LẬP TRÌNH ĐA TUYẾN (MULTITHREADING)

PGS.TS. Huỳnh Công Pháp

I. Tóm tắt lý thuyết

- Mỗi chương trình đều có một tuyến xuyên suốt từ khi chương trình bắt đầu cho đến khi chương trình kết thúc, đó là tuyến chính (main thread).
- Chương trình đa tuyến (thread) là chương trình gồm có nhiều tuyến thực thi đồng thời. Mỗi tuyến thực hiện một công việc nào đó.
- Do vậy, một tuyến (Thread) được hiểu như một nhánh của chương trình, chạy đồng thời với tuyến chính và các tuyến khác.
- Mỗi tuyến gồm 4 trạng thái chính: vừa mới tạo (born), hoạt động (runnable), dừng (blocked) và hủy (dead).
- Một tuyến vừa mới tạo, chưa hoạt động cho đến khi phương thức start() được gọi.
- Nội dung hoạt động của một tuyến được thực hiện trong phương thức run(), được gọi tự động bởi phương thức start().
- Trong Java, tạo một tuyến bằng cách tạo một đối tượng của lớp Thread, có hai cách:
    - Tạo lớp con của lớp Thread, định nghĩa công việc của tuyến trong phương thức run(), rồi tạo đối tượng của lớp vừa tạo.
    - Tạo lớp hiện thực interface Runnable, định nghĩa công việc của tuyến trong phương thức run(), rồi tạo đối tượng Thread với đối số là đối tượng của lớp vừa khai báo.
- Các tuyến có cùng độ ưu tiên sẽ thực hiện đồng thời. Tuyến có độ ưu tiên thấp sẽ thực hiện sau các tuyến có độ ưu tiên cao hơn.
- Để ngăn cấm không cho nhiều tuyến đồng thời thực thi một phương thức hay đoạn lệnh, Java cung cấp từ khóa synchronized để thiết lập chế độ thực thi đồng bộ. Khi một tuyến thực thi một phương thức hay một đoạn lệnh đã được thiết lập đồng bộ thì các tuyến khác phải chờ cho đến tuyến đó hoàn tất.
- Phương thức sleep() được dùng để dừng một tuyến trong một thời gian cố định, còn phương thức wait() được dùng để dừng một tuyến cho đến khi được đánh thức bằng phương thức notify() hay notifyAll().

II. Bài tập thực hành mẫu

1. Viết chương trình tạo 2 tuyến, tuyến thứ nhất in các số lẻ từ 1 đến 9, tuyến thứ 2 in các số chẵn từ 2 đến 10.

PGS.TS. Huỳnh Công Pháp

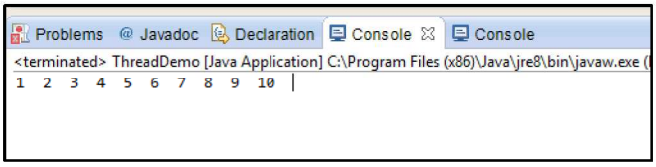
Đáp án 1:

```
    class Thread_1 extends Thread
    {

/*định nghĩa lại hàm run()của lớp Thread để in ra màn hình các số lẻ từ 1
đến 9*/
     public void run()
          {
             for(int i = 1;i<10;i+=2)
             {
                System.out.print(i+"  ");
                try{

       /*tạm dừng 100 mili giây để tuyến khác hoạt động*/
                    Thread.sleep(100);
                }
             catch(Exception e){}
          }
          }
    }
    class Thread_2 extends Thread
    {

    /*định nghĩa lại hàm run()của lớp Thread
          để in ra màn hình các số chẵn từ 2 đến 10*/
       public void run()
          {
             for(int i = 2;i<=10;i+=2)
             {
                System.out.print(i+"  ");
                try{

      /*tạm dừng 100 mili giây để tuyến khác hoạt động*/
                 Thread.sleep(100);
                }
             catch(Exception e){}
          }
          }
    }
    public class ThreadDemo
    {
       public static void main(String[] arg)
       {

     /*tạo 2 tuyến con */
              Thread t1 = new Thread_1();
              Thread t2 = new Thread_2();

     /*khởi động tuyến và tự động gọi hàm run()*/
              t1.start();
              t2.start();
       }
     }
```

Đáp án 2:

```
class MyThread extends Thread
{

/*sử dụng start để nhận giá trị khởi đầu, nếu start =1 thì in các số lẻ,
nếu start=2 thì in các số chẵn*/
  int start;

  public MyThread(int s)
   {

  /*Khởi tạo giá trị cho start*/
   start =s;
   }

/*Hàm static này sẽ được dùng chung cho tất cả các tuyến*/
  public static void go(int s)
  {

/*tùy vào giá trị s (1 hoặc 2) hàm go(...) sẽ in ra màn hình các số lẻ hay
chẵn từ 1 đến 10*/
    for(int i = s;i<10;i+=2)
      {
         System.out.print(i+"  ");
         try{

        /*tạm ngừng tuyến hiện tại để tuyến khác hoạt động*/
              Thread.sleep(100);
         }
      catch(Exception e){}
   }

  }

/*hàm run() sẽ được gọi tự động bởi hàm start()*/
  public void run()
   {
     go(start);
   }
}

public class ThreadDemo
{
 public static void main(String[] arg)
  {

  /*Tạo 2 tuyến con, tuyến t1 sẽ in các số lẻ từ 1 đến 9*/
     Thread t1 = new MyThread(1);

   /*Tuyến t2 sẽ in các số chẵn từ 2 đến 10*/
     Thread t2 = new MyThread(2);

   /*khởi động 2 tuyến và hàm run() sẽ được gọi tự động*/
     t1.start();
     t2.start();
   }
 }
```

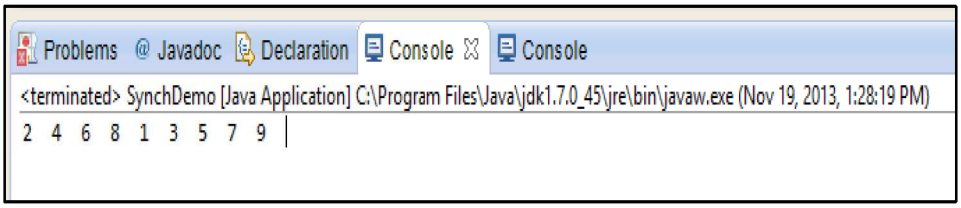
Kết quả chương trình:



2. Sửa lại đáp án 2 ở bài tập 1 bằng cách thêm từ khóa ***synchronized*** trước hàm go(). Kết quả sẽ minh chứng khi một tài nguyên (cụ thể phương thức go()) được thiết lập truy cập đồng bộ sẽ không được phép chia sẽ đồng thời cho nhiều tuyến.

```
    class MyThread extends Thread
    {
      int start;
      public MyThread(int s)
       {
       start =s;
       }

/*Thêm từ khóa synchronized sẽ ngăn cấm nhiều tuyến thực thi hàm go() đồng
thời. Các tuyến lần lược thực thi cho đến hết hàm go()*/
      public static synchronized void go(int s)
      {
       for(int i = s;i<10;i+=2)
         {
          System.out.print(i+"  ");
          try{
               Thread.sleep(100);
             }
          catch(Exception e){}
       }

      }
     public void run()
     {
       go(start);
      }
    }

public class SynchDemo
 {
 public static void main(String[] arg)
  {

  /*Tạo 2 tuyến con, tuyến t1 sẽ in các số lẻ từ 1 đến 9*/
     Thread t1 = new MyThread(1);

   /*Tuyến t2 sẽ in các số chẵn từ 2 đến 10*/
     Thread t2 = new MyThread(2);

   /*khởi động 2 tuyến và hàm run() sẽ được gọi tự động*/
     t1.start();
     t2.start();
  }
}
```

Kết quả chương trình:



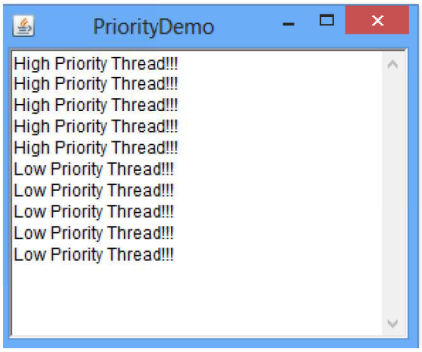
3. Viết chương trình minh chứng một tuyến có độ ưu tiên cao hơn sẽ thực thi trước các tuyến có độ ưu tiên thấp hơn.

```
import java.awt.*;
import java.awt.event.*;
public class PriorityDemo extends Frame {

/* 2 tuyến có tên là high và low*/
   private HighThread high;
   private LowThread low;

/*dùng để hiển thị kết quả*/
   private TextArea output;

   public PriorityDemo()
   {

 /*gọi phương thức khởi tạo của lớp Frame, thiết lập tiêu đề cho
     Frame*/
      super( "PriorityDemo" );

  /*tạo một TextArea có kích thước 10x20 để hiển thị kết quả*/
      output = new TextArea( 10, 20 );
      add( output );

  /*thiết lập giao diện kích thước 250x200*/
      setSize( 250, 200 );
      setVisible( true );

 /*tạo và khởi động tuyến có độ ưu tiên cao hơn*/
      high = new HighThread( output );
      high.start();

/*tạo và khởi động tuyến có độ ưu tiên thấp hơn*/
      low = new LowThread( output );
      low.start();
   }


public static void main( String args[] )
   {

   /*Khởi động chương trình*/
      PriorityDemo app = new PriorityDemo();

/*Bắt sự kiện cho phép đóng cửa sổ (bằng cách ấn dấu X ở cửa sổ)*/
      app.addWindowListener(
         new WindowAdapter() {
```

```
                public void windowClosing( WindowEvent e )
                {
                   System.exit( 0 );
                }
             }
          );

       }
    }

/*tạo tuyến và thiết lập độ ưu tiên cao hơn*/
    class HighThread extends Thread {

    /*Dùng để nhận đối tượng TextArea ở chương trình chính để hiển thị kết
    quả*/
       private TextArea display;

       public HighThread( TextArea a )
       {

    /*Nhận đối tượng TextArea từ chương trình chính truyền đến*/
       display = a;

    /*thiết lập độ ưu tiên cao nhất*/
       setPriority( Thread.MAX_PRIORITY );
       }

    /*định nghĩa lại hàm run() của lớp Thread để in kết quả*/
    public void run()
      {

    /*Chèn thêm 5 chuỗi High...vào TextArea*/
      for ( int x = 1; x <= 5; x++ )
       display.append( "High Priority Thread!!!\n" );
       }
    }

    /*tạo tuyến và thiết lập độ ưu tiên thấp*/
    class LowThread extends Thread {

    /*Dùng để nhận đối tượng TextArea ở chương trình chính để hiển thị kết
    quả*/
       private TextArea display;

       public LowThread( TextArea a )
       {

    /*Nhận đối tượng TextArea từ chương trình chính truyền đến*/
          display = a;

    /*thiết lập độ ưu tiên thấp nhất*/
          setPriority( Thread.MIN_PRIORITY );
       }

    /*định nghĩa lại hàm run() của lớp Thread để in kết quả*/
       public void run()
       {

    /*Chèn thêm 5 chuỗi Low...vào TextArea*/
```

```
         for ( int y = 1; y <= 5; y++ )
            display.append( "Low Priority Thread!!!\n" );
      }

   }
```

Kết quả chương trình:



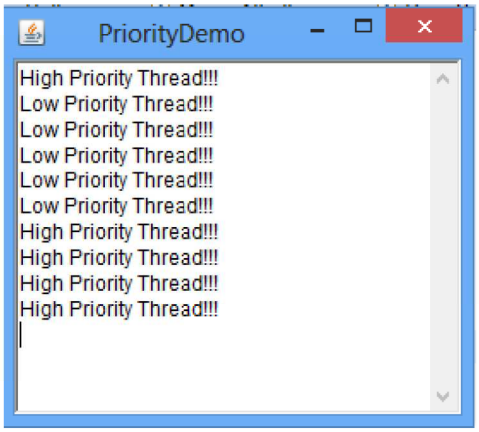
4. Viết chương trình minh chứng khi tuyến có độ ưu tiên cao hơn rơi vào trạng thái tạm dừng (sử dụng phương thức sleep), thì các tuyến có độ ưu tiên thấp hơn sẽ thực thi.

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

public class PriorityDemo extends Frame {

/* 2 tuyến có tên là high và low*/
   private HighThread high;
   private LowThread low;

/*dùng để hiển thị kết quả*/
   private TextArea output;

   public PriorityDemo()
   {
      super( "PriorityDemo" );
      output = new TextArea( 10, 20 );
      add( output );
      setSize( 250, 200 );
      setVisible( true );

/*tạo và khởi động tuyến có độ ưu tiên cao hơn*/
      high = new HighThread( output );
      high.start();

/*tạo và khởi động tuyến có độ ưu tiên thấp hơn*/
      low = new LowThread( output );
      low.start();
   }

   public static void main( String args[] )
   {
      PriorityDemo app = new PriorityDemo();
```

```
   /*Bắt sự kiện cho phép đóng cửa sổ (bằng cách ấn dấu X ở cửa sổ)*/
        app.addWindowListener(
            new WindowAdapter() {
               public void windowClosing( WindowEvent e )
               {
                  System.exit( 0 );
               }
            }
         );

      }
   }

/*tạo tuyến và thiết lập độ ưu tiên cao hơn*/
   class HighThread extends Thread {
      private TextArea display;

      public HighThread( TextArea a )
      {
         display = a;

      /*thiết lập độ ưu tiên cao nhất*/
         setPriority( Thread.MAX_PRIORITY );
      }

      public void run()
      {
         for ( int x = 1; x <= 5; x++ )
         {
          display.append( "High Priority Thread!!!\n" );
         try{

   /*cho tuyến có độ ưu tiên cao tạm dừng để các tuyến khác thực hiện*/
             Thread.sleep(10);
         }catch(Exception e){}
      }
      }
   }

/*tạo tuyến và thiết lập độ ưu tiên thấp*/
   class LowThread extends Thread {
      private TextArea display;

      public LowThread( TextArea a )
      {
         display = a;

   /*thiết lập độ ưu tiên thấp nhất*/
         setPriority( Thread.MIN_PRIORITY );
      }

      public void run()
      {
         for ( int y = 1; y <= 5; y++ )
            display.append( "Low Priority Thread!!!\n" );
      }
```

```
}
```

Kết quả chương trình:



5. Viết chương trình tạo ra 3 tuyến, mỗi tuyến thực hiện tính giá trị cho một phần tử của biểu thức bên dưới, chương trình chính chờ 3 tuyến hoàn thành và lấy kết quả để tính biểu thức tổng:

$S = F1(n) + F2(n) + F3(x,n)$.

Trong đó:

$F1(n) = 1 * 2 * 3... * n$

$F2(n) = 1 + 2 + 3...+ n$

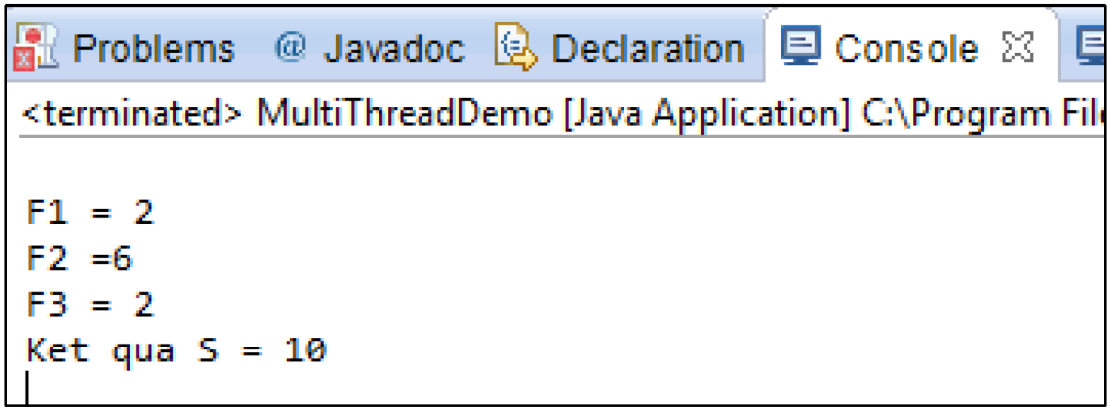
$F3(x,n) = x^1 + x^2 + x^3 ....+ x^n$

```
public class MultiThreadDemo  {

  /*tao 3 tuyến */
       FacThread t1 = new FacThread(2);
       SumThread t2 = new SumThread(3);
       SumPowThread t3 = new SumPowThread(2,1);

       public MultiThreadDemo()
       {
         //khởi động 3 tuyến
             t1.start();
             t2.start();
             t3.start();
             try{

         //mỗi tuyến thực hiện cho đến hết
            t1.join();
             t2.join();
             t3.join();

        /*tính kết quả tổng khi các tuyến con đã hoàn tất (nhờ hàm join())
         và in ra màn hình*/
          long S = t1.getResult()+t2.getResult()+t3.getResult();
```

```
            System.out.println("  \nKet qua = "+S);

         }catch (Exception e)
                  {
                        e.printStackTrace();
                  }
      }

      public static void main(String[] a)
      {

    /*khởi động chương trình*/
       new MultiThreadDemo();

      }
}

/*Định nghĩa tuyến tính F1(n)*/
class FacThread extends Thread
{
      long gt=1;
      int n;

      public FacThread(int k)
      {

   /*nhận giá trị n được truyền vào từ chương trình chính*/
       n=k;
      }

 /*Định nghĩa lại hàm run()*/
      public void run()
      {

     /*Tính F1(n)*/
        for (int i = 2;i<=n;i++)
         {
          gt *=i;
         }
         System.out.print("\nF1 = "+ gt);
      }

  /*Hàm trả kết quả mà tuyến tính được*/
      public long getResult()
      {
            return gt;
      }
}

/*Định nghĩa tuyến tính F2(n)*/
class SumThread extends Thread
{
      long S=0;
      int n;

      public SumThread(int k)
      {

   /*nhận giá trị n được truyền vào từ chương trình chính*/
```

```
        n=k;
       }

 /*Định nghĩa lại hàm run()*/
       public void run()
       {

     /*Tính F2(n)*/
        for (int i = 1;i<=n;i++)
         {
         S +=i;
         }
        System.out.print(" \nF2 ="+S);
       }

   /*Hàm trả kết quả mà tuyến tính được*/
       public long getResult()
       {
              return S;
       }
}

/*Định nghĩa tuyến tính F3(x,n)*/
class SumPowThread extends Thread
{
       long S=0;
       int x,n;

       public SumPowThread(int y,int k)
       {

    /*nhận giá trị x, n được truyền vào từ chương trình chính*/
         x=y;
         n=k;
       }

 /*Định nghĩa lại hàm run()*/
       public void run()
       {

     /*Tính F3(x,n)*/
        for (int i = 1;i<=n;i++)
         {

     /*sử dụng hàm tính mũ pow của lớp Math*/
          S +=Math.pow(x, i);
         }
        System.out.print(" \nF3 = "+S);
       }

/*Hàm trả kết quả mà tuyến tính được*/
       public long getResult()
       {
              return S;
       }
}
```

Kết quả chương trình:

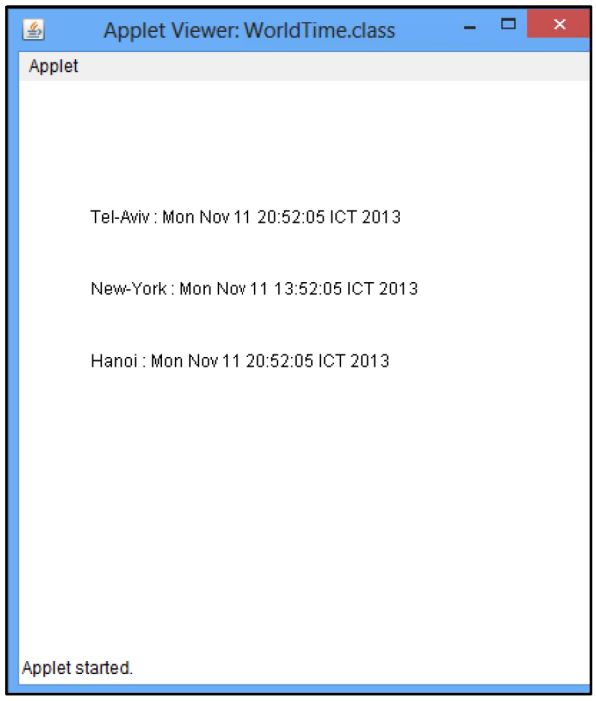


6. Viết chương trình Applet hiển thị thời gian ở 3 thủ đô của 3 quốc gia khác nhau.

```
import java.awt.*;
import java.applet.*;
import java.util.*;
public class WorldTime extends Applet implements Runnable
{

/*lớp Date của gói java.util cho phép lấy thông tin thời gian*/
 private Date toDay = new Date();

/*Tuyến con có tên th*/
 private Thread th = null;

 public void start()
   {
   if(th==null)
    {

  /*Tạo tuyến bằng đối số là đối tượng của
   lớp dẫn xuất interface Runnable*/
     th = new Thread(this);

 /*khởi động tuyến và thực thi hàm run()*/
     th.start();
     }
   }

/*Định nghĩa lại hàm run() của interface Runnable*/
 public void run()
   {

 /*Cho tuyến thực thi vô thời hạn*/
   while(true)
    {

/*Lấy thông tin thời gia*/
    toDay = new Date();

/*xóa màn hình và gọi lại hàm paint(...) để cập nhật lại màn hình*/
    repaint();
```

```
    try
      {

/*Dừng lại 1 giây (1000 mili giây) rồi cập nhật lại thời gian*/
      Thread.sleep(1000);
       }
     catch(Exception e) {}
     }

}
 public void paint(Graphics g)
   {

   /*Hiển thị thời gian tại thành phố Tel-Aviv*/
      g.drawString("Tel-Aviv : " + toDay,50,100);

   /*Hiển thị thời gian tại thành phố New-York (chậm hơn 7h)*/
      toDay.setTime(toDay.getTime()-7*3600*1000);
      g.drawString("New-York : " + toDay,50,150);

    /*Hiển thị thời gian tại Hà Nội (Nhanh hơn 7h)*/
      toDay.setTime(toDay.getTime()+7*3600*1000);
      g.drawString("Hanoi : " + toDay,50,200);
  }
}
```

Kết quả chương trình:

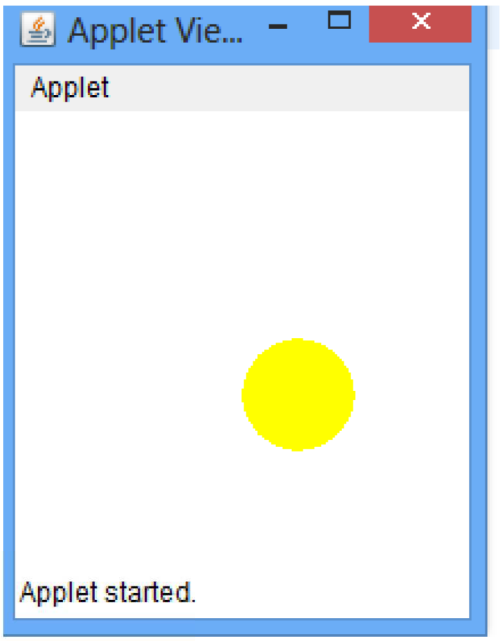


7. Viết chương trình biểu diễn hệ thống đèn giao thông, mỗi thời điểm hệ thống chỉ hiển thị một trong 3 màu đỏ, vàng, xanh.

```
    import javax.swing.*;
```

```
import java.awt.*;
public class TrafficLight extends JApplet implements Runnable {

  /*khoi tao den xanh*/
    String light ="green";

    public void init()
    {

    //tạo 3 tuyến
        Thread one=new Thread(this);
        Thread two=new Thread(this);
        Thread three=new Thread(this);

   //đặt tên cho 3 tuyến
        one.setName("red");
        two.setName("yellow");
        three.setName("green");

  //khởi động 3 tuyến
        one.start();
        two.start();
        three.start();

    }

    public void paint(Graphics g)
    {

      //Thiết lập màu tương ứng tuyến đang hoạt động
          if (light.equals("green")) g.setColor(Color.green);
          if (light.equals("yellow")) g.setColor(Color.yellow);
          if (light.equals("red")) g.setColor(Color.red);


       //vẽ đèn giao thông với màu tương ứng
          g.fillOval(100, 100, 50, 50);
    }

    public void showLight()
    {

   //Lấy tên tuyến đang hoạt động
        light= Thread.currentThread().getName();

    //xóa màn hình và gọi lại hàm paint(...)
        repaint();
    }

    public void run()
    {

      while (true)
        {

      //Gọi hàm để vẽ đèn giao thông
        showLight();

         try {
```

```
        //hiển thị đèn với màu hiện tại 500 mili giây
             Thread.sleep(500);

              }
            catch (InterruptedException e)
                  {
                  e.printStackTrace();
                  }
          }
        }
    }
```

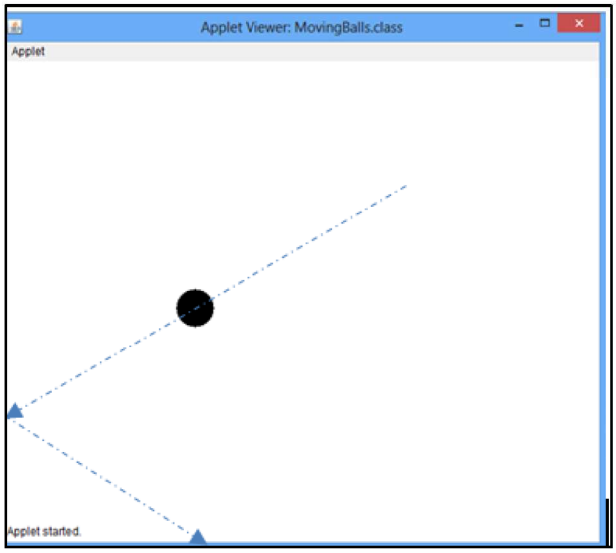


8. Viết chương trình hiển thị 01 hình tròn tô màu di chuyển và phản xạ theo quy luật di chuyển và phản xạ của quả bóng bi da.

```
import java.awt.*;
import java.applet.*;
public class MovingBall extends Applet implements Runnable
{

//Tuyến con có tên t
 Thread t;

//Vị trí khởi tạo của quả bóng
 int x=34,y=14;

//Bước di chuyển
 int dx=5, dy=5;

/*Hàm init được gọi tự động đầu tiên khi Applet chạy*/
public void init()
{

/*tạo và khởi động tuyến con với đối số là đối tượng của lớp dẫn xuất
interface Runnable*/
```

```
 t=new Thread(this);

/*khởi động tuyến và thực thi hàm run()*/
 t.start();
}

/*định nghĩa hàm run() của interface Runnable*/
public void run()
{
 while(true)
   {

/*nếu quả bóng va vào các đường biên*/
    if (x+dx>this.getWidth()||x+dx<0) dx=-dx;
    if (y+dy>this.getHeight()||y+dy<0) dy=-dy;

/*Thay đổi vị trí quả bóng*/
              x=x+dx;
              y=y+dy;

/*Xóa màn hình và gọi lại hàm paint để vẽ lại quả bóng tại vị trí mới*/
              repaint();

              try
              {

/* Dừng lại 10 mili giây để nhìn quả bóng trước khi xóa và vẽ lại */
              Thread.sleep(10);
              }
              catch(Exception e)
              {}
       }
}

public void paint(Graphics g)
{

/* Vẽ quả bóng tại vị trí hiện tại */
  g.fillOval(x,y,40,40);
}
}
```

Kết quả chương trình:



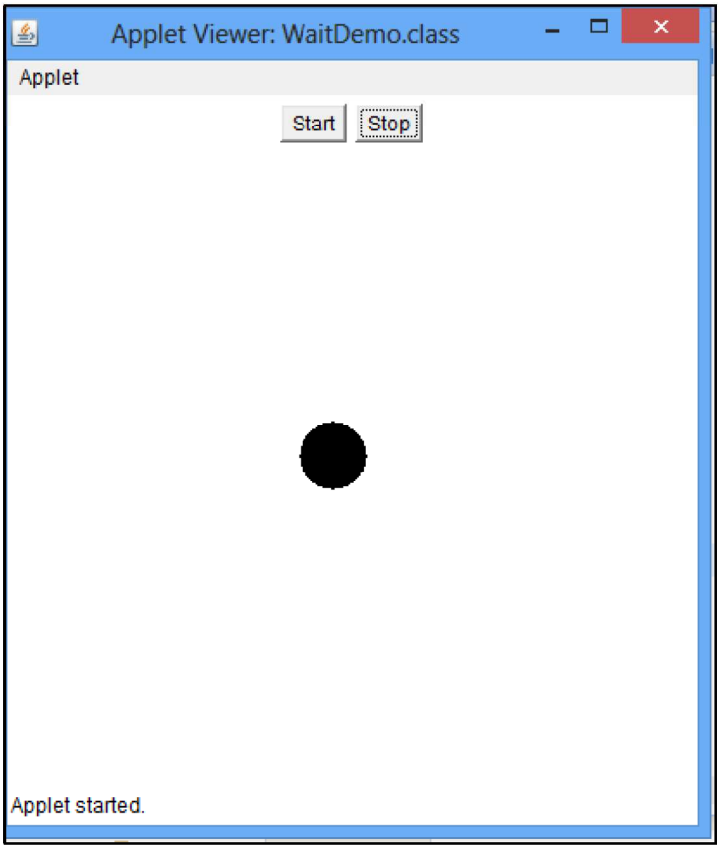
9. Cải tiến chương trình ở bài 8 bằng cách thêm 2 nút "Start" và "Stop" vào chương trình, quả bóng sẽ dừng nếu người dùng ấn nút "Stop" và tiếp tục di chuyển nếu người dùng ấn nút "Start". (Sử dụng hàm wait() và notify() để dừng và cho quả bóng tiếp tục di chuyển).

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

    public class WaitDemo extends Applet implements Runnable,
    ActionListener
    {

//Tuyến con có tên t
     Thread t;

//Vị trí khởi tạo của quả bóng
     int x=34,y=14;

//Bước di chuyển
     int dx=5, dy=5;

//Hai nút có tên Start và Stop
     Button start, stop;

/*Dùng để biểu diễn trạng thái di chuyển hay dừng quả bóng*/
     String status ="Go";

    public void init()
    {

    //Thêm 2 nút Start, Stop ở giao diện
     start=new Button("Start");
     stop=new Button("Stop");
     add(start);
     add(stop);
```

```
    //Gắn ống nghe cho 2 nút
     start.addActionListener(this);
     stop.addActionListener(this);

    //tạo tuyến và khởi động
     t=new Thread(this);
     t.start();
    }

/*Thực hiện việc di chuyển quả bóng*/
    public void run()
    {

/*Quả bóng di chuyển mãi mãi*/
     while(true)
       {

/*Nếu trạng thái quả bóng là Stop thì dừng tạm thời tuyến*/
        if (status.equals("Stop"))
        {
/*gọi hàm wait trong khối synchronized*/
        synchronized(t)
          {
          try
           {

          /*tuyến rơi vào trạng thái ngừng hoạt động*/
              t.wait();
           }
           catch(Exception e1)
             {
              e1.printStackTrace();
             }
         }
     }

    //Nếu va vào đường biên
        if (x+dx>this.getWidth()||x+dx<0) dx=-dx;
        if (y+dy>this.getHeight()||y+dy<0) dy=-dy;

       //thay đổi vị trí
              x=x+dx;
              y=y+dy;

      //xóa màn hình và gọi lại hàm paint(...)
              repaint();
              try
              {

       /*dừng lại 10 mili giây để có thể nhìn thấy quả bóng*/
              Thread.sleep(10);

              }
              catch(Exception e)
              {}
       }
    }

    public void paint(Graphics g)
```

```
    {

      // Vẽ hình tròn tô màu ở vị trí hiện tại
      g.fillOval(x,y,40,40);
    }
    public void actionPerformed(ActionEvent e)
    {

       //Nếu ấn nút Stop
       if (e.getSource()==stop)
       {
              status="Stop";
       }

        //Nếu ấn nút Start
       if (e.getSource()==start)
       {
             status ="Go";
           synchronized(t)
             {

       //Đánh thức và tiếp tục tuyến
              t.notify();

              }
       }
    }
    }
```

Kết quả chương trình:

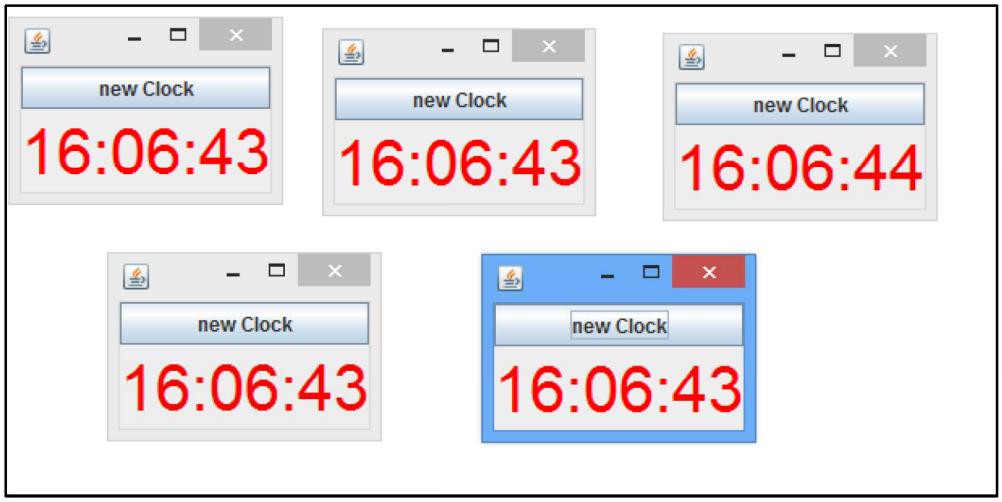


10. Viết chương trình tạo một tuyến biểu diễn đồng hồ điện tử và chứa nút "New Clock". Khi người dùng ấn nút "New Clock" thì chương trình tạo một tuyến để hiển thị một đồng hồ mới.

```
import java.util.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;

public class MultiClock extends JFrame implements ActionListener,
Runnable
{

/*Nút để người dùng ấn và tạo ra tuyến mới*/
   JButton createClock = new JButton("new Clock");

/*dùng để hiển thị thời gian*/
   JLabel clock = new JLabel();

/*Tuyến có tên t*/
   Thread t;

public MultiClock()
 {

 /*Lấy tầng ContentPane của Jframe để chứa các đối tượng hiển thị*/
 Container cont = this.getContentPane();

        /*Lấy thời gian hiện tại*/
     Calendar cal = Calendar.getInstance();

/*thiết lập định dạng hiển thị thời gian giờ:phút:giây*/
     SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");

/*Hiển thị thời gian nhờ Jlable clock, canh lề giữa*/
     clock = new JLabel(sdf.format(cal.getTime()),JLabel.CENTER);

/*Thiết lập kích thước và màu chữ*/
    clock.setFont(new Font(clock.getFont().getName(), Font.PLAIN, 40));
    clock.setForeground(Color.RED);

/*Đặt nút vào phía trên cùng của giao diện*/
    cont.add(createClock,"North");

/*Đặt Jlabel clock ở vùng trung tâm của giao diện*/
    cont.add(clock);
    this.pack();
    this.setVisible(true);

/*Gắn ống nghe cho nút ấn*/
    createClock.addActionListener(this);

/*Tao mot tuyen de hien thi dong ho*/
    Thread t = new Thread(this);
```

```
    /*Khởi động tuyến và gọi hàm run()*/
        t.start();
       }

    /*Hàm tick() dùng để cap nhat lai thoi gian sau 1 giay*/
       public void tick()
       {
        try{

    /*Lấy thời gian hiện tại và thiết lập định dạng hiển thị thời gian*/
         Calendar cal = Calendar.getInstance();
         SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");

    /*Cập nhật lại giá trị ở Jlabel clock tức là cập nhật thời gian*/
         clock.setText(sdf.format(cal.getTime()));


    /*Dừng 1000 mili giây tức 1 giây trước khi cập nhật thời gian mới*/
          Thread.sleep(1000);


        }catch(Exception e)
         {
           e.printStackTrace();
         }

       }

   /*Tuyen hoat dong*/
       public void run()
       {

    /*Vòng lặp vô hạn*/
      while (true)
       {

    /*Cập nhật lại thời gian*/
       tick();
       }

       }

/*Tao mot tuyến mới (dong ho moi) khi an nut New Clock*/
       public void actionPerformed(ActionEvent e)
       {

       /*Tạo một tuyến biểu diễn đồng hồ*/
       Thread t = new Thread(new MultiClock());
       t.start();

       }
       public static void main(String[] a)
       {
             new MultiClock();
       }

    }
```

Kết quả chương trình:



III. Bài tập tự giải

1. Viết chương trình tạo 3 tuyến: tuyến 1 cho phép người dùng nhập vào 2 số thực là hai cạnh của hình chữ nhật, tuyến 2 và tuyến 3 sẽ đồng thời tính diện tích và chu vi của hình chữ nhật với 2 cạnh nhập được ở tuyến 1. Tuyến chính chờ các tuyến hoàn thành rồi in kết quả chu vi và diện tích của hình chữ nhật ra màn hình.
2. Viết chương trình Applet hiển thị chuỗi "Multithread programming" chạy từ bên trái cửa sổ sang phải và ngược lại.
3. Viết chương trình tạo 7 tuyến, mỗi tuyến tạo ngẫu nhiên một số từ 1 đến 30. Tuyến chính chờ tất cả các tuyến hoàn thành, tính tổng các số đã được tạo ra từ các tuyến rồi in ra màn hình.
4. Viết chương trình hiển thị 10 hình tròn tô màu khác nhau và xuất phát từ các vị trí ngẫu nhiên, di chuyển và phản xạ theo một quy luật di chuyển và phản xạ của quả bóng bi da. Mỗi quả bóng tương ứng có 2 nút nhấn "Stop" và "Start" để điều khiển việc dừng và di chuyển tương ứng.
5. Phát triển bài tập 10 của phần bài tập thực hành mẫu bằng cách thêm một lựa chọn hiển thị thời gian của một thành phố ở các quốc gia khác nhau.

Viết chương trình nhân 2 ma trận vuông NxN, chương trình có NxN tuyến, mỗi tuyến thực hiện việc tính giá trị một phần tử của ma trận kết quả. Tuyến chính sẽ nhận kết quả của các tuyến con để tạo ma trận kết quả và in ra màn hình.