课程设计题目

一、课程设计目的和要求

课程设计是巩固所学理论知识、提高程序设计能力的重要实践环节。课程设计的目的是，使学生能够综合应用 java 基础知识和基本方法，编写实用有效的应用程序，体会程序设计的全过程，深入理解和进一步巩固所学知识，培养自学能力，培养独立分析问题和解决问题的作风和能力，提高软件设计能力，为今后进行系统软件和应用软件的开发研究打下坚实基础，培养刻苦专研精神和严谨的治学作风。

课程设计的任务是，综合运用 java 语言的基础知识和面向对象设计的基本原则，独立编制一个具有中等规模的、一定难度的、解决实际问题的应用程序。进行课题的需求分析、设计方案准备、编程、运行、调试、完善等软件设计的各环节。撰写实验报告，说明设计目的、题目、题意分析、设计特点、设计方案、功能说明、实现手段、源程序清单、运行结果及结果分析、设计经验和教训总结、存在问题及改进措施等。

课程设计功能要求：①采用 swing 组件设计图形用户界面，使用组件数组；使用菜单；使用多文档界面技术。②响应事件。③处理异常，当输入数据错误时，弹出对话框，提示重新输入信息。④使用线程演示算法执行过程中数组或其他变量数据的动态变化情况。⑤从指定文件中读取原始数据，并将运算结果写入文件，使用带有过滤器的打开和保存文件对话框选择文件名。说明采用什么格式的文件，采用什么流，比较多种格式文件的操作区别。⑥实现基于 Socket 通信的网络应用程序。

二、 题意说明及分析

基于线程同步的多窗口弹弹球。

① 已知多个弹弹球运行在 A 窗口。

② 创建 B 窗口，其中没有球；移动 B 窗口，当 B 窗口的一边与 A 窗口接触时，若干球从 A 窗口运行到 B 窗口；当 A、B 窗口相邻时，球在 A 和 B 窗口范围内运行；当 B 窗口离开 A 窗口时，若干球分别在 A 或 B 窗口中运行。

③ 再创建 C、D 等窗口，具有上述功能。

三、算法设计与分析

本次课程设计，采用多线程技术，要正确处理好线程的并发机制以及线程之间的通信。就对于 A 窗口和 B 窗口来说，要想实现小球在两个窗口之间运动，由 A 窗口向 B 窗口运动时，需要建立一个缓冲区，A 作为发送者，B 作为接受者；小球由 B 窗口向 A

窗口运动时，也需要建立一个缓冲区，B 作为发送者，A 作为接受者，以此而推，建立 C、D 窗口。因为并发线程的不确定性，使得以上两种不同的过程不能同时使用一个缓冲区，所以，需要两个缓冲区。

在本次课设中，我还添加了文件功能，储存多个球，可以实现文件的打开与保存。当开始程序运行时，就进行一次文件的读取功能；在程序运行过程中，可以进行多次的文件保存。

四、源程序

```
public class Ballbuffer  //缓冲区
{
    private Ball ball;
    boolean isEmpty=true;
    public Ballbuffer()
    {}
    public synchronized void put( Ball ball)
    {
        while(!isEmpty)
        try{this.wait();}
catch(InterruptedException e) {}
        this.ball=ball;
        isEmpty=false;
        this.notify();
    }
    public synchronized Ball get()
    {
          while(isEmpty)
           try
{this.wait();}catch(InterruptedException e) {}
          isEmpty=true;
          this.notifyAll();
          return this.ball;
    }
}
class send extends Thread
{
    private Ballbuffer buffer;
    private ArrayList<Ball> balls;
    public send(Ballbuffer buffer,ArrayList<Ball>
balls)
    {
        this.buffer=buffer;
        this.balls=new ArrayList<Ball>(balls);
    }
    public void run()
    {
        for(int i=0;i<this.balls.size();i++)
           {
             System.out.println("发送球 2");
               buffer.put(balls.get(i));
               try
                  {
                       Thread.sleep(100);
                  }catch(InterruptedException  e)
{}
           }
    }}
```

```
//发送球 1
public void paint(Graphics g)
         {
              try
              {
                 lost=new ArrayList<Ball>();
                 for(int i=0;i<balls.size();i++)
                 {

     g.setColor(this.balls.get(i).color);

     this.balls.get(i).x=this.balls.get(i).left?th
is.balls.get(i).x-this.balls.get(i).v:this.balls.
get(i).x+this.balls.get(i).v;  //控制球的水平速度
                   if(this.balls.get(i).x<=0)
                   {

     this.balls.get(i).left=!this.balls.get(i).lef
t;
                        this.balls.get(i).x=0;
                   }

     if(balls.get(i).x>=this.getWidth())   //接收
者的位置
                   {
                       Ball n=new
Ball(balls.get(i));
                       this.lost.add(n);
                        new
send(this.buffer,this.lost).start();
                        this.balls.remove(i);
                   }
     this.balls.get(i).y=this.balls.get(i).up?this
.balls.get(i).y-this.balls.get(i).v:this.balls.ge
t(i).y+this.balls.get(i).v;  //控制球的垂直速度
                   if(this.balls.get(i).y<=0)
                   {

     this.balls.get(i).up=!this.balls.get(i).up;
                        this.balls.get(i).y=0;
                   }

     if(this.balls.get(i).y>=this.getHeight())
                   {

     this.balls.get(i).up=!this.balls.get(i).up;

     this.balls.get(i).y=this.getHeight();
                   }

     g.fillOval(this.balls.get(i).x,this.balls.get
(i).y,
this.balls.get(i).size,this.balls.get(i).size);
//控制球的大小和方向
                }
              }catch(IndexOutOfBoundsException e)
{}
         }

     public void run()                    //线程
     {
          while(true)
          try
          {
               repaint();
               Thread.sleep(250);
          }
          catch(InterruptedException ex)
          {
               break;
          }
     }
//接收球 1
public void run()
     {
     while(true)
     {
        for(int i=0;i<3;i++)
          {
this.canvas.addBalls(buffer.get());
          }
        thread=new Thread(this.canvas);
        thread.start();
        try{Thread.sleep(1);}
        catch(InterruptedException e)
        {
                break;
        }
     }
}
//发送球 2
public void paint(Graphics g)
     {
          lost=new ArrayList<Ball>();
          try
          {
             for(int i=0;i<this.balls.size();i++)
            {
                g.setColor(balls.get(i).color);

this.balls.get(i).x=this.balls.get(i).left?this.b
alls.get(i).x+this.balls.get(i).v:this.balls.get(
i).x-this.balls.get(i).v;
                if(balls.get(i).x<=0)
                {
                    Ball n=new Ball(balls.get(i));
                   this.lost.add(n);
                   new
send(this.buffer,this.lost).start();
                   this.balls.remove(i);
                }

if(balls.get(i).x>=this.getWidth())
                {

     balls.get(i).left=!balls.get(i).left;
```

```
 }

this.balls.get(i).y=this.balls.get(i).up?this.balls.get(i).y+this.balls.get(i).v:this.balls.get(i).y-this.balls.get(i).v;

if(balls.get(i).y<=0||balls.get(i).y>=this.getHeight())
               {

     balls.get(i).up=!balls.get(i).up;
                }
               g.fillOval(balls.get(i).x, balls.get(i).y, balls.get(i).size, balls.get(i).size);
             }
          }catch(IndexOutOfBoundsException e) {}
     }
     public void run()
     {
          while(true)
          try
          {
             repaint();
             Thread.sleep(250);
          }
          catch(InterruptedException e)
          {
               break;
          }
     }
//接收球2
public void run()
     {
     while(true)
          {
             for(int i=0;i<3;i++)
               {

this.canvas.addBall(buffer2.get());
               }
             thread1=new Thread(this.canvas);
             thread1.start();
             try{Thread.sleep(1);}
             catch(InterruptedException e)
             {
                     break;
             }
          }
     }
```

//文件读取

```
public void readFrom(File file)
    {
     try
     {
          FileInputStream fin=new FileInputStream(file);
          ObjectInputStream objin=new ObjectInputStream(fin);
          while(true)
               try
               {
                    this.canvas.balls.add((Ball)objin.readObject());
               }catch(EOFException eofx) {  break;}
          objin.close();
          fin.close();
     }
     catch(Exception e) {}
    }
```

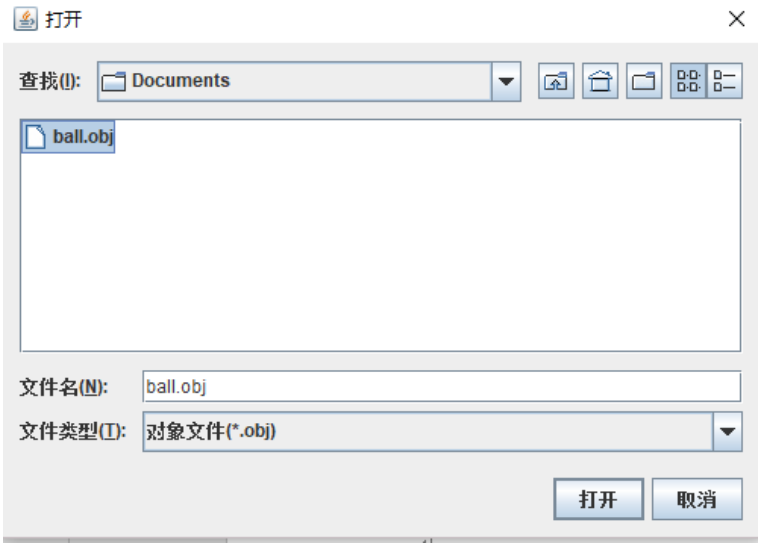
//文件写入

```
public void writeTo(File file)
    {
     try
     {
          FileOutputStream fout=new FileOutputStream(file);
          ObjectOutputStream objout=new ObjectOutputStream(fout);
          for(int i=0;i<this.canvas.balls.size();i++)
          {
                 objout.writeObject(this.canvas.balls.get(i));
          }
          objout.close();
          fout.close();
     }catch(Exception e) {}
    }
```

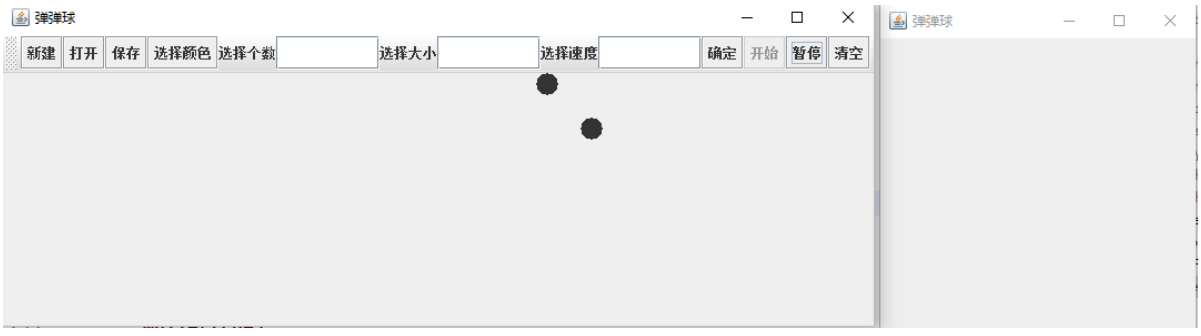
五、结果及分析

1、基于线程同步的弹弹球

2、打开文件



3、弹弹球开始运行



经过修改，文件部分被加以限制了，可以根据自己的要求修改文件技术部分。

六、总结与思考

本次课设，在于使得弹弹球在多个窗口运行，这里采用的是线程同步技术，通过两个窗口共用一个缓冲区，使得弹弹球可以在两个窗口之间自由移动。因为在球由左边窗口向右边窗口移动时，需要的缓冲区与球由右边窗口向左边窗口移动的缓冲区一致，所以一开始，我采用一个缓冲区，结果导致从左向右、从右向左两个线程无法同步，所以两个接收线程共用一个缓冲区的想法是错误的。之后，我开始使用两个不同的缓冲区（性质一样），这样，球从左边向右边移动，从右边向左边移动，就可以实现。

因为，一开始，我想的太多，两个球同时穿过窗口边界，结果导致算法变得复杂、难懂。经过老师指点，我可以先选择一个球的接收和一个球的发送，暂时不用考虑多个球同时穿过窗口边界问题。

本次课设，我采用了文件存储功能，因为我使用的是对象字节流，将每个球对象存入文件中，当球全部读入时，在 obj.close()时，文件会自动给它文件一个文件结束异常，然后在文件读出时，在读取结束时，会 catch 到文件结束流异常，然后文件读取结束。