实验内容	第 14 周 实验　Java 多线程编程			成绩	
姓名	王秋锋	学号	2015111948	班级	计算机 2015-03 班
专业	计算机科学与技术			日期	2017 年 12 月 8 日

【实验目的】--多线程与动画编程

- 理解掌握 Thread 类和编写多线程应用程序的两种方式：继承 Thread，实现 Runnable 接口
- 理解掌握基于多线程编程的数据共享和同步机制
- 掌握基于多线程的编程的应用，如动画处理

【实验内容】

1、Thread 子类的方法实现多线程

源代码如下：

```
class SimpleThread extends Thread {
  public SimpleThread(String str) {
      super(str);
  }
  public void run() {
for (int i = 0; i < 10; i++) {
          System.out.println(i + "" + getName());
try {
              sleep((int)(Math.random() * 1000));
          } catch (InterruptedException e) {}
      }
      System.out.println("DONE! " + getName());
  }
 }
  public class TwoThreadsTest {
   public static void main (String[] args) {
      new SimpleThread("Go to Beijing??").start();
      new SimpleThread("Stay here!!").start();
 }
}
```


说明：1、编译运行上面的程序，理解掌握通过继承Thread类实现多线程编程，编辑并运行程序
2、修个程序，如果建立更多个线程对象，运行结果是什么？

【实验结果与分析】

实验结果：

```
0Go to Beijing??
0Stay here!!
1Go to Beijing??
1Stay here!!
2Stay here!!
2Go to Beijing??
3Stay here!!
3Go to Beijing??
4Go to Beijing??
4Stay here!!
5Go to Beijing??
5Stay here!!
6Go to Beijing??
6Stay here!!
7Go to Beijing??
7Stay here!!
8Go to Beijing??
9Go to Beijing??
8Stay here!!
9Stay here!!
DONE! Go to Beijing??
DONE! Stay here!!
```

实验分析：

创建两个多线程同时运行，因为 sleep 函数里面的时间是随机数，因为存在抢占机制，所以"Go to Beijing??"的线程里面 for 内容执行完之后，被"Stay here!!"线程抢占执行了 for 内容。

再加入一个"Leave now!!"线程，实验结果如下：

```
0Go to Beijing??
0Stay here!!
0Leave now!!
1Leave now!!
1Stay here!!
2Stay here!!
3Stay here!!
4Stay here!!
1Go to Beijing??
5Stay here!!
2Leave now!!
2Go to Beijing??
3Leave now!!
6Stay here!!
3Go to Beijing??
4Leave now!!
7Stay here!!
4Go to Beijing??
5Leave now!!
8Stay here!!
5Go to Beijing??
6Leave now!!
6Go to Beijing??
7Leave now!!
9Stay here!!
7Go to Beijing??
DONE! Stay here!!
8Leave now!!
9Leave now!!
8Go to Beijing??
9Go to Beijing??
DONE! Go to Beijing??
DONE! Leave now!!
```


2、实现 Runnable 接口的方法实现多线程

1. 程序功能：一个时钟 Applet，它显示当前时间并逐秒进行更新
2. 编写 KY11_2.java 程序文件，源代码如下。

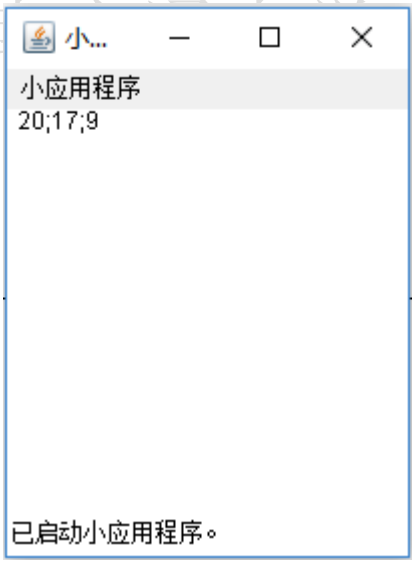
```
import java.awt.*;
import java.applet.*;
import java.util.*;
public class Clock extends Applet implements Runnable{
Thread clockThread;
        public void start(){
                if(clockThread==null){
                        clockThread=new Thread(this,"Clock");
                        clockThread.start();
                }
        }
        public void run(){
                while(clockThread !=null){
                        repaint();
```

```
                    try{
                        clockThread.sleep(1000);
                    }catch(InterruptedException e){}
                }
        }
        public void paint(Graphics g){
            Date now=new Date();
            g.drawString(now.getHours()+";"+now.getMinutes()+";"+now.getSeconds(),5,10);
            // g.drawString(now.toString(),5,600)
        }
        public void stop(){
            clockThread.stop();
            clockThread=null;
        }
    }
```

说明：编译运行上面的程序，该程序体现了基于多线程动画编程

【实验结果与分析】

实验结果：



实验分析：

在 java 中可有两种方式实现多线程，一种是继承 Thread 类，一种是实现 Runnable 接口。这里创建 Clock 程序调用 Runnable 接口，然后用 date 类显示出当前时间。

3、利用多线程编程，编写火车站售票程序,3个窗口共卖10张票，用多线程模拟窗口买票，运行结果类似下图

```
窗口1卖出票：票号＝1
窗口2卖出票：票号＝2
……
```

说明：解决多线程同步问题；给出源码与运行结果

【实验结果与分析】

实验代码：

```
class Seller extends Thread {
    protected static int i = 0;
    public Seller() {}
    public void run() {
     for (int k = 100; k > 0; k--) {
        synchronized (this) {
           if (i < 10) {
              if (Thread.currentThread().getName().equals("Thread-1")) {
                 System.out.println("窗口1卖出票：票号 = " + ++i);
              } else if
(Thread.currentThread().getName().equals("Thread-2")){
                 System.out.println("窗口2卖出票：票号 = " + ++i);
              }
              else{
                 System.out.println("窗口3卖出票：票号 = " + ++i);
              }
           }
           else {
              System.exit(0);
           }
        }
        try {
           Thread.sleep(200);
        } catch (InterruptedException e) {
           e.printStackTrace();
        }
      }
    }
}

public class TestSeller {
   public static void main(String[] args) {
      Seller s = new Seller();
      Thread th1 = new Thread(s);
      Thread th2 = new Thread(s);
      Thread th3 = new Thread(s);
```

```
        th1.start();
        th2.start();
        th3.start();
    }
}
```

实验结果：

```
窗口1卖出票：票号 = 1
窗口3卖出票：票号 = 2
窗口2卖出票：票号 = 3
窗口2卖出票：票号 = 4
窗口3卖出票：票号 = 5
窗口1卖出票：票号 = 6
窗口2卖出票：票号 = 7
窗口1卖出票：票号 = 8
窗口3卖出票：票号 = 9
窗口1卖出票：票号 = 10
```

4、编辑运行下面动画程序


```
import java.awt.Color;
import java.util.*;
import java.awt.*;
import java.applet.*;
public class LxDemo    extends Applet implements Runnable
{
    Thread timer = null;
    Label a;
    int lastxs=50, lastys=30, lastxm=50, lastym=30, lastxh=50, lastyh=30;
    public void init(){ //初始化方法
        setBackground(Color.white); //设置 Applet  背景
        a=new Label("");
        add(a);}
    public void paint(Graphics g)    //显示数字和图形时钟的方法
    {
        int xh, yh, xm, ym, xs, ys, s, m, h, xcenter, ycenter;
        Date rightnow = new Date(); //获取当前日期和时间
        String today = rightnow.toLocaleString(); //时间对应的字符串
        a.setText(today); //显示数字时钟
        s = rightnow.getSeconds();
        m = rightnow.getMinutes();
        h = rightnow.getHours();
        xcenter=100; ycenter=80; //图形钟的原点
```

```
        //以下计算秒针、分针、时针位置
        xs = (int)(Math.cos(s * 3.14f/30 - 3.14f/2) * 45 + xcenter);
        ys = (int)(Math.sin(s * 3.14f/30 - 3.14f/2) * 45 + ycenter);
        xm = (int)(Math.cos(m * 3.14f/30 - 3.14f/2) * 40 + xcenter);
        ym = (int)(Math.sin(m * 3.14f/30 - 3.14f/2) * 40 + ycenter);
        xh = (int)(Math.cos((h*30+m/2)*3.14f/180-3.14f/2)*30+xcenter);
        yh = (int)(Math.sin((h*30+m/2)*3.14f/180-3.14f/2)*30+ycenter);
        g.setFont(new Font("TimesRoman", Font.PLAIN, 14));
        g.setColor(Color.orange); //设置表盘颜色
        g.fill3DRect(xcenter-50,ycenter-50,100,100,true); //画表盘
        g.setColor(Color.darkGray); //设置表盘数字颜色
        g.drawString("9",xcenter-45,ycenter+3); //画表盘上的数字
        g.drawString("3",xcenter+40,ycenter+3);
        g.drawString("12",xcenter-5,ycenter-37);
        g.drawString("6",xcenter-3,ycenter+45);
        //时间变化时，需要重新画各个指针，即先消除原有指针，然后画新指针
        g.setColor(Color.orange); //用表的填充色画线，可以消除原来画的线
        if (xs != lastxs || ys != lastys){ //秒针变化
        g.drawLine(xcenter, ycenter, lastxs, lastys); }
        if (xm != lastxm || ym != lastym) { //分针变化
        g.drawLine(xcenter, ycenter-1, lastxm, lastym);
        g.drawLine(xcenter-1, ycenter, lastxm, lastym); }
        if (xh != lastxh || yh != lastyh) { //时针变化
        g.drawLine(xcenter, ycenter-1, lastxh, lastyh);
        g.drawLine(xcenter-1, ycenter, lastxh, lastyh); }
        g.setColor(Color.red); //使用红色画新指针
        g.drawLine(xcenter, ycenter, xs, ys);
        g.drawLine(xcenter, ycenter-1, xm, ym);
        g.drawLine(xcenter-1, ycenter, xm, ym);
        g.drawLine(xcenter, ycenter-1, xh, yh);
        g.drawLine(xcenter-1, ycenter, xh, yh);
        lastxs=xs; lastys=ys; //保存指针位置
        lastxm=xm; lastym=ym;
        lastxh=xh; lastyh=yh; }
    public void start() { //启动线程的方法
        if(timer == null) {
            timer = new Thread(this);
            timer.start();
        }
    }
    public void stop() //停止线程方法
    {
        timer = null;
    }
    public void run(){ //每隔一秒钟，刷新一次画面的方法
        while (timer != null)
```

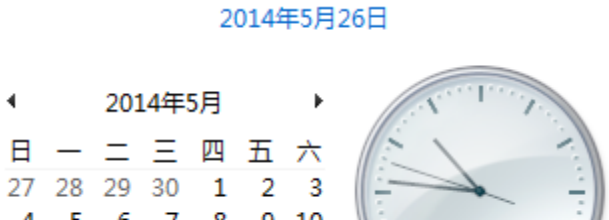
```
        {
            try { Thread.sleep(1000); }
            catch (InterruptedException e) {}
            repaint(); //调用 paint()方法重画时钟
        }
        timer = null;
    }
    public void update(Graphics g) //重写该方法是为了消除抖动现象
    {
        paint(g);
    }
     }
```

（3）【实验步骤】

① 在编辑区先选中 LxDemo　.java，编译 java 源文件，生成 LxDemo　.class。

② 再编辑 Hello.html，运行该程序，也可以在 Eclipse 以小应用程序形式直接运行

③ 理解掌握利用多线程编程实现动画，修改该程序，实现 Windows 的时钟动画和日历程序，如图



【实验结果与分析】

实验结果：

修改后的实验代码：

```
import java.util.*;
import java.util.Date;
import java.awt.Font;
import java.awt.event.*;
import java.awt.Color;
import java.awt.Label;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.BorderLayout;
import javax.swing.*;
import javax.swing.border.TitledBorder;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;

public class Test extends JFrame
{
   public static void main(String[] args)
   {
      try
      {
         Test frame = new Test();
         frame.setTitle("日历");
      }
      catch (Exception e){
          System.out.print("run error!");
      }
   }
   private static final long serialVersionUID = 1L;
   public Test()
   {
       Clock clock =new Clock();
       Calender cal = new Calender();
       JPanel jp2 = new JPanel();
       setLocationRelativeTo(null);
       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       setSize(450,300);
       setVisible(true);
       this.setContentPane(clock);
       this.getContentPane().add(cal,BorderLayout.WEST);
```

```
        setResizable(false);
    }

    //画时钟
    public class LxDemo extends JPanel implements Runnable{
        private static final long serialVersionUID = 1L;
        Thread timer = null;  //线程
        int lastxs=50, lastys=30, lastxm=50, lastym=30, lastxh=50, lastyh=30;
        public void paint(Graphics g){
            super.paint(g);
            int xh, yh, xm, ym, xs, ys, s, m, h, xcenter, ycenter;
            Date rightnow = new Date(); //获取当前日期和时间
            g.drawString(rightnow.toLocaleString(), 55,60);
            s = rightnow.getSeconds();
            m = rightnow.getMinutes();
            h = rightnow.getHours();
            xcenter=110; ycenter=150; //图形钟的原点
            //以下计算秒针、分针、时针位置
            xs = (int)(Math.cos(s * 3.14f/30 - 3.14f/2) * 45 + xcenter);
            ys = (int)(Math.sin(s * 3.14f/30 - 3.14f/2) * 45 + ycenter);
            xm = (int)(Math.cos(m * 3.14f/30 - 3.14f/2) * 40 + xcenter);
            ym = (int)(Math.sin(m * 3.14f/30 - 3.14f/2) * 40 + ycenter);
            xh = (int)(Math.cos((h*30+m/2)*3.14f/180-3.14f/2)*30+xcenter);
            yh = (int)(Math.sin((h*30+m/2)*3.14f/180-3.14f/2)*30+ycenter);
            g.setFont(new Font("TimesRoman", Font.PLAIN, 14));
            g.setColor(Color.orange); //设置表盘颜色
            g.fill3DRect(xcenter-50,ycenter-50,100,100,true); //画表盘
            g.setColor(Color.darkGray); //设置表盘数字颜色
            g.drawString("9",xcenter-45,ycenter+3); //画表盘上的数字
            g.drawString("3",xcenter+40,ycenter+3);
            g.drawString("12",xcenter-5,ycenter-37);
            g.drawString("6",xcenter-3,ycenter+45);
            //时间变化时，需要重新画各个指针，即先消除原有指针，然后画新指针
            g.setColor(Color.orange); //用表的填充色画线，可以消除原来画的线
            if (xs != lastxs || ys != lastys){ //秒针变化
            g.drawLine(xcenter, ycenter, lastxs, lastys); }
            if (xm != lastxm || ym != lastym) { //分针变化
                g.drawLine(xcenter, ycenter-1, lastxm, lastym);
                g.drawLine(xcenter-1, ycenter, lastxm, lastym); }
            if (xh != lastxh || yh != lastyh) { //时针变化
                g.drawLine(xcenter, ycenter-1, lastxh, lastyh);
                g.drawLine(xcenter-1, ycenter, lastxh, lastyh); }
            g.setColor(Color.red); //使用红色画新指针
            g.drawLine(xcenter, ycenter, xs, ys);
            g.drawLine(xcenter, ycenter-1, xm, ym);
            g.drawLine(xcenter-1, ycenter, xm, ym);
```

```
            g.drawLine(xcenter, ycenter-1, xh, yh);
            g.drawLine(xcenter-1, ycenter, xh, yh);
            lastxs=xs; lastys=ys; //保存指针位置
            lastxm=xm; lastym=ym;
            lastxh=xh; lastyh=yh;
             setBorder(new TitledBorder("时间"));
             setBackground(Color.white); // 定义颜色
        }
        public Date getDate()
        {
            Date timeNow = new Date();
            return timeNow;
        }
        // 刷新图层
        public void update(Graphics g)
        {
            paint(g);
        }
         // 画出一个帧的图像
        public void start() { //启动线程的方法
          if(timer == null) {
             timer = new Thread(this);
             timer.start();
          }
        }
        public void stop() //停止线程方法
        {
          timer = null;
        }
        public void run(){ //每隔一秒钟，刷新一次画面的方法
           while (timer != null)
           {
              try { Thread.sleep(1000); }
              catch (InterruptedException e) {}
              repaint(); //调用paint()方法重画时钟
           }
           timer = null;
        }
    }

    //时钟
    public class Clock extends JPanel{
         private static final long serialVersionUID = 1L;
         private UIManager.LookAndFeelInfo looks[];
         private LxDemo clock ;
         private JPanel pane_clock ;
```

```
        JPanel pane_cal;
        public Clock(){
            super();
             looks = UIManager.getInstalledLookAndFeels();
             changeTheLookAndFeel(2);
             clock = new LxDemo();
             clock.start();
             this.setBackground(Color.GRAY);
             this.setLayout(new BorderLayout());
             this.setOpaque(false);
             this.add(clock);
             this.setBorder(new TitledBorder("时间日期"));
             setSize( 300, 300 );
             setVisible( true );
        }
       private void changeTheLookAndFeel(int i){
            try{
             UIManager.setLookAndFeel(looks[i].getClassName());
             SwingUtilities.updateComponentTreeUI(this);
            }
            catch(Exception exception){
             exception.printStackTrace();
            }
       }
    }

   //设计日历
   public class Calender extends JPanel implements ActionListener
   {
        private static final long serialVersionUID = 1L;
        public  final String HOUR_OF_DAY = null;
        //定义
        JComboBox Month = new JComboBox();
        JComboBox Year = new JComboBox();
        JLabel Year_l = new JLabel("年");
        JLabel Month_l = new JLabel("月");
        Date now_date = new Date();
        JLabel[] Label_day = new JLabel[49];
        int now_year = now_date.getYear() + 1900;
        int now_month = now_date.getMonth();
        boolean bool = false;
        String year_int = null;
        int month_int;
        JPanel pane_ym = new JPanel();
        JPanel pane_day = new JPanel();
        public Calender(){
```

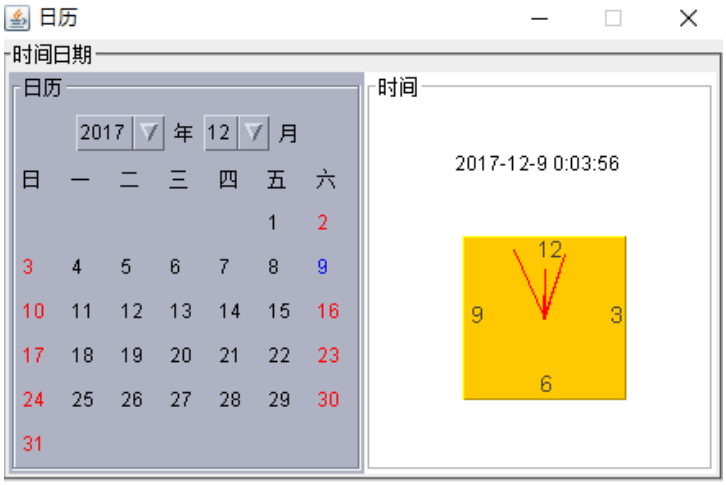
```
        super();
        //设定年月
        for (int i = now_year - 10; i <= now_year + 20; i++)
        {
            Year.addItem(i + "");
        }
        for (int i = 1; i < 13; i++)
        {
            Month.addItem(i + "");
        }
        Year.setSelectedIndex(10);
        pane_ym.add(new  JLabel("        "));
        pane_ym.add(Year);
        pane_ym.add(Year_l);
        Month.setSelectedIndex(now_month);
        pane_ym.add(Month);
        pane_ym.add(Month_l);
        pane_ym.add(new  JLabel("        "));

        Month.addActionListener(this);
        Year.addActionListener(this);
        //初始化日期并绘制
        pane_day.setLayout(new GridLayout(7, 7, 10, 10));
        for (int i = 0; i < 49; i++) {
            Label_day[i] = new JLabel(" ");
            pane_day.add(Label_day[i]);
        }
        this.setDay();
        this.setLayout(new BorderLayout());
        this.add(pane_day, BorderLayout.CENTER);
        this.add(pane_ym, BorderLayout.NORTH);
        this.setSize(100,200);
        this.setBorder(new TitledBorder("日历"));
        setSize(300,300);
    }
    void setDay()
    {
        if (bool)
        {
            year_int = now_year + "";
            month_int = now_month;
        }
        else
        {
            year_int = Year.getSelectedItem().toString();
            month_int = Month.getSelectedIndex();
```

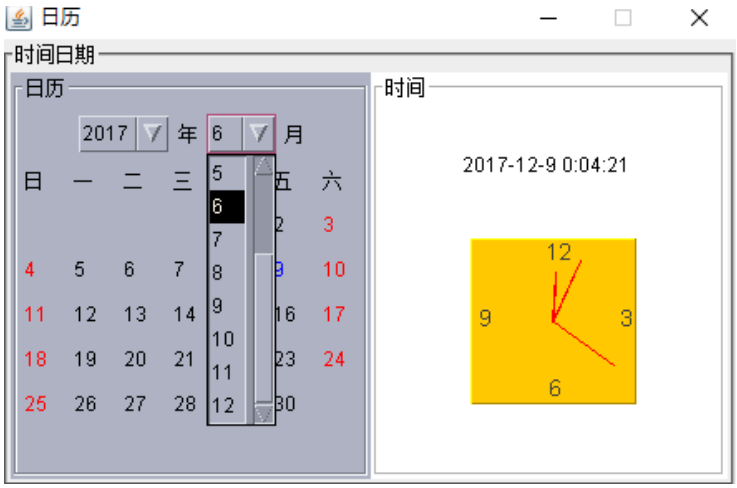
```
        }

        int year_sel = Integer.parseInt(year_int) - 1900; //获得年份值
        Date dt = new Date(year_sel, month_int, 1); //构造一个日期
        GregorianCalendar cal = new GregorianCalendar(); //创建一个Calendar
实例
        cal.setTime(dt);
        String week[] = { "日", "一","二", "三", "四", "五", "六" };
        int day = 0;
        int day_week = 0;
        for (int i = 0; i < 7; i++) {
            Label_day[i].setText(week[i]);
        }
        //月份
        if (month_int == 0||month_int == 2 ||month_int == 4 ||
          month_int == 6 ||
          month_int == 9 ||month_int == 11)
        {
            day = 31;
        }
        else if (month_int == 3 ||month_int == 5 || month_int == 7||
         month_int == 8 ||month_int == 10|| month_int == 1)
        {
            day = 30;
        }
        else
        {
             if (cal.isLeapYear(year_sel))
             {
              day = 29;
             }
             else
             {
              day = 28;
             }
        }
        day_week = 7 + dt.getDay();
        int count = 1;
        for (int i = day_week; i < day_week + day; count++, i++)
        {
            if (i % 7 == 0 ||i == 13||i == 20||i == 27||
                  i == 48 ||i == 34 ||i == 41)
            {
                if (i == day_week + now_date.getDate() - 1)
                {
                    Label_day[i].setForeground(Color.blue);
```

```
                Label_day[i].setText(count + "");
            }
            else
            {
                Label_day[i].setForeground(Color.red);
                Label_day[i].setText(count + "");
            }
        }
        else
        {
            if (i == day_week + now_date.getDate() - 1)
            {
                Label_day[i].setForeground(Color.blue);
                 Label_day[i].setText(count + "");
            }
            else
            {
                Label_day[i].setForeground(Color.black);
                Label_day[i].setText(count + "");
            }
        }
    }
    if (day_week == 0)
    {
        for (int i = day; i < 49; i++)
        {
            Label_day[i].setText(" ");
        }
    }
    else
    {
        for (int i = 7; i < day_week; i++)
        {
            Label_day[i].setText(" ");
        }
        for (int i = day_week + day; i < 49; i++)
        {
            Label_day[i].setText("  ");
        }
    }
}
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == Year || e.getSource() == Month) {
        bool = false;
        this.setDay();
    }
```

```
        }
    }
}
```

修改后的实验结果：





5、编辑、编译、运行下面 Applet 程序

```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import java.util.*;
import javax.swing.*;
public class BounceThread
{
    public static void main(String[] args)
    {
        JFrame frame = new BounceFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.show();
    }
}
class BounceFrame extends JFrame
{
    public BounceFrame()
    {
        setSize(WIDTH, HEIGHT);
        setTitle("BounceThread");
        Container contentPane = getContentPane();
        canvas = new BallCanvas();
        contentPane.add(canvas, BorderLayout.CENTER);
        JPanel buttonPanel = new JPanel();
        addButton(buttonPanel, "Start",
            new ActionListener()
                {
                    public void actionPerformed(ActionEvent evt)
                    {
                        addBall();
                    }
                });
        addButton(buttonPanel, "Close",
            new ActionListener()
                {
                    public void actionPerformed(ActionEvent evt)
                    {
                        System.exit(0);
                    }
                });
```

```
        contentPane.add(buttonPanel, BorderLayout.SOUTH);
    }

    public void addButton(Container c, String title,
        ActionListener listener)
    {
        JButton button = new JButton(title);
        c.add(button);
        button.addActionListener(listener);
    }

    public void addBall()
    {   Ball b = new Ball(canvas);
        canvas.add(b);
        BallThread thread = new BallThread(b);
        thread.start();
    }

    private BallCanvas canvas;
    public static final int WIDTH = 450;
    public static final int HEIGHT = 350;
}
class BallThread extends Thread
{
    public BallThread(Ball aBall)
    {
   b = aBall;
    }
    public void run()
    {
        try
        {
            for (int i = 1; i <= 1000; i++)
            {
                b.move();
                sleep(5);
            }
        }
        catch (InterruptedException exception) { }
    }
```

```
    private Ball b;
}
class BallCanvas extends JPanel
{
    public void add(Ball b)
    {
        balls.add(b);
    }
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D)g;
        for (int i = 0; i < balls.size(); i++)
        {
            Ball b = (Ball)balls.get(i);
            b.draw(g2);
        }
    }
    private ArrayList balls = new ArrayList();
}
class Ball
{
    public Ball(Component c)
{
canvas = c;
}
    public void draw(Graphics2D g2)
    {
        g2.fill(new Ellipse2D.Double(x, y, XSIZE, YSIZE));
    }
    public void move()
    {
        x += dx;
        y += dy;
        if (x < 0)
        {
            x = 0;
            dx = -dx;
        }
        if (x + XSIZE >= canvas.getWidth())
```

```
            {
                x = canvas.getWidth() - XSIZE;
                dx = -dx;
            }
            if (y < 0)
            {
                y = 0;
                dy = -dy;
            }
            if (y + YSIZE >= canvas.getHeight())
            {
                y = canvas.getHeight() - YSIZE;
                dy = -dy;
            }
            canvas.repaint();
        }
        private Component canvas;
        private static final int XSIZE = 15;
        private static final int YSIZE = 15;
        private int x = 0;
        private int y = 0;
        private int dx = 2;
        private int dy = 2;
    }
```

说明与要求：

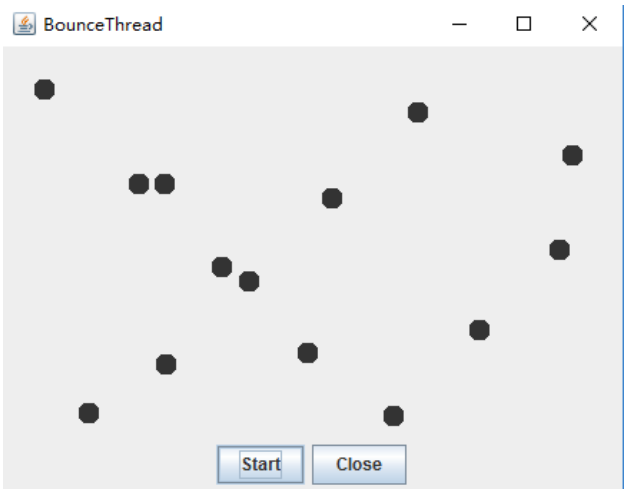
（1）编辑运行上面的程序，理解基于多线程的动画编程

（2）修改程序，**实现有 2 个或多个不同颜色的小球在运动的动画程序，并给出源程序与程序运行结果**

【实验结果与分析】

（1）

实验结果：



实验分析：

本程序主函数创建 BounceFrame 类，BounceFrame 继承自 Jframe，中间实现了界面的设计，其中 addButton 函数实现了每次按下"Start"，新建一个球，用多线程 BallThread 实现各个小球的独立运动。BallCanvas 类实现每次小球移动到新的位置重新绘制出小球。Ball 类存入小球的基本数据。

（2）

实验程序：

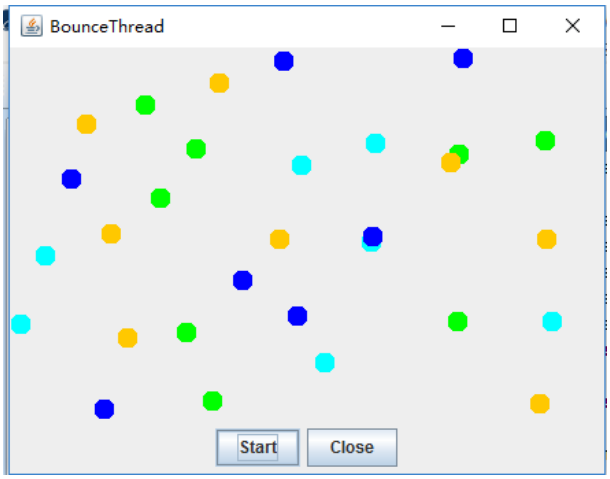
```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import java.util.*;
import javax.swing.*;
public class BounceThread
{
   public static void main(String[] args)
   {
      JFrame frame = new BounceFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.show();
   }
}

class BounceFrame extends JFrame
{
   static int i = 0;
    public BounceFrame()
   {
     setSize(WIDTH, HEIGHT);
      setTitle("BounceThread");
      Container contentPane = getContentPane();
      canvas = new BallCanvas();
      contentPane.add(canvas, BorderLayout.CENTER);
```

```
      JPanel buttonPanel = new JPanel();
      addButton(buttonPanel, "Start",
         new ActionListener()
            {
               public void actionPerformed(ActionEvent evt)
               {
                  addBall((i++)%4); //判断第几个线程
               }
            });
      addButton(buttonPanel, "Close",
         new ActionListener()
            {
               public void actionPerformed(ActionEvent evt)
               {
                  System.exit(0);
               }
            });
      contentPane.add(buttonPanel, BorderLayout.SOUTH);
   }

   public void addButton(Container c, String title,
      ActionListener listener)
   {
      JButton button = new JButton(title);
      c.add(button);
      button.addActionListener(listener);
   }

   public void addBall(int _id)
   {  Ball b1 = new Ball(canvas, _id);
      canvas.add(b1);
      BallThread thread1 = new BallThread(b1);
      thread1.start();
   }

   private BallCanvas canvas;
   public static final int WIDTH = 450;
   public static final int HEIGHT = 350;
}

class BallThread extends Thread
{
   public BallThread(Ball aBall)
   {
     b = aBall;
   }
```

```
   public void run()
   {
      try
      {
         for (int i = 1; i <= 1000; i++)
         {
            b.move();
            sleep(5);
         }
      }
      catch (InterruptedException exception) { }
   }
   private Ball b;
}

class BallCanvas extends JPanel
{
    public void add(Ball b)
   {
      balls.add(b);

   }
   public void paintComponent(Graphics g)
   {
      super.paintComponent(g);
      Graphics2D g2 = (Graphics2D)g;

      for (int i = 0; i < balls.size(); i++)
      {
         Ball b = (Ball)balls.get(i);
         b.draw(g2);
      }
   }
   private ArrayList balls = new ArrayList();
}


class Ball
{
   int id = 0;
   public Ball(Component c, int _id){
       canvas = c;
       id = _id;
   }
   public void draw(Graphics2D g2)
   {
```

```
      g2.fill(new Ellipse2D.Double(x, y, XSIZE, YSIZE));
      if(id == 0)g2.setColor(Color.orange); //设置颜色
      else if(id == 1)g2.setColor(Color.cyan);
      else if(id == 2)g2.setColor(Color.blue);
      else g2.setColor(Color.green);
   }
   public void move()
   {
      x += dx;
      y += dy;
      if (x < 0)
      {
         x = 0;
         dx = -dx;
      }
      if (x + XSIZE >= canvas.getWidth())
      {
         x = canvas.getWidth() - XSIZE;
         dx = -dx;
      }
      if (y < 0)
      {
         y = 0;
         dy = -dy;
      }
      if (y + YSIZE >= canvas.getHeight())
      {
         y = canvas.getHeight() - YSIZE;
         dy = -dy;
      }
      canvas.repaint();
   }
   private Component canvas;
   private static final int XSIZE = 15;
   private static final int YSIZE = 15;
   private int x = 0;
   private int y = 0;
   private int dx = 2;
   private int dy = 2;
}
```


实验结果:



6、编辑、编译、运行下面动画程序

```
import java.applet.Applet;
import java.awt.*;
import java.awt.image.MemoryImageSource;
import java.util.Random;
public class Fire2 extends Applet implements Runnable
{
    final int ITEM_COUNT = 10000;  // 粒子数
    final int PIXEL_LIFE = 100;    // 粒子生命期最大值
    private Thread thread;
    int[] particleLife;            // 生命期
    int[] particleColor;           // 粒子颜色
    byte[] particleState;          // 粒子运动状态
    int[] liftPosX, liftPosY;      // 上升过程位置
    double[] explodePosX, explodePosY;  // 爆炸过程位置
    double[] vx, vy;               // 爆炸过程的速度分量
    int scrWidth, scrHeight;       // 屏幕宽高
    int[] pixels;                  // 屏幕像素（用来显示粒子）
    int pixelCount;                // 粒子数量
    MemoryImageSource offImage;    // 用来显示粒子的内存位图
    Image dbImage;                 // 用来显示的图像
    Graphics dispGraph;
    Image mapImage, dispImage;
    Random random;
    boolean isInited = false;
    public Fire2()
    {
        particleLife = new int[ITEM_COUNT];
        particleColor = new int[ITEM_COUNT];
        particleState = new byte[ITEM_COUNT];
        liftPosX = new int[ITEM_COUNT];
```

```
        liftPosY = new int[ITEM_COUNT];
        explodePosX = new double[ITEM_COUNT];
        explodePosY = new double[ITEM_COUNT];
        vx = new double[ITEM_COUNT];
        vy = new double[ITEM_COUNT];
        random = new Random();
        isInited = true;
    }
    public void init()
    {
        particleLife = new int[ITEM_COUNT];
        particleColor = new int[ITEM_COUNT];
        particleState = new byte[ITEM_COUNT];
        liftPosX = new int[ITEM_COUNT];
        liftPosY = new int[ITEM_COUNT];
        explodePosX = new double[ITEM_COUNT];
        explodePosY = new double[ITEM_COUNT];
        vx = new double[ITEM_COUNT];
        vy = new double[ITEM_COUNT];
        scrWidth = size().width;
        scrHeight = size().height;
        pixelCount = scrWidth * scrHeight;
        pixels = new int[pixelCount];// 很重要，不然以后的操作很慢
        for (int i=0; i< pixelCount; ++i)
        pixels[i] = 0xff000000;
        offImage = new MemoryImageSource(scrWidth, scrHeight, pixels, 0, scrWidth);
        offImage.setAnimated(true);
        dbImage = createImage(offImage);
        mapImage = getImage(getDocumentBase(), "smile.jpg");
        dispImage = createImage(scrWidth, scrHeight);
        dispGraph = dispImage.getGraphics();
    }
    public void start()
    {
        if (thread == null) thread = new Thread(this);
        thread.start();
    }
    public void stop()
    {
        if (thread != null){
            thread.stop();
            thread = null;
        }
    }
    public boolean mouseDown(Event e, int x, int y)
    { //产生一簇粒子
        double speed;
        byte r, g, b;
```

```
        r = (byte)(Math.random() * 0xff);
        g = (byte)(Math.random() * 0xff);
        b = (byte)(Math.random() * 0xff);
        int color = r << 16 | g << 8 | b | 0xff000000;
        for (int i=0; i< ITEM_COUNT; ++i)
        {
            if (particleLife[i] != 0) continue;
            particleLife[i] = (int)(Math.random() * PIXEL_LIFE);
            speed = Math.random() * 6.28D;
            vx[i] = Math.cos(speed);
            vy[i] = Math.sin(speed);
            liftPosX[i] = x;
            liftPosY[i] = scrHeight - 5;
            explodePosX[i] = x;
            explodePosY[i] = y;
            particleColor[i] = color;
            particleState[i] = 1;
        }
        return true;
    }
    private void setPixel(int x, int y, int color)
    {   // 粒子运动控制
        pixels[y * scrWidth + x] = color;
    }
    private void moveParticles()
    {
        for (int i=0; i< ITEM_COUNT; ++i)
        {
            switch(particleState[i])
            {
              case 1: // 粒子上升运动
                liftPosY[i] -= 5;
                if (liftPosY[i] <= explodePosY[i]) particleState[i] = 2;
                else if (Math.random() > 0.9)
                {
                    int randX = (int)(Math.random() * 2.0);
                                        // 在上升过程制造摇晃
                    int randY = (int)(Math.random() * 5.0);
                    setPixel(liftPosX[i] + randX, liftPosY[i] + randY, -1);
                }
                break;
              case 2: // 粒子爆炸过程
                particleLife[i] --;
                vy[i] += Math.random() / 20D; // 使得粒子的下降不一样
                vx[i] += Math.random() / 50D;
                explodePosX[i] += vx[i];
                explodePosY[i] += vy[i];
                if (particleLife[i] == 0 || explodePosX[i] < 5 || explodePosX[i] > scrWidth -
```

```
                    5 || explodePosY[i] < 5 || explodePosY[i] > scrHeight - 5)
            {
                particleState[i] = 0;
                particleLife[i] = 0;
            }
            else setPixel((int)explodePosX[i], (int)explodePosY[i], particleColor[i]);
            break;
         default: break;
        }
    }
}
private void soften()
{//图像柔化
    byte avgR, avgG, avgB;
    int[] color = new int[9];
    for (int i=scrWidth + 1; i< pixelCount - scrWidth * 2; ++i)
    {
        int sumR = 0, sumG = 0, sumB = 0;
        color[0] = pixels[i-scrWidth-1];
        color[1] = pixels[i-scrWidth];
        color[2] = pixels[i-scrWidth+1];
        color[3] = pixels[i-1];
        color[4] = pixels[i];
        color[5] = pixels[i+1];
        color[6] = pixels[i+scrWidth-1];
        color[7] = pixels[i+scrWidth];
        color[8] = pixels[i+scrWidth+1];
        for (int j=0; j< 9; ++j)
        {
            sumR += (color[j] & 0xff0000) >> 16;
            sumG += (color[j] & 0xff00) >> 8;
            sumB += color[j] & 0xff;
        }
        avgR = (byte)(sumR / 9);
        avgG = (byte)(sumG / 9);
        avgB = (byte)(sumB / 9);
        pixels[i] = 0xff000000 | avgR << 16 | avgG << 8 | avgB;
    }
}
public void run()
{
    while (!isInited)
    {
        try
        {
          Thread.sleep(200);
        }
        catch (InterruptedException e){}
```

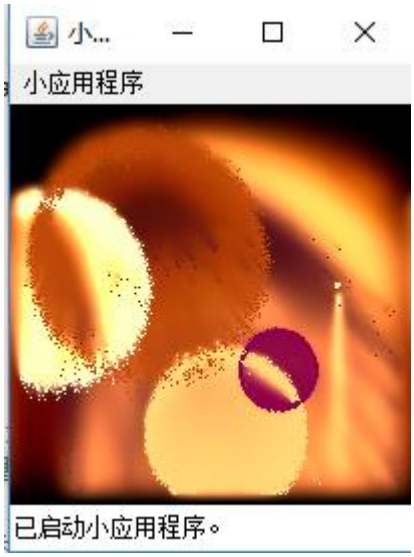
```
        }

        while (thread == Thread.currentThread())
        {
            soften();
            moveParticles();
            offImage.newPixels(0, 0, scrWidth, scrHeight);
            try
            {
              Thread.sleep(10);
            } catch (InterruptedException e){}
        }
    }
    public void update(Graphics g)
    {
        paint(g);
    }
    public void paint(Graphics g)
    {
        dispGraph.drawImage(mapImage, 0, 0, this);
        dispGraph.drawImage(dbImage, 0, 0, this);
        g.drawImage(dispImage, 0, 0, this);
    }
}
```


说明：1、该程序的运行，点击鼠标，可产生漂亮的烟花动画；

2、通过调试程序，进一步理解掌握动画编程

【实验结果与分析】

1.实验结果：



2.实验分析：

mouseDown 函数检测鼠标点击位置，原型是 boolean mouseDown(Event e, int x, int y)，三个参数分别是鼠标响应时间，烟花爆炸的横纵坐标。

setPixel 函数控制粒子上升运动和爆炸过程，其中使用随机数产生偏移位置使得烟花出现上升时摇晃的效果。

Run 函数是用来实现多线程控制产生不同的烟花。