choose=newJTextField("5");
    panel1.add(text_choose);
```

```
button_confirm=new JButton("确认");
button_confirm.setFont(f2);
button_confirm.setBackground(Color.cyan);
button_confirm.setOpaque(true);
        //setOpaqueIture)方法的目的是让组件变成不透明，
        //这样我们在 button_confirm 上所设置的颜色才能显示出来。
button_confirm.addActionListener(this);
                //为“确认”按钮注册动作事件监听器
panel1.add(button_confirm);
num_choose=Integer.parseInt(text_choose.getText());

//显示剩余题数的面板
JPanel panel2=newJPanel(new GridLayout(1,3,10,5));
label_rest=new JLabel("剩余题数",JLabel.CENTER);
panel2.add(label_rest);
label_rest.setFont(f2);
text_rest=newJTextField();
panel2.add(text_rest);
panel2.add(new JLabel(""));
num_rest=num_choose-1;
text_rest.setText(""+num_rest);   //显示剩余题数
text_rest.setEditable(false);      //设置不可编辑，只用于显示

//出题并输入答案的面板
JPanel panel3=newJPanel(new GridLayout(1,5,10,5));
label_num1=newJLabel(String.valueOf(
                        (int)(Math.random()*100)),
                          JLabel.CENTER);
                              //随机产生一个 0-100 的整数，居中
panel3.add(label_num1);
char[]ch={'+','-','*','/'};   //字符数组中存有
                                          //“+、-、*、/”四个运算符
r=(int)(Math.random()*4);     //随机产生一个 0-4 的整数
label_operator=newJLabel(String.valueOf(ch[r]),
                                  JLabel.CENTER);
                        //随机产生“+、-、*、/”四个运算符，居中
label_operator.setFont(f2);     //设置运算符字体，便于看清
panel3.add(label_operator);
label_num2=newJLabel(String.valueOf(
                        (int)(Math.random()*100)+1),
                            JLabel.CENTER);
                    //随机产生一个 1-100 的整数，居中
panel3.add(label_num2);
panel3.add(new JLabel("=",JLabel.CENTER));
```

```
        text_result=new JTextField();     //输入答案的文本行
        panel3.add(text_result);

        //下一题按钮及提交按钮
        JPanel panel4=newJPanel(new GridLayout(1,5));
        panel4.add(new JLabel(""));
        panel4.add(new JLabel(""));
        button_next=new JButton("下一题");
        button_next.setFont(f2);
        button_next.setBackground(Color.cyan);
        button_next.setOpaque(true);
        button_next.addActionListener(this);
                                  //为“下一题”按钮注册动作事件监听器
        panel4.add(button_next);
        panel4.add(new JLabel(""));
        panel4.add(new JLabel(""));

        this.add(panel1);
        this.add(panel2);
        this.add(panel3);
        this.add(panel4);
        this.setVisible(true);    //设置框架可视
}

//判断字符串是否为整数
publicstaticboolean isInteger(String value)
 {
      try
       {
            Integer.parseInt(value);
      returntrue;
         }
      catch (NumberFormatException e)
       {
      returnfalse;
       }
   }

//判断字符串是否为浮点数
publicstaticboolean isDouble(String value)
 {
      try
       {
            Double.parseDouble(value);
```

```
            if (value.contains("."))
                returntrue;
            else
                returnfalse;
              }
            catch (NumberFormatException e)
             {
            returnfalse;
             }
          }

        //判断字符串是否为数字
        publicstaticboolean isNumber(String value)
         {
            returnisInteger(value) || isDouble(value);
         }

        //判断字符串是否为非负整数
            publicstaticboolean isNonNagativeInteger(String value)
             {
                try
                 {
                    int temp=Integer.parseInt(value);
                    if(temp>=0)
                        returntrue;
                    else
                        returnfalse;
                  }
                catch (NumberFormatException e)
                 {
                returnfalse;
                 }
              }

        //实现按钮的动作事件监听器接口
        publicvoid actionPerformed(ActionEvent ev)
        {
    if(ev.getSource()==button_confirm) //点击“确认”按钮时
            {
        String str_choose=text_choose.getText();
                                        //获得选择区输入的字符串

        //只有输入的答题数是不大于 1000 的非负整数时，
        //才执行相关步骤，否则弹出提示对话框
```

```
        if(isNonNagativeInteger(str_choose)
            &&Integer.parseInt(str_choose)<=1000)
        {
        num_choose=Integer.parseInt(str_choose);
        num_rest=num_choose-1;
            text_rest.setText(""+num_rest); //显示剩余题数
            text_result.setText("");            //答题区清空
            count=0;                                //答对题数置 0
        }
        else
            JOptionPane.showMessageDialog(this, "请输入题数，" +
                    "输入必须为不大于 1000 的正整数！");

            }

elseif(ev.getSource()==button_next) //点击“下一题”按钮
            {
        String str_result=text_result.getText();
                                    //获得答案输入区输入的字符串

//只有输入数字打答案时，才执行相关步骤，否则弹出提示对话框
        if(isNumber(str_result))
        {
        double temple=Double.parseDouble(str_result);
double num_result=Math.round(temple*100)/100.0;
//答题者输入的答案，保留小数点后两位
int num1,num2;
                if(num_rest>=0)
                {
                    num1=Integer.parseInt(label_num1.getText());
                    num2=Integer.parseInt(label_num2.getText());
                    switch(r)
                    {
                    case 0:if(num_result==(num1+num2))
                                {
                        JOptionPane.showMessageDialog(this,
                                                "真不错，回答正确！");
                                 text_result.setText("");  //清空答题区
                    count++;
                                }
                    else
                            {
                        JOptionPane.showMessageDialog(this,
                                "真可惜，回答错误！正确答案为："
```

```
                        +(num1+num2));
                                                    //保留小数点后两位
                            text_result.setText("");//清空答题区
                        }   break;
                    case 1:if(num_result==(num1-num2))
                                {
                JOptionPane.showMessageDialog(this,
                                        "真不错，回答正确！");
                                 text_result.setText("");
            count++;
                                }
            else
                                {
                JOptionPane.showMessageDialog(this,
                            "真可惜，回答错误！正确答案为："
                +(num1-num2));
                                text_result.setText("");
                }   break;
                    case 2:if(num_result==(num1*num2))
                                {
        JOptionPane.showMessageDialog(this,
                                "真不错，回答正确！");
                                    text_result.setText("");
count++;
                                }
else
                                {
        JOptionPane.showMessageDialog(this,
                    "真可惜，回答错误！正确答案为："
        +(num1*num2));
                                text_result.setText("");
        }   break;
                    case 3:if(num_result==
                            Math.round(((double)num1/
                                    (double)num2)*100)/100.0)
                                {
        JOptionPane.showMessageDialog(this,
                                "真不错，回答正确！");
                                    text_result.setText("");
count++;
                                }
else
                                {
        JOptionPane.showMessageDialog(this,
```

```
                "真可惜，回答错误！正确答案为："
                        +Math.round(((double)num1/
                (double)num2)*100)/100.0);
                                text_result.setText("");
        }   break;
                }

                if(num_rest==0)   //如果所答之题是最后一题
                {
                    JOptionPane.showMessageDialog(this,
                "你一共回答了"+num_choose+"道题，"+"其中答对了"
                        +count+"道，"+"最终得分为"
                        +Math.round(((double)count/
                            (double)num_choose)*100)
                        +"分。");
                    text_result.setText("答题结束");
                }

                num_rest--;
                text_rest.setText(""+num_rest);

                num1=(int)(Math.random()*100); //随机产生一个 0-100
                                                //的整数
                char[]ch={'+','-','*','/'};      //字符数组中存有四则
                                                //运算的四个运算符
                r=(int)(Math.random()*4);         //随机产生一个 0-4 的
                                                  //整数
                num2=(int)(Math.random()*100); //随机产生一个 0-100
                                                //的整数
                label_num1.setText(String.valueOf(num1));
                label_operator.setText(String.valueOf(ch[r]));
                            //随机产生“+、-、*、/”其中的一个运算符
                label_num2.setText(String.valueOf(num2));
            }
            else
                JOptionPane.showMessageDialog(this,
                                        "答题已结束，" +
                                       "请重新设置答题数后再答题！");
        }
else                    //如果答案输入区输入的字符串不是数字
    JOptionPane.showMessageDialog(this, "请输入数字答案！");
        }
    }
```

```
    publicstaticvoid main(String arg[])
    {
        new Sizeyunsuan();      //新建框架对象
    }
}
```

（6）设计总结

四则运算出题器设计的难点在于随机产生一个运算符显示出来，并且要求程序内部能够知道产生的是哪个运算符。程序中用一个长度为 4 的一维数组 ch[]来存放四个运算符，然后随机产生一个位于 1-4 之间的整数 r，从而显示的 ch[r]就是一个随机的运算符；程序内部计算则根据 r 的值来确定具体是哪个运算符。

3. 健康计算器

设计流程为：了解“健康计算器”功能→进行窗口布局→美化界面→实现窗口中各个组件的预设功能→调试→完成选题。现在就设计结果说明如下。

（1）程序初始运行界面图 17。其中标题为“四则运算出题器”，界面采用分割窗格。分割窗格上下排列，上面为“基本信息面板”，用于输入参数，初始界面中已经设置了初始参数，用户可以重新设置；下面为“计算”面板，用于计算两个指数，显示指数的文本区设为不可编辑，即只能用于显示。

图 17　初始界面

（2）求 BMI 指数（有对话框提示）


健康计算器
基本信息
体重 71 kg
身高 1.76 m
腰围 90 cm
性别 男 女
计算
BMI指数 22.921 kg/m2
体脂率 %
求BMI指数
求体脂率
消息
体重正常，继续保持哈^_^
确定


健康计算器
基本信息
体重 45 kg
身高 1.65 m
腰围 60 cm
性别 男 女
计算
BMI指数 16.529 kg/m2
体脂率 %
求BMI指数
求体脂率
消息
体重过轻，记得不要太瘦哦！
确定


健康计算器
基本信息
体重 90 kg
身高 1.80 m
腰围 100 cm
性别 男 女
计算
BMI指数 27.778 kg/m2
体脂率 %
求BMI指数
求体脂率
消息
你超重啦，注意要减肥了！
确定

图 18　四个不同范围的 BMI 指数对应四个不同的提示

（3）求体脂率

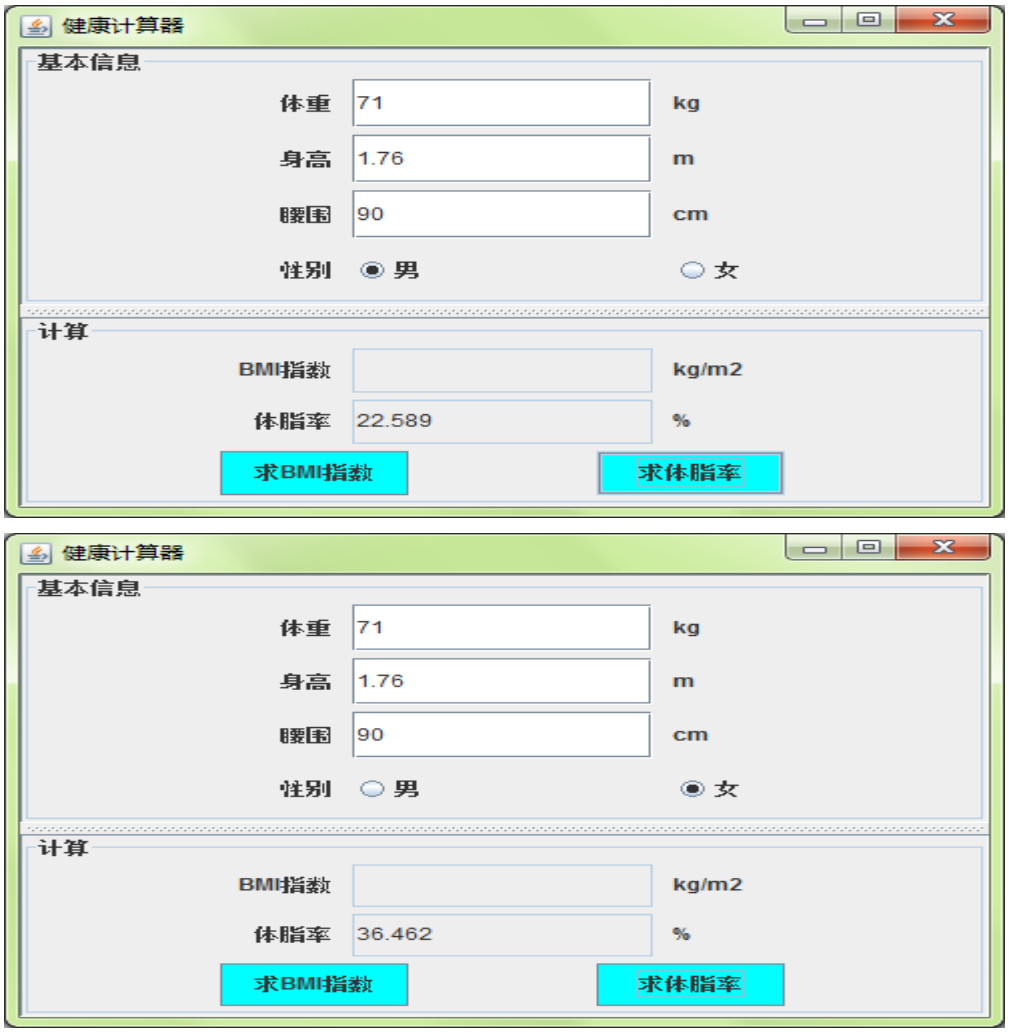


图 19　男、女的体脂率公式不同

（4）异常处理

图 20　输入非数字字符串或不输入

（5）源代码（包含三个类）及注释

①第一个类（基本信息面板）

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.TitledBorder;
publicclassPersonJPanelextends JPanel    //基本信息面板
{
    static JTextField text_weight,text_height,text_yao;
    static JRadioButton rbm,rbw;     //单选按钮
    public PersonJPanel()
    {
```

```
        this.setBorder(new TitledBorder("基本信息"));
                                        //设置面板带有含标题的边框线
        this.setLayout(new GridLayout(4,3,10,5));
                                    //面板布局，4 行 3 列，
                                    //组件水平间距 10 像素，垂直间距 5 像素

        //与体重相关的组件添加到面板中，第 1 行
        this.add(new JLabel("体重",JLabel.RIGHT));
        text_weight=newJTextField("71");
        this.add(text_weight);
        this.add(new JLabel("kg"));

        //与身高相关的组件添加到面板中，第 2 行
        this.add(new JLabel("身高",JLabel.RIGHT));
        text_height=newJTextField("1.76");
        this.add(text_height);
        this.add(new JLabel("m"));

        //与腰围相关的组件添加到面板中，第 3 行
        this.add(new JLabel("腰围",JLabel.RIGHT));
        text_yao=newJTextField("90");
        this.add(text_yao);
        this.add(new JLabel("cm"));

        //与性别相关的组件添加到面板中，第 4 行
        this.add(new JLabel("性别",JLabel.RIGHT));
        rbm=new JRadioButton("男");
        rbw=new JRadioButton("女");
        rbm.setSelected(true);    //设置默认“男”按钮选中
        ButtonGroup bg=newButtonGroup();
        bg.add(rbm);
        bg.add(rbw);      //rbm 与 rbw 包含到一个 ButtonGroup 按钮组中，
                          //实现 rbm 与 rbw 互斥，即不能同时选中
    this.add(rbm);
    this.add(rbw);

    }
}
```

②第二个类（计算面板）

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.TitledBorder;
```

```
publicclassCalculateextends JPanel implements ActionListener
{
    private JTextField text_BMI,text_tizhi;
    private JButton button_BMI,button_tizhi;
    public Calculate()          //计算功能面板
    {
        this.setBorder(new TitledBorder("计算"));
                                //设置面板带有标题的边框线
        this.setLayout(new GridLayout(3,1,10,5));
                                //面板布局，3 行 1 列，
                                //组件水平间距 10 像素，垂直间距 5 像素

        //BMI 指数计算结果显示面板
        JPanel panel1=newJPanel(new GridLayout(1,3,10,5));
        panel1.add(new JLabel("BMI 指数",JLabel.RIGHT));
        text_BMI=newJTextField();
        text_BMI.setEditable(false);  //设置不可编辑，只用于显示
        panel1.add(text_BMI);
        panel1.add(new JLabel("kg/m2"));

        //体脂率计算结果显示面板
        JPanel panel2=newJPanel(new GridLayout(1,3,10,5));
        panel2.add(new JLabel("体脂率",JLabel.RIGHT));
        text_tizhi=newJTextField();
        text_tizhi.setEditable(false);
        panel2.add(text_tizhi);
        panel2.add(new JLabel("%"));

        //按钮子面板
        JPanel panel3=newJPanel(new GridLayout(1,5));
                                //子面板布局，1 行 5 列，
                                //两个按钮放在第 2 列及第 4 列，显得布局合理
        panel3.add(new JLabel(""));
        button_BMI=new JButton("求 BMI 指数");
        panel3.add(button_BMI);
        button_BMI.setBackground(Color.cyan);
        button_BMI.setOpaque(true);
                    //setOpaqueIture)方法的目的是让组件变成不透明，
                    //这样在 button_BMI 上所设置的颜色才能显示出来。
        panel3.add(new JLabel(""));
        button_tizhi=new JButton("求体脂率");
        button_tizhi.setBackground(Color.cyan);
        button_tizhi.setOpaque(true);
        panel3.add(button_tizhi);
```

```
        panel3.add(new JLabel(""));
        button_BMI.addActionListener(this);
                    //为“求 BMI 指数”按钮注册动作事件监听器
        button_tizhi.addActionListener(this);
                    //为“求体脂率”按钮注册动作事件监听器

        //主面板添加三个子面板
        this.add(panel1);
        this.add(panel2);
        this.add(panel3);
    }

    //判断字符串是否为整数
publicstaticboolean isInteger(String value)
         {
            try
             {
                Integer.parseInt(value);
            returntrue;
               }
            catch (NumberFormatException e)
             {
            returnfalse;
             }
          }

        //判断字符串是否为浮点数
        publicstaticboolean isDouble(String value)
         {
            try
             {
                Double.parseDouble(value);
            if (value.contains("."))
                returntrue;
            else
                returnfalse;
              }
            catch (NumberFormatException e)
             {
            returnfalse;
             }
          }

        //判断字符串是否为数字
```

```
    publicstaticboolean isNumber(String value)
     {
         returnisInteger(value) || isDouble(value);
     }

//实现按钮的动作事件监听器接口
publicvoid actionPerformed(ActionEvent ev)
{
    String str_weight,str_height,str_yao;
    str_weight=PersonJPanel.text_weight.getText();
    str_height=PersonJPanel.text_height.getText();
    str_yao=PersonJPanel.text_yao.getText();

    //只有在输入数字数据时，才执行相关语句，否则弹出提示对话框
    if(isNumber(str_weight)&&isNumber(str_height)&&isNumber(str_yao))
    {
    double w=Double.parseDouble(str_weight);
    double h=Double.parseDouble(str_height);
    double y=Double.parseDouble(str_yao);
    if(ev.getSource()==button_BMI)              //如果点击按钮“求 BMI 指数”
    {
        double bmi=w/(h*h);
        text_BMI.setText(""+Math.round(bmi*1000)/1000.0);
                                                  //计算结果保留三位小数

        //弹出标准消息对话框，进行健康提示
        if(bmi<=18.5)
            JOptionPane.showMessageDialog(null,
                                    "体重过轻，记得不要太瘦哦！");
        elseif(bmi>18.5&&bmi<=23.9)
            JOptionPane.showMessageDialog(null, "体重正常，继续保持哈^_^");
        elseif(bmi>23.9&&bmi<=27.9)
            JOptionPane.showMessageDialog(null, "你超重啦，注意要减肥了！");
        elseif(bmi>27.9)
            JOptionPane.showMessageDialog(null, "你已经患上肥胖症！");
    }
    elseif(ev.getSource()==button_tizhi) //如果点击按钮“求体脂率”
    {
        if(PersonJPanel.rbm.isSelected()) //如果性别单选按钮选中的是“男”
        {
            double a,b,c,tizhi;
            a=y*0.74;
            b=w*0.082+44.74;
            c=a-b;
```

```
                tizhi=(c/w)*100;                //计算结果乘以 100 作为百分数的分子
                text_tizhi.setText(""+Math.round(tizhi*1000)/1000.0);
                                                    //百分数分子保留三位小数
            }
            elseif(PersonJPanel.rbw.isSelected()) //如果性别单选按钮选中的是“女”
            {
                double a,b,c,tizhi;
                a=y*0.74;
                b=w*0.082+34.89;
                c=a-b;
                tizhi=(c/w)*100;                //计算结果乘以 100 作为百分数的分子
                text_tizhi.setText(""+Math.round(tizhi*1000)/1000.0);
                                                    //百分数分子保留三位小数
            }
        }
    }
        else
            JOptionPane.showMessageDialog(null, "请输入数据，数据是数字！");
    }
}
```

③第三个类（分割窗格）

```
import javax.swing.*;
publicclassHealthJFrameextends JFrame
{
    protected PersonJPanel person=new PersonJPanel();
    protected Calculate calculate=new Calculate();
    public HealthJFrame()                       //顶层框架的构造方法
    {
        super("健康计算器");                        //框架标题
        this.setSize(500,360);                  //框架大小，单位是像素
        this.setLocationRelativeTo(null);   //窗口居中
        this.setBackground(java.awt.Color.lightGray);   //框架背景色
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);   //关闭框架

        JSplitPane split=new
        JSplitPane(JSplitPane.VERTICAL_SPLIT,true,person,calculate);
                                                    //垂直分割窗格为上下两部分，
                                                    //上部为 PersonJPanel()的一个对象，
                                                    //下部为 Calculate()的一个对象，
         //窗格中组件随分割线移动而改变大小
        split.setDividerLocation(180);      //分割线位置
        this.getContentPane().add(split);   //框架添加分割窗格
        this.setVisible(true);                  //设置框架可视
```

```
    }

    publicstaticvoid main(String arg[])
    {
        new HealthJFrame();                    //新建框架对象
    }
}
```

（6）设计总结

本题较为简单，在得到 BMI 指数后，判断一下范围，弹出一个对话框进行健康提示。

4. flash 动画

以自己为原型，设计了一套三级跳的动作。描述了自己经过数年的努力训练，终于在运动会上取得了冠军的故事。

5. 矩阵计算器

设计流程为：了解“矩阵计算器”功能→进行窗口布局→美化界面→实现窗口中各个组件的预设功能→调试→完成选题。现在就设计结果说明如下。

（1）程序初始运行界面图 21。其中标题为“矩阵计算器”，界面采用分割窗格。分割窗格上下排列，上面为“矩阵 a”面板，用于输入矩阵 a；下面为“矩阵 b”面板，10*10 的文本区用于输入矩阵 b，最下方的 6 个按钮分别为“a+b”，“a*b”，“b 的转置”，“b 的秩”，“清空 a”，“清空 b”。

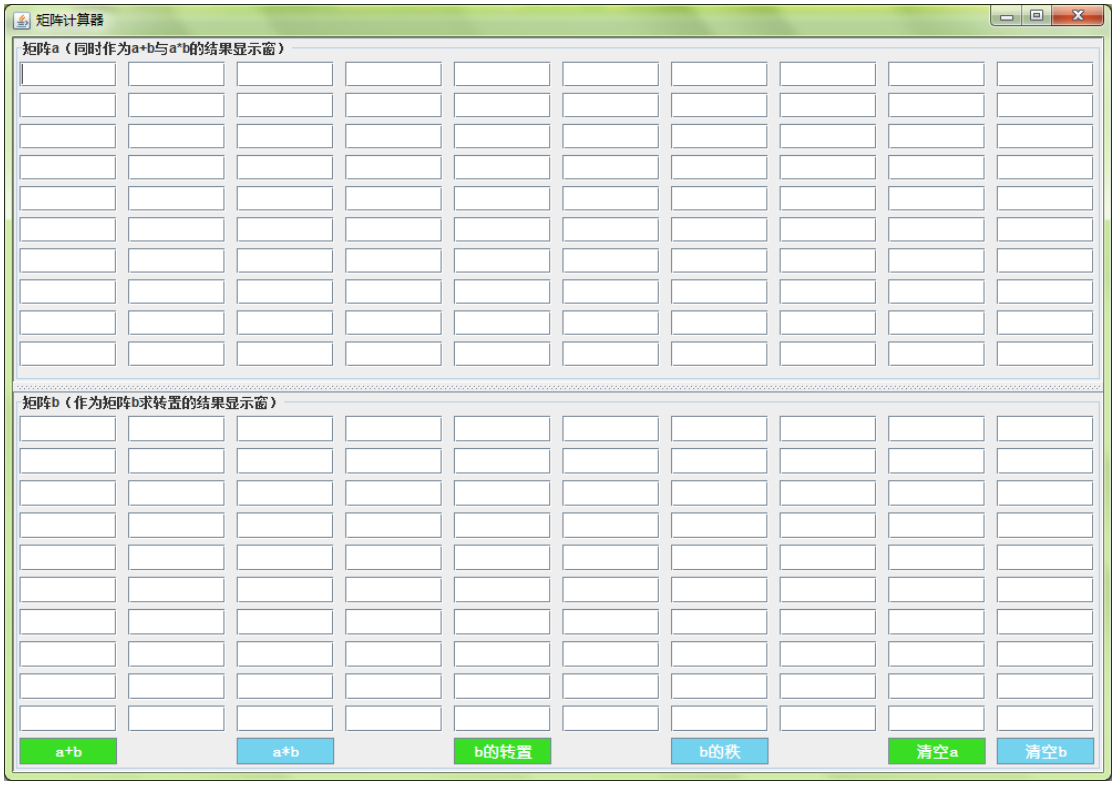
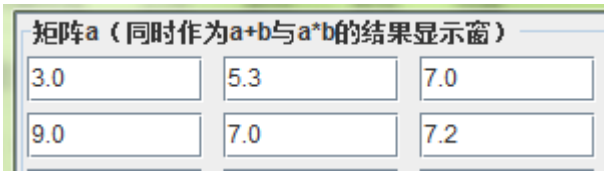


图 21　初始界面

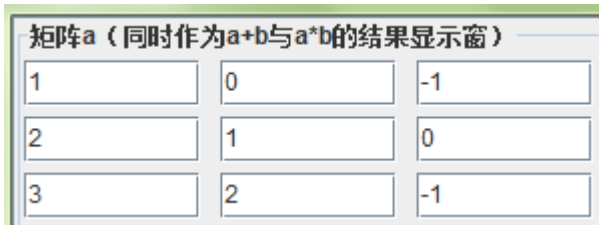
(2) 加法

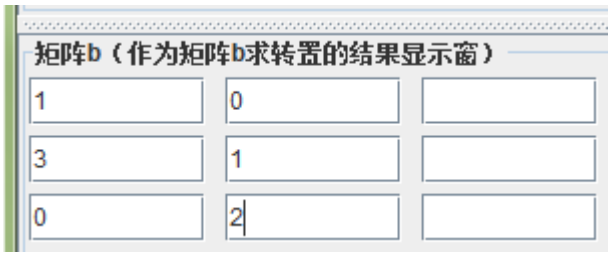


加法结果

图 22　矩阵加法运算

（3）乘法





结果为

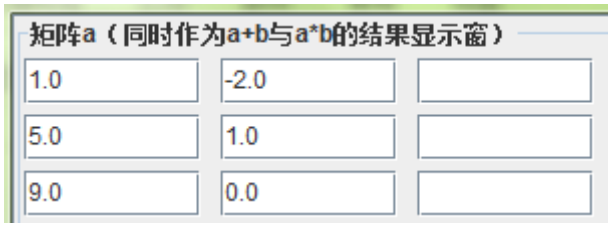


图 23　矩阵乘法运算

（4）求矩阵 b 的转置

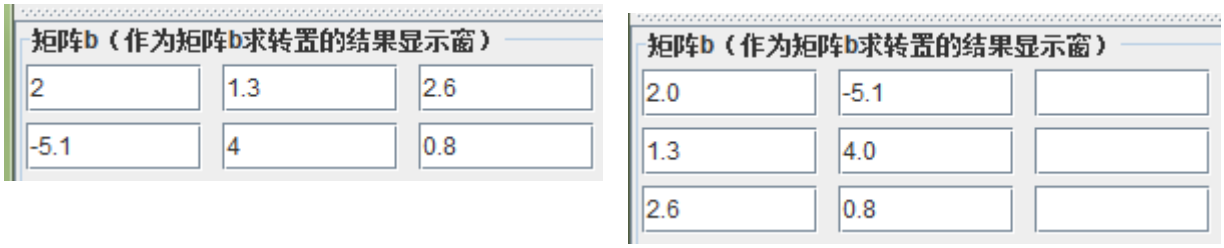


图 24　矩阵转置运算

（5）求矩阵 b 的秩

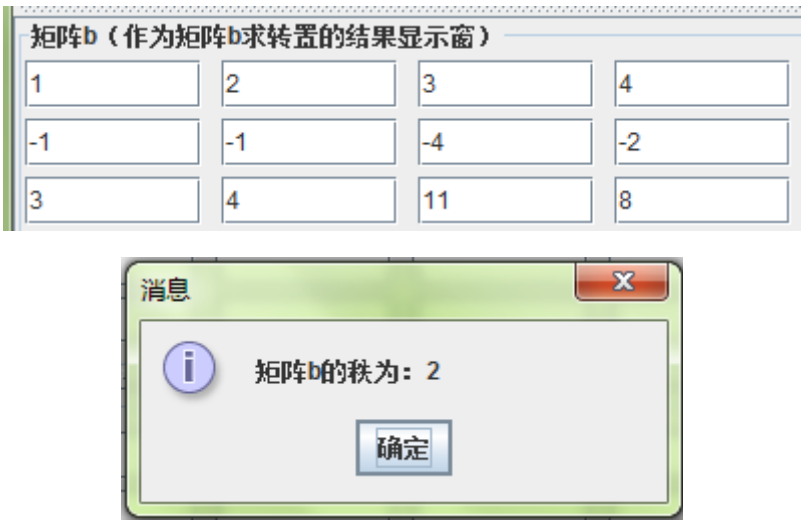
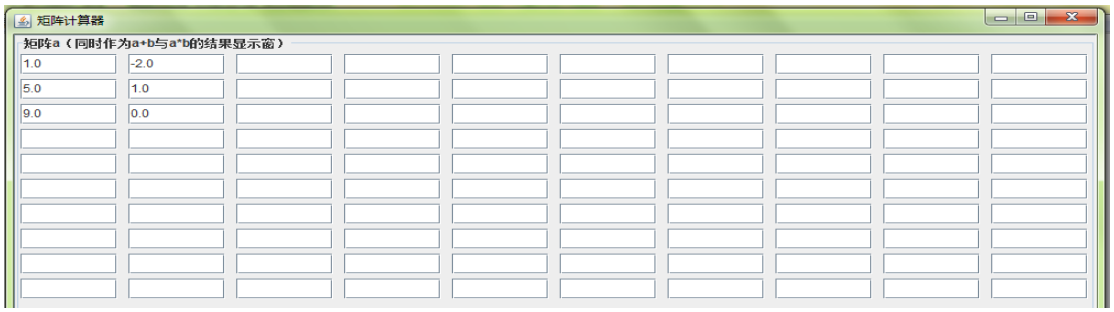


图 25　矩阵求秩运算

（6）清空 a



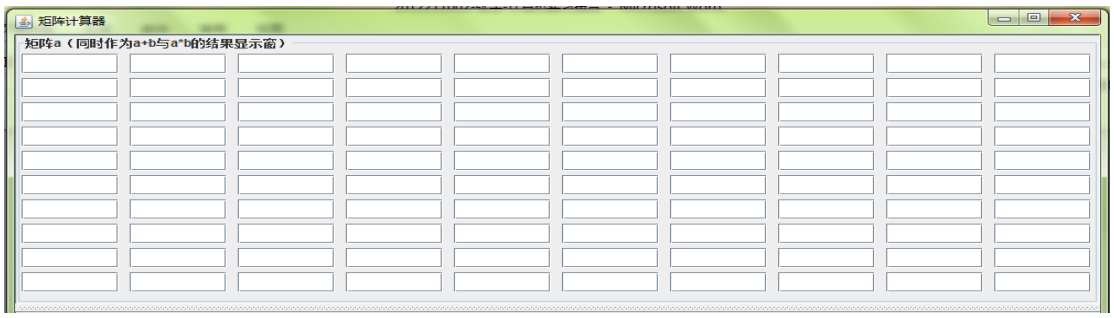


图 26　清空矩阵 a

(7)清空 b

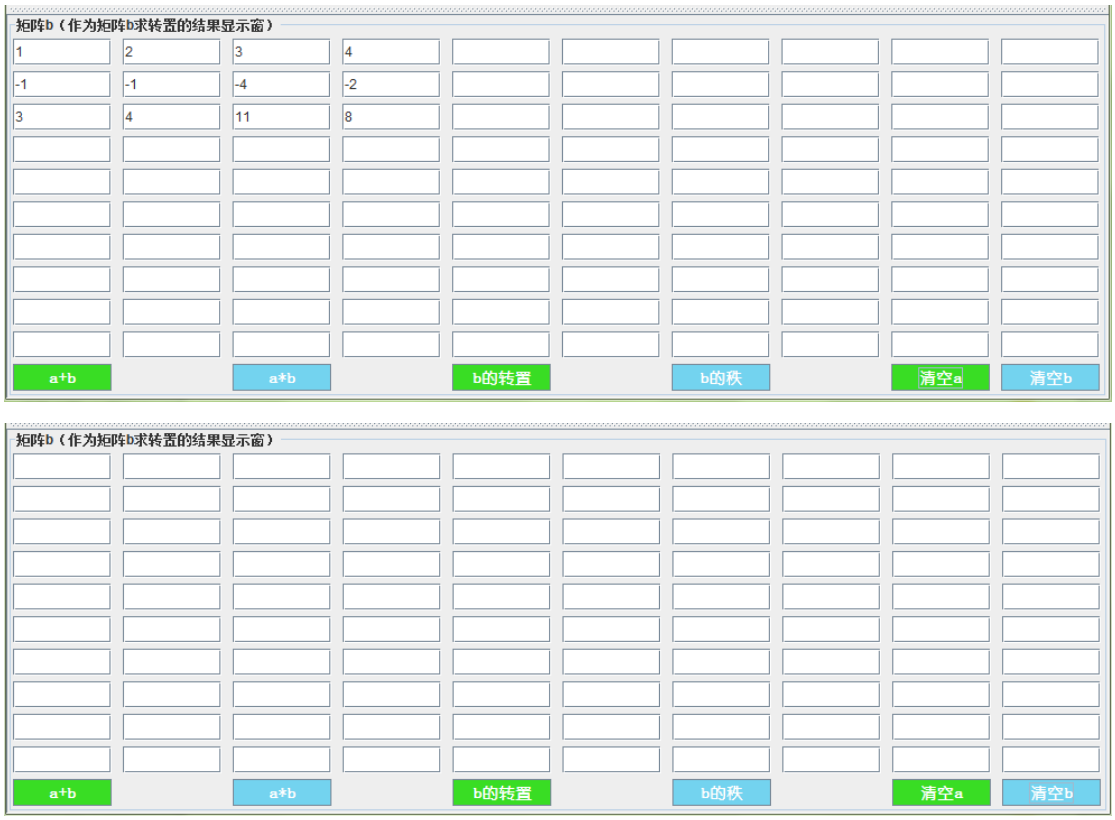
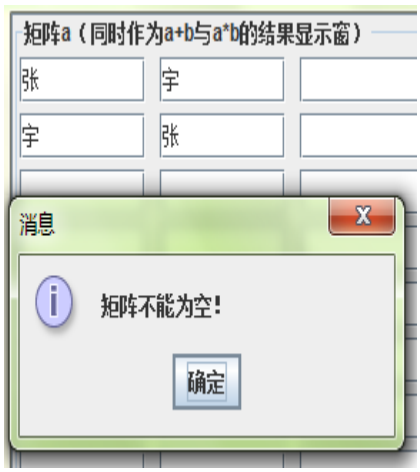


图 27　清空矩阵 b

(8)异常处理



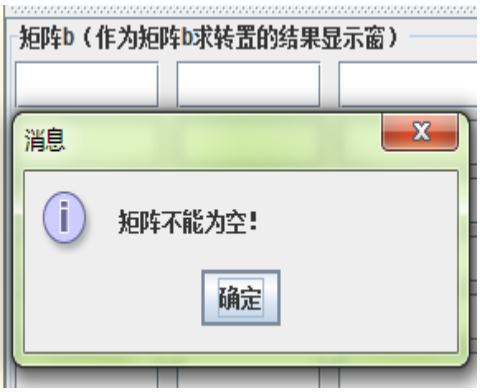


图 28　未输入数据就点击前 4 个按钮中的一个

图 29　输入的不是矩阵或矩阵不能相加

图 30　输入的矩阵不能进行乘法运算

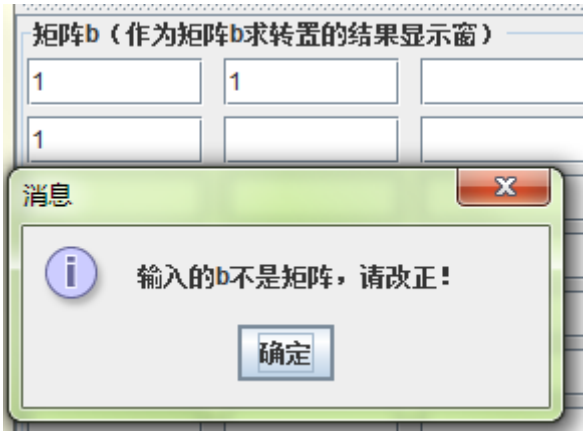


图 31　求矩阵转置时输入的不是矩阵

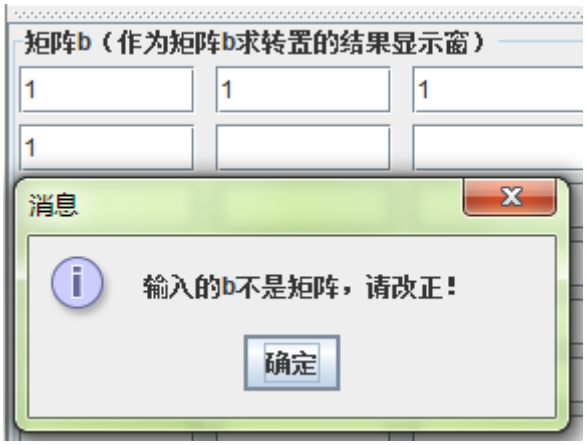


图 32　求矩阵的逆时输入的不是矩阵

（9）源代码（包含 3 个类）及注释

①第一个类（矩阵 a 面板）

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.TitledBorder;

publicclassMatrixaextends JPanel
{
        JTextField [][]a;

        //构造方法
        public Matrixa()
        {
            this.setBorder(new  TitledBorder("矩阵 a（同时作为 a+b 与 a*b 的结果显示窗）
"));
                                        //设置面板带有含标题的边框线
            this.setLayout(new GridLayout(10,10,10,5));
                                //面板布局，10 行 10 列，
                                //组件水平间距 10 像素，垂直间距 5 像素
            a = newJTextField[10][10];
            for(int m=0;m<10;m++)
                for(int n=0;n<10;n++)
                {
                    a[m][n]=new JTextField();
                    this.add(a[m][n]);
                }          //10*10 的矩阵输入区
        }
}
```

②第二个类（矩阵 b 面板）

```
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
import javax.swing.border.TitledBorder;

publicclassMatrixbextends JPanel implements ActionListener
{
        private Font f1;
        private Color c1,c2;
        private JTextField [][]b;
        private JButton button_jia;
```

```
private JButton button_cheng;
private JButton button_zhuanzhi;
private JButton button_zhi;
private JButton button_cleara;
private JButton button_clearb;
private Matrixa matrixa;

//构造方法
public Matrixb(Matrixa matrixa)
{
    this.setBorder(new TitledBorder("矩阵 b（作为矩阵 b 求转置的结果显示窗）"));
                        //设置面板带有含标题的边框线
    this.setLayout(new GridLayout(11,10,10,5));
                        //面板布局，10 行 10 列，
                        //组件水平间距 10 像素，垂直间距 5 像素
    this.matrixa=matrixa; //成员对象

    b = newJTextField[10][10];
    for(int i=0;i<10;i++)
        for(int j=0;j<10;j++)
        {
            b[i][j]=new JTextField();
            this.add(b[i][j]);
        }           //10*10 的矩阵输入区

    f1=new Font("幼圆",Font.BOLD,14); //设置新字体
    c1=new Color(57,221,36);            //浅绿色
    c2=new Color(115,211,239);          //浅蓝色

    button_jia=newJButton("a+b");
    button_jia.setFont(f1);
    button_jia.setForeground(Color.white);
                                        //按钮字体颜色为白色
    button_jia.setBackground(c1);
    button_jia.setOpaque(true);

    button_cheng=newJButton("a*b");
    button_cheng.setFont(f1);
    button_cheng.setForeground(Color.white);
    button_cheng.setBackground(c2);
    button_cheng.setOpaque(true);

    button_zhuanzhi=new JButton("b 的转置");
    button_zhuanzhi.setFont(f1);
```

```
        button_zhuanzhi.setForeground(Color.white);
        button_zhuanzhi.setBackground(c1);
        button_zhuanzhi.setOpaque(true);

        button_zhi=new JButton("b 的秩");
        button_zhi.setFont(f1);
        button_zhi.setForeground(Color.white);
        button_zhi.setBackground(c2);
        button_zhi.setOpaque(true);

        button_cleara=new JButton("清空 a");
        button_cleara.setFont(f1);
        button_cleara.setForeground(Color.white);
        button_cleara.setBackground(c1);
        button_cleara.setOpaque(true);

        button_clearb=new JButton("清空 b");
        button_clearb.setFont(f1);
        button_clearb.setForeground(Color.white);
        button_clearb.setBackground(c2);
        button_clearb.setOpaque(true);

        button_jia.addActionListener(this);
                    //为“a+b”按钮注册动作事件监听器
    button_cheng.addActionListener(this);
    button_zhuanzhi.addActionListener(this);
    button_zhi.addActionListener(this);
    button_cleara.addActionListener(this);
    button_clearb.addActionListener(this);

        this.add(button_jia);
        this.add(new JLabel(""));
        this.add(button_cheng);
        this.add(new JLabel(""));
        this.add(button_zhuanzhi);
        this.add(new JLabel(""));
        this.add(button_zhi);
        this.add(new JLabel(""));
        this.add(button_cleara);
        this.add(button_clearb);
      }

    //判断字符串是否为整数
publicstaticboolean isInteger(String value)
```

```
 {
    try
     {
          Integer.parseInt(value);
    returntrue;
       }
    catch (NumberFormatException e)
     {
    returnfalse;
     }
  }

//判断字符串是否为浮点数
publicstaticboolean isDouble(String value)
 {
    try
     {
          Double.parseDouble(value);
    if (value.contains("."))
          returntrue;
    else
          returnfalse;
       }
    catch (NumberFormatException e)
     {
    returnfalse;
     }
  }

//判断字符串是否为数字
publicstaticboolean isNumber(String value)
 {
    returnisInteger(value) || isDouble(value);
 }

//实现按钮的动作事件监听器接口
publicvoid actionPerformed(ActionEvent ev)
 {
    //如果点击按钮“a+b”
    if(ev.getSource()==button_jia)
     {
          int ma,na,mb,nb;
           ma=na=mb=nb=0;        //ma、na 是矩阵 a 的行、列数
                                 //mb、nb 是矩阵 b 的行、列数
```

```
int i,j,count=0;
double x,y;
double z[][]=newdouble[10][10];

 //四个 for 语句实现分别求出矩阵 a、b 的行、列数
for(i=0;isNumber(matrixa.a[i][0].getText());i++)
      ma++;     //以矩阵 a 第一列的数字个数作为矩阵 a 的行数
for(j=0;isNumber(matrixa.a[0][j].getText());j++)
      na++;     //以矩阵 a 第一行的数字个数作为矩阵 a 的列数
for(i=0;isNumber(b[i][0].getText());i++)
      mb++;     //以矩阵 b 第一列的数字个数作为矩阵 b 的行数
for(j=0;isNumber(b[0][j].getText());j++)
      nb++;     //以矩阵 b 第一行的数字个数作为矩阵 b 的列数

if((ma==0&&na==0)||(mb==0&&nb==0))
      JOptionPane.showMessageDialog(null, "矩阵不能为空！");
                  //未输入数据就点击“a+b”按钮时，进行提示
elseif((ma!=mb)||(na!=nb))
      JOptionPane.showMessageDialog(null, "矩阵相加，" +
                        "必须保证相加两矩阵行数相等、列数也相等！");
            //输入的矩阵行、列数不匹配时，进行提示
else
 {
     sometag:
     for(i=0;i<ma;i++)
          for(j=0;j<na;j++)
               try
                {
                     x=Double.parseDouble(
                               matrixa.a[i][j].getText());
                     y=Double.parseDouble(
                               b[i][j].getText());
                                    //获得矩阵第 i 行第 j 列的数字
                     z[i][j]=x+y; //保存矩阵相加的结果
                     count++;       //记录矩阵中输入数字的文本区的总数
                }
     catch (NumberFormatException e)
                {
          JOptionPane.showMessageDialog(null,
                     "输入的不是矩阵，请改正！");
                     //输入的必须为矩阵形式，
                     //即 m 行 n 列必须全有数据，否则报错
          break sometag;   //跳出两层循环
                }
```

```
            //若输入的矩阵符合规范，则在矩阵 a 中显示相加结果
            if(count==ma*na)
             {
                 for(i=0;i<10;i++)
                      for(j=0;j<10;j++)
                           matrixa.a[i][j].setText("");
                                //先将作为显示区的矩阵 a 的
                                //10*10 个文本区清空，便于观察结果

                 for(i=0;i<ma;i++)
                      for(j=0;j<na;j++)
                           matrixa.a[i][j].setText(""+(z[i][j]));
                                //显示相加结果
             }
         }
    }

    //如果点击按钮“a*b”
if(ev.getSource()==button_cheng)
    {
        int ma,na,mb,nb;
          ma=na=mb=nb=0;        //ma、na 是矩阵 a 的行、列数
                                //mb、nb 是矩阵 b 的行、列数
        int i,j,count=0;
        double x[][]=newdouble[10][10];
        double y[][]=newdouble[10][10];
        double z[][]=newdouble[10][10];

         //四个 for 语句实现分别求出矩阵 a、b 的行、列数
        for(i=0;isNumber(matrixa.a[i][0].getText());i++)
             ma++;
        for(j=0;isNumber(matrixa.a[0][j].getText());j++)
             na++;
        for(i=0;isNumber(b[i][0].getText());i++)
             mb++;
        for(j=0;isNumber(b[0][j].getText());j++)
             nb++;

        if((ma==0&&na==0)||(mb==0&&nb==0))
              JOptionPane.showMessageDialog(null, "矩阵不能为空！ ");
                    //未输入数据就点击“a*b”按钮时，进行提示
        elseif(na!=mb)
              JOptionPane.showMessageDialog(null, "矩阵 a*b， " +
```

```
                    "必须保证 a 的列数等于 b 的行数！");
            //输入的矩阵不能相乘时，进行提示
else
 {
    sometag:
    for(i=0;i<ma;i++)
        for(j=0;j<na;j++)
            try
             {
                x[i][j]=Double.parseDouble(
                            matrixa.a[i][j].getText());
                    //获得矩阵 a 中的数据
                count++;
             }
    catch (NumberFormatException e)
             {
        JOptionPane.showMessageDialog(null,
                 "输入的 a 不是矩阵，请改正！");
            //输入的必须为矩阵形式，
            //即 m 行 n 列必须全有数据，否则报错
        break sometag;
             }

    sometag:
    for(i=0;i<mb;i++)
        for(j=0;j<nb;j++)
            try
             {
                y[i][j]=Double.parseDouble(
                            b[i][j].getText());
                    //获得矩阵 b 中的数据
                count++;
             }
    catch (NumberFormatException e)
             {
        JOptionPane.showMessageDialog(null,
                 "输入的 b 不是矩阵，请改正！");
                  //输入的必须为矩阵形式，
                  //即 m 行 n 列必须全有数据，否则报错
        break sometag;
             }

   //若输入的矩阵符合规范，则在矩阵 a 中显示相加结果
    if(count==ma*na+mb*nb)
```

```
            {
                for(i=0;i<10;i++)
                    for(j=0;j<10;j++)
                        matrixa.a[i][j].setText("");
                            //先将作为显示区的矩阵 a 的
                            //10*10 个文本区清空，便于观察结果

                 //相乘后的矩阵行数与 a 矩阵相同，列数与 b 矩阵相同
                for(i=0;i<ma;i++)
                    for(j=0;j<nb;j++)
                     {
                        for(int k=0;k<na;k++)
                        z[i][j]+=x[i][k]*y[k][j];
                          matrixa.a[i][j].setText(""+(z[i][j]));
                     }
            }
        }
    }

    //如果点击按钮“b 的转置”
if(ev.getSource()==button_zhuanzhi)
    {
        int mb,nb;        //mb、nb 是矩阵 b 的行、列数
        mb=nb=0;
        int i,j,count=0;
        double x[][]=newdouble[10][10];

        //两个 for 语句实现分别求出矩阵 b 的行、列数
        for(i=0;isNumber(b[i][0].getText());i++)
            mb++;
        for(j=0;isNumber(b[0][j].getText());j++)
            nb++;

        if(mb==0&&nb==0)
            JOptionPane.showMessageDialog(null,
                            "矩阵 b 不能为空！ ");
                        //未输入数据就点击“b 的转置”按钮时，进行提示
        else
         {
            sometag:
            for(i=0;i<mb;i++)
                for(j=0;j<nb;j++)
                    try
                     {
```

```
                        x[i][j]=Double.parseDouble(
                                b[i][j].getText());
                            //获得矩阵 b 中的数据
                        count++;
                    }
                catch (NumberFormatException e)
                    {
                    JOptionPane.showMessageDialog(null,
                            "输入的 b 不是矩阵，请改正！");
                    //输入的必须为矩阵形式，
                    //即 m 行 n 列必须全有数据，否则报错
                    break sometag;
                    }

                //若输入的矩阵符合规范，则在矩阵 b 中显示 b 的转置
                if(count==mb*nb)
                {
                    for(i=0;i<10;i++)
                        for(j=0;j<10;j++)
                            b[i][j].setText("");
                                //先将作为显示区的矩阵 b 的
                                //10*10 个文本区清空，便于观察结果
                    for(i=0;i<mb;i++)
                        for(j=0;j<nb;j++)
                            b[j][i].setText(""+(x[i][j]));
                }
            }
        }

        //如果点击按钮“b 的秩”
        if(ev.getSource()==button_zhi)
        {
            int mb,nb;          //mb、nb 是矩阵 b 的行、列数
            mb=nb=0;
            int i,j,count=0;
            int rank=0;         //矩阵 b 的秩
            double max;
            double x[][]=newdouble[10][10];

            //两个 for 语句实现分别求出矩阵 b 的行、列数
            for(i=0;isNumber(b[i][0].getText());i++)
                mb++;
            for(j=0;isNumber(b[0][j].getText());j++)
                nb++;
```

```
if(mb==0&&nb==0)
    JOptionPane.showMessageDialog(null,
                    "矩阵 b 不能为空！");
                //未输入数据就点击“b 的秩”按钮时，进行提示
else
 {
    sometag:
    for(i=0;i<mb;i++)
        for(j=0;j<nb;j++)
            try
             {
                x[i][j]=Double.parseDouble(
                            b[i][j].getText());
                    //获得矩阵 b 中的数据
                count++;
             }
    catch (NumberFormatException e)
             {
        JOptionPane.showMessageDialog(null,
                    "输入的 b 不是矩阵，请改正！");
            //输入的必须为矩阵形式，
            //即 m 行 n 列必须全有数据，否则报错
        break sometag;
             }

     //若输入的矩阵符合规范，则用高斯-约当法求出矩阵 b 的秩
    if(count==mb*nb)
     {
        double temp;
        int m,zhu;
        for(i=0;i<mb;i++)
        {
            //选列主元（即第 i 列最大的元素）
            max=Math.abs(x[i][i]);
            zhu=i;
            for(m=i+1;m<mb;m++)
            {
                if(Math.abs(x[m][i])>max)
                {
                    max=Math.abs(x[m][i]);
                    zhu=m;   //记录主元所在行数
                }
            }
```

```
                    //如果主元为一个极小的数，则不用再消元
                    if(Math.abs(max)<Math.pow(10,-10))
                        continue;

                    for(j=0;j<nb&&zhu!=i;j++)
                    {
                        temp=x[zhu][j];
                        x[zhu][j]=x[i][j];
                        x[i][j]=temp;
                    }

                    //高斯-约当消元法
                    double w=x[i][i];
                    for(j=0;j<nb;j++)
                    {
                        x[i][j]=x[i][j]/w;
                    }
                    for(m=i+1;m<mb;m++)
                    {
                        temp=x[m][i];
                        for(j=i;j<nb;j++)
                        {
                            x[m][j]-=x[i][j]*temp;   //消元
                            if(x[m][j]<Math.pow(10,-5))
                                x[m][j]=0;
                        }
                    }
                }

                for(i=0;i<mb;i++)
                    if(x[i][i]==1)
                        rank++;
                JOptionPane.showMessageDialog(null,
                    "矩阵 b 的秩为："+rank);
            }
        }
    }

    //如果点击按钮“清空 a”
if(ev.getSource()==button_cleara)
    {
        for(int i=0;i<10;i++)
            for(int j=0;j<10;j++)
```

```
                    matrixa.a[i][j].setText("");
                        //清空 a 矩阵
                }

               //如果点击按钮“清空 a”
               if(ev.getSource()==button_clearb)
                {
                    for(int i=0;i<10;i++)
                        for(int j=0;j<10;j++)
                            b[i][j].setText("");
                                //清空 b 矩阵
                }
            }
}
```

③第三个类（分割窗格）

```
import javax.swing.*;

publicclassMatrixCalculatorextends JFrame
{
        private Matrixa matrixa=new Matrixa();
        private Matrixb matrixb=new Matrixb(matrixa);

        public MatrixCalculator()                    //顶层框架的构造方法
        {
            super("矩阵计算器");                        //框架标题
            this.setSize(1000,700);                 //框架大小，单位是像素
            this.setLocationRelativeTo(null);   //窗口居中
            this.setBackground(java.awt.Color.lightGray);   //框架背景色
            this.setDefaultCloseOperation(EXIT_ON_CLOSE);   //关闭框架

            JSplitPane split=new
            JSplitPane(JSplitPane.VERTICAL_SPLIT,true,
                    matrixa,matrixb);
                                                    //垂直分割窗格为上下两部分，
                                                    //上部为 matrixa 对象，
                                                    //下部为 matrixb 对象，
            //窗格中组件随分割线移动而改变大小
            split.setDividerLocation(310);        //分割线位置
            this.getContentPane().add(split);   //框架添加分割窗格
            this.setVisible(true);                  //设置框架可视
        }

        publicstaticvoid main(String arg[])
```

```
        {
            new MatrixCalculator();                    //新建框架对象
        }
}
```

（10）设计总结

主要难点在于对矩阵秩的求法，因此上网查了高斯-约当消元法，此法对于小型矩阵的求秩还是很有效的。另外，矩阵布局采用了两个面板，分别存放矩阵 a 与矩阵 b，每个面板中添加 10*10 用于输入矩阵元素的文本区。