PECL2

PROGRAMA EXPOSICIÓN, CON CONTROL REMOTO.

Juan Casado Ballesteros 0908762A
GII Programación Avanzada
Laboratorio 08:00 a 12:00

Índice

PECL1

Tal y como se especificaba para la realización de esta práctica se parte de la PECL1 realizada con anterioridad, sobre ella no se ha hecho ningún cambio significativo ni en cuanto a estructura ni en cuanto a código.
Para permitir las nuevas funcionalidades requeridas ahora en la interface sobre la que se muestra el estado de los productores, los consumidores y el buffer se muestra también la IP que tenga el equipo que esté ejecutando el código en ese momento, así los usuarios remotos podrán saber dónde conectarse.

PECL2

En base a lo realizado en la práctica anterior se han añadido nuevas funcionalidades a la práctica para que otros usuarios puedan acceder a ella de forma remota y controlarla a partir de un módulo de control.
Se ha creado la interface gráfica para permitir la conexión, el módulo de control desde el cual parar o reanudar la ejecución del código y desde el que también se puede ver el estado de ejecución de los productores y consumidores ya que se pueden parar todos a la vez o de forma independiente cada uno. Se han creado también seis clases desde las que se se controla el envío de datos, la recepción de los mismos y el cierre y reapertura de los streams y sockets necesarios para la realización de la comunicación.

Para realizar la práctica se utilizan sockets y se permite hasta un máximo de 10 conexiones simultáneas.

Server

Esta clase es la encargada de administrar las conexiones, a grandes rasgos se compone de un hilo que se queda escuchando sobre un puerto para crear objetos del tipo ServerSocket cada vez que un cliente intente conectarse a él.
Una vez conectado se crearán nuevos objetos que explicaremos a continuación cuya finalizar es gestionar la entrada y salida de datos. Dichos objetos heredan de Thread y se ejecutan en un pool de hilos contenido en el servidor.
A cada nueva conexión creada se le asigna un identificador que la referencia de forma única, de este modo nos podremos comunicar con cada cliente sabiendo con quien lo hacemos.
Se incluyen métodos para llamar desde el programa de la PECL1 al envío de datos, para desde el Servidor comunicarnos con el código de la PECL1 así como algunos otros para gestionar la conexión como eliminar un cliente cuando decida desconectarse o eliminarlos a todos cuando termina la ejecución.

ServerSender

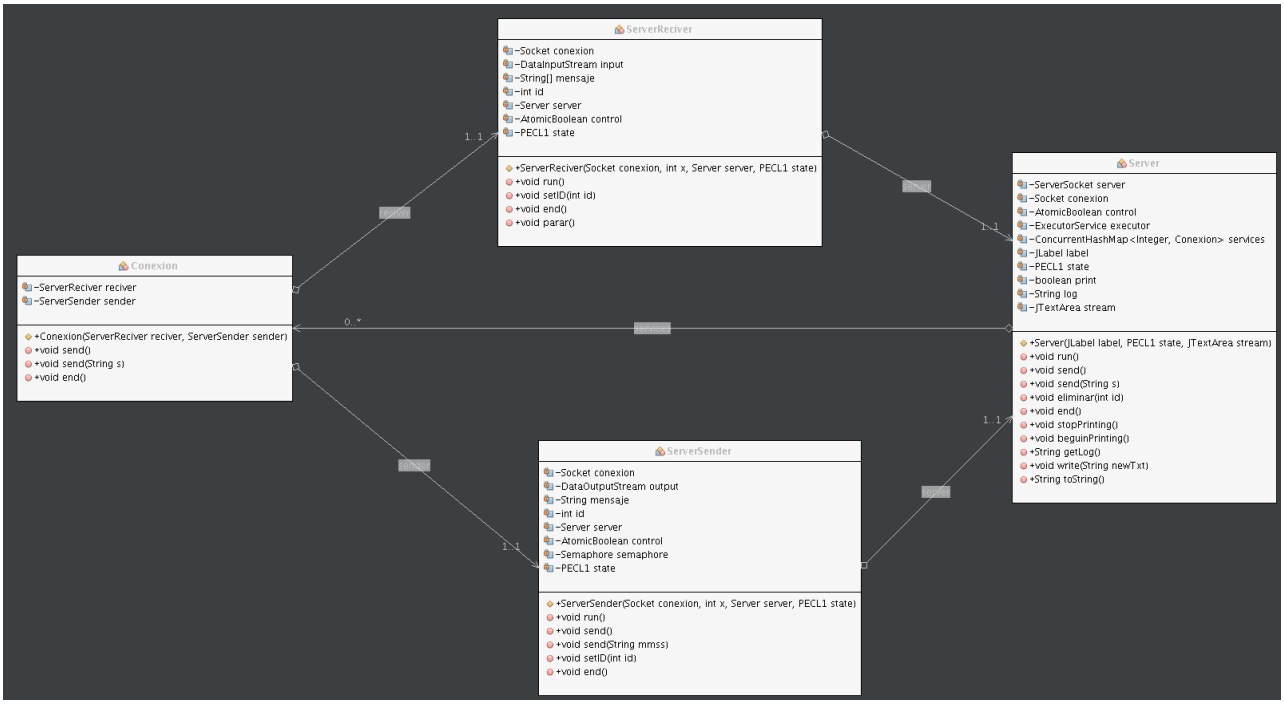
Desde esta clase gestionamos los envíos de datos, cuando se crea se inician los canales de comunicación para el envío de datos.
Se ha implementado como un hilo que espera una llamada desde el código de la PECL1 al método send(), este método es llamado cada vez que se modifican los botones. De forma concurrente al resto del programa se tomará el estado de los botones y se enviará al cliente correspondiente.
También permite enviar textos concretos para controlar al cliente como “X” para finalizar su ejecución.

ServerReciver

Desde esta clase gestionamos la recepción de datos, cuando se crea se inician los canales de comunicación necesarios para ello.
Consta de un hilo que espera la llegada de datos desde un cliente concreto, cuando estos llegan se procesan y se envían al código de la PECL1 para que cambie el estado de sus botones y pare o reanude los productores y consumidores que sean necesarios.
Podrá recibir una "X" para finalizar los canales que nos comunican con ese cliente concreto.

Conexión

Esta clase contiene al emisor y el receptor hermanos de una misma comunicación con un cliente de forma que no tengamos que usarlos ambos como objetos separados si no que podamos utilizar esta clase para que actúe de interface entra ambas.



La idea es que el Server crea un ServerSender y un ServerReciver, los hermana mediante una id única y común, lanza los hilos que contienen en un pool de hilos y une en una clase Conexión, de este modo el servidor se comunica simultáneamente con el emisor y el receptor simultáneamente a través de esta clase. El ServerSender y el ServerReciver pueden comiscares con el servidor pues contienen una referencia de este.

ClientReciver

Cuando se crea desde la clase PECL2 inicia todos los canales necesarios para la recepción de datos.
Contiene un holo que espera la recepción de datos por parte del cliente, los procesa y se los pasa a PECL2 para que los muestre sobre el menú de control.
Adicionalmente tiene un método para cerrar los canales de comunicación y finalizar su hilo de forma adecuada si se cierra PECL2 o si se recibe una "X" del servidor, el ClientReciver cierra también al ClientSender

ClientSender

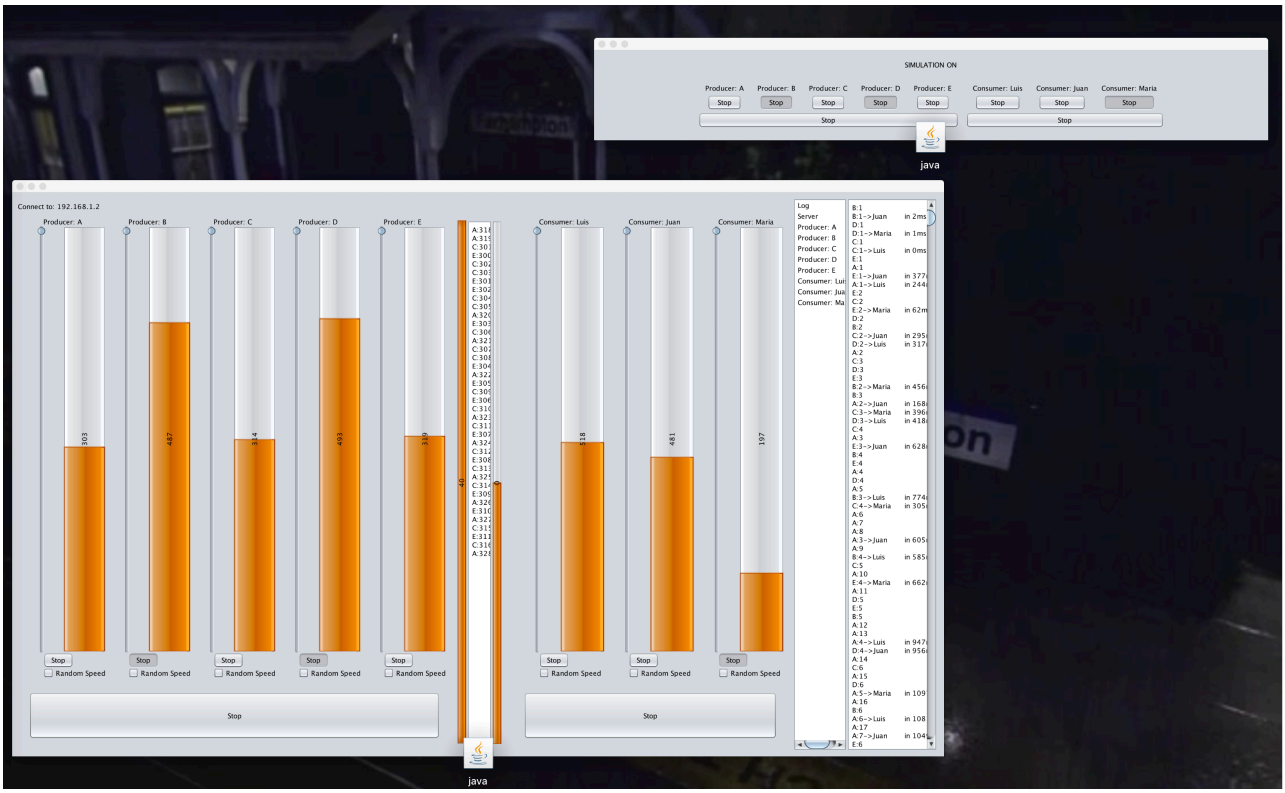
Cuando se crea desde la clase PECL2 inicia todos los canales necesarios para el envío de datos.
Contiene un hilo que espera un llamada a send() para enviar el estado de los botones de PECL2 al servidor.

PECL2

Esta clase se corresponde con la interface gráfica en la que se muestra el estado de los botones.
Esta clase además crea y controla el ClientSender y el ClientReciver con los que interactúa.

- El procesamiento de la aplicación está en el servidor, el cliente solo dice qué botones se pulsan en el caso de pulsar el botón de pausar todos los productores en el cliente solo quedaría marcado como pulsado ese, se envía esa información al servidor quien la procesa y devuelve al cliente el estado de todos los botones, en este caso "hundir" todos los botones de los productores.
- Cuando la simulación termina los clientes no son desconectados, es más, se siguen aceptando nuevos clientes hasta que se cierra la interface gráfica que muestra el estado del programa en el servidor, esto está hecho así a posta para poder seguir viendo como la comunicación funciona correctamente. Está comentado en el código la parte que desconectaría a los clientes cuando la simulación termina, por ahora solo les avisa de que lo ha hecho para que lo muestren

Este es un ejemplo de cliente y servidor ejecutándose a la vez.
Recordemos que el programa puede llegar a admitir hasta 10 clientes conectados de forma simultánea.



RemoteConexion

Esta clase crea una conexión remota, es decir, comprueba que la IP introducida se alcanzable, (puede durar un rato si no lo es) y se conecta a ella, lanza entonces una instancia de PECL2

Anexo de Código

Buffer

```
package pecl2.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;
import pecl2.screens.PECL1;

/**
 *
 * @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;
    private int ended;
    private final PECL1 state;
```

```
    /**
     *
     * @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, PECL1 state)
    {
        this.ended = 0;
        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;
        this.state = state;
    }

    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;
    }

    public synchronized void end()
    {
        ended++;
        if (ended >= 8)
            state.finish();
    }
}
```

ClientReciver

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pecl2.classes;

import java.io.DataInputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.concurrent.atomic.AtomicBoolean;
import pecl2.screens.PECL2;

/**
 *
 * @author mr.blissfulgrin
 */
public class ClientReciver extends Thread
{
    private final Socket client;
    private DataInputStream input;
    private final AtomicBoolean control;
    private final PECL2 state;

    public ClientReciver (Socket socket, PECL2 state)
    {
        this.client = socket;
        this.control = new AtomicBoolean (true);
        this.state = state;
        try
        {
            input = new DataInputStream(client.getInputStream());
        }
        catch (IOException e){}
    }

    /**
     * Hilo que espera a la recepción de datos, los procesa y se los comunica a la interface gráfica
     */
    @Override
    public void run ()
```

```
    {
        String [] ms;
        boolean [] data = new boolean [11];
        while(control.get())
        {
            try
            {
                ms = input.readUTF().split(" ");
                switch (ms[0])
                {
                    case "X":
                        state.end();
                        break;
                    default:
                        for (int x = 0; x < ms.length; x++)
                        {
                            data [x] = ms [x].equals("1");
                        }
                        state.setState(data);
                        break;
                }
            }
            catch (IOException e){}
        }
    }

    /**
     * Cierra los canales de comunicación y finaliza el hilo
     */
    public void parar ()
    {
        try
        {
            control.set(false);
            input.close();
            client.close();
        }
        catch (IOException ex){}
    }
}
```

ClientSender

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pecl2.classes;

import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.concurrent.Semaphore;
import java.util.concurrent.atomic.AtomicBoolean;
import pecl2.screens.PECL2;

/**
 *
 * @author mr.blissfulgrin
 */
public class ClientSender extends Thread
{
    private final Socket client;
    private DataOutputStream output;
    private final AtomicBoolean control;
    private final PECL2 state;
    private final Semaphore semaphore;

    public ClientSender (Socket socket, PECL2 state)
    {
        this.client = socket;
        this.control = new AtomicBoolean (true);
        this.state = state;
        this.semaphore = new Semaphore(0);
        try
        {
            output = new DataOutputStream(client.getOutputStream());
        }
        catch (IOException e){}
    }

    /**
     * Espera una llamada a send() para enviar los datos del estado de los botones de PECL2
     */
```

```
@Override
public void run ()
{
   String  mensaje;
   boolean [] data;
   while (control.get())
   {
      try
      {
         semaphore.acquire(1);
         mensaje = "";
         data = state.getButtonState();

         for (boolean b : data)
         {
            if (b)
               mensaje += "1";
            else
               mensaje += "0";
            mensaje += " ";
         }
         output.writeUTF(mensaje);
      }
      catch (IOException | InterruptedException e){}
   }
}

/**
 * Produce el envío de datos
 */
public void send ()
{
   semaphore.release(1);
}

/**
 * Finaliza la comunicación y termina el hilo
 */
public void parar ()
{
   try
   {
      if (!state.isEnded())
         output.writeUTF("X");
```

```
            control.set(false);
            semaphore.release(1);
            output.close();
            client.close();
        }
        catch (IOException ex){}
    }
}
```

Conexion

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pecl2.classes;

/**
 *
 * @author mr.blissfulgrin
 */
public class Conexion
{
    private final ServerReciver reciver;
    private final ServerSender sender;

    public Conexion (ServerReciver reciver, ServerSender sender)
    {
        this.reciver = reciver;
        this.sender = sender;
    }

    /*
    * Métodos desde los que controlar al emisor
    */
    public void send()
    {
        sender.send();
    }
    public void send(String s)
    {
        sender.send(s);
    }

    /**
     * Finaliza la comunicación
     */
    public void end()
    {
        sender.send("X");
        reciver.end();
        sender.end();
```

```
    }
}
```

Consumer

```
package pecl2.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");
        buffer.end();
    }

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

Package

```
package pecl2.classes;

import java.util.Date;

/**
 *
 * @author mr.blissfulgrin
 */
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 pecl2.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");
   buffer.end();
}
```

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

Server

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pecl2.classes;

import java.io.IOException;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicBoolean;
import javax.swing.JLabel;
import javax.swing.JTextArea;
import pecl2.screens.PECL1;

/**
 *
 * @author mr.blissfulgrin
 */
public class Server extends Thread
{
    private ServerSocket server;
    private Socket conexion;
    private final AtomicBoolean control;
    private final ExecutorService executor;
    private final ConcurrentHashMap <Integer,Conexion> services;

    private final JLabel label;
    private final PECL1 state;

    private boolean print;
    private String log;
    private final JTextArea stream;

    public Server (JLabel label, PECL1 state, JTextArea stream)
    {
        control = new AtomicBoolean(true);
```

```
        executor = Executors.newFixedThreadPool(20);
        services = new ConcurrentHashMap <> ();
        this.label = label;
        this.state = state;
        this.print = false;
        this.log = "";
        this.stream = stream;
    }

    /**
     * Escuchamos nuevas conexiones de clientes y creamos los objetos necesarios para gestionarlas.
     * La comunicación será full duplex controlado el envío de datos y su recepción por hilos distintos.
     * A cada cliente se le asigna una identidad aleatoria que lo identifica para poder comunicarnos con cada uno de forma individual
     */
    @Override
    public void run()
    {
        try
        {
            int id;
            server = new ServerSocket(4444);
            label.setText("Connect to: " + InetAddress.getLocalHost().getHostAddress());
            while (control.get())
            {
                do
                {
                    id = (int)(Math.random()*(Integer.MAX_VALUE-1)+1);
                }
                while (services.containsKey(id));
                conexion = server.accept();
                write("NEW CLIENT "+id);
                ServerSender s = new ServerSender (conexion,id,this,state);
                ServerReciver r = new ServerReciver (conexion,id,this,state);
                Conexion c = new Conexion (r,s);
                services.put(id, c);
                executor.execute(r);
                executor.execute(s);
            }
        }
        catch (IOException e)
        {
```

```
            label.setText("SERVER ERROR");
        }
    }

    /**
    * Enviamos los datos del estado de los botones a los clientes conectados
    */
    public synchronized void send ()
    {
        services.values().forEach((c) ->
        {
            c.send();
        });
    }

    /**
     * Enviamos un texto a los clientes que será decodificado para controlarlos desde el Servidor
    * X -> cerrar el cliente
    * @param s Texto a enviar a los clientes
    */
    public synchronized void send (String s)
    {
        services.values().forEach((c) ->
        {
            c.send(s);
        });
    }

    /**
     * Elimina un cliente con la identidad dada
      * @param id Identidad del cliente que se desea eliminar, es asiganada de forma aleatoria por el servidor
     */
    public synchronized void eliminar (int id)
    {
        if (services.containsKey(id))
        {
            services.get(id).end();
            services.remove(id);
            write("END CONNECTION "+id);
        }
        else
        {
```

```
            System.out.println("ERROR");
        }
    }

    /**
     * Finaliza la ejecición del servidor cerrando los streams y sockets abiertos así como finalizando las tareas
     */
    public synchronized void end ()
    {
        try
        {
            services.keySet().forEach((c) ->
            {
                services.get(c).end();
                write("END CONNECTION " + c);
            });
            control.set(false);
            server.close();
            executor.shutdown();
            executor.awaitTermination(30, TimeUnit.SECONDS);
            executor.shutdownNow();
        }
        catch (IOException | InterruptedException ex)
        {
            label.setText("SERVER COULDN'T STOP");
        }
    }

    /*
     * Métodos para controlar la escitura de datos desde el código de la PECL1
     */
    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)
    {
        log += newTxt+"\n";
        if (print)
            stream.setText(log);
    }

    @Override
    public String toString()
    {
        return "Server";
    }
}
```

ServerReciver

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pecl2.classes;

import java.io.DataInputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.concurrent.atomic.AtomicBoolean;
import pecl2.screens.PECL1;

/**
 *
 * @author mr.blissfulgrin
 */
public class ServerReciver extends Thread
{
    private final Socket conexion;
    private final DataInputStream input;
    private String [] mensaje;
    private int id;
    private final Server server;
    private final AtomicBoolean control;
    private final PECL1 state;

        public ServerReciver (Socket conexion, int x,Server server,PECL1 state) throws
IOException
    {
        input = new DataInputStream(conexion.getInputStream());
        this.id = x;
        this .conexion = conexion;
        this.server = server;
        this.control = new AtomicBoolean(true);

        this.state = state;
    }
    /**
     * Este hilo escucha los datos que los clientes envían al servidor cada vez que pulsan un
botón
     */
```

```
@Override
public void run ()
{
   boolean [] data = new boolean [11];
   while (control.get())
   {
      try
      {
         mensaje = input.readUTF().split(" ");
         if (mensaje[0].equals("X"))
         {
            this.parar();
         }
         else
         {
            for (int x = 0; x < mensaje.length; x++)
            {
               data [x] = mensaje [x].equals("1");
            }
            state.setButtonState(data);
         }
      }
      catch (IOException e){}
   }
}

public void setID(int id)
{
   this.id = id;
}

/**
 * Cierra el canal de comunicación y finaliza el hilo
 */
public void end ()
{
   control.set(false);
   try
   {
      input.close();
      conexion.close();
   } catch (IOException ex){}
}
```

```
    /**
     * Notifica a un remisor que debe cerrarse, el emisor hermanado con este recerptor (Misma ID)
     */
    public void parar ()
    {
        server.eliminar(id);
    }
}
```

ServerSender

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pecl2.classes;

import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.concurrent.Semaphore;
import java.util.concurrent.atomic.AtomicBoolean;
import pecl2.screens.PECL1;

/**
 *
 * @author mr.blissfulgrin
 */
public class ServerSender extends Thread
{
    private final Socket conexion;
    private final DataOutputStream output;
    private String mensaje;
    private int id;
    private final Server server;
    private final AtomicBoolean control;
    private final Semaphore semaphore;

    private final PECL1 state;

        public ServerSender (Socket conexion, int x,Server server, PECL1 state) throws
IOException
    {
        output = new DataOutputStream(conexion.getOutputStream());
        this.id = x;
        this .conexion = conexion;
        this.server = server;
        this.control = new AtomicBoolean(true);
        this.semaphore = new Semaphore(0);
        this.state = state;
    }
```

```
/**
 * Se envían datos al cliente enlazado a esta clase que describen el estado de los botones
 * El envío se produce a cada llamada de send()
 */
@Override
public void run ()
{
   boolean [] data;

   while (control.get())
   {
      try
      {
         mensaje = "";
         data = state.getButtonState();
         for (boolean b : data)
         {
            if (b)
               mensaje += "1";
            else
               mensaje += "0";
            mensaje += " ";
         }
         output.writeUTF(mensaje);

         semaphore.acquire(1);
      }
      catch (InterruptedException | IOException ex){}
   }
}

/**
 * Cada vez que se lanza este método se envían datos
 */
public void send ()
{
   semaphore.release(1);
}

/**
 * Permite el envío de datos concretos para controlar al cliente
 * @param mmss String de datos a enviar
 */
public void send (String mmss)
```

```
    {
        try
        {
            output.writeUTF(mmss);
        }
        catch (IOException ex){}
    }

    /**
     * Dota de identidad a la clase
     * @param id Identificador de la comunicación
     */
    public void setID(int id)
    {
        this.id = id;
    }

    /**
     * Finaliza la ejecición del hilo y cierra los sockets y streams utilizados
     */
    public void end ()
    {
        control.set(false);
        semaphore.release(1);
        try
        {
            output.close();
            conexion.close();
        } catch (IOException ex){}
    }
}
```

Stop

```
package pecl2.classes;

import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

/**
 *
 * @author mr.blissfulgrin
 */
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();
    }
}
}
```

Generator

```
package pecl2.screens;

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

    public Generator()
    {
        initComponents();
    }

    @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();
        jSlider1 = new javax.swing.JSlider();
        jPanel3 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        remote = 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));
        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.GridBagLayout());

jButton1.setText("Generar: 99 paquetes");
jButton1.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      jButton1ActionPerformed(evt);
   }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.weightx = 0.1;
gridBagConstraints.weighty = 0.1;
jPanel3.add(jButton1, gridBagConstraints);

remote.setText("Conexion Remota");
remote.setToolTipText("");
remote.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      remoteActionPerformed(evt);
   }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
```

```
    gridBagConstraints.gridwidth = 2;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.weightx = 0.1;
    gridBagConstraints.weighty = 0.1;
    jPanel3.add(remote, gridBagConstraints);

    jButton2.setText("Salir");
    jButton2.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
            jButton2ActionPerformed(evt);
        }
    });
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.weightx = 0.1;
    gridBagConstraints.weighty = 0.1;
    jPanel3.add(jButton2, gridBagConstraints);

    jPanel1.add(jPanel3);

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

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

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

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");
    }

    private void remoteActionPerformed(java.awt.event.ActionEvent evt)
    {
        RemoteConexion c = new RemoteConexion();
        c.setVisible(true);
        this.dispose();
    }

    /**
     * @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;
    private javax.swing.JButton remote;
    // End of variables declaration
}
```

Load

```
package pecl2.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 pecl2.screens;

import java.util.concurrent.Semaphore;
import java.util.concurrent.atomic.AtomicBoolean;
import pecl2.classes.Stop;
import pecl2.classes.Buffer;
import pecl2.classes.Producer;
import pecl2.classes.Consumer;
import javax.swing.JProgressBar;
import pecl2.classes.Server;


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

    private final Producer [] producer;
    private final Consumer [] consumer;
    private Buffer buffer;
    private final Stop [] stop;
    private final Server server;
    private final Semaphore semaphrore;
    private final AtomicBoolean finishd;

    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,this);

    //CREAR
    this.server = new Server (ip, this, txt);
    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+2];
    data[0] = "Log";
    data[1] = "Server";

    for (int i = 2; i < producer_id.length+2; i++)
    {
       data[i] = producer[i-2].toString();
    }
    for (int i = producer_id.length+2; i < producer_id.length+2+consumer_id.length; i++)
    {
       data[i] = consumer[(i-2)-producer_id.length].toString();
    }

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

```
        this.semaphrore = new Semaphore (1);

        this.finishd = new AtomicBoolean(false);
        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();
        }
        server.start();
    }

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

```
}

public synchronized boolean [] getButtonState ()
{
   try
   {
      semaphrore.acquire(1);
   }
   catch (InterruptedException ex){}

   boolean [] state = new boolean [11];
   state [0] = producer0Btt.isSelected();
   state [1] = producer1Btt.isSelected();
   state [2] = producer2Btt.isSelected();
   state [3] = producer3Btt.isSelected();
   state [4] = producer4Btt.isSelected();
   state [5] = consumer0Btt.isSelected();
   state [6] = consumer1Btt.isSelected();
   state [7] = consumer2Btt.isSelected();
   state [8] = producerStop.isSelected();
   state [9] = consumerStop.isSelected();
   state[10] = finishd.get();
   semaphrore.release(1);

   return state;
}

public synchronized void setButtonState (boolean [] state)
{
   try
   {
      semaphrore.acquire(1);
   }
   catch (InterruptedException ex){}

   producer0Btt.setSelected(state[10]?state[0]:state[8]);
   producer1Btt.setSelected(state[10]?state[1]:state[8]);
   producer2Btt.setSelected(state[10]?state[2]:state[8]);
   producer3Btt.setSelected(state[10]?state[3]:state[8]);
   producer4Btt.setSelected(state[10]?state[4]:state[8]);
   consumer0Btt.setSelected(state[10]?state[5]:state[9]);
   consumer1Btt.setSelected(state[10]?state[6]:state[9]);
   consumer2Btt.setSelected(state[10]?state[7]:state[9]);
```

```
                                        producerStop.setSelected(state[10]?
(state[0]&&state[1]&&state[2]&&state[3]&&state[4]):state[8]);
    consumerStop.setSelected(state[10]?(state[5]&&state[6]&&state[7]):state[9]);

    if(state[10]?state[0]:state[8])
    {
       stop[consumer.length].close();
    }
    else
    {
       stop[consumer.length].open();
    }
    if(state[10]?state[1]:state[8])
    {
       stop[consumer.length+1].close();
    }
    else
    {
       stop[consumer.length+1].open();
    }
    if(state[10]?state[2]:state[8])
    {
       stop[consumer.length+2].close();
    }
    else
    {
       stop[consumer.length+2].open();
    }
    if(state[10]?state[3]:state[8])
    {
       stop[consumer.length+3].close();
    }
    else
    {
       stop[consumer.length+3].open();
    }
    if(state[10]?state[4]:state[8])
    {
       stop[consumer.length+4].close();
    }
    else
    {
       stop[consumer.length+4].open();
    }
```

```
        if(state[10]?state[5]:state[9])
        {
            stop[0].close();
        }
        else
        {
            stop[0].open();
        }
        if(state[10]?state[6]:state[9])
        {
            stop[1].close();
        }
        else
        {
            stop[1].open();
        }
        if(state[10]?state[7]:state[9])
        {
            stop[2].close();
        }
        else
        {
            stop[2].open();
        }

        semaphrore.release(1);

        server.send();
    }

    public void finish()
    {
        finishd.set(true);
        server.send();
        //server.end();
        /*
        Terminaría la ejecución del servidor (impidiendo nuevas conexiones)
         Ahora notifica a los clientes que la ejecución ha finalizado pero los mantiene activos cuando la
        simulación finaliza de modo que siguen pudiendo modificar el estado de los botones
        */
    }

    /**
```

```
     * 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();
        ip = new javax.swing.JLabel();
        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.gridx = 0;
        gridBagConstraints.gridy = 1;
        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.HORIZONTA
L_SCROLLBAR_NEVER);

jScrollPane3.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.VERTICAL_SCR
OLLBAR_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.gridx = 1;
        gridBagConstraints.gridy = 1;
        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, 499, Short.MAX_VALUE)
         .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
                                  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel9Layout.createSequentialGroup()
                                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(jPanel27, javax.swing.GroupLayout.PREFERRED_SIZE, 486,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addContainerGap(7, Short.MAX_VALUE)))
    );
    jPanel9Layout.setVerticalGroup(

jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 944, Short.MAX_VALUE)
         .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
                  .addComponent(jPanel27, javax.swing.GroupLayout.DEFAULT_SIZE, 944,
Short.MAX_VALUE))
    );

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 2;
    gridBagConstraints.gridy = 1;
```

```
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.weighty = 1.2;
jPanel4.add(jPanel9, gridBagConstraints);

ip.setText("IP:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.gridwidth = 3;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.insets = new java.awt.Insets(7, 0, 7, 0);
jPanel4.add(ip, 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..");
        server.end();
        Generator g = new Generator ();
        g.setVisible(true);
    }

    private void producer0BttActionPerformed(java.awt.event.ActionEvent evt)
    {
        try
        {
            semaphrore.acquire(1);
        }
```

```
    catch (InterruptedException ex){}

    if(producer0Btt.isSelected())
    {
       stop[consumer.length].close();
    }
    else
    {
       stop[consumer.length].open();
    }
    if (bttProducer())
       producerStop.setSelected(true);
    else
       producerStop.setSelected(false);

    semaphrore.release(1);

    server.send();
}

private void producer1BttActionPerformed(java.awt.event.ActionEvent evt)
{
    try
    {
       semaphrore.acquire(1);
    }
    catch (InterruptedException ex){}

    if(producer1Btt.isSelected())
    {
       stop[consumer.length+1].close();
    }
    else
    {
       stop[consumer.length+1].open();
    }
    if (bttProducer())
       producerStop.setSelected(true);
    else
       producerStop.setSelected(false);

    semaphrore.release(1);

    server.send();
```

```
}

private void producer2BttActionPerformed(java.awt.event.ActionEvent evt)
{
   try
   {
      semaphrore.acquire(1);
   }
   catch (InterruptedException ex){}

   if(producer2Btt.isSelected())
   {
      stop[consumer.length+2].close();
   }
   else
   {
      stop[consumer.length+2].open();
   }
   if (bttProducer())
      producerStop.setSelected(true);
   else
      producerStop.setSelected(false);

   semaphrore.release(1);

   server.send();
}

private void producer3BttActionPerformed(java.awt.event.ActionEvent evt)
{
   try
   {
      semaphrore.acquire(1);
   }
   catch (InterruptedException ex){}

   if(producer3Btt.isSelected())
   {
      stop[consumer.length+3].close();
   }
   else
   {
      stop[consumer.length+3].open();
   }
```

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

    semaphrore.release(1);

    server.send();
}

private void producer4BttActionPerformed(java.awt.event.ActionEvent evt)
{
    try
    {
        semaphrore.acquire(1);
    }
    catch (InterruptedException ex){}

    if(producer4Btt.isSelected())
    {
        stop[consumer.length+4].close();
    }
    else
    {
        stop[consumer.length+4].open();
    }
    if (bttProducer())
        producerStop.setSelected(true);
    else
        producerStop.setSelected(false);

    semaphrore.release(1);

    server.send();
}

private void consumer0BttActionPerformed(java.awt.event.ActionEvent evt)
{
    try
    {
        semaphrore.acquire(1);
    }
    catch (InterruptedException ex){}
```

```
    if(consumer0Btt.isSelected())
    {
        stop[0].close();
    }
    else
    {
        stop[0].open();
    }
    if (bttConsumer())
        consumerStop.setSelected(true);
    else
        consumerStop.setSelected(false);

    semaphrore.release(1);

    server.send();
}

private void consumer1BttActionPerformed(java.awt.event.ActionEvent evt)
{
    try
    {
        semaphrore.acquire(1);
    }
    catch (InterruptedException ex){}

    if(consumer1Btt.isSelected())
    {
        stop[1].close();
    }
    else
    {
        stop[1].open();
    }
    if (bttConsumer())
        consumerStop.setSelected(true);
    else
        consumerStop.setSelected(false);

    semaphrore.release(1);

    server.send();
}
```

```
private void consumer2BttActionPerformed(java.awt.event.ActionEvent evt)
{
   try
   {
      semaphrore.acquire(1);
   }
   catch (InterruptedException ex){}

   if(consumer2Btt.isSelected())
   {
      stop[2].close();
   }
   else
   {
      stop[2].open();
   }
   if (bttConsumer())
      consumerStop.setSelected(true);
   else
      consumerStop.setSelected(false);

   semaphrore.release(1);

   server.send();
}

private void producerStopActionPerformed(java.awt.event.ActionEvent evt)
{
   try
   {
      semaphrore.acquire(1);
   }
   catch (InterruptedException ex){}

   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);
        }

        semaphrore.release(1);

        server.send();
    }

    private void consumerStopActionPerformed(java.awt.event.ActionEvent evt)
    {
        try
        {
            semaphrore.acquire(1);
        }
        catch (InterruptedException ex){}

        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);
        }

        semaphrore.release(1);

        server.send();
    }

    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:
         server.beguinPrinting();
      break;
      case 2:
         producer[0].beguinPrinting();
      break;
      case 3:
         producer[1].beguinPrinting();
      break;
      case 4:
         producer[2].beguinPrinting();
      break;
      case 5:
         producer[3].beguinPrinting();
      break;
      case 6:
         producer[4].beguinPrinting();
      break;
```

```
            case 7:
                consumer[0].beguinPrinting();
            break;
            case 8:
                consumer[1].beguinPrinting();
            break;
            case 9:
                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.JLabel ip;
    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
}
```

PECL2

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pecl2.screens;

import java.net.Socket;
import java.util.concurrent.atomic.AtomicBoolean;
import pecl2.classes.ClientReciver;
import pecl2.classes.ClientSender;

/**
 *
 * @author mr.blissfulgrin
 */
public class PECL2 extends javax.swing.JFrame
{
    private final ClientSender s;
    private final ClientReciver r;
    private boolean lastClicked;
    private final AtomicBoolean ended;

    public PECL2(Socket socket)
    {
        new Load().go("Starting...");
        initComponents();

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

        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]);

        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");

    s = new ClientSender(socket,this);
    r = new ClientReciver(socket,this);

    lastClicked = true;
    ended = new AtomicBoolean(false);

    r.start();
    s.start();
}

public synchronized void setState (boolean [] data)
{
    producer0Btt.setSelected(data[0]);
    producer1Btt.setSelected(data[1]);
    producer2Btt.setSelected(data[2]);
    producer3Btt.setSelected(data[3]);
    producer4Btt.setSelected(data[4]);
    consumer0Btt.setSelected(data[5]);
    consumer1Btt.setSelected(data[6]);
    consumer2Btt.setSelected(data[7]);
    producerStop.setSelected(data[8]);
    consumerStop.setSelected(data[9]);
    if (data[10])
    {
        txt.setText("SIMULATION FINISHD");
    }
}

public synchronized boolean [] getButtonState ()
{
    boolean [] state = new boolean [11];
    state [0] = producer0Btt.isSelected();
    state [1] = producer1Btt.isSelected();
    state [2] = producer2Btt.isSelected();
    state [3] = producer3Btt.isSelected();
```

```
    state [4] = producer4Btt.isSelected();
    state [5] = consumer0Btt.isSelected();
    state [6] = consumer1Btt.isSelected();
    state [7] = consumer2Btt.isSelected();
    state [8] = producerStop.isSelected();
    state [9] = consumerStop.isSelected();
    state[10] = lastClicked;
    lastClicked = true;
    return state;
}

public synchronized void end ()
{
    s.parar();
    r.parar();
    txt.setText("CONNECTION CLOSED");
}

public synchronized void simulationEND ()
{
    txt.setText("SIMULATION FINISHD");
    ended.set(true);
}

public boolean isEnded()
{
    return ended.get();
}

/**
 * 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();
    producer0Btt = new javax.swing.JToggleButton();
    producer1Btt = new javax.swing.JToggleButton();
```

```
producer2Btt = new javax.swing.JToggleButton();
producer3Btt = new javax.swing.JToggleButton();
producer4Btt = new javax.swing.JToggleButton();
consumer0Btt = new javax.swing.JToggleButton();
consumer1Btt = new javax.swing.JToggleButton();
consumer2Btt = new javax.swing.JToggleButton();
producerStop = new javax.swing.JToggleButton();
consumerStop = new javax.swing.JToggleButton();
producer0Txt = new javax.swing.JLabel();
producer1Txt = new javax.swing.JLabel();
producer2Txt = new javax.swing.JLabel();
producer3Txt = new javax.swing.JLabel();
producer4Txt = new javax.swing.JLabel();
consumer0Txt = new javax.swing.JLabel();
consumer1Txt = new javax.swing.JLabel();
consumer2Txt = new javax.swing.JLabel();
txt = new javax.swing.JLabel();

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());

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

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

producer0Btt.setText("jToggleButton1");
producer0Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer0BttActionPerformed(evt);
   }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
```

```
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(4, 16, 4, 16);
jPanel2.add(producer0Btt, gridBagConstraints);

producer1Btt.setText("jToggleButton2");
producer1Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer1BttActionPerformed(evt);
   }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(4, 16, 4, 16);
jPanel2.add(producer1Btt, gridBagConstraints);

producer2Btt.setText("jToggleButton3");
producer2Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer2BttActionPerformed(evt);
   }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 2;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(4, 16, 4, 16);
jPanel2.add(producer2Btt, gridBagConstraints);

producer3Btt.setText("jToggleButton4");
producer3Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer3BttActionPerformed(evt);
   }
});
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 2;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(4, 16, 4, 16);
jPanel2.add(producer3Btt, gridBagConstraints);

producer4Btt.setText("jToggleButton5");
producer4Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      producer4BttActionPerformed(evt);
   }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 2;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(4, 16, 4, 29);
jPanel2.add(producer4Btt, gridBagConstraints);

consumer0Btt.setText("jToggleButton6");
consumer0Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      consumer0BttActionPerformed(evt);
   }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 2;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(4, 16, 4, 16);
jPanel2.add(consumer0Btt, gridBagConstraints);

consumer1Btt.setText("jToggleButton7");
consumer1Btt.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
```

```
    {
        consumer1BttActionPerformed(evt);
    }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 6;
gridBagConstraints.gridy = 2;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(4, 16, 4, 16);
jPanel2.add(consumer1Btt, gridBagConstraints);

consumer2Btt.setText("jToggleButton8");
consumer2Btt.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt)
    {
        consumer2BttActionPerformed(evt);
    }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 7;
gridBagConstraints.gridy = 2;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(4, 16, 4, 16);
jPanel2.add(consumer2Btt, gridBagConstraints);

producerStop.setText("jToggleButton9");
producerStop.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt)
    {
        producerStopActionPerformed(evt);
    }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.gridwidth = 5;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(0, 0, 22, 13);
jPanel2.add(producerStop, gridBagConstraints);
```

```
consumerStop.setText("jToggleButton10");
consumerStop.addActionListener(new java.awt.event.ActionListener()
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      consumerStopActionPerformed(evt);
   }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 3;
gridBagConstraints.gridwidth = 3;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(0, 0, 22, 0);
jPanel2.add(consumerStop, gridBagConstraints);

producer0Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
producer0Txt.setText("jLabel1");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(8, 12, 0, 12);
jPanel2.add(producer0Txt, gridBagConstraints);

producer1Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
producer1Txt.setText("jLabel2");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(8, 12, 0, 12);
jPanel2.add(producer1Txt, gridBagConstraints);

producer2Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
producer2Txt.setText("jLabel3");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
```

```
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(8, 12, 0, 12);
jPanel2.add(producer2Txt, gridBagConstraints);

producer3Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
producer3Txt.setText("jLabel4");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 1;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(8, 12, 0, 12);
jPanel2.add(producer3Txt, gridBagConstraints);

producer4Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
producer4Txt.setText("jLabel5");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 1;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(8, 12, 0, 25);
jPanel2.add(producer4Txt, gridBagConstraints);

consumer0Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
consumer0Txt.setText("jLabel6");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 1;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(8, 12, 0, 12);
jPanel2.add(consumer0Txt, gridBagConstraints);

consumer1Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
consumer1Txt.setText("jLabel7");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 6;
gridBagConstraints.gridy = 1;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(8, 12, 0, 12);
jPanel2.add(consumer1Txt, gridBagConstraints);
```

```
    consumer2Txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    consumer2Txt.setText("jLabel8");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 7;
    gridBagConstraints.gridy = 1;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
    gridBagConstraints.insets = new java.awt.Insets(8, 12, 0, 12);
    jPanel2.add(consumer2Txt, gridBagConstraints);

    txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    txt.setText("SIMULATION ON");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 0;
    gridBagConstraints.gridwidth = 8;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.insets = new java.awt.Insets(18, 18, 18, 18);
    jPanel2.add(txt, gridBagConstraints);

    jPanel1.add(jPanel2, "card2");

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

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

private void formWindowClosing(java.awt.event.WindowEvent evt)
{
    end();
    new Load().go("Quiting..");
    Generator g = new Generator ();
    g.setVisible(true);
}

private void producer0BttActionPerformed(java.awt.event.ActionEvent evt)
{
    s.send();
}

private void producer1BttActionPerformed(java.awt.event.ActionEvent evt)
{
    s.send();
```

```
}

private void producer2BttActionPerformed(java.awt.event.ActionEvent evt)
{
   s.send();
}

private void producer3BttActionPerformed(java.awt.event.ActionEvent evt)
{
   s.send();
}

private void producer4BttActionPerformed(java.awt.event.ActionEvent evt)
{
   s.send();
}

private void consumer0BttActionPerformed(java.awt.event.ActionEvent evt)
{
   s.send();
}

private void consumer1BttActionPerformed(java.awt.event.ActionEvent evt)
{
   s.send();
}

private void consumer2BttActionPerformed(java.awt.event.ActionEvent evt)
{
   s.send();
}

private void producerStopActionPerformed(java.awt.event.ActionEvent evt)
{
   s.send();
   lastClicked = false;
}

private void consumerStopActionPerformed(java.awt.event.ActionEvent evt)
{
   s.send();
   lastClicked = false;
}
```

```
    // Variables declaration - do not modify
    private javax.swing.JToggleButton consumer0Btt;
    private javax.swing.JLabel consumer0Txt;
    private javax.swing.JToggleButton consumer1Btt;
    private javax.swing.JLabel consumer1Txt;
    private javax.swing.JToggleButton consumer2Btt;
    private javax.swing.JLabel consumer2Txt;
    private javax.swing.JToggleButton consumerStop;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JToggleButton producer0Btt;
    private javax.swing.JLabel producer0Txt;
    private javax.swing.JToggleButton producer1Btt;
    private javax.swing.JLabel producer1Txt;
    private javax.swing.JToggleButton producer2Btt;
    private javax.swing.JLabel producer2Txt;
    private javax.swing.JToggleButton producer3Btt;
    private javax.swing.JLabel producer3Txt;
    private javax.swing.JToggleButton producer4Btt;
    private javax.swing.JLabel producer4Txt;
    private javax.swing.JToggleButton producerStop;
    private javax.swing.JLabel txt;
    // End of variables declaration
}
```

RemoteConexion

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pecl2.screens;

import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;

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

    /**
     * Creates new form RemoteConexion
     */
    public RemoteConexion()
    {
        initComponents();
    }

    /**
     * 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();
        ip1 = new javax.swing.JTextField();
        ip2 = new javax.swing.JTextField();
        ip3 = new javax.swing.JTextField();
```

```
ip4 = new javax.swing.JTextField();
txt = new javax.swing.JLabel();
btt = new javax.swing.JButton();

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());

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

jPanel2.setLayout(new java.awt.GridBagLayout());
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.ipadx = 70;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(42, 9, 0, 9);
jPanel2.add(ip1, gridBagConstraints);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.ipadx = 70;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(42, 9, 0, 9);
jPanel2.add(ip2, gridBagConstraints);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 0;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.ipadx = 70;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(42, 9, 0, 9);
jPanel2.add(ip3, gridBagConstraints);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 0;
```

```
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.ipadx = 70;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
    gridBagConstraints.insets = new java.awt.Insets(42, 9, 0, 9);
    jPanel2.add(ip4, gridBagConstraints);

    txt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 1;
    gridBagConstraints.gridwidth = 2;
    gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
    gridBagConstraints.insets = new java.awt.Insets(22, 9, 9, 9);
    jPanel2.add(txt, gridBagConstraints);

    btt.setText("CONECTAR");
    btt.addActionListener(new java.awt.event.ActionListener()
    {
       public void actionPerformed(java.awt.event.ActionEvent evt)
       {
          bttActionPerformed(evt);
       }
    });
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 2;
    gridBagConstraints.gridwidth = 2;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
    gridBagConstraints.insets = new java.awt.Insets(8, 8, 8, 8);
    jPanel2.add(btt, gridBagConstraints);

    jPanel1.add(jPanel2, "card2");

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

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

private void formWindowClosing(java.awt.event.WindowEvent evt)
{
    Generator g = new Generator ();
```

```
    g.setVisible(true);
}

/**
 * Intenta conectarse a la IP dada y de poder hacerlo lanza el cliente
 * @param evt
 */
private void bttActionPerformed(java.awt.event.ActionEvent evt)
{
    try
    {
        byte [] IP = new byte [4];
        IP [0] = (byte)Integer.parseInt(ip1.getText());
        IP [1] = (byte)Integer.parseInt(ip2.getText());
        IP [2] = (byte)Integer.parseInt(ip3.getText());
        IP [3] = (byte)Integer.parseInt(ip4.getText());

        InetAddress add = InetAddress.getByAddress(IP);
        if (!add.isReachable(10000))
            throw new IOException();

        Socket client = new Socket(add,4444);

        PECL2 p = new PECL2 (client);
        p.setVisible(true);
        this.dispose();
    }
    catch (NumberFormatException e)
    {
        txt.setText("INVALID IP");
    }
    catch (IOException ex)
    {
        txt.setText("SERVER NOT RESPONDING");
    }
}

// Variables declaration - do not modify
private javax.swing.JButton btt;
private javax.swing.JTextField ip1;
private javax.swing.JTextField ip2;
private javax.swing.JTextField ip3;
private javax.swing.JTextField ip4;
private javax.swing.JPanel jPanel1;
```

```
    private javax.swing.JPanel jPanel2;
    private javax.swing.JLabel txt;
    // End of variables declaration
}
```