GRADO EN INGENIERÍA INFORMÁTICA

CURSO 2017/2018 – CONVOCATORIA ORDINARIA

03146098C – Zamorano Ortega, Álvaro
03201906F – López Cuenca, Gabriel

1. ANÁLISIS DE ALTO NIVEL

La simulación de la gasolinera básicamente se compone de una cola de espera a la que van entrando los vehículos, y si hay hueco en algún surtidor entran a él. Una vez dentro del surtidor esperan a que vaya un operario y les atienda para poder salir de la gasolinera y dejar hueco a otro coche. Estos eventos se muestran gráficamente en una interfaz donde se puede parar y reanudar el funcionamiento, así como en un archivo de texto.

Para la realización de la parte distribuida se ha optado por usar RMI ya que es necesario el uso remotamente de dos métodos de la clase EstadoGasolinera, y además se permite la conexión de varios clientes simultáneamente sin necesidad de gestionar el canal de comunicación ni atender las peticiones desde el servidor.

Eventos gasolinera:

-- REGISTRO ACCIONES GASOLINERA --

12:22:20 -- El Coche 1 tarda en llegar a la gasolinera 1 segundos

12:22:21 -- El Coche 2 tarda en llegar a la gasolinera 2 segundos

12:22:22 -- Llega Coche 1 a la gasolinera

12:22:22 -- Entra Coche 1 surtidor 1

12:22:22 -- El Operario 2 reposta al Coche 1 en el surtidor 1, tarda en repostar 4 segundos

12:22:22 -- El Coche 3 tarda en llegar a la gasolinera 3 segundos

12:22:23 -- El Coche 4 tarda en llegar a la gasolinera 3 segundos

12:22:23 -- Llega Coche 2 a la gasolinera

12:22:23 -- Entra Coche 2 surtidor 2

12:22:23 -- El Operario 1 reposta al Coche 2 en el surtidor 2, tarda en repostar 7 segundos

12:22:24 -- El Coche 5 tarda en llegar a la gasolinera 3 segundos

12:22:25 -- El Coche 6 tarda en llegar a la gasolinera 4 segundos

12:22:26 -- Llega Coche 3 a la gasolinera

12:22:26 -- Entra Coche 3 surtidor 3

12:22:26 -- El Operario 3 reposta al Coche 3 en el surtidor 3, tarda en repostar 7 segundos

12:22:26 -- El Coche 7 tarda en llegar a la gasolinera 2 segundos

12:22:26 -- Llega Coche 4 a la gasolinera

12:22:26 -- Entra Coche 4 surtidor 4

12:22:27 -- Sale Coche 1 surtidor 1

12:22:27 -- Se va el Coche 1 de la gasolinera

12:22:27 -- El Operario 2 reposta al Coche 4 en el surtidor 4, tarda en repostar 6 segundos

.

.

.

12:30:0 -- Se va el Coche 197 de la gasolinera

12:30:4 -- Sale Coche 200 surtidor 3

12:30:4 -- Se va el Coche 200 de la gasolinera

12:30:5 -- Sale Coche 199 surtidor 7

12:30:5 -- Se va el Coche 199 de la gasolinera

-- FIN SIMULACION GASOLINERA --

2. DISEÑO DEL SISTEMA Y HERRAMIENTAS DE SINCRONIZACIÓN

En la solución de la simulación se ha optado por usar diferentes herramientas de sincronización dependiendo del problema a resolver:

- Semáforo de 8 permisos (8 surtidores) para entrar a un surtidor una vez se está en la cola de espera y haya alguno libre, es decir, el semáforo tenga permisos. Además, asegura que la entrada a la gasolinera sea en orden FIFO, el primero que llega es el primero en entrar.
- Monitores: método synchronized para elegir el surtidor en el que entra cada vehículo y así estos no se puedan solapar.
- Cerrojo con condición para elegir en que surtidor debe atender el operario. Esta elección únicamente se hará en caso de que haya coches esperando y algún operario no esté repostando en otro surtidor.
- Cerrojo con condición para parar y reanudar la simulación del programa.

3. DESCRIPCIÓN DE CLASES

Coche

Esta clase extiende de la clase Thread, por lo que cada coche será representado mediante un hilo. Tiene como atributos: número identificativo de coche (entero), gasolinera a la que reposta (objeto de la clase Gasolinera), un booleano para comprobar si ha sido atendido (inicialmente a false), el surtidor al que repostará (inicialmente a nulo) y un objeto de la clase Paso que usará para detener su ejecución en caso de que el atributo cerrado de esta clase esté en true.

Cuando desde el programa principal se lance este hilo, se ejecutará el método run(). Dentro de este método se esperará un tiempo aleatorio entre 0,5 y 6 segundos antes de llamar al método entrar() de la clase Gasolinera. Desde ahí irá haciendo diferentes acciones(entrar a la gasolinera, esperar en caso necesario en la cola de espera, entrar al surtidor, esperar a que le reposte un hilo de la clase Operario, y salir de la gasolinera) hasta terminar su ejecución.

Comunicacion

Esta clase extiende de la clase UnicastRemoteObject e implementa la interfaz InterfaceComunica. Tiene como atributo un objeto de la clase EstadoGasolinera, la cual usará para obtener los JTextField de la clase principal y un booleano para saber si sigue activo el programa principal. A través de sus métodos getTextos(),el cual retornará un ArrayList de String de lo que contienen los JTextField, y isAlive(), que retorna el atributo booleano isAlive del objeto Gasolinera , el programa principal (servidor) enviará remotamente al cliente para visualizar los eventos que ocurren en la gasolinera, y, en caso de que haya terminado la ejecución del programa principal, cerrar el cliente.

CrearCoche

Esta clase extiende de la clase Thread, por lo que actuará como un hilo que ejecutará su método run() cuando se lance desde el programa principal. Tiene como atributos un objeto de la clase Gasolinera y otro de la clase Paso para parar en caso necesario la creación de los hilos Coche. Este hilo creará un hilo Coche y lo lanzará cada segundo, intentando llevar a cabo una simulación más realista.

EstadoGasolinera

Esta clase tendrá como misión principal el guardado de los JTextField que visualizan la cola de espera, los operarios y los vehículos de la clase principal, por lo que tiene como atributos dos dos ArrayList de JTextField tanto para escribir vehículos como para escribir el operario que los atiende, un JTextField de los vehículos que estarán en la cola de espera, y la gasolinera a la que todo esto hará referencia.

Sus métodos retornan los valores de los JTextField y el atributo isAlive de la clase Gasolinera.

Gasolinera

Esta clase tiene como misión principal el control tanto de las salidas como de las entradas de los coches en ella, así como la generación del documento para el análisis del comportamiento del programa.

Tiene como atributos: un objeto de la clase ListaThreadFuera (cola de espera de los coches que están fuera de la gasolinera)**,** un objeto de la clase ListaThreadFuera(cola de dentro de la

gasolinera), un semáforo para acceder a los surtidores de la gasolinera, un contador para indicar los coches que ya han repostado y salido, una ventana principal para visualizar las acciones, un objeto de la clase Paso para parar o reanudar la simulación, un entero para definir los coches que van a ir a la gasolinera y otro para definir en número de operarios que repostan y un booleano para indicar si ha acabado la ejecución de todos los hilos Coche (alive).

Además de los métodos getter de los atributos que definen el número de coches (*getN_coches()*) y operarios (*getN_operarios()*), del atributo cola de espera de dentro(*getColaEsperaDentro()*), y para obtener y establecer el atributo booleano alive, tiene los siguientes métodos:

- public void entrar(Coche c) : este método simula la entrada de los coches a la gasolinera. Lo primero que hace es insertar el coche en la cola de espera de fuera, y si hubiera sitio en un surtidor dentro de la gasolinera, restaría un permiso del semáforo y llamaría al método *dondeEntrar* de la cola de espera de dentro para obtener el surtidor al que debe ir, para luego insertarlo en este surtidor llamando al método *meter.*

- public void salir(Coche c): este método simula la salida de los coches de la gasolinera. Este es llamado desde la clase ListaThreadDentro cuando un hilo Operario ha acabado de repostar a un coche. Una vez repostado, el coche sale de la gasolinera y se libera un permiso del semáforo. Finalmente, suma en uno el contador de los coches salidos, y si este es igual al numero de coches definidos en el inicio, se pone a false el atributo alive y se cierra la ventana principal.

- private void generarDocumento(): este método captura todo lo que sale por la salida y lo escribe en el documento *evolucionGasolinera.txt*.

InterfaceComunica

Se trata de la interfaz que declara los métodos isAlive() y getTextos() que usará la clase Comunicación para enviar de manera remota el estado del programa principal y el contenido de los JTextField respectivamente.

ListaThreadDentro

Esta clase tiene como misión principal llevar a cabo las acciones de elegir en que surtidor entrar, entrar en ese surtidor, elegir a que surtidor debe ir el operario, y repostar al coche para su posterior salida de la gasolinera.

Tiene como atributos: un ArrayList de los surtidores que hay en la gasolinera, un ArrayList de enteros (número de surtidor) para indicar el orden en el que se debe atender a los coches, dos ArrayList de JTextField para escribir en el programa principal que coche y que operario hay en cada surtidor, la gasolinera a la que pertenece, un cerrojo concurrente para la elección del surtidor por parte de los hilos Operario con su correspondiente condition (en caso de que esté

vacía la gasolinera) y un entero que guarda en número de operarios que hay atendiendo en cada momento.

Dispone de los siguientes métodos:

- public synchronized int dondeEntrar(Coche c): este método es llamado desde el método *entrar* de la clase Gasolinera y busca el primer surtidor que esté libre. Para ello recorre el ArrayList de surtidores, y devuelve el número del primer surtidor que esté libre. Además, añade el número de surtidor al final del ArrayList que indica el orden en el que se debe atender a los coches (nSurtidor).

- public void meter(Coche vehiculo, int i): este método es llamado desde el método *entrar* de la clase Gasolinera e introduce el coche al surtidor que se pasa como parámetro. Para ello se utiliza un cerrojo para asegurar la exclusión mutua y añade al JTextField (del programa principal) el nombre identificativo del coche correspondiente a ese surtidor.

- public int dondeSacar(): este método es llamado desde el método *run()* del hilo Operario para obtener el número del surtidor al que debe ir el operario para después repostar en él. Para ello utiliza un cerrojo para asegurar la exclusión mutua y espera en caso de que no haya ningún coche dentro de la gasolinera. Si hay algún coche dentro de ella, extrae el primer número identificativo del ArrayList *nSurtidor* (que indica el orden en el que se debe atender a los coches) y lo retorna.

- public void sacar(Operario op, int i): este método es llamado desde el método *run()* del hilo Operario para simular el repostaje y sacar al coche de la gasolinera. Para ello, añade el operario al JTextField (del programa principal) correspondiente, simula un tiempo aleatorio entre 4 y 8 segundos y posteriormente llama al método *salir(Coche c)* de la clase Gasolinera. Finalmente, pone a false y a null los atributos *ocupado* y *coche* del surtidor y elimina el contenido de los JTextField (del programa principal) correspondientes al vehículo y al operario del surtidor.

- public void imprimir(String accion, Coche c): este método es llamado desde los métodos anteriores para imprimir por pantalla cuando entran y salen los coches de los surtidores.

ListaThreadFuera

Esta clase gestiona los coches que están en la cola de espera para entrar a la gasolinera, y muestra el contenido de la misma. Tiene como atributos un ArrayList de objetos Coche (simula la cola de espera) y un JTextField para visualizar la cola por pantalla. Sus métodos principales son:

- public synchronized void meter(Coche c): método synchronized que sirve para introducir un coche en la cola de espera para entrar a la gasolinera y una llamada al método imprimir() para mostrar en el jTextField correspondiente, el contenido de la misma.

- public synchronized void sacar(Coche c): similar al método anterior pero elimina el coche de la cola de espera para que este entre a un surtidor.

Operario

Extiende de la clase Thread, es decir, cada operario se representa mediante un hilo. Tiene como atributos un objeto de la clase Gasolinera, un objeto de la clase Paso para reanudar o parar su ejecución y un entero identificativo.

- run(): en caso de que el paso esté abierto, buscará el surtidor al que debe atender, volverá a mirar el paso, y finalmente atenderá al coche que esté en dicho surtidor.

Paso

Indicar a los componentes del programa si pueden continuar con sus tareas. Se basa en un cerrojo con condition sobre un booleano que abre o cierra el paso, en función de si está a false o no.

- mirar(): en caso de que un hilo entre a él y el paso esté cerrado, se quedará parado hasta que se modifique está condición.

- abrir() y cerrar(): únicamente cambiarán el estado del booleano “cerrado”, serán invocados una vez pulsados los botones de la interfaz gráfica destinados a este fin.

Principal

Se trata de la interfaz gráfica principal para visualizar las acciones de la gasolinera, así como iniciar todos aquellos componentes necesarios para la simulación. También se registra el objeto estado para realizar el intercambio de datos mediante RMI con los posibles clientes que se conecten.

- cerrar(): método a la espera del evento del cierre de la ventana, y cuando este ocurre, indicar que la simulación de la gasolinera va a terminar para así cerrar las ventanas de los posibles clientes conectados

- pasoBotonActionPerformed(): sirve para reanudar o cerrar el paso de la clase Paso, dependiendo de cómo esté este. Dependiendo de la acción realizada se imprimirá un mensaje u otro en el log.

- Se adjunta una captura del diseño de la interfaz donde su muestran los eventos que ocurren en la gasolinera, así como un botón para Parar o Reanudar la ejecución del programa.

RecibirDatos

Esta clase extiende de la clase Thread, y pertenece a la parte del cliente y tiene como misión principal el recibir los datos remotos de la clase principal para visualizar por pantalla los eventos de la gasolinera.

- run(): gestiona la recepción del objeto remoto, ejecutar sus métodos y actualizar los datos correspondientes para poder visualizar en la interfaz del cliente los eventos que ocurren en la gasolinera. Esta actualización se llevará a cabo hasta que la gasolinera esté abierta, con una frecuencia de 1 segundo.
  Cuando termina la simulación de la gasolinera, cierra la ventana del cliente.

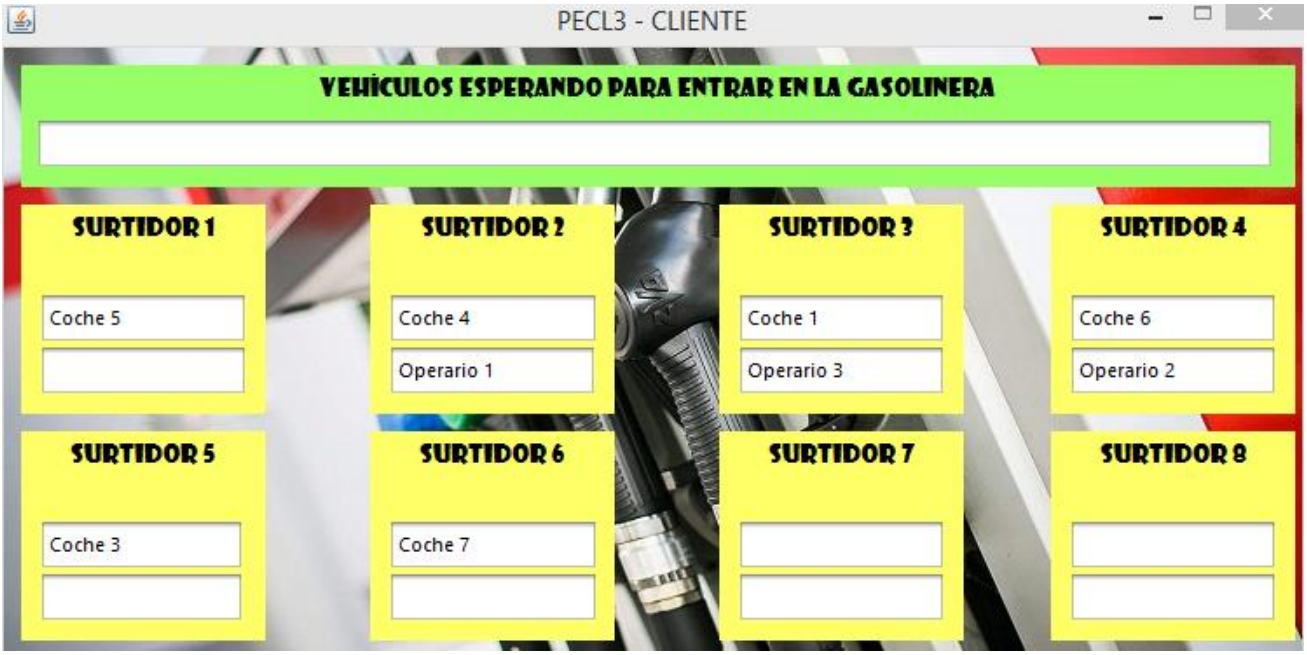
Surtidor

Esta clase únicamente nos servirá para almacenar, consultar y modificar datos de los surtidores. Tiene como atributos un entero identificativo del surtidor, un booleano indicando si hay algún vehículo en él, y en caso afirmativo, indicar el coche que hay.

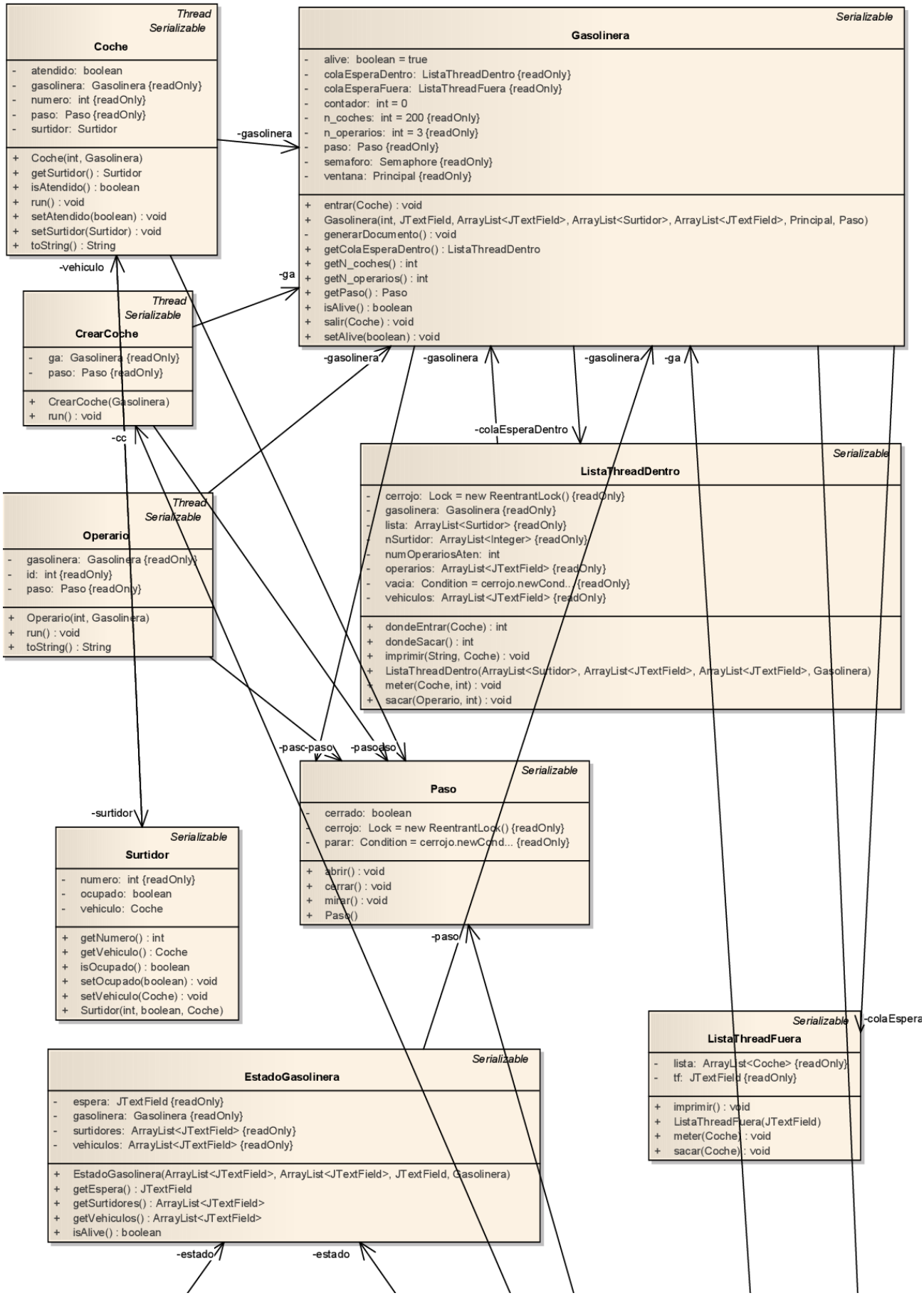
VentanaCliente

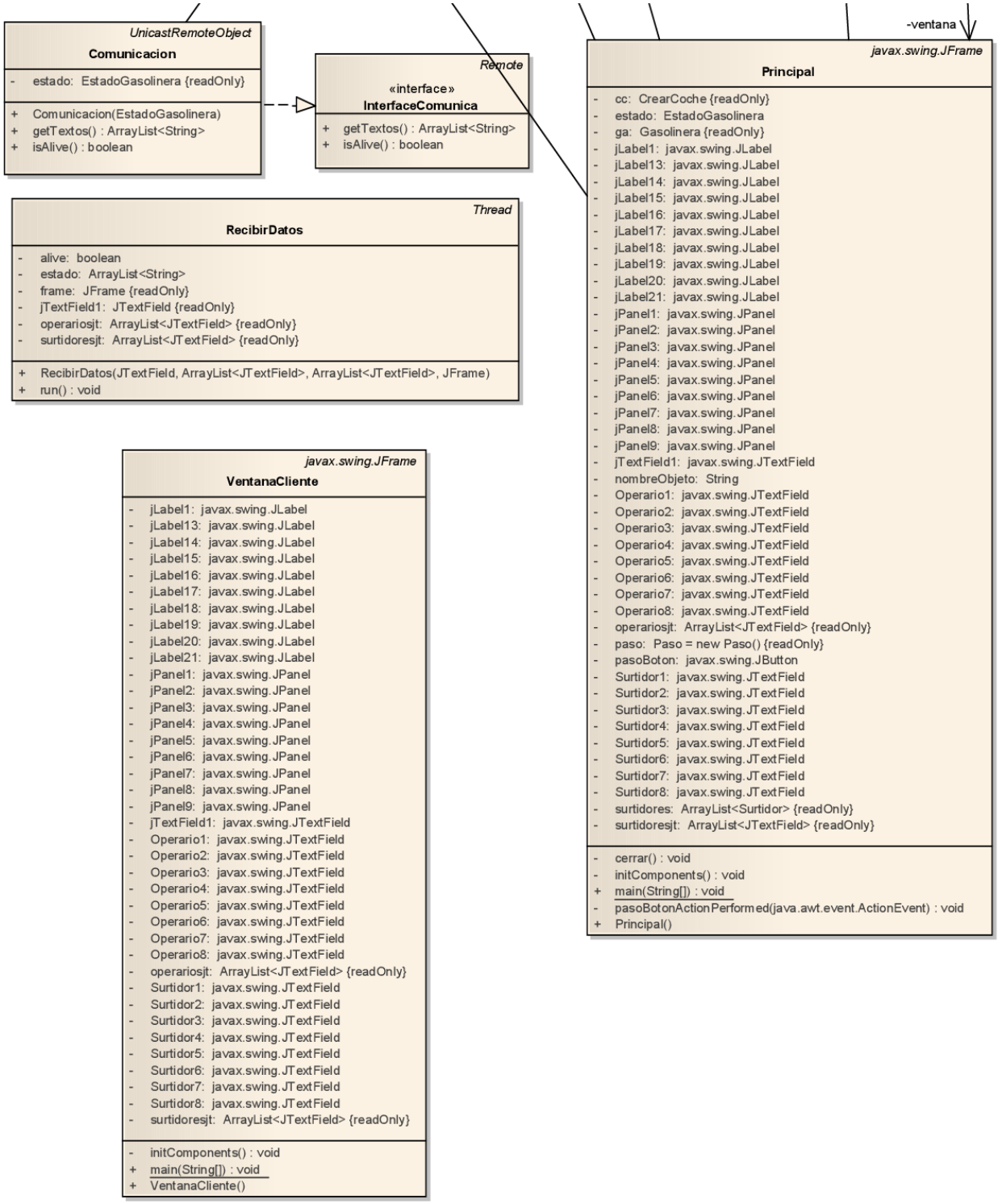
Sirve para visualizar de forma remota las acciones que se llevan a cabo en la gasolinera con un retardo de 1 segundo. En ella se inicia el hilo que actualiza estos eventos (RecibirDatos).

- Se adjunta una captura del diseño de la interfaz donde su muestran los eventos que ocurren en la gasolinera.


PECL3 - CLIENTE
VEHÍCULOS ESPERANDO PARA ENTRAR EN LA GASOLINERA
SURTIDOR 1
Coche 5
SURTIDOR 2
Coche 4
Operario 1
SURTIDOR 3
Coche 1
Operario 3
SURTIDOR 4
Coche 6
Operario 2
SURTIDOR 5
Coche 3
SURTIDOR 6
Coche 7
SURTIDOR 7
SURTIDOR 8

4. DIAGRAMA DE CLASES




UnicastRemoteObject
Comunicacion
- estado: EstadoGasolinera {readOnly}
+ Comunicacion(EstadoGasolinera)
+ getTextos() : ArrayList<String>
+ isAlive() : boolean
Remote
«interface»
InterfaceComunica
+ getTextos() : ArrayList<String>
+ isAlive() : boolean
-ventana
javax.swing.JFrame
Principal
- cc: CrearCoche {readOnly}
- estado: EstadoGasolinera
- ga: Gasolinera {readOnly}
- jLabel1: javax.swing.JLabel
- jLabel13: javax.swing.JLabel
- jLabel14: javax.swing.JLabel
- jLabel15: javax.swing.JLabel
- jLabel16: javax.swing.JLabel
- jLabel17: javax.swing.JLabel
- jLabel18: javax.swing.JLabel
- jLabel19: javax.swing.JLabel
- jLabel20: javax.swing.JLabel
- jLabel21: javax.swing.JLabel
- jPanel1: javax.swing.JPanel
- jPanel2: javax.swing.JPanel
- jPanel3: javax.swing.JPanel
- jPanel4: javax.swing.JPanel
- jPanel5: javax.swing.JPanel
- jPanel6: javax.swing.JPanel
- jPanel7: javax.swing.JPanel
- jPanel8: javax.swing.JPanel
- jPanel9: javax.swing.JPanel
- jTextField1: javax.swing.JTextField
- nombreObjeto: String
- Operario1: javax.swing.JTextField
- Operario2: javax.swing.JTextField
- Operario3: javax.swing.JTextField
- Operario4: javax.swing.JTextField
- Operario5: javax.swing.JTextField
- Operario6: javax.swing.JTextField
- Operario7: javax.swing.JTextField
- Operario8: javax.swing.JTextField
- operariosjt: ArrayList<JTextField> {readOnly}
- paso: Paso = new Paso() {readOnly}
- pasoBoton: javax.swing.JButton
- Surtidor1: javax.swing.JTextField
- Surtidor2: javax.swing.JTextField
- Surtidor3: javax.swing.JTextField
- Surtidor4: javax.swing.JTextField
- Surtidor5: javax.swing.JTextField
- Surtidor6: javax.swing.JTextField
- Surtidor7: javax.swing.JTextField
- Surtidor8: javax.swing.JTextField
- surtidores: ArrayList<Surtidor> {readOnly}
- surtidoresjt: ArrayList<JTextField> {readOnly}
- cerrar() : void
- initComponents() : void
+ main(String[]) : void
- pasoBotonActionPerformed(java.awt.event.ActionEvent) : void
+ Principal()
Thread
RecibirDatos
- alive: boolean
- estado: ArrayList<String>
- frame: JFrame {readOnly}
- jTextField1: JTextField {readOnly}
- operariosjt: ArrayList<JTextField> {readOnly}
- surtidoresjt: ArrayList<JTextField> {readOnly}
+ RecibirDatos(JTextField, ArrayList<JTextField>, ArrayList<JTextField>, JFrame)
+ run() : void
javax.swing.JFrame
VentanaCliente
- jLabel1: javax.swing.JLabel
- jLabel13: javax.swing.JLabel
- jLabel14: javax.swing.JLabel
- jLabel15: javax.swing.JLabel
- jLabel16: javax.swing.JLabel
- jLabel17: javax.swing.JLabel
- jLabel18: javax.swing.JLabel
- jLabel19: javax.swing.JLabel
- jLabel20: javax.swing.JLabel
- jLabel21: javax.swing.JLabel
- jPanel1: javax.swing.JPanel
- jPanel2: javax.swing.JPanel
- jPanel3: javax.swing.JPanel
- jPanel4: javax.swing.JPanel
- jPanel5: javax.swing.JPanel
- jPanel6: javax.swing.JPanel
- jPanel7: javax.swing.JPanel
- jPanel8: javax.swing.JPanel
- jPanel9: javax.swing.JPanel
- jTextField1: javax.swing.JTextField
- Operario1: javax.swing.JTextField
- Operario2: javax.swing.JTextField
- Operario3: javax.swing.JTextField
- Operario4: javax.swing.JTextField
- Operario5: javax.swing.JTextField
- Operario6: javax.swing.JTextField
- Operario7: javax.swing.JTextField
- Operario8: javax.swing.JTextField
- operariosjt: ArrayList<JTextField> {readOnly}
- Surtidor1: javax.swing.JTextField
- Surtidor2: javax.swing.JTextField
- Surtidor3: javax.swing.JTextField
- Surtidor4: javax.swing.JTextField
- Surtidor5: javax.swing.JTextField
- Surtidor6: javax.swing.JTextField
- Surtidor7: javax.swing.JTextField
- Surtidor8: javax.swing.JTextField
- surtidoresjt: ArrayList<JTextField> {readOnly}
- initComponents() : void
+ main(String[]) : void
+ VentanaCliente()

5. CÓDIGO FUENTE

Coche:

```
public class Coche extends Thread implements Serializable {


   //Atributos de la clase
   private final int numero;
   private final Gasolinera gasolinera;
   private boolean atendido;
   private Surtidor surtidor;
   private final Paso paso;


   public Coche(int p_n, Gasolinera p_g) {
      numero = p_n;
      gasolinera = p_g;
      surtidor = null;
      atendido = false;
      paso = gasolinera.getPaso();
   }
   public String toString() {
      return "Coche " + numero + " ";
   }


   public Surtidor getSurtidor() {
      return surtidor;
   }
   public boolean isAtendido() {
      return atendido;
   }
   public void setSurtidor(Surtidor surtidor) {
      this.surtidor = surtidor;
```

```
    }
    public void setAtendido(boolean atendido) {
        this.atendido = atendido;
    }
    public void run() {
        paso.mirar();
        try {
            int rnd = (int) (5500 * Math.random() + 500);
            Calendar calendario = new GregorianCalendar();
            int hora = calendario.get(Calendar.HOUR_OF_DAY);
            int minutos = calendario.get(Calendar.MINUTE);
            int segundos = calendario.get(Calendar.SECOND);
            System.out.println(hora + ":" + minutos + ":" + segundos + " -- " + "El " + this.toString() + " tarda en
llegar a la gasolinera " + rnd / 1000 + " segundos");
            sleep(rnd); //Tiempo de llegada a la gasolinera
        } catch (InterruptedException e) {
        }
        paso.mirar();
        gasolinera.entrar(this); //Entra si hay hueco; y sino espera en la cola
    }
}
```

Comunicación:

```
public class Comunicacion extends UnicastRemoteObject implements InterfaceComunica {

    //Atributos de la clase
    private final EstadoGasolinera estado;

    public Comunicacion(EstadoGasolinera p_estado) throws RemoteException {
        estado = p_estado;
    }
```

```
    @Override
    public ArrayList<String> getTextos() throws RemoteException {
        ArrayList<String> textos = new ArrayList<>();
        textos.add(estado.getEspera().getText());
        for (int i = 0; i < 8; i++) {
            textos.add(estado.getSurtidores().get(i).getText());
        }
        for (int i = 0; i < 8; i++) {
            textos.add(estado.getVehiculos().get(i).getText());
        }
        return textos;
    }
    @Override
    public boolean isAlive() throws RemoteException {
        return estado.isAlive();
    }
}
```

CrearCoche:

```
public class CrearCoche extends Thread implements Serializable {


    //Atributos de la clase
    private final Gasolinera ga;
    private final Paso paso;
    public CrearCoche(Gasolinera p_g) {
        ga = p_g;
        paso = ga.getPaso();
    }
    public void run() {
        for (int k = 1; k <= ga.getN_coches(); k++) {
            paso.mirar();
```

```
            Coche c = new Coche(k, ga);
            c.start();
            try {
               sleep(1000); //Se crean coches cada segundo
            } catch (InterruptedException ex) {
               Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
            }
         }
      }
   }
```

EstadoGasolinera:

```
public class EstadoGasolinera implements Serializable {

   //Atributos de la clase
   private final ArrayList<JTextField> vehiculos;
   private final ArrayList<JTextField> surtidores;
   private final JTextField espera;
   private final Gasolinera gasolinera;
   public EstadoGasolinera(ArrayList<JTextField> p_v, ArrayList<JTextField> p_s,
         JTextField p_e, Gasolinera ga) {
      vehiculos = p_v;
      surtidores = p_s;
      espera = p_e;
      gasolinera = ga;
   }
   public boolean isAlive() {
      return gasolinera.isAlive();
   }
   public ArrayList<JTextField> getVehiculos() {
      return vehiculos;
```

```
    }
    public ArrayList<JTextField> getSurtidores() {
        return surtidores;
    }
    public JTextField getEspera() {
        return espera;
    }
}
```

Gasolinera:

```
public class Gasolinera implements Serializable {

    //Atributos de la clase
    private final ListaThreadDentro colaEsperaDentro;
    private final ListaThreadFuera colaEsperaFuera;
    private final Semaphore semaforo;
    private int contador = 0;
    private final Principal ventana;
    private final Paso paso;
    private final int n_coches = 200;
    private final int n_operarios = 3;
    private boolean alive = true;
    public Gasolinera(int p_n_surtidores, JTextField tfEsperan, ArrayList<JTextField> v,
            ArrayList<Surtidor> surtidores, ArrayList<JTextField> p_operarios, Principal p_v,
            Paso p_paso) {
        semaforo = new Semaphore(p_n_surtidores, true);
        colaEsperaFuera = new ListaThreadFuera(tfEsperan);
        colaEsperaDentro = new ListaThreadDentro(surtidores, v, p_operarios, this);
        ventana = p_v;
        paso = p_paso;
        generarDocumento(); //Llamada al método para crar el documento con la
```

```
                //evolución de la gasolinera
    }
    public ListaThreadDentro getColaEsperaDentro() {
        return colaEsperaDentro;
    }
    public boolean isAlive() {
        return alive;
    }
    public Paso getPaso() {
        return paso;
    }
    public int getN_coches() {
        return n_coches;
    }
    public int getN_operarios() {
        return n_operarios;
    }
    public void setAlive(boolean alive) {
        this.alive = alive;
    }
    public void entrar(Coche c) {
        paso.mirar();
        colaEsperaFuera.meter(c);
        Calendar calendario = new GregorianCalendar();
        int hora = calendario.get(Calendar.HOUR_OF_DAY);
        int minutos = calendario.get(Calendar.MINUTE);
        int segundos = calendario.get(Calendar.SECOND);
        System.out.println(hora + ":" + minutos + ":" + segundos + " -- " + "Llega " + c.toString() + " a la
gasolinera");
        try {
            semaforo.acquire();
```

```
        } catch (InterruptedException e) {
        }
        paso.mirar();
        colaEsperaFuera.sacar(c);
        paso.mirar();
        colaEsperaDentro.meter(c, colaEsperaDentro.dondeEntrar(c));
    }
    public void salir(Coche c) {
        paso.mirar();
        semaforo.release();
        Calendar calendario = new GregorianCalendar();
        int hora = calendario.get(Calendar.HOUR_OF_DAY);
        int minutos = calendario.get(Calendar.MINUTE);
        int segundos = calendario.get(Calendar.SECOND);
        System.out.println(hora + ":" + minutos + ":" + segundos + " -- " + "Se va el " + c.toString() + " de la
gasolinera");
        contador++;
        paso.mirar();
        if (contador == n_coches) {
            alive = false;
            try {
                sleep(1500); //Fin gasolinera
            } catch (InterruptedException ex) {
                Logger.getLogger(Gasolinera.class.getName()).log(Level.SEVERE, null, ex);
            }
            System.out.println("");
            System.out.println("-- FIN SIMULACION GASOLINERA --");
            ventana.dispose();
            exit(0);
        }
```

```
    }
    private void generarDocumento() {
        File ev = new File("./evolucionGasolinera.txt");
        FileOutputStream salida = null;
        try {
            salida = new FileOutputStream(ev);
        } catch (FileNotFoundException ex) {
            Logger.getLogger(Gasolinera.class.getName()).log(Level.SEVERE, null, ex);
        }
        PrintStream print = new PrintStream(salida);
        System.setOut(print);
        System.out.println("-- REGISTRO ACCIONES GASOLINERA --");
        System.out.println("");
    }
}
```

InterfazComunica:

```
public interface InterfaceComunica extends Remote {
    ArrayList<String> getTextos() throws RemoteException;

    boolean isAlive() throws RemoteException;
}
```

ListaThreadDentro:

```
public class ListaThreadDentro implements Serializable {

    //Atributos de la clase
    private final ArrayList<Surtidor> lista;
    private final ArrayList<Integer> nSurtidor;
    private final ArrayList<JTextField> vehiculos, operarios;
    private final Gasolinera gasolinera;
```

```
private final Lock cerrojo = new ReentrantLock();
private final Condition vacia = cerrojo.newCondition();
private int numOperariosAten;
public ListaThreadDentro(ArrayList<Surtidor> surtidores, ArrayList<JTextField> JTextV,
      ArrayList<JTextField> p_otext, Gasolinera p_g) {
   numOperariosAten = 0;
   lista = surtidores;
   vehiculos = JTextV;
   operarios = p_otext;
   gasolinera = p_g;
   nSurtidor = new ArrayList<>();
}
public synchronized int dondeEntrar(Coche c) {
   int i = 0;
   boolean sitioEncontrado = false;
   while (!sitioEncontrado && i < lista.size()) {
      if (!lista.get(i).isOcupado()) {
         sitioEncontrado = true;
      } else {
         i++;
      }
   }

   nSurtidor.add(i);
   return i;
}
public void meter(Coche vehiculo, int i) {
   cerrojo.lock();
   Surtidor s = lista.get(i);
   s.setVehiculo(vehiculo);
   s.setOcupado(true);
```

```
    vehiculo.setSurtidor(s);
    imprimir("Entra ", vehiculo);
    vehiculos.get(i).setText(s.getVehiculo().toString());
    try {
        vacia.signal();
    } finally {
        cerrojo.unlock();
    }
}
public int dondeSacar() {
    cerrojo.lock();
    int i = 0;
    try {
        while (nSurtidor.isEmpty() || numOperariosAten == gasolinera.getN_operarios()) {
            vacia.await();
        }

        i = nSurtidor.get(0);
        nSurtidor.remove(0);

    } catch (InterruptedException ex) {
        Logger.getLogger(ListaThreadDentro.class.getName()).log(Level.SEVERE, null, ex);
    } finally {
        cerrojo.unlock();
    }
    return i;
}
public void sacar(Operario op, int i) throws InterruptedException {
    numOperariosAten++;
    Coche c = lista.get(i).getVehiculo();
    c.setAtendido(true);
```

```
        operarios.get(i).setText(op.toString());
        Calendar calendario = new GregorianCalendar();
        int hora = calendario.get(Calendar.HOUR_OF_DAY);
        int minutos = calendario.get(Calendar.MINUTE);
        int segundos = calendario.get(Calendar.SECOND);
        int rnd = (int) (4000 + Math.random() * 4000);
        System.out.println(hora + ":" + minutos + ":" + segundos + " -- " + "El " + op.toString() + " reposta al " +
c.toString()
                + " en el surtidor " + c.getSurtidor().getNumero() + ", tarda en repostar " + rnd / 1000 + "
segundos");

        //Tiempo de repostaje
        try {
           sleep(rnd);
        } catch (InterruptedException e) {
        }

        lista.get(i).setOcupado(false);
        lista.get(i).setVehiculo(null);
        imprimir("Sale ", c);
        gasolinera.salir(c);
        vehiculos.get(i).setText("");
        operarios.get(i).setText("");
        numOperariosAten--;
    }
    public void imprimir(String accion, Coche c) {
        Calendar calendario = new GregorianCalendar();
        int hora = calendario.get(Calendar.HOUR_OF_DAY);
        int minutos = calendario.get(Calendar.MINUTE);
        int segundos = calendario.get(Calendar.SECOND);
        System.out.println(hora + ":" + minutos + ":" + segundos + " -- " + accion + c.toString() + " surtidor " +
c.getSurtidor().getNumero());
```

```
    }
}
```

ListaThreadFuera:

```
public class ListaThreadFuera implements Serializable {

    //Atributos de la clase
    private final ArrayList<Coche> lista;
    private final JTextField tf;
    public ListaThreadFuera(JTextField p_tf) {
        lista = new ArrayList<>();
        tf = p_tf;
    }
    public synchronized void meter(Coche c) {
        lista.add(c);
        imprimir();
    }
    public synchronized void sacar(Coche c) {
        lista.remove(c);
        imprimir();
    }
    public void imprimir() {
        String contenido = "";
        for (int i = 0; i < lista.size(); i++) {
            contenido = contenido + lista.get(i).toString() + " ";
        }
        tf.setText(contenido);
    }
}
```

Operario:

```
public class Operario extends Thread implements Serializable {


   //Atributos de la clase
   private final int id;
   private final Gasolinera gasolinera;
   private final Paso paso;
   public Operario(int p_id, Gasolinera p_g) {
      id = p_id;
      gasolinera = p_g;
      paso = gasolinera.getPaso();
   }
   public String toString() {
      return "Operario " + id;
   }
   public void run() {
      while (true) {
         try {
            paso.mirar();
            //Buscar el surtidor en el que repostar
            int pos = gasolinera.getColaEsperaDentro().dondeSacar();
            paso.mirar();
            if (pos < 8) {
               //Repostar
               gasolinera.getColaEsperaDentro().sacar(this, pos);
               paso.mirar();
            }


         } catch (InterruptedException ex) {
            Logger.getLogger(Operario.class.getName()).log(Level.SEVERE, null, ex);
         }
```

```
        }
    }
}
```

Paso:

```
public class Paso implements Serializable {

    //Atributos de la clase
    private boolean cerrado;
    private final Lock cerrojo = new ReentrantLock();
    private final Condition parar = cerrojo.newCondition();

    public Paso() {
        cerrado = false;
    }
    public void mirar() {
        cerrojo.lock();
        try {
            while (cerrado) {
                try {
                    parar.await();
                } catch (InterruptedException ie) {
                }
            }
        } finally {
            cerrojo.unlock();
        }

    }
    public void abrir() {
        cerrojo.lock();
```

```
        try {
            cerrado = false;
            parar.signalAll();
        } finally {
            cerrojo.unlock();
        }
    }
    public void cerrar() {
        cerrojo.lock();
        try {
            cerrado = true;
            parar.signalAll();
        } finally {
            cerrojo.unlock();
        }
    }
}
```

Principal:

```
public class Principal extends javax.swing.JFrame {

    //Atributos de la clase
    private final ArrayList<JTextField> surtidoresjt;
    private final ArrayList<Surtidor> surtidores;
    private final Gasolinera ga;
    private final ArrayList<JTextField> operariosjt;
    private final Paso paso = new Paso();
    private EstadoGasolinera estado;
    private final CrearCoche cc;
    private String nombreObjeto;
```

```
public Principal() {
    initComponents();

    surtidoresjt = new ArrayList<>(8);
    surtidores = new ArrayList<>(8);
    operariosjt = new ArrayList<>(8);

    surtidoresjt.add(0, Surtidor1);
    surtidoresjt.add(1, Surtidor2);
    surtidoresjt.add(2, Surtidor3);
    surtidoresjt.add(3, Surtidor4);
    surtidoresjt.add(4, Surtidor5);
    surtidoresjt.add(5, Surtidor6);
    surtidoresjt.add(6, Surtidor7);
    surtidoresjt.add(7, Surtidor8);

    operariosjt.add(0, Operario1);
    operariosjt.add(1, Operario2);
    operariosjt.add(2, Operario3);
    operariosjt.add(3, Operario4);
    operariosjt.add(4, Operario5);
    operariosjt.add(5, Operario6);
    operariosjt.add(6, Operario7);
    operariosjt.add(7, Operario8);

    for (int i = 1; i < 9; i++) {
        Surtidor surtidor = new Surtidor(i, false, null);
        surtidores.add(i - 1, surtidor);
    }

    ga = new Gasolinera(8, jTextField1, surtidoresjt, surtidores, operariosjt, this, paso);
```

```
    cc = new CrearCoche(ga);
    cc.start();

    for (int j = 1; j < ga.getN_operarios() + 1; j++) {
        Operario o = new Operario(j, ga);
        o.start();
    }

    nombreObjeto = "//localhost/ObGasolinera";
    estado = new EstadoGasolinera(surtidoresjt, operariosjt, jTextField1, ga);
    try {
        Comunicacion com = new Comunicacion(estado);
        Registry registry = LocateRegistry.createRegistry(1099);
        Naming.rebind(nombreObjeto, com);
    } catch (MalformedURLException | RemoteException e) {
        System.out.println(" Error: " + e.getMessage());
    }
    cerrar();
}
private void initComponents() {

    jTextField1 = new javax.swing.JTextField();
    Surtidor1 = new javax.swing.JTextField();
    Surtidor2 = new javax.swing.JTextField();
    Surtidor3 = new javax.swing.JTextField();
    Surtidor4 = new javax.swing.JTextField();
    Surtidor5 = new javax.swing.JTextField();
    Surtidor6 = new javax.swing.JTextField();
    Surtidor7 = new javax.swing.JTextField();
    Surtidor8 = new javax.swing.JTextField();
```

```
Operario1 = new javax.swing.JTextField();
Operario2 = new javax.swing.JTextField();
Operario3 = new javax.swing.JTextField();
Operario4 = new javax.swing.JTextField();
Operario5 = new javax.swing.JTextField();
Operario7 = new javax.swing.JTextField();
Operario6 = new javax.swing.JTextField();
Operario8 = new javax.swing.JTextField();
pasoBoton = new javax.swing.JButton();
jPanel2 = new javax.swing.JPanel();
jLabel20 = new javax.swing.JLabel();
jPanel3 = new javax.swing.JPanel();
jLabel13 = new javax.swing.JLabel();
jPanel4 = new javax.swing.JPanel();
jLabel14 = new javax.swing.JLabel();
jPanel5 = new javax.swing.JPanel();
jLabel15 = new javax.swing.JLabel();
jPanel6 = new javax.swing.JPanel();
jLabel16 = new javax.swing.JLabel();
jPanel7 = new javax.swing.JPanel();
jLabel17 = new javax.swing.JLabel();
jPanel8 = new javax.swing.JPanel();
jLabel18 = new javax.swing.JLabel();
jPanel9 = new javax.swing.JPanel();
jLabel19 = new javax.swing.JLabel();
jPanel1 = new javax.swing.JPanel();
jLabel21 = new javax.swing.JLabel();
jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("PECL3 - Alvaro Zamorano y Gabriel Lopez");
```

```
setAlwaysOnTop(true);
setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
setMinimumSize(new java.awt.Dimension(750, 450));
setResizable(false);
getContentPane().setLayout(null);

jTextField1.setEditable(false);
jTextField1.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(jTextField1);
jTextField1.setBounds(18, 40, 710, 30);

Surtidor1.setEditable(false);
Surtidor1.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor1);
Surtidor1.setBounds(20, 140, 120, 30);

Surtidor2.setEditable(false);
Surtidor2.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor2);
Surtidor2.setBounds(220, 140, 121, 30);

Surtidor3.setEditable(false);
Surtidor3.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor3);
Surtidor3.setBounds(420, 140, 120, 30);

Surtidor4.setEditable(false);
Surtidor4.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor4);
Surtidor4.setBounds(610, 140, 120, 30);
```

```
Surtidor5.setEditable(false);
Surtidor5.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor5);
Surtidor5.setBounds(20, 270, 118, 30);

Surtidor6.setEditable(false);
Surtidor6.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor6);
Surtidor6.setBounds(220, 270, 121, 30);

Surtidor7.setEditable(false);
Surtidor7.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor7);
Surtidor7.setBounds(420, 270, 120, 30);

Surtidor8.setEditable(false);
Surtidor8.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor8);
Surtidor8.setBounds(610, 270, 120, 30);

Operario1.setEditable(false);
Operario1.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario1);
Operario1.setBounds(20, 170, 120, 30);

Operario2.setEditable(false);
Operario2.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario2);
Operario2.setBounds(220, 170, 120, 30);

Operario3.setEditable(false);
```

```
Operario3.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario3);
Operario3.setBounds(420, 170, 120, 30);

Operario4.setEditable(false);
Operario4.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario4);
Operario4.setBounds(610, 170, 120, 30);

Operario5.setEditable(false);
Operario5.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario5);
Operario5.setBounds(20, 300, 118, 30);

Operario7.setEditable(false);
Operario7.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario7);
Operario7.setBounds(420, 300, 120, 30);

Operario6.setEditable(false);
Operario6.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario6);
Operario6.setBounds(220, 300, 121, 30);

Operario8.setEditable(false);
Operario8.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario8);
Operario8.setBounds(610, 300, 120, 30);

pasoBoton.setBackground(new java.awt.Color(255, 51, 0));
pasoBoton.setFont(new java.awt.Font("Showcard Gothic", 0, 11)); // NOI18N
```

```
pasoBoton.setText("Pausar");
pasoBoton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      pasoBotonActionPerformed(evt);
   }
});
getContentPane().add(pasoBoton);
pasoBoton.setBounds(340, 380, 92, 23);

jPanel2.setBackground(new java.awt.Color(255, 255, 102));

jLabel20.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel20.setText("Surtidor 8");
jPanel2.add(jLabel20);

getContentPane().add(jPanel2);
jPanel2.setBounds(600, 220, 140, 120);

jPanel3.setBackground(new java.awt.Color(255, 255, 102));

jLabel13.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel13.setText("Surtidor 1");
jPanel3.add(jLabel13);

getContentPane().add(jPanel3);
jPanel3.setBounds(10, 90, 140, 120);

jPanel4.setBackground(new java.awt.Color(255, 255, 102));

jLabel14.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel14.setText("Surtidor 2");
```

```
jPanel4.add(jLabel14);

getContentPane().add(jPanel4);
jPanel4.setBounds(210, 90, 140, 120);

jPanel5.setBackground(new java.awt.Color(255, 255, 102));

jLabel15.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel15.setText("Surtidor 3");
jPanel5.add(jLabel15);

getContentPane().add(jPanel5);
jPanel5.setBounds(410, 90, 140, 120);

jPanel6.setBackground(new java.awt.Color(255, 255, 102));

jLabel16.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel16.setText("Surtidor 4");
jPanel6.add(jLabel16);

getContentPane().add(jPanel6);
jPanel6.setBounds(600, 90, 140, 120);

jPanel7.setBackground(new java.awt.Color(255, 255, 102));

jLabel17.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel17.setText("Surtidor 5");
jPanel7.add(jLabel17);

getContentPane().add(jPanel7);
jPanel7.setBounds(10, 220, 140, 120);
```

```
        jPanel8.setBackground(new java.awt.Color(255, 255, 102));

        jLabel18.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
        jLabel18.setText("Surtidor 6");
        jPanel8.add(jLabel18);

        getContentPane().add(jPanel8);
        jPanel8.setBounds(210, 220, 140, 120);

        jPanel9.setBackground(new java.awt.Color(255, 255, 102));

        jLabel19.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
        jLabel19.setText("Surtidor 7");
        jPanel9.add(jLabel19);

        getContentPane().add(jPanel9);
        jPanel9.setBounds(410, 220, 140, 120);

        jPanel1.setBackground(new java.awt.Color(153, 255, 102));

        jLabel21.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
        jLabel21.setText("VEHÍCULOS ESPERANDO PARA ENTRAR EN LA GASOLINERA");
        jPanel1.add(jLabel21);

        getContentPane().add(jPanel1);
        jPanel1.setBounds(10, 10, 730, 70);

        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/pecl3/gas-station.jpg"))); //
NOI18N
```

```
    getContentPane().add(jLabel1);
    jLabel1.setBounds(0, 0, 750, 430);

    pack();
}

private void pasoBotonActionPerformed(java.awt.event.ActionEvent evt) {
    if (pasoBoton.getText().equals("Pausar")) {
        paso.cerrar();
        Calendar calendario = new GregorianCalendar();
        int hora = calendario.get(Calendar.HOUR_OF_DAY);
        int minutos = calendario.get(Calendar.MINUTE);
        int segundos = calendario.get(Calendar.SECOND);
        System.out.println(hora + ":" + minutos + ":" + segundos + " -- " + "PASO CERRADO");
        pasoBoton.setText("Reanudar");
    } else {
        paso.abrir();
        Calendar calendario = new GregorianCalendar();
        int hora = calendario.get(Calendar.HOUR_OF_DAY);
        int minutos = calendario.get(Calendar.MINUTE);
        int segundos = calendario.get(Calendar.SECOND);
        System.out.println(hora + ":" + minutos + ":" + segundos + " -- " + "PASO ABIERTO");
        pasoBoton.setText("Pausar");
    }
}

private void cerrar() {
    try {
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
```

```
                ga.setAlive(false);
                try {
                    Thread.sleep(1500); //Espera a que llegue el cierre a los clientes que hay
                    Naming.unbind(nombreObjeto); //Quitamos del registro el objeto remoto
                } catch (InterruptedException | RemoteException | NotBoundException |
MalformedURLException ex) {
                    Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
                }
                dispose();
                System.exit(0);
            }
        });
    } catch (Exception e) {
    }
  }
  public static void main(String args[]) {
    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(Principal.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Principal.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Principal.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
```

```
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Principal.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Principal().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JTextField Operario1;
    private javax.swing.JTextField Operario2;
    private javax.swing.JTextField Operario3;
    private javax.swing.JTextField Operario4;
    private javax.swing.JTextField Operario5;
    private javax.swing.JTextField Operario6;
    private javax.swing.JTextField Operario7;
    private javax.swing.JTextField Operario8;
    private javax.swing.JTextField Surtidor1;
    private javax.swing.JTextField Surtidor2;
    private javax.swing.JTextField Surtidor3;
    private javax.swing.JTextField Surtidor4;
    private javax.swing.JTextField Surtidor5;
    private javax.swing.JTextField Surtidor6;
    private javax.swing.JTextField Surtidor7;
    private javax.swing.JTextField Surtidor8;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel13;
```

```
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel18;
    private javax.swing.JLabel jLabel19;
    private javax.swing.JLabel jLabel20;
    private javax.swing.JLabel jLabel21;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JPanel jPanel6;
    private javax.swing.JPanel jPanel7;
    private javax.swing.JPanel jPanel8;
    private javax.swing.JPanel jPanel9;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JButton pasoBoton;
}
```

RecibirDatos:

```
public class RecibirDatos extends Thread {

    //Atributos de la clase
    private ArrayList<String> estado;
    private boolean alive;
    private final JTextField jTextField1;
    private final ArrayList<JTextField> surtidoresjt;
    private final ArrayList<JTextField> operariosjt;
    private final JFrame frame;
```

```
    public RecibirDatos(JTextField jt, ArrayList<JTextField> s,
            ArrayList<JTextField> o, JFrame jf) {
        jTextField1 = jt;
        surtidoresjt = s;
        operariosjt = o;
        frame = jf;
    }
    public void run() {
        do {
            try {
                InterfaceComunica obj;
                obj = (InterfaceComunica) Naming.lookup("//127.0.0.1/ObGasolinera");
                estado = obj.getTextos();
                alive = obj.isAlive();

                jTextField1.setText(estado.get(0));

                for (int i = 0; i < 8; i++) {
                    operariosjt.get(i).setText(estado.get(i + 1));
                }
                for (int i = 0; i < 8; i++) {
                    surtidoresjt.get(i).setText(estado.get(i + 9));
                }

                sleep(1000); //Se actualiza cada segundo
            } catch (NotBoundException | MalformedURLException | RemoteException | InterruptedException
ex) {
                Logger.getLogger(RecibirDatos.class.getName()).log(Level.SEVERE, null, ex);
            }
        } while (alive);
        frame.dispose();
```

```
        exit(0);
    }
}
```

Surtidor:

```
public class Surtidor implements Serializable {

    //Atributos de la clase
    private final int numero;
    private boolean ocupado;
    private Coche vehiculo;
    public Surtidor(int numero, boolean ocupado, Coche vehiculo) {
        this.numero = numero;
        this.ocupado = ocupado;
        this.vehiculo = vehiculo;
    }
    public int getNumero() {
        return numero;
    }
    public boolean isOcupado() {
        return ocupado;
    }
    public void setOcupado(boolean ocupado) {
        this.ocupado = ocupado;
    }
    public Coche getVehiculo() {
        return vehiculo;
    }
    public void setVehiculo(Coche vehiculo) {
        this.vehiculo = vehiculo;
    }
```

```
}
```

VentanaCliente:

```
public class VentanaCliente extends javax.swing.JFrame {

   //Atributos de la clase
   private final ArrayList<JTextField> surtidoresjt;

   public VentanaCliente() {
      initComponents();

      surtidoresjt = new ArrayList<>(8);
      operariosjt = new ArrayList<>(8);

      surtidoresjt.add(0, Surtidor1);
      surtidoresjt.add(1, Surtidor2);
      surtidoresjt.add(2, Surtidor3);
      surtidoresjt.add(3, Surtidor4);
      surtidoresjt.add(4, Surtidor5);
      surtidoresjt.add(5, Surtidor6);
      surtidoresjt.add(6, Surtidor7);
      surtidoresjt.add(7, Surtidor8);

      operariosjt.add(0, Operario1);
      operariosjt.add(1, Operario2);
      operariosjt.add(2, Operario3);
      operariosjt.add(3, Operario4);
      operariosjt.add(4, Operario5);
      operariosjt.add(5, Operario6);
      operariosjt.add(6, Operario7);
      operariosjt.add(7, Operario8);
```

```
    RecibirDatos rd = new RecibirDatos(jTextField1, surtidoresjt, operariosjt, this);
    rd.start();
}
private void initComponents() {

    jTextField1 = new javax.swing.JTextField();
    Surtidor1 = new javax.swing.JTextField();
    Surtidor2 = new javax.swing.JTextField();
    Surtidor3 = new javax.swing.JTextField();
    Surtidor4 = new javax.swing.JTextField();
    Surtidor5 = new javax.swing.JTextField();
    Surtidor6 = new javax.swing.JTextField();
    Surtidor7 = new javax.swing.JTextField();
    Surtidor8 = new javax.swing.JTextField();
    Operario1 = new javax.swing.JTextField();
    Operario2 = new javax.swing.JTextField();
    Operario3 = new javax.swing.JTextField();
    Operario4 = new javax.swing.JTextField();
    Operario5 = new javax.swing.JTextField();
    Operario7 = new javax.swing.JTextField();
    Operario6 = new javax.swing.JTextField();
    Operario8 = new javax.swing.JTextField();
    jPanel2 = new javax.swing.JPanel();
    jLabel20 = new javax.swing.JLabel();
    jPanel3 = new javax.swing.JPanel();
    jLabel13 = new javax.swing.JLabel();
    jPanel4 = new javax.swing.JPanel();
    jLabel14 = new javax.swing.JLabel();
    jPanel5 = new javax.swing.JPanel();
    jLabel15 = new javax.swing.JLabel();
```

```
jPanel6 = new javax.swing.JPanel();
jLabel16 = new javax.swing.JLabel();
jPanel7 = new javax.swing.JPanel();
jLabel17 = new javax.swing.JLabel();
jPanel8 = new javax.swing.JPanel();
jLabel18 = new javax.swing.JLabel();
jPanel9 = new javax.swing.JPanel();
jLabel19 = new javax.swing.JLabel();
jPanel1 = new javax.swing.JPanel();
jLabel21 = new javax.swing.JLabel();
jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("PECL3 - CLIENTE");
setAlwaysOnTop(true);
setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
setMinimumSize(new java.awt.Dimension(750, 375));
setResizable(false);
getContentPane().setLayout(null);

jTextField1.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(jTextField1);
jTextField1.setBounds(18, 40, 710, 30);

Surtidor1.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor1);
Surtidor1.setBounds(20, 140, 120, 30);

Surtidor2.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor2);
Surtidor2.setBounds(220, 140, 121, 30);
```

```
Surtidor3.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor3);
Surtidor3.setBounds(420, 140, 120, 30);

Surtidor4.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor4);
Surtidor4.setBounds(610, 140, 120, 30);

Surtidor5.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor5);
Surtidor5.setBounds(20, 270, 118, 30);

Surtidor6.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor6);
Surtidor6.setBounds(220, 270, 121, 30);

Surtidor7.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor7);
Surtidor7.setBounds(420, 270, 120, 30);

Surtidor8.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Surtidor8);
Surtidor8.setBounds(610, 270, 120, 30);

Operario1.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario1);
Operario1.setBounds(20, 170, 120, 30);

Operario2.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario2);
```

```
Operario2.setBounds(220, 170, 120, 30);

Operario3.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario3);
Operario3.setBounds(420, 170, 120, 30);

Operario4.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario4);
Operario4.setBounds(610, 170, 120, 30);

Operario5.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario5);
Operario5.setBounds(20, 300, 118, 30);

Operario7.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario7);
Operario7.setBounds(420, 300, 120, 30);

Operario6.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario6);
Operario6.setBounds(220, 300, 121, 30);

Operario8.setFont(new java.awt.Font("Nirmala UI", 0, 11)); // NOI18N
getContentPane().add(Operario8);
Operario8.setBounds(610, 300, 120, 30);

jPanel2.setBackground(new java.awt.Color(255, 255, 102));

jLabel20.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel20.setText("Surtidor 8");
jPanel2.add(jLabel20);
```

```
getContentPane().add(jPanel2);
jPanel2.setBounds(600, 220, 140, 120);

jPanel3.setBackground(new java.awt.Color(255, 255, 102));

jLabel13.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel13.setText("Surtidor 1");
jPanel3.add(jLabel13);

getContentPane().add(jPanel3);
jPanel3.setBounds(10, 90, 140, 120);

jPanel4.setBackground(new java.awt.Color(255, 255, 102));

jLabel14.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel14.setText("Surtidor 2");
jPanel4.add(jLabel14);

getContentPane().add(jPanel4);
jPanel4.setBounds(210, 90, 140, 120);

jPanel5.setBackground(new java.awt.Color(255, 255, 102));

jLabel15.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel15.setText("Surtidor 3");
jPanel5.add(jLabel15);

getContentPane().add(jPanel5);
jPanel5.setBounds(410, 90, 140, 120);
```

```
jPanel6.setBackground(new java.awt.Color(255, 255, 102));

jLabel16.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel16.setText("Surtidor 4");
jPanel6.add(jLabel16);

getContentPane().add(jPanel6);
jPanel6.setBounds(600, 90, 140, 120);

jPanel7.setBackground(new java.awt.Color(255, 255, 102));

jLabel17.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel17.setText("Surtidor 5");
jPanel7.add(jLabel17);

getContentPane().add(jPanel7);
jPanel7.setBounds(10, 220, 140, 120);

jPanel8.setBackground(new java.awt.Color(255, 255, 102));

jLabel18.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel18.setText("Surtidor 6");
jPanel8.add(jLabel18);

getContentPane().add(jPanel8);
jPanel8.setBounds(210, 220, 140, 120);

jPanel9.setBackground(new java.awt.Color(255, 255, 102));

jLabel19.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
jLabel19.setText("Surtidor 7");
```

```
        jPanel9.add(jLabel19);

        getContentPane().add(jPanel9);
        jPanel9.setBounds(410, 220, 140, 120);

        jPanel1.setBackground(new java.awt.Color(153, 255, 102));

        jLabel21.setFont(new java.awt.Font("Showcard Gothic", 0, 14)); // NOI18N
        jLabel21.setText("VEHÍCULOS ESPERANDO PARA ENTRAR EN LA GASOLINERA");
        jPanel1.add(jLabel21);

        getContentPane().add(jPanel1);
        jPanel1.setBounds(10, 10, 730, 70);

        jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/pecl3/gas-station.jpg")));
        getContentPane().add(jLabel1);
        jLabel1.setBounds(0, 0, 750, 350);

        pack();
    }
    public static void main(String args[]) {
        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(VentanaCliente.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
```

```
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(VentanaCliente.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(VentanaCliente.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(VentanaCliente.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new VentanaCliente().setVisible(true);
            }
        });
    }
    private javax.swing.JTextField Operario1;
    private javax.swing.JTextField Operario2;
    private javax.swing.JTextField Operario3;
    private javax.swing.JTextField Operario4;
    private javax.swing.JTextField Operario5;
    private javax.swing.JTextField Operario6;
    private javax.swing.JTextField Operario7;
    private javax.swing.JTextField Operario8;
    private javax.swing.JTextField Surtidor1;
    private javax.swing.JTextField Surtidor2;
    private javax.swing.JTextField Surtidor3;
    private javax.swing.JTextField Surtidor4;
    private javax.swing.JTextField Surtidor5;
    private javax.swing.JTextField Surtidor6;
    private javax.swing.JTextField Surtidor7;
```

```
    private javax.swing.JTextField Surtidor8;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel18;
    private javax.swing.JLabel jLabel19;
    private javax.swing.JLabel jLabel20;
    private javax.swing.JLabel jLabel21;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JPanel jPanel6;
    private javax.swing.JPanel jPanel7;
    private javax.swing.JPanel jPanel8;
    private javax.swing.JPanel jPanel9;
    private javax.swing.JTextField jTextField1;
}
```