PECL1

Practica de productores consumidores.

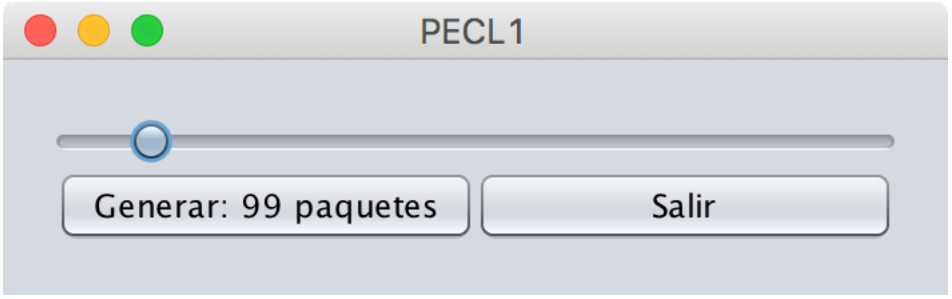
Juan Casado Ballesteros - 09108762A

GII - Programación Avanzada - Laboratorio 8:00 a 10:00

Generator

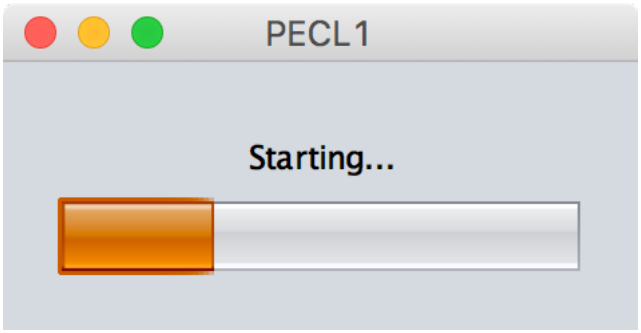
Es la primera clase en ser ejecutada, consiste en una UI que permite ajustar los parámetros del constructor de la clase PECL1. Simplemente se selecciona el número de paquetes que cada productor a generar.

También permite terminar la ejecución del programa.



Load

Es un JFrame que nos muestra una barra de carga. Se lanza cada vez que pasamos del generador a PECL1 o viceversa, es un mero elemento visual.



PECL1

Design

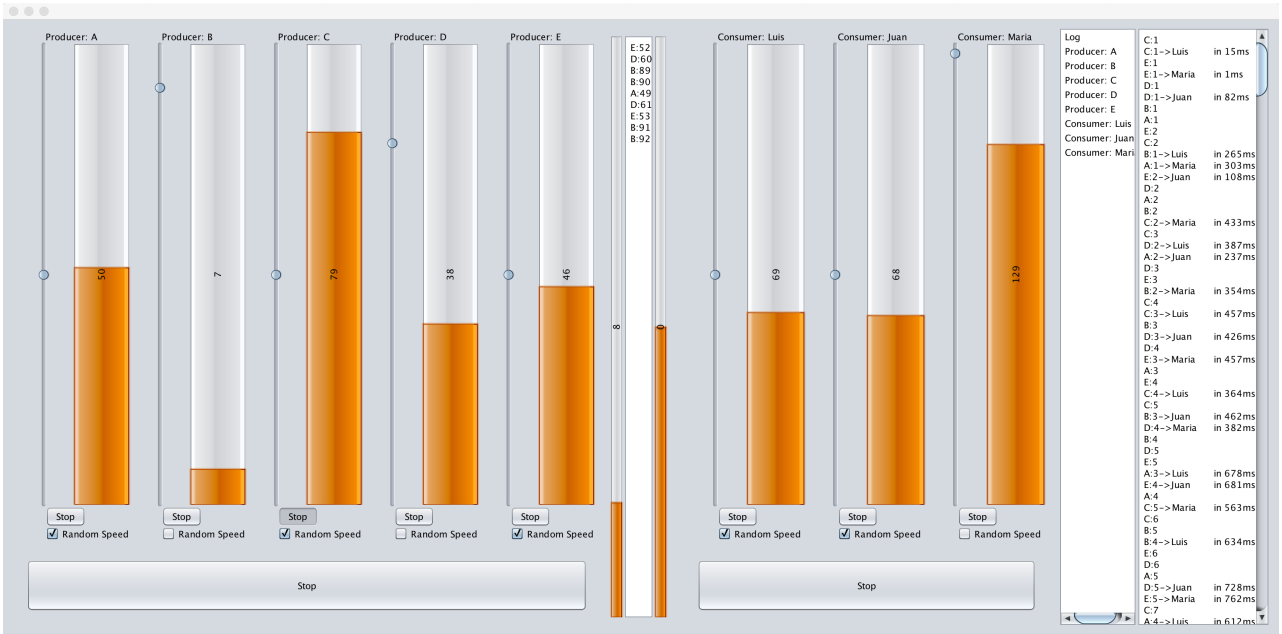
Es la parte principal de la aplicación en cuanto a UI se refiere, consta de cuatro partes, de izquierda a derecha:

Productores: en esta zona se muestran los paquetes que a cada productor le quedan mediante el número concreto y una barra vertical. Se puede ajustar su velocidad con un Slider o volverla aleatoria según lo definido en la práctica. Adicionalmente se podrá parar su producción en conjunto o individualmente.

Buffer: se muestra el estado del buffer, cómo de lleno está mediante una barra y los paquetes que en cada momento contiene. También se muestra otra barra que representa el estado de crecimiento del buffer cada cuatro paquete manejados, cuanto ha crecido o disminuido la cantidad de paquetes que contenía.

Consumidores: Muestran la cantidad de paquetes que contienen con respecto del ideal que deberían contener en función de los que se van a generar. Al igual que con los productores se puede cambiar su velocidad o pararlos en conjunto o por separado.

Log: Muestra un registro en tiempo real del estado del sistema, nos dice qué paquetes a creado un productor en cada momento, los que un consumidor ha recibido y un registro general donde en orden cronológico aparece todo lo que ha ido sucediendo.



Source

En esta clase iniciaremos los objetos que forman la interface así como las clases de negocio. Pondremos el texto de los Label, definiremos el comportamiento de los botones y Sliders para que paren los productores o consumidores según corresponda, les cambien la velocidad… También definimos como se mostrará el Log y como interactuaremos con él.

Lo principal que se realiza en esta clase es instancia el objeto compartido Buffer que también contiene el Log general, este objeto compartido se le pasa tanto a los productores como a los consumidores. Ambas clases recibirán también una infancia propia de la clase Stop lo cual nos va a permitir detenerlas desde la interface al actuar sobre dichas clases. A cada productor o consumidor, así como al Buffer se les pasa también como parámetro en el constructor aquellos elementos de la interface que van a modificar.

Stop

Esta clase contiene un ReentrantLock con una Condition implementada en base al estado de una variable booleana. Adicionalmente contiene al Buffer con la única finalidad de que en caso de error este sea escrito en el Log que dicho Buffer contiene.

Contiene tres métodos que en **exclusión mutua** gracias al Lock modifican la variable boleada, un método es para ponerla a true con seguridad y el otro es para poner a false. El tercer método sirve para “consultar” la variable de forma que si esta está a true forzamos a hilo que la consultó a pasar al estado de espera voluntaria hasta que el estado de la variable vuelva a ser false en cuyo caso no se hace nada.

Package

El objeto que los productores crean, el buffer almacena y los consumidores reciben.

Contiene un id de paquete, el id del productor que lo creó y el tiempo en le que se creó, cuando el consumidor lo extrae del buffer lo debe marcar como enviado para añadir su id a la información de este y dar el “tiempo de vida” del mismo.

Mediante un toString podemos en todo momento saber el estado del paquete.

Producer

Tiene un id y un número de paquetes a crear. Además toma un Buffer como variable compartida donde almacenar dichos paquetes. También toma un ProgressBar y un TextArea como elementos de UI que actualizan para mostrar en la barra el número de paquetes que le quedan por crear y en el área de texto la variable String que contiene si es que le toca mostrarla. Toma un objeto de tipo Stop para parase se es necesario.

Tiene distinto métodos para cambiar entre si mostrar. Sus datos en el área de texto o no y también para modificar su velocidad de producción.

Su ejecución como **hilo** consiste de los siguientes pasos:

1. Esperar el tiempo necesario según el usuario haya indicado en la UI.
2. Parar la producción si así el usuario lo indica.
3. Crear un paquete nuevo.
4. Añadir el paquete al buffer cuando en este haya espacio.
5. Actualizar la UI según sea necesario.

Buffer

Esta es la **variable compartida** sobre la que los productores dejan sus paquetes y de donde los consumidores los toman. Tiene un número máximo de paquetes que puede almacenar y una cantidad de paquetes que lo va a atravesar para saber cuando la simulación termina.

Para almacenar los paquetes se utiliza una LinkedList implementada sobre la interface de Queue para que los paquetes salgan en orden FIFO del buffer.

Se utiliza un ReentrantLock para **garantizar la exclusión mutua a los datos compartidos**, la cola y el String que actúa como Log. Sobre ese Lock se crean dos Condition reguladas por el estado de la cola, vacía o llena para hacer parar a los consumidores y que no retiren paquetes si la cola está vacía o evitar que los productores añadan paquetes nuevos si esta está llena.

Existen distintos métodos para actualizar la UI, mostrar los paquetes dentro del buffer, escribir el Log, decir cómo de llano está este o indicar cómo está creciendo (si se llena o se vacía o está estable).

Consumer

Tiene un id y una cantidad de paquetes que espera recoger. Esta cantidad es idílica, los paquetes que debería recoger en condiciones perfectas de igual de comportamiento de todos los consumidores, solo se usa para actualizar la UI que consta de un TextArea y una ProgressBar. En el constructor toma la variable compartida Buffer y una clase Stop para parar el consumidor desde la UI.

Esta clase tiene un String que actúa como su Log propio del mismo modo que ocurría en los consumidores. Tiene métodos para determinar cuando y cómo debe modificar la UI y también otros para modificar la velocidad de consumo.

Su ejecución como **hilo** consiste de los siguientes pasos:

1. Esperar el tiempo necesario según el usuario haya indicado en la UI.
2. Parar la producción si así el usuario lo indica.
3. Extraer un paquete del buffer cuando en este haya alguno.
4. Actualizar la UI según sea necesario.

Código

Stop

```
package pecl1.classes;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;


public class Stop

{

   private boolean close=false;

   private final Lock lock = new ReentrantLock();

   private final Condition stop = lock.newCondition();

   private final Buffer buffer;


   public Stop (Buffer buffer)

   {

      this.buffer = buffer;

   }


   /**

    * If this object was closed before when we use this method well be kept waiting until the open method is called.

    * If it was open we do nothing and continue the normal execution path

    */

   public void look()

   {

      try

      {

         lock.lock();

         while(close)

         {

            try

            {
```

```
            stop.await();
        }
        catch(InterruptedException i)
        {
            buffer.write("ERROR --> STOP " + i.toString());
        }
      }
   }
   finally
   {
      lock.unlock();
   }
}

/**
 * This makes the thread available to continue the execution whenever the look method is called
 */
public void open()
{
   try
   {
      lock.lock();
      close=false;
      stop.signalAll();
   }
   finally
   {
      lock.unlock();
   }
}

/**
 * This makes the thread stop until we reopen whenever the look method is called
 */
```

```
    public void close()
    {
        try
        {
            lock.lock();
            close=true;
            stop.signalAll();
        }
        finally
        {
            lock.unlock();
        }
    }
}
```

Package

```
package pecl1.classes;

import java.util.Date;


public class Package
{
   private final long startTime;
   private long timeAlive;
   private final String producer;
   private final int number;
   private String consumer;
   private boolean onItsWay;


   /**
    * The creation time is set
    * @param producer Who made the package?
    * @param number Which package is this?
    */
   public Package (String producer, int number)
   {
      this.producer = producer;
      this.number = number;
      this.startTime = new Date().getTime();
      this.onItsWay = true;
   }


   /**
    * The end of cycle time is set
    * @param consumer Who took the package from the buffer?
    */
   public void delivered (String consumer)
   {
      if (onItsWay)
```

```
        {
            this.consumer = consumer;
            this.timeAlive = new Date().getTime() - startTime;
            onItsWay = false;
        }
    }

    /**
     * We can see the status of each package
     * @return The current information about the package, it changes over time
     */
    @Override
    public String toString ()
    {
        if (onItsWay)
            return String.valueOf(producer + ":" + number);
        else
            return String.valueOf(producer + ":" + number + "->" + consumer + "\t in " + timeAlive + "ms");
    }
}
```

Producer

```
package pecl1.classes;


import javax.swing.JProgressBar;
import javax.swing.JTextArea;


/**
 *
 * @author mr.blissfulgrin
 */
public class Producer extends Thread
{
    private final String id;
    private int nextDelivery;
    private final Buffer buffer;
    private int productionRate;
    private boolean random;
    private final int total;
    private String log = "";
    private final JProgressBar bar;
    private final JTextArea stream;
    private Boolean print;
    private final Stop stop;


    /**
     *
     * @param id
     * @param buffer The shared variable where to store the packages produced
     * @param total The number of packages that the producer need to create
     * @param bar UI
     * @param stream UI
     * @param stop The class used to Stop the production of this producer
```

```
 */
public Producer (String id, Buffer buffer, int total, JProgressBar bar, JTextArea stream, Stop stop)
{
   this.id = id;
   this.buffer = buffer;
   this.nextDelivery = 1;
   this.random = true;
   this.total = total+1;
   this.bar = bar;
   this.stream = stream;
   this.print = false;
   this.stop = stop;
}

public void stopPrinting()
{
   print = false;
}

public void beguinPrinting()
{
   print = true;
   stream.setText(log);
}

public String getLog ()
{
   return log;
}

/**
 * We update the log string and if needed the UI
 * Also we update the log of the buffer
 * @param newTxt
```

```
 */
public void write (String newTxt)
{
   buffer.write(newTxt+"\n");
   log += newTxt+"\n";
   if (print)
      stream.setText(log);
}

public void setFixedProductionRate (int productionRate)
{
   random = false;
   this.productionRate = productionRate;
}
public void setRandomProductionRate ()
{
   random = true;
}

/**
 *
 * @return The id of the next package we are going to create
 */
public int getNextDelivery ()
{
   return nextDelivery;
}

/**
 * 1. wait the time as set
 * 2. stop if needed
 * 3. create the package
 * 4. add it to the buffer when possible
 * 5. update UI
```

```
     */
    @Override
    public void run ()
    {
       for (;nextDelivery < total; nextDelivery ++)
       {
          if (random)
          {
             productionRate = (int)(Math.random()*400+400);
          }
          try
          {
             Thread.sleep(productionRate);
          }
          catch (InterruptedException i)
          {
             write("ERROR --> SLEEP IN PRODUCER " + i.toString());
          }
          stop.look();
          Package pack = new Package (id,nextDelivery);
          write(pack.toString());
          buffer.add(pack);
          bar.setValue(total-nextDelivery);
          bar.setString(String.valueOf((total-nextDelivery)-1));
       }
       write(this.toString() + " FINISHED");
    }

    @Override
    public String toString ()
    {
       return String.valueOf("Producer: " + id);
    }
}
```

Buffer

```
package pecl1.classes;

import java.util.Queue;
import java.util.LinkedList ;
import java.util.NoSuchElementException;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import javax.swing.JProgressBar;
import javax.swing.JTextArea;

/**
 *
 * @author mr.blissfulgrin
 */
public class Buffer
{
    private final Queue <Package> buffer;
    private int MAX;
    private final int MIN;
    private final int numberOfDeliveriesTotal;
    private int numberOfDeliveries;

    private String log = "";

    private final Lock lock;
    private final Condition full;
    private final Condition empty;

    private final JProgressBar bufferBar;
    private final JProgressBar bufferDerivate;
```

```
    private final JTextArea bufferTxt;
    private int changes;
    private int previous;
    private final JTextArea stream;
    private Boolean print;


    /**
     *
     * @param MAX Amount of packets the buffer can store at a time
     * @param numberOfDeliveriesTotal Number of packets the producers are gonna produce in total
     * @param bufferTxt UI
     * @param bufferBar UI
     * @param bufferDerivate UI
     * @param stream  UI
     */
      public Buffer (int MAX, int numberOfDeliveriesTotal, JTextArea bufferTxt, JProgressBar bufferBar, JProgressBar
bufferDerivate, JTextArea stream)
    {
        this.MIN = 0;
        this.MAX = MAX;
        this.lock = new ReentrantLock();
        this.full = lock.newCondition();
        this.empty = lock.newCondition();
        this.buffer = new LinkedList();
        this.numberOfDeliveriesTotal = numberOfDeliveriesTotal;
        bufferBar.setMaximum(MAX);
        bufferBar.setMinimum(MIN);
        bufferDerivate.setMaximum(10);
        bufferDerivate.setMinimum(-10);
        this.bufferBar = bufferBar;
        this.bufferDerivate = bufferDerivate;
        this.bufferTxt = bufferTxt;
        this.stream = stream;
        this.print = true;
```

```
}

public void stopPrinting()
{
   print = false;
}

public void beguinPrinting()
{
   print = true;
   stream.setText(log);
}

/**
 *
 * @return it says if all the packages have been delivered or no
 */
public boolean deliveriesLeft()
{
   lock.lock();
   try
   {
      if (numberOfDeliveries >= numberOfDeliveriesTotal)
      {
         bufferDerivate.setValue(0);
         bufferDerivate.setString("0");
         bufferTxt.setText("FINISH");
      }
      return numberOfDeliveries < numberOfDeliveriesTotal;
   }
   finally
   {
      lock.unlock();
   }
```

```
}

/**
 *
 * @return The amount of packages in the Buffer
 */
public int getStatus ()
{
   lock.lock();
   try
   {
      return buffer.size();
   }
   finally
   {
      lock.unlock();
   }
}

/**
 * If there's space we add packages, if not we wait until more space is available
 * @param pack The new package of the Buffer
 */
public void add (Package pack)
{
   lock.lock();
   while (isFull())
   {
      try
      {
         full.await();
      }
      catch (InterruptedException i)
      {
```

```
            write("ERROR --> AWAIT IN ADD " + i.toString());
        }
    }
    try
    {
        buffer.add(pack);
        bufferTxt.setText(this.toString());
        bufferBar.setValue(this.getStatus());
        bufferBar.setString(String.valueOf(this.getStatus()));
        this.derivate();
        empty.signal();
    }
    finally
    {
        lock.unlock();
    }
}

/**
 *
 * @return used to update the UI
 */
public String getLog ()
{
    try
    {
        lock.lock();
        return log;
    }
    finally
    {
        lock.unlock();
    }
}
```

```
/**
 *
 * @param newTxt The new information added to the log
 */
public void write (String newTxt)
{
   try
   {
      lock.lock();
      log += newTxt;
      if (print)
         stream.setText(log);
   }
   finally
   {
      lock.unlock();
   }
}

/**
 *
 * @return used to update the UI with the content of the buffer
 */
@Override
public String toString()
{
   try
   {
      lock.lock();

      String str = "";
      str = buffer.stream().map((p) -> p.toString()+"\n").reduce(str, String::concat);
      return String.valueOf(str);
```

```
    }
    finally
    {
        lock.unlock();
    }
}

/**
 *
 * @return The package been removed from the buffer
 */
public Package get ()
{
    Package pack = null;
    lock.lock();
    while (isEmpty())
    {
        try
        {
            empty.await();
        }
        catch (InterruptedException i)
        {
            write("ERROR --> AWAIT IN GET " + i.toString());
        }
    }
    try
    {
        pack = buffer.remove();
        numberOfDeliveries++;
        bufferTxt.setText(this.toString());
        bufferBar.setValue(this.getStatus());
        bufferBar.setString(String.valueOf(this.getStatus()));
        this.derivate();
```

```
            full.signal();
        }
        catch(NoSuchElementException e)
        {
            write("ERROR --> BUFFER VACIO " + e.toString());
        }
        finally
        {
            lock.unlock();
        }
        return pack;
    }

    /**
     * Used tho calculate the variation in packages every four of them modified, it says if the buffer grows or decrements and the rete of change
     */
    private void derivate ()
    {
        if(changes%4 == 0)
        {
            int value = ((this.getStatus() - previous))%10;
            bufferDerivate.setValue(value);
            bufferDerivate.setString(String.valueOf(value));
            previous = this.getStatus();
        }
        changes++;
    }

    public void setMax (int MAX)
    {
        this.MAX = MAX;
    }
    private boolean isEmpty()
```

```
    {
        return buffer.size() <= MIN;
    }
    private boolean isFull()
    {
        return buffer.size() >= MAX;
    }
}
```

Consumer

```
package pecl1.classes;


import javax.swing.JProgressBar;
import javax.swing.JTextArea;


/**
 *
 * @author mr.blissfulgrin
 */
public class Consumer extends Thread
{
    private final String id;
    private final Buffer buffer;
    private int consumptionRate;
    private boolean random;
    private int deliveriesDone;
    String log = "";
    private final int packetsExpected;
    private final JProgressBar bar;
    private final JTextArea stream;
    private Boolean print;


    private final Stop stop;


    /**
     *
     * @param id
     * @param buffer The shared variable from where the packages are extracted
      * @param packetsExpected The expected amount of packages each producer should in ideal conditions get from the buffer
     * @param bar UI
     * @param stream UI
```

```
 * @param stop The class used to Stop the production of this consumer
 */
public Consumer (String id, Buffer buffer, int packetsExpected, JProgressBar bar, JTextArea stream, Stop stop)
{
   this.id = id;
   this.buffer = buffer;
   this.random = true;
   this.packetsExpected = packetsExpected;
   this.bar = bar;
   this.stream = stream;
   this.print = false;
   this.stop = stop;
}

public void stopPrinting()
{
   print = false;
}

public void beguinPrinting()
{
   print = true;
   stream.setText(log);
}

public String getLog ()
{
   return log;
}

/**
 * We update the log string and if needed the UI
 * Also we update the log of the buffer
 * @param newTxt
```

```
 */
public void write (String newTxt)
{
   buffer.write(newTxt+"\n");
   log += newTxt+"\n";
   if (print)
      stream.setText(log);
}

public void setFixedProductionRate (int consumptionRate)
{
   random = false;
   this.consumptionRate = consumptionRate;
}
public void setRandomProductionRate ()
{
   random = true;
}

public int getDeliveriesDone ()
{
   return deliveriesDone;
}

/**
 * 1. wait the time as set
 * 2. stop if needed
 * 3. extract the package from the buffer when possible
 * 4. update UI
 */
@Override
public void run ()
{
   Package pack;
```

```
    while (buffer.deliveriesLeft())
    {
        if (random)
        {
            consumptionRate = (int)(Math.random()*300+300);
        }
        try
        {
            Thread.sleep(consumptionRate);
        }
        catch (InterruptedException i)
        {
            write("ERROR --> SLEEP IN CONSUMER " + i.toString());
        }
        if (buffer.deliveriesLeft())
        {
            stop.look();
            pack = buffer.get();
            pack.delivered(id);
            write(pack.toString());
            deliveriesDone ++;
            if (deliveriesDone >= packetsExpected)
            {
                bar.setMaximum(deliveriesDone);
            }
            bar.setValue(deliveriesDone);
            bar.setString(String.valueOf(deliveriesDone));
        }
    }
    write(this.toString() + " FINISHED " + deliveriesDone + " deliveries done");
}

@Override
public String toString ()
```

```
    {
        return String.valueOf("Consumer: " + id);
    }
}
```

Generator

```
package pecl1.screens;


/**
 *
 * @author mr.blissfulgrin
 */
public class Generator extends javax.swing.JFrame
{


    PECL1 p;
    public Generator()
    {
        initComponents();
    }


    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents()
    {


        jPanel1 = new javax.swing.JPanel();
        jPanel2 = new javax.swing.JPanel();
        jSlider1 = new javax.swing.JSlider();
        jPanel3 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();


        setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        setTitle("PECL1");
        setMinimumSize(new java.awt.Dimension(500, 100));
        setPreferredSize(new java.awt.Dimension(500, 150));
```

```
setSize(new java.awt.Dimension(500, 150));
getContentPane().setLayout(new java.awt.CardLayout(20, 20));

jPanel1.setLayout(new javax.swing.BoxLayout(jPanel1, javax.swing.BoxLayout.Y_AXIS));

jPanel2.setLayout(new java.awt.CardLayout());

jSlider1.setMaximum(1000);
jSlider1.setMinimum(10);
jSlider1.setValue(99);
jSlider1.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
jSlider1.addChangeListener(new javax.swing.event.ChangeListener()
{
   public void stateChanged(javax.swing.event.ChangeEvent evt)
   {
      jSlider1StateChanged(evt);
   }
});
jPanel2.add(jSlider1, "card2");

jPanel1.add(jPanel2);

jPanel3.setLayout(new java.awt.GridLayout(1, 0));

jButton1.setText("Generar: 99 paquetes");
jButton1.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      jButton1ActionPerformed(evt);
   }
});
jPanel3.add(jButton1);
```

```
    jButton2.setText("Salir");
    jButton2.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
            jButton2ActionPerformed(evt);
        }
    });
    jPanel3.add(jButton2);

    jPanel1.add(jPanel3);

    getContentPane().add(jPanel1, "card2");

    pack();
    setLocationRelativeTo(null);
}// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    p = new PECL1(jSlider1.getValue());
    p.setVisible(true);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
    this.setVisible(false);
    this.dispose();
    System.exit(0);
}

private void jSlider1StateChanged(javax.swing.event.ChangeEvent evt)
{
    jButton1.setText("Generate "+ jSlider1.getValue()+" packets");
```

```
}

/**
 * @param args the command line arguments
 */
public static void main(String args[])
{
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try
    {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels())
        {
            if ("Nimbus".equals(info.getName()))
            {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex)
    {
        java.util.logging.Logger.getLogger(Generator.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex)
    {
        java.util.logging.Logger.getLogger(Generator.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex)
    {
        java.util.logging.Logger.getLogger(Generator.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex)
    {
        java.util.logging.Logger.getLogger(Generator.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(() ->
        {
            new Generator().setVisible(true);
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JSlider jSlider1;
    // End of variables declaration
}
```

Load

```
package pecl1.screens;


/**
 *
 * @author mr.blissfulgrin
 */
public class Load extends javax.swing.JFrame {


    public Load()
    {
        initComponents();
    }


    public void go (String txt)
    {
        this.txt.setText(txt);
        this.setVisible(true);
        this.update(this.getGraphics());


        for(int x = 0; x<100; x++)
        {
            try
            {
                Thread.sleep(5);
            }
            catch (InterruptedException i)
            {
                System.out.println("ERROR LOADING!!! " + i.toString());
            }
            bar.setValue(x);
            bar.setString(x + "%");
```

```
        bar.update(bar.getGraphics());
    }
    this.setVisible(false);
    this.dispose();
}


@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents()
{

    jPanel1 = new javax.swing.JPanel();
    jPanel2 = new javax.swing.JPanel();
    txt = new javax.swing.JLabel();
    bar = new javax.swing.JProgressBar();

    setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
    setTitle("PECL1");

    jPanel1.setLayout(new java.awt.CardLayout(20, 20));

    jPanel2.setLayout(new java.awt.GridLayout(0, 1));

    txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    txt.setToolTipText("");
    jPanel2.add(txt);
    jPanel2.add(bar);

    jPanel1.add(jPanel2, "card2");

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
```

```
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 238, javax.swing.GroupLayout.PREFERRED_SIZE)
        );
        layout.setVerticalGroup(
          layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>

    // Variables declaration - do not modify
    private javax.swing.JProgressBar bar;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JLabel txt;
    // End of variables declaration
}
```

PECL1

```
package pecl1.screens;

import pecl1.classes.Stop;
import pecl1.classes.Buffer;
import pecl1.classes.Producer;
import pecl1.classes.Consumer;
import javax.swing.JProgressBar;



/**
 *
 * @author mr.blissfulgrin
 */
public class PECL1 extends javax.swing.JFrame {

    private final Producer [] producer;
    private final Consumer [] consumer;
    Buffer buffer;
    private final Stop [] stop;

    public PECL1(int p)
    {
        new Load().go("Starting...");
        initComponents();

        int numerOfPackets = p;
        String [] producer_id = {"A","B","C","D","E"};
        String [] consumer_id = {"Luis","Juan","Maria"};

        int packetsExpected = numerOfPackets*producer_id.length/consumer_id.length;
```

```
producer0Txt.setText("Producer: "+producer_id[0]);
producer1Txt.setText("Producer: "+producer_id[1]);
producer2Txt.setText("Producer: "+producer_id[2]);
producer3Txt.setText("Producer: "+producer_id[3]);
producer4Txt.setText("Producer: "+producer_id[4]);
consumer0Txt.setText("Consumer: "+consumer_id[0]);
consumer1Txt.setText("Consumer: "+consumer_id[1]);
consumer2Txt.setText("Consumer: "+consumer_id[2]);

producer0Bar.setMaximum(numerOfPackets);
producer1Bar.setMaximum(numerOfPackets);
producer2Bar.setMaximum(numerOfPackets);
producer3Bar.setMaximum(numerOfPackets);
producer4Bar.setMaximum(numerOfPackets);
consumer0Bar.setMaximum(packetsExpected);
consumer1Bar.setMaximum(packetsExpected);
consumer2Bar.setMaximum(packetsExpected);

producer0Chk.setSelected(true);
producer1Chk.setSelected(true);
producer2Chk.setSelected(true);
producer3Chk.setSelected(true);
producer4Chk.setSelected(true);
consumer0Chk.setSelected(true);
consumer1Chk.setSelected(true);
consumer2Chk.setSelected(true);

producer0Chk.setText("Random Speed");
producer1Chk.setText("Random Speed");
producer2Chk.setText("Random Speed");
producer3Chk.setText("Random Speed");
producer4Chk.setText("Random Speed");
consumer0Chk.setText("Random Speed");
consumer1Chk.setText("Random Speed");
```

```
consumer2Chk.setText("Random Speed");

producer0Btt.setText("Stop");
producer1Btt.setText("Stop");
producer2Btt.setText("Stop");
producer3Btt.setText("Stop");
producer4Btt.setText("Stop");
producerStop.setText("Stop");
consumerStop.setText("Stop");
consumer0Btt.setText("Stop");
consumer1Btt.setText("Stop");
consumer2Btt.setText("Stop");

JProgressBar [] bp = {producer0Bar,producer1Bar,producer2Bar,producer3Bar,producer4Bar};
JProgressBar [] bc = {consumer0Bar,consumer1Bar,consumer2Bar};

producer = new Producer[producer_id.length];
consumer = new Consumer[consumer_id.length];
stop = new Stop [producer_id.length+consumer_id.length];
for (int i = 0; i < stop.length; i++)
{
   stop[i] = new Stop(buffer);
}

buffer = new Buffer(40, producer.length*numerOfPackets, bufferTxt, bufferBar, bufferDerivate,txt);

//CREAR
for (int x = 0; x<consumer.length; x++)
{
   consumer[x] = new Consumer(consumer_id[x],buffer,packetsExpected,bc[x],txt,stop[x]);
}
for (int x = 0; x<producer.length; x++)
{
   producer[x] = new Producer(producer_id[x],buffer,numerOfPackets,bp[x],txt,stop[x+consumer.length]);
```

```
        }

        String [] data = new String [producer_id.length+consumer_id.length+1];
        data[0] = "Log";

        for (int i = 1; i < producer_id.length+1; i++)
        {
            data[i] = producer[i-1].toString();
        }
        for (int i = producer_id.length+1; i < producer_id.length+1+consumer_id.length; i++)
        {
            data[i] = consumer[(i-1)-producer_id.length].toString();
        }

        list.setListData(data);
        stopPrinting();
        buffer.beguinPrinting();

        run();
    }

    private Boolean bttProducer ()
    {
                    return producer0Btt.isSelected() && producer1Btt.isSelected() && producer2Btt.isSelected() &&
producer3Btt.isSelected() && producer4Btt.isSelected();
    }

    private Boolean bttConsumer ()
    {
        return consumer0Btt.isSelected() && consumer1Btt.isSelected() && consumer2Btt.isSelected();
    }

    private void run()
    {
```

```
    //INICIAR
    for (Consumer c : consumer)
    {
        c.start();
    }
    for (Producer p : producer)
    {
        p.start();
    }
}

private void stopPrinting ()
{
    for (Consumer c : consumer)
    {
        c.stopPrinting();
    }
    for (Producer p : producer)
    {
        p.stopPrinting();
    }
    buffer.stopPrinting();
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents()
{
    java.awt.GridBagConstraints gridBagConstraints;
```

```
jPanel1 = new javax.swing.JPanel();
jPanel2 = new javax.swing.JPanel();
jPanel4 = new javax.swing.JPanel();
jPanel7 = new javax.swing.JPanel();
jPanel10 = new javax.swing.JPanel();
jPanel6 = new javax.swing.JPanel();
jPanel15 = new javax.swing.JPanel();
producer0Txt = new javax.swing.JLabel();
jPanel17 = new javax.swing.JPanel();
jPanel16 = new javax.swing.JPanel();
producer0Sld = new javax.swing.JSlider();
producer0Bar = new javax.swing.JProgressBar();
jPanel18 = new javax.swing.JPanel();
producer0Btt = new javax.swing.JToggleButton();
producer0Chk = new javax.swing.JCheckBox();
jPanel19 = new javax.swing.JPanel();
producer1Txt = new javax.swing.JLabel();
jPanel20 = new javax.swing.JPanel();
jPanel21 = new javax.swing.JPanel();
producer1Sld = new javax.swing.JSlider();
producer1Bar = new javax.swing.JProgressBar();
jPanel22 = new javax.swing.JPanel();
producer1Btt = new javax.swing.JToggleButton();
producer1Chk = new javax.swing.JCheckBox();
jPanel23 = new javax.swing.JPanel();
producer2Txt = new javax.swing.JLabel();
jPanel24 = new javax.swing.JPanel();
jPanel25 = new javax.swing.JPanel();
producer2Sld = new javax.swing.JSlider();
producer2Bar = new javax.swing.JProgressBar();
jPanel26 = new javax.swing.JPanel();
producer2Btt = new javax.swing.JToggleButton();
producer2Chk = new javax.swing.JCheckBox();
```

```
jPanel43 = new javax.swing.JPanel();
producer3Txt = new javax.swing.JLabel();
jPanel44 = new javax.swing.JPanel();
jPanel45 = new javax.swing.JPanel();
producer3Sld = new javax.swing.JSlider();
producer3Bar = new javax.swing.JProgressBar();
jPanel46 = new javax.swing.JPanel();
producer3Btt = new javax.swing.JToggleButton();
producer3Chk = new javax.swing.JCheckBox();
jPanel47 = new javax.swing.JPanel();
producer4Txt = new javax.swing.JLabel();
jPanel48 = new javax.swing.JPanel();
jPanel49 = new javax.swing.JPanel();
producer4Sld = new javax.swing.JSlider();
producer4Bar = new javax.swing.JProgressBar();
jPanel50 = new javax.swing.JPanel();
producer4Btt = new javax.swing.JToggleButton();
producer4Chk = new javax.swing.JCheckBox();
jPanel11 = new javax.swing.JPanel();
producerStop = new javax.swing.JToggleButton();
jPanel8 = new javax.swing.JPanel();
jPanel14 = new javax.swing.JPanel();
bufferBar = new javax.swing.JProgressBar();
jScrollPane3 = new javax.swing.JScrollPane();
bufferTxt = new javax.swing.JTextArea();
bufferDerivate = new javax.swing.JProgressBar();
jPanel9 = new javax.swing.JPanel();
jPanel27 = new javax.swing.JPanel();
jPanel28 = new javax.swing.JPanel();
jPanel29 = new javax.swing.JPanel();
jPanel30 = new javax.swing.JPanel();
consumer0Txt = new javax.swing.JLabel();
jPanel31 = new javax.swing.JPanel();
jPanel32 = new javax.swing.JPanel();
```

```
consumer0Sld = new javax.swing.JSlider();
consumer0Bar = new javax.swing.JProgressBar();
jPanel33 = new javax.swing.JPanel();
consumer0Btt = new javax.swing.JToggleButton();
consumer0Chk = new javax.swing.JCheckBox();
jPanel34 = new javax.swing.JPanel();
consumer1Txt = new javax.swing.JLabel();
jPanel35 = new javax.swing.JPanel();
jPanel36 = new javax.swing.JPanel();
consumer1Sld = new javax.swing.JSlider();
consumer1Bar = new javax.swing.JProgressBar();
jPanel37 = new javax.swing.JPanel();
consumer1Btt = new javax.swing.JToggleButton();
consumer1Chk = new javax.swing.JCheckBox();
jPanel38 = new javax.swing.JPanel();
consumer2Txt = new javax.swing.JLabel();
jPanel39 = new javax.swing.JPanel();
jPanel40 = new javax.swing.JPanel();
consumer2Sld = new javax.swing.JSlider();
consumer2Bar = new javax.swing.JProgressBar();
jPanel41 = new javax.swing.JPanel();
consumer2Btt = new javax.swing.JToggleButton();
consumer2Chk = new javax.swing.JCheckBox();
jPanel42 = new javax.swing.JPanel();
consumerStop = new javax.swing.JToggleButton();
jPanel3 = new javax.swing.JPanel();
jPanel5 = new javax.swing.JPanel();
jScrollPane2 = new javax.swing.JScrollPane();
list = new javax.swing.JList<>();
jScrollPane1 = new javax.swing.JScrollPane();
txt = new javax.swing.JTextArea();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
addWindowListener(new java.awt.event.WindowAdapter()
```

```
{
   public void windowClosing(java.awt.event.WindowEvent evt)
   {
      formWindowClosing(evt);
   }
});
getContentPane().setLayout(new java.awt.CardLayout(10, 10));

jPanel1.setLayout(new java.awt.GridBagLayout());

jPanel2.setLayout(new java.awt.CardLayout());

jPanel4.setLayout(new java.awt.GridBagLayout());

jPanel7.setLayout(new java.awt.GridBagLayout());

jPanel10.setLayout(new java.awt.CardLayout(5, 5));

jPanel6.setLayout(new javax.swing.BoxLayout(jPanel6, javax.swing.BoxLayout.LINE_AXIS));

jPanel15.setLayout(new javax.swing.BoxLayout(jPanel15, javax.swing.BoxLayout.Y_AXIS));

producer0Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
producer0Txt.setText("producer0Txt");
jPanel15.add(producer0Txt);

jPanel17.setLayout(new java.awt.CardLayout());

jPanel16.setLayout(new java.awt.GridLayout(1, 0));

producer0Sld.setMaximum(-100);
producer0Sld.setMinimum(-2000);
producer0Sld.setOrientation(javax.swing.JSlider.VERTICAL);
producer0Sld.setValue(-1050);
```

```
producer0Sld.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
producer0Sld.addChangeListener(new javax.swing.event.ChangeListener()
{
   public void stateChanged(javax.swing.event.ChangeEvent evt)
   {
      producer0SldStateChanged(evt);
   }
});
jPanel16.add(producer0Sld);

producer0Bar.setOrientation(1);
producer0Bar.setStringPainted(true);
jPanel16.add(producer0Bar);

jPanel17.add(jPanel16, "card2");

jPanel15.add(jPanel17);

jPanel18.setLayout(new javax.swing.BoxLayout(jPanel18, javax.swing.BoxLayout.Y_AXIS));

producer0Btt.setText("producer0Btt");
producer0Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer0BttActionPerformed(evt);
   }
});
jPanel18.add(producer0Btt);

producer0Chk.setText("producer0Chk");
producer0Chk.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
```

```
    {
        producer0ChkActionPerformed(evt);
    }
});
jPanel18.add(producer0Chk);

jPanel15.add(jPanel18);

jPanel6.add(jPanel15);

jPanel19.setLayout(new javax.swing.BoxLayout(jPanel19, javax.swing.BoxLayout.Y_AXIS));

producer1Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
producer1Txt.setText("producer1Txt");
jPanel19.add(producer1Txt);

jPanel20.setLayout(new java.awt.CardLayout());

jPanel21.setLayout(new java.awt.GridLayout(1, 0));

producer1Sld.setMaximum(-100);
producer1Sld.setMinimum(-2000);
producer1Sld.setOrientation(javax.swing.JSlider.VERTICAL);
producer1Sld.setValue(-1050);
producer1Sld.addChangeListener(new javax.swing.event.ChangeListener()
{
    public void stateChanged(javax.swing.event.ChangeEvent evt)
    {
        producer1SldStateChanged(evt);
    }
});
jPanel21.add(producer1Sld);

producer1Bar.setOrientation(1);
```

```
producer1Bar.setStringPainted(true);
jPanel21.add(producer1Bar);

jPanel20.add(jPanel21, "card2");

jPanel19.add(jPanel20);

jPanel22.setLayout(new javax.swing.BoxLayout(jPanel22, javax.swing.BoxLayout.Y_AXIS));

producer1Btt.setText("producer1Btt");
producer1Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer1BttActionPerformed(evt);
   }
});
jPanel22.add(producer1Btt);

producer1Chk.setText("producer1Chk");
producer1Chk.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer1ChkActionPerformed(evt);
   }
});
jPanel22.add(producer1Chk);

jPanel19.add(jPanel22);

jPanel6.add(jPanel19);

jPanel23.setLayout(new javax.swing.BoxLayout(jPanel23, javax.swing.BoxLayout.Y_AXIS));
```

```
producer2Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
producer2Txt.setText("producer2Txt");
jPanel23.add(producer2Txt);

jPanel24.setLayout(new java.awt.CardLayout());

jPanel25.setLayout(new java.awt.GridLayout(1, 0));

producer2Sld.setMaximum(-100);
producer2Sld.setMinimum(-2000);
producer2Sld.setOrientation(javax.swing.JSlider.VERTICAL);
producer2Sld.setValue(-1050);
producer2Sld.addChangeListener(new javax.swing.event.ChangeListener()
{
   public void stateChanged(javax.swing.event.ChangeEvent evt)
   {
      producer2SldStateChanged(evt);
   }
});
jPanel25.add(producer2Sld);

producer2Bar.setOrientation(1);
producer2Bar.setStringPainted(true);
jPanel25.add(producer2Bar);

jPanel24.add(jPanel25, "card2");

jPanel23.add(jPanel24);

jPanel26.setLayout(new javax.swing.BoxLayout(jPanel26, javax.swing.BoxLayout.Y_AXIS));

producer2Btt.setText("producer2Btt");
producer2Btt.addActionListener(new java.awt.event.ActionListener()
```

```
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer2BttActionPerformed(evt);
   }
});
jPanel26.add(producer2Btt);

producer2Chk.setText("producer2Chk");
producer2Chk.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer2ChkActionPerformed(evt);
   }
});
jPanel26.add(producer2Chk);

jPanel23.add(jPanel26);

jPanel6.add(jPanel23);

jPanel43.setLayout(new javax.swing.BoxLayout(jPanel43, javax.swing.BoxLayout.Y_AXIS));

producer3Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
producer3Txt.setText("producer3Txt");
jPanel43.add(producer3Txt);

jPanel44.setLayout(new java.awt.CardLayout());

jPanel45.setLayout(new java.awt.GridLayout(1, 0));

producer3Sld.setMaximum(-100);
producer3Sld.setMinimum(-2000);
```

```
producer3Sld.setOrientation(javax.swing.JSlider.VERTICAL);
producer3Sld.setValue(-1050);
producer3Sld.addChangeListener(new javax.swing.event.ChangeListener()
{
   public void stateChanged(javax.swing.event.ChangeEvent evt)
   {
      producer3SldStateChanged(evt);
   }
});
jPanel45.add(producer3Sld);

producer3Bar.setOrientation(1);
producer3Bar.setStringPainted(true);
jPanel45.add(producer3Bar);

jPanel44.add(jPanel45, "card2");

jPanel43.add(jPanel44);

jPanel46.setLayout(new javax.swing.BoxLayout(jPanel46, javax.swing.BoxLayout.Y_AXIS));

producer3Btt.setText("producer3Btt");
producer3Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer3BttActionPerformed(evt);
   }
});
jPanel46.add(producer3Btt);

producer3Chk.setText("producer3Chk");
producer3Chk.addActionListener(new java.awt.event.ActionListener()
{
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt)
    {
        producer3ChkActionPerformed(evt);
    }
});
jPanel46.add(producer3Chk);

jPanel43.add(jPanel46);

jPanel6.add(jPanel43);

jPanel47.setLayout(new javax.swing.BoxLayout(jPanel47, javax.swing.BoxLayout.Y_AXIS));

producer4Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
producer4Txt.setText("producer4Txt");
jPanel47.add(producer4Txt);

jPanel48.setLayout(new java.awt.CardLayout());

jPanel49.setLayout(new java.awt.GridLayout(1, 0));

producer4Sld.setMaximum(-100);
producer4Sld.setMinimum(-2000);
producer4Sld.setOrientation(javax.swing.JSlider.VERTICAL);
producer4Sld.setValue(-1050);
producer4Sld.addChangeListener(new javax.swing.event.ChangeListener()
{
    public void stateChanged(javax.swing.event.ChangeEvent evt)
    {
        producer4SldStateChanged(evt);
    }
});
jPanel49.add(producer4Sld);
```

```
producer4Bar.setOrientation(1);
producer4Bar.setStringPainted(true);
jPanel49.add(producer4Bar);

jPanel48.add(jPanel49, "card2");

jPanel47.add(jPanel48);

jPanel50.setLayout(new javax.swing.BoxLayout(jPanel50, javax.swing.BoxLayout.Y_AXIS));

producer4Btt.setText("producer4Btt");
producer4Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer4BttActionPerformed(evt);
   }
});
jPanel50.add(producer4Btt);

producer4Chk.setText("producer4Chk");
producer4Chk.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer4ChkActionPerformed(evt);
   }
});
jPanel50.add(producer4Chk);

jPanel47.add(jPanel50);

jPanel6.add(jPanel47);
```

```
jPanel10.add(jPanel6, "card2");

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.weighty = 10.0;
jPanel7.add(jPanel10, gridBagConstraints);

jPanel11.setLayout(new java.awt.CardLayout(20, 20));

producerStop.setText("producerStop");
producerStop.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producerStopActionPerformed(evt);
   }
});
jPanel11.add(producerStop, "card3");

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.weighty = 1.0;
jPanel7.add(jPanel11, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.weighty = 1.0;
jPanel4.add(jPanel7, gridBagConstraints);
```

```
        jPanel8.setLayout(new java.awt.CardLayout(10, 10));

        jPanel14.setLayout(new javax.swing.BoxLayout(jPanel14, javax.swing.BoxLayout.LINE_AXIS));

        bufferBar.setOrientation(1);
        bufferBar.setStringPainted(true);
        jPanel14.add(bufferBar);

jScrollPane3.setHorizontalScrollBarPolicy(javax.swing.ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
        jScrollPane3.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.VERTICAL_SCROLLBAR_NEVER);

        bufferTxt.setColumns(20);
        bufferTxt.setRows(5);
        bufferTxt.setAutoscrolls(false);
        bufferTxt.setDragEnabled(false);
        bufferTxt.setFocusable(false);
        bufferTxt.setMinimumSize(new java.awt.Dimension(0, 5));
        bufferTxt.setPreferredSize(new java.awt.Dimension(5, 5));
        bufferTxt.setRequestFocusEnabled(false);
        jScrollPane3.setViewportView(bufferTxt);

        jPanel14.add(jScrollPane3);

        bufferDerivate.setOrientation(1);
        bufferDerivate.setStringPainted(true);
        jPanel14.add(bufferDerivate);

        jPanel8.add(jPanel14, "card2");

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.weightx = 8.0;
        gridBagConstraints.weighty = 1.0;
```

```
jPanel4.add(jPanel8, gridBagConstraints);

jPanel27.setLayout(new java.awt.GridBagLayout());

jPanel28.setLayout(new java.awt.CardLayout(5, 5));

jPanel29.setLayout(new javax.swing.BoxLayout(jPanel29, javax.swing.BoxLayout.LINE_AXIS));

jPanel30.setLayout(new javax.swing.BoxLayout(jPanel30, javax.swing.BoxLayout.Y_AXIS));

consumer0Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
consumer0Txt.setText("consumer0Txt");
jPanel30.add(consumer0Txt);

jPanel31.setLayout(new java.awt.CardLayout());

jPanel32.setLayout(new java.awt.GridLayout(1, 0));

consumer0Sld.setMaximum(-100);
consumer0Sld.setMinimum(-2000);
consumer0Sld.setOrientation(javax.swing.JSlider.VERTICAL);
consumer0Sld.setValue(-1050);
consumer0Sld.addChangeListener(new javax.swing.event.ChangeListener()
{
   public void stateChanged(javax.swing.event.ChangeEvent evt)
   {
      consumer0SldStateChanged(evt);
   }
});
jPanel32.add(consumer0Sld);

consumer0Bar.setOrientation(1);
consumer0Bar.setStringPainted(true);
jPanel32.add(consumer0Bar);
```

```
jPanel31.add(jPanel32, "card2");

jPanel30.add(jPanel31);

jPanel33.setLayout(new javax.swing.BoxLayout(jPanel33, javax.swing.BoxLayout.Y_AXIS));

consumer0Btt.setText("consumer0Btt");
consumer0Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      consumer0BttActionPerformed(evt);
   }
});
jPanel33.add(consumer0Btt);

consumer0Chk.setText("consumer0Chk");
consumer0Chk.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      consumer0ChkActionPerformed(evt);
   }
});
jPanel33.add(consumer0Chk);

jPanel30.add(jPanel33);

jPanel29.add(jPanel30);

jPanel34.setLayout(new javax.swing.BoxLayout(jPanel34, javax.swing.BoxLayout.Y_AXIS));

consumer1Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
```

```
consumer1Txt.setText("consumer1Txt");
jPanel34.add(consumer1Txt);

jPanel35.setLayout(new java.awt.CardLayout());

jPanel36.setLayout(new java.awt.GridLayout(1, 0));

consumer1Sld.setMaximum(-100);
consumer1Sld.setMinimum(-2000);
consumer1Sld.setOrientation(javax.swing.JSlider.VERTICAL);
consumer1Sld.setValue(-1050);
consumer1Sld.addChangeListener(new javax.swing.event.ChangeListener()
{
   public void stateChanged(javax.swing.event.ChangeEvent evt)
   {
      consumer1SldStateChanged(evt);
   }
});
jPanel36.add(consumer1Sld);

consumer1Bar.setOrientation(1);
consumer1Bar.setStringPainted(true);
jPanel36.add(consumer1Bar);

jPanel35.add(jPanel36, "card2");

jPanel34.add(jPanel35);

jPanel37.setLayout(new javax.swing.BoxLayout(jPanel37, javax.swing.BoxLayout.Y_AXIS));

consumer1Btt.setText("consumer1Btt");
consumer1Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
```

```
    {
        consumer1BttActionPerformed(evt);
    }
});
jPanel37.add(consumer1Btt);

consumer1Chk.setText("consumer1Chk");
consumer1Chk.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt)
    {
        consumer1ChkActionPerformed(evt);
    }
});
jPanel37.add(consumer1Chk);

jPanel34.add(jPanel37);

jPanel29.add(jPanel34);

jPanel38.setLayout(new javax.swing.BoxLayout(jPanel38, javax.swing.BoxLayout.Y_AXIS));

consumer2Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
consumer2Txt.setText("consumer2Txt");
jPanel38.add(consumer2Txt);

jPanel39.setLayout(new java.awt.CardLayout());

jPanel40.setLayout(new java.awt.GridLayout(1, 0));

consumer2Sld.setMaximum(-100);
consumer2Sld.setMinimum(-2000);
consumer2Sld.setOrientation(javax.swing.JSlider.VERTICAL);
consumer2Sld.setValue(-1050);
```

```
consumer2Sld.addChangeListener(new javax.swing.event.ChangeListener()
{
   public void stateChanged(javax.swing.event.ChangeEvent evt)
   {
      consumer2SldStateChanged(evt);
   }
});
jPanel40.add(consumer2Sld);

consumer2Bar.setOrientation(1);
consumer2Bar.setStringPainted(true);
jPanel40.add(consumer2Bar);

jPanel39.add(jPanel40, "card2");

jPanel38.add(jPanel39);

jPanel41.setLayout(new javax.swing.BoxLayout(jPanel41, javax.swing.BoxLayout.Y_AXIS));

consumer2Btt.setText("consumer2Btt");
consumer2Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      consumer2BttActionPerformed(evt);
   }
});
jPanel41.add(consumer2Btt);

consumer2Chk.setText("consumer2Chk");
consumer2Chk.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
```

```
        consumer2ChkActionPerformed(evt);
    }
});
jPanel41.add(consumer2Chk);

jPanel38.add(jPanel41);

jPanel29.add(jPanel38);

jPanel28.add(jPanel29, "card2");

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.weighty = 10.0;
jPanel27.add(jPanel28, gridBagConstraints);

jPanel42.setLayout(new java.awt.CardLayout(20, 20));

consumerStop.setText("consumerStop");
consumerStop.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt)
    {
        consumerStopActionPerformed(evt);
    }
});
jPanel42.add(consumerStop, "card3");

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.weightx = 1.0;
```

```
        gridBagConstraints.weighty = 1.0;
        jPanel27.add(jPanel42, gridBagConstraints);

        javax.swing.GroupLayout jPanel9Layout = new javax.swing.GroupLayout(jPanel9);
        jPanel9.setLayout(jPanel9Layout);
        jPanel9Layout.setHorizontalGroup(
            jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 487, Short.MAX_VALUE)
            .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel9Layout.createSequentialGroup()
                    .addGap(0, 0, Short.MAX_VALUE)
                                                .addComponent(jPanel27, javax.swing.GroupLayout.PREFERRED_SIZE, 486, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(0, 0, Short.MAX_VALUE)))
        );
        jPanel9Layout.setVerticalGroup(
            jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 788, Short.MAX_VALUE)
            .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel9Layout.createSequentialGroup()
                    .addGap(0, 0, Short.MAX_VALUE)
                                                .addComponent(jPanel27, javax.swing.GroupLayout.PREFERRED_SIZE, 788, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(0, 0, Short.MAX_VALUE)))
        );

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
        jPanel4.add(jPanel9, gridBagConstraints);

        jPanel2.add(jPanel4, "card2");

        gridBagConstraints = new java.awt.GridBagConstraints();
```

```
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.weightx = 3.0;
        gridBagConstraints.weighty = 1.0;
        jPanel1.add(jPanel2, gridBagConstraints);

        jPanel3.setLayout(new java.awt.CardLayout());

        jPanel5.setLayout(new java.awt.GridBagLayout());

        list.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        list.addListSelectionListener(new javax.swing.event.ListSelectionListener()
        {
           public void valueChanged(javax.swing.event.ListSelectionEvent evt)
           {
              listValueChanged(evt);
           }
        });
        jScrollPane2.setViewportView(list);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
        jPanel5.add(jScrollPane2, gridBagConstraints);

jScrollPane1.setHorizontalScrollBarPolicy(javax.swing.ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
        jScrollPane1.setAutoscrolls(true);

        txt.setEditable(false);
        txt.setColumns(20);
        txt.setRows(5);
        txt.setFocusable(false);
        jScrollPane1.setViewportView(txt);
```

```
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.weightx = 1.9;
    gridBagConstraints.weighty = 1.0;
    jPanel5.add(jScrollPane1, gridBagConstraints);

    jPanel3.add(jPanel5, "card2");

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.weightx = 14.0;
    gridBagConstraints.weighty = 1.0;
    jPanel1.add(jPanel3, gridBagConstraints);

    getContentPane().add(jPanel1, "card2");

    pack();
    setLocationRelativeTo(null);
}// </editor-fold>

private void formWindowClosing(java.awt.event.WindowEvent evt) {
    new Load().go("Quiting..");
}

private void producer0BttActionPerformed(java.awt.event.ActionEvent evt)
{
    if(producer0Btt.isSelected())
    {
        stop[consumer.length].close();
    }
    else
    {
        stop[consumer.length].open();
```

```
    }
    if (bttProducer())
        producerStop.setSelected(true);
    else
        producerStop.setSelected(false);
}

private void producer1BttActionPerformed(java.awt.event.ActionEvent evt)
{
    if(producer1Btt.isSelected())
    {
        stop[consumer.length+1].close();
    }
    else
    {
        stop[consumer.length+1].open();
    }
    if (bttProducer())
        producerStop.setSelected(true);
    else
        producerStop.setSelected(false);
}

private void producer2BttActionPerformed(java.awt.event.ActionEvent evt)
{
    if(producer2Btt.isSelected())
    {
        stop[consumer.length+2].close();
    }
    else
    {
        stop[consumer.length+2].open();
    }
    if (bttProducer())
```

```
        producerStop.setSelected(true);
    else
        producerStop.setSelected(false);
}

private void producer3BttActionPerformed(java.awt.event.ActionEvent evt)
{
    if(producer3Btt.isSelected())
    {
        stop[consumer.length+3].close();
    }
    else
    {
        stop[consumer.length+3].open();
    }
    if (bttProducer())
        producerStop.setSelected(true);
    else
        producerStop.setSelected(false);
}

private void producer4BttActionPerformed(java.awt.event.ActionEvent evt)
{
    if(producer4Btt.isSelected())
    {
        stop[consumer.length+4].close();
    }
    else
    {
        stop[consumer.length+4].open();
    }
    if (bttProducer())
        producerStop.setSelected(true);
    else
```

```
        producerStop.setSelected(false);
    }

    private void consumer0BttActionPerformed(java.awt.event.ActionEvent evt)
    {
       if(consumer0Btt.isSelected())
       {
          stop[0].close();
       }
       else
       {
          stop[0].open();
       }
       if (bttConsumer())
          consumerStop.setSelected(true);
       else
          consumerStop.setSelected(false);
    }

    private void consumer1BttActionPerformed(java.awt.event.ActionEvent evt)
    {
       if(consumer1Btt.isSelected())
       {
          stop[1].close();
       }
       else
       {
          stop[1].open();
       }
       if (bttConsumer())
          consumerStop.setSelected(true);
       else
          consumerStop.setSelected(false);
    }
```

```
private void consumer2BttActionPerformed(java.awt.event.ActionEvent evt)
{
   if(consumer2Btt.isSelected())
   {
      stop[2].close();
   }
   else
   {
      stop[2].open();
   }
   if (bttConsumer())
      consumerStop.setSelected(true);
   else
      consumerStop.setSelected(false);
}

private void producerStopActionPerformed(java.awt.event.ActionEvent evt)
{
   if (producerStop.isSelected())
   {
      for (int i = consumer.length; i < consumer.length+producer.length; i++)
      {
         stop[i].close();
      }
      producer0Btt.setSelected(true);
      producer1Btt.setSelected(true);
      producer2Btt.setSelected(true);
      producer3Btt.setSelected(true);
      producer4Btt.setSelected(true);
   }
   else
   {
      for (int i = consumer.length; i < consumer.length+producer.length; i++)
```

```
        {
            stop[i].open();
        }
        producer0Btt.setSelected(false);
        producer1Btt.setSelected(false);
        producer2Btt.setSelected(false);
        producer3Btt.setSelected(false);
        producer4Btt.setSelected(false);
    }
}

private void consumerStopActionPerformed(java.awt.event.ActionEvent evt)
{
    if (consumerStop.isSelected())
    {
        for (int i = 0; i < consumer.length; i++)
        {
            stop[i].close();
        }
        consumer0Btt.setSelected(true);
        consumer1Btt.setSelected(true);
        consumer2Btt.setSelected(true);
    }
    else
    {
        for (int i = 0; i < consumer.length; i++)
        {
            stop[i].open();
        }
        consumer0Btt.setSelected(false);
        consumer1Btt.setSelected(false);
        consumer2Btt.setSelected(false);
    }
}
```

```
private void producer0ChkActionPerformed(java.awt.event.ActionEvent evt)
{
   producer0Chk.setSelected(true);
   producer[0].setRandomProductionRate();
}

private void producer1ChkActionPerformed(java.awt.event.ActionEvent evt)
{
   producer1Chk.setSelected(true);
   producer[1].setRandomProductionRate();
}

private void producer2ChkActionPerformed(java.awt.event.ActionEvent evt)
{
   producer2Chk.setSelected(true);
   producer[2].setRandomProductionRate();
}

private void producer3ChkActionPerformed(java.awt.event.ActionEvent evt)
{
   producer3Chk.setSelected(true);
   producer[3].setRandomProductionRate();
}

private void producer4ChkActionPerformed(java.awt.event.ActionEvent evt)
{
   producer4Chk.setSelected(true);
   producer[4].setRandomProductionRate();
}

private void consumer0ChkActionPerformed(java.awt.event.ActionEvent evt)
{
   consumer0Chk.setSelected(true);
```

```
    consumer[0].setRandomProductionRate();
}

private void consumer1ChkActionPerformed(java.awt.event.ActionEvent evt)
{
    consumer1Chk.setSelected(true);
    consumer[1].setRandomProductionRate();
}

private void consumer2ChkActionPerformed(java.awt.event.ActionEvent evt)
{
    consumer2Chk.setSelected(true);
    consumer[2].setRandomProductionRate();
}

private void producer0SldStateChanged(javax.swing.event.ChangeEvent evt)
{
    producer0Chk.setSelected(false);
    producer[0].setFixedProductionRate(-producer0Sld.getValue());
}

private void producer1SldStateChanged(javax.swing.event.ChangeEvent evt)
{
    producer1Chk.setSelected(false);
    producer[1].setFixedProductionRate(-producer1Sld.getValue());
}

private void producer2SldStateChanged(javax.swing.event.ChangeEvent evt)
{
    producer2Chk.setSelected(false);
    producer[2].setFixedProductionRate(-producer2Sld.getValue());
}

private void producer3SldStateChanged(javax.swing.event.ChangeEvent evt)
```

```
{
    producer3Chk.setSelected(false);
    producer[3].setFixedProductionRate(-producer3Sld.getValue());
}

private void producer4SldStateChanged(javax.swing.event.ChangeEvent evt)
{
    producer4Chk.setSelected(false);
    producer[4].setFixedProductionRate(-producer4Sld.getValue());
}

private void consumer0SldStateChanged(javax.swing.event.ChangeEvent evt)
{
    consumer0Chk.setSelected(false);
    consumer[0].setFixedProductionRate(-consumer0Sld.getValue());
}

private void consumer1SldStateChanged(javax.swing.event.ChangeEvent evt)
{
    consumer1Chk.setSelected(false);
    consumer[1].setFixedProductionRate(-consumer1Sld.getValue());
}

private void consumer2SldStateChanged(javax.swing.event.ChangeEvent evt)
{
    consumer2Chk.setSelected(false);
    consumer[2].setFixedProductionRate(-consumer2Sld.getValue());
}

private void listValueChanged(javax.swing.event.ListSelectionEvent evt)
{
    stopPrinting();
    switch(list.getSelectedIndex())
    {
```

```
        case 0:
            buffer.beguinPrinting();
        break;
        case 1:
            producer[0].beguinPrinting();
        break;
        case 2:
            producer[1].beguinPrinting();
        break;
        case 3:
            producer[2].beguinPrinting();
        break;
        case 4:
            producer[3].beguinPrinting();
        break;
        case 5:
            producer[4].beguinPrinting();
        break;
        case 6:
            consumer[0].beguinPrinting();
        break;
        case 7:
            consumer[1].beguinPrinting();
        break;
        case 8:
            consumer[2].beguinPrinting();
        break;
    }
}



// Variables declaration - do not modify
private javax.swing.JProgressBar bufferBar;
private javax.swing.JProgressBar bufferDerivate;
```

```
private javax.swing.JTextArea bufferTxt;
private javax.swing.JProgressBar consumer0Bar;
private javax.swing.JToggleButton consumer0Btt;
private javax.swing.JCheckBox consumer0Chk;
private javax.swing.JSlider consumer0Sld;
private javax.swing.JLabel consumer0Txt;
private javax.swing.JProgressBar consumer1Bar;
private javax.swing.JToggleButton consumer1Btt;
private javax.swing.JCheckBox consumer1Chk;
private javax.swing.JSlider consumer1Sld;
private javax.swing.JLabel consumer1Txt;
private javax.swing.JProgressBar consumer2Bar;
private javax.swing.JToggleButton consumer2Btt;
private javax.swing.JCheckBox consumer2Chk;
private javax.swing.JSlider consumer2Sld;
private javax.swing.JLabel consumer2Txt;
private javax.swing.JToggleButton consumerStop;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel10;
private javax.swing.JPanel jPanel11;
private javax.swing.JPanel jPanel14;
private javax.swing.JPanel jPanel15;
private javax.swing.JPanel jPanel16;
private javax.swing.JPanel jPanel17;
private javax.swing.JPanel jPanel18;
private javax.swing.JPanel jPanel19;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel20;
private javax.swing.JPanel jPanel21;
private javax.swing.JPanel jPanel22;
private javax.swing.JPanel jPanel23;
private javax.swing.JPanel jPanel24;
private javax.swing.JPanel jPanel25;
private javax.swing.JPanel jPanel26;
```

```
private javax.swing.JPanel jPanel27;
private javax.swing.JPanel jPanel28;
private javax.swing.JPanel jPanel29;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel30;
private javax.swing.JPanel jPanel31;
private javax.swing.JPanel jPanel32;
private javax.swing.JPanel jPanel33;
private javax.swing.JPanel jPanel34;
private javax.swing.JPanel jPanel35;
private javax.swing.JPanel jPanel36;
private javax.swing.JPanel jPanel37;
private javax.swing.JPanel jPanel38;
private javax.swing.JPanel jPanel39;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel40;
private javax.swing.JPanel jPanel41;
private javax.swing.JPanel jPanel42;
private javax.swing.JPanel jPanel43;
private javax.swing.JPanel jPanel44;
private javax.swing.JPanel jPanel45;
private javax.swing.JPanel jPanel46;
private javax.swing.JPanel jPanel47;
private javax.swing.JPanel jPanel48;
private javax.swing.JPanel jPanel49;
private javax.swing.JPanel jPanel5;
private javax.swing.JPanel jPanel50;
private javax.swing.JPanel jPanel6;
private javax.swing.JPanel jPanel7;
private javax.swing.JPanel jPanel8;
private javax.swing.JPanel jPanel9;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
```

```
    private javax.swing.JList<String> list;
    private javax.swing.JProgressBar producer0Bar;
    private javax.swing.JToggleButton producer0Btt;
    private javax.swing.JCheckBox producer0Chk;
    private javax.swing.JSlider producer0Sld;
    private javax.swing.JLabel producer0Txt;
    private javax.swing.JProgressBar producer1Bar;
    private javax.swing.JToggleButton producer1Btt;
    private javax.swing.JCheckBox producer1Chk;
    private javax.swing.JSlider producer1Sld;
    private javax.swing.JLabel producer1Txt;
    private javax.swing.JProgressBar producer2Bar;
    private javax.swing.JToggleButton producer2Btt;
    private javax.swing.JCheckBox producer2Chk;
    private javax.swing.JSlider producer2Sld;
    private javax.swing.JLabel producer2Txt;
    private javax.swing.JProgressBar producer3Bar;
    private javax.swing.JToggleButton producer3Btt;
    private javax.swing.JCheckBox producer3Chk;
    private javax.swing.JSlider producer3Sld;
    private javax.swing.JLabel producer3Txt;
    private javax.swing.JProgressBar producer4Bar;
    private javax.swing.JToggleButton producer4Btt;
    private javax.swing.JCheckBox producer4Chk;
    private javax.swing.JSlider producer4Sld;
    private javax.swing.JLabel producer4Txt;
    private javax.swing.JToggleButton producerStop;
    private javax.swing.JTextArea txt;
    // End of variables declaration
}
```