Práctica Final Laboratorio
Convocatoria Ordinaria
2022-2023

Asignatura: Paradigmas de Programación

Titulación: Ingeniería en Sistemas de Información

Autores: Iván Sánchez Ranz - Walter Huatay Rodríguez

DNI: 51546661G-06027011E

ÍNDICE

1. Análisis de alto nivel 1

2. Diseño general del sistema y herramientas de sincronización utilizadas 2

3. Clases principales que intervienen con su descripción (atributos y métodos)… 8

3.1. Clase Persona 8

3.2. Clase Empleado 9

3.3. Clase Buzón. 10

3.4. Clase Furgoneta. 11

3.5. Clase Interfaz 12

3.6. Clase Pausar. 14

3.7. Clase Registro. 14

3.8. Clase Distribuida (Interface remota) 15

3.9. Clase InterfazDistribuida. 15

3.10. Clase ObjetoRemoto. 16

4. Diagrama de clases 18

5. Anexo, código fuente. 19

1. Análisis de alto nivel

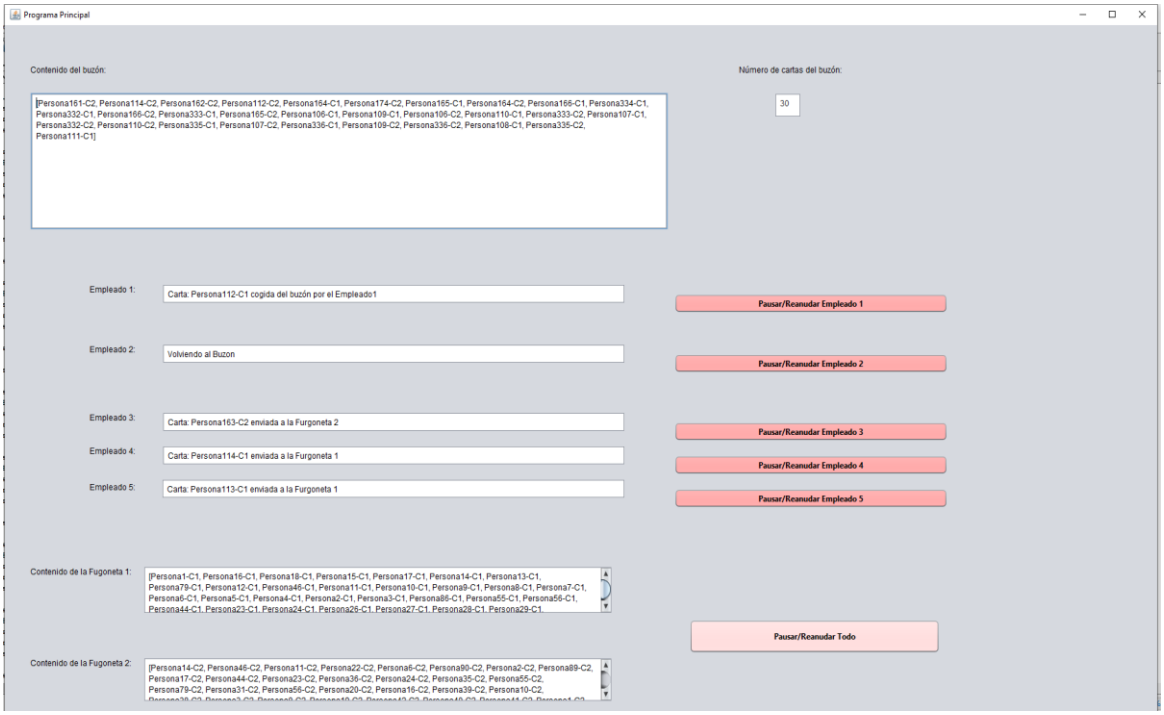
En este problema que se nos ha planteado, nos encontramos con un sistema de reparto de cartas, donde se generan **400 persona**s y a su vez cada persona generará **2 cartas**, las cuales tendrán su propio **identificador**. Es decir, no habrá una identificación igual en las **800 cartas** que tendrá el problema. Por ejemplo: **Persona80-C1, Persona80-C2**. Esta identificación será la primera y segunda carta de la persona 80.
También contamos con **5 empleados** que irán recogiendo las cartas que las personas depositen en el **buzón** y al recogerlas, las enviarán a **2 furgonetas** diferentes dependiendo de si es la primera o la segunda carta. Este **buzón será un buffer compartido entre Persona y Empleado**, el cual tendrá una capacidad máxima de **30 cartas.** Cabe destacar que en la furgoneta 1 se colocarán todas las primeras cartas de todas las personas y en la furgoneta 2 se colocarán todas las segundas cartas de las personas. Todas estas cartas serán dejadas por los 5 empleados, los cuales después de dejar la carta en la furgoneta, volverán al buzón a por más.
El programa cuenta con **2 interfaces ejecutables**. La primera **(Interfaz.java)** será el servidor de nuestra aplicación y generará y enviará los datos a la otra interfaz ejecutable **(IntefazDistribuida.java)** la cual será el cliente del programa. Todo este envío de datos se hará mediante un sistema cliente-servidor basado en RMI. No se podrá ejecutar la segunda Interfaz sin haber antes ejecutado la primera.
A continuación, se explica paso por paso la aplicación:

1. Se generan 2 cartas por persona que se van a enviar. En total **800 cartas**
2. Las cartas se envían al **buzón,** no sin antes identificar si es la primera carta o la segunda carta la que se está depositando en el buzón
3. Cuando se deja la primera o segunda carta, se espera un tiempo aleatorio entre 400 y 800 milisegundos para seguir depositando cartas.
4. La carta ya está en el buzón y se guarda en una lista. Aparte se muestra en la interfaz cuantas cartas hay en ese momento en el buzón y cuales son. Recordar que no puede haber más de **30 cartas en el buzón.**
5. Los **5 empleados** van cogiendo las cartas del buzón y comprueban si es la primera carta o la segunda carta de la persona.
    1. Si es la primera carta, el empleado la enviará a la **furgoneta 1**
    2. Si es la segunda carta, el empleado la enviará a la **furgoneta 2**
6. Al dejar la carta los empleados en la furgoneta esperarán un tiempo aleatorio entre 700 y 400 milisegundos y volverán al buzón a recoger otra carta.
7. En las furgonetas se irán dejando las cartas. **En cada furgoneta** se irá controlando que **solamente 1 empleado,** de los 5 posibles, esté dejando cartas en ese momento. Aparte se mostrará en la interfaz qué cartas están ya depositadas en la furgoneta por los empleados.
8. Se finalizará el trabajo cuando las **800 cartas** hayan sido depositadas por las personas en el buzón, recogidas y diferenciadas por los empleados y finalmente depositadas en sus debidas furgonetas. Todo este proceso mientras se monitorea y genera en la interfaz servidor **(Interfaz.java)**, siendo estos datos enviados a la interfaz cliente y siendo visualizados si así se desea **(interfazDistribuida.java)**.

2. Diseño general del sistema y herramientas de sincronización utilizadas

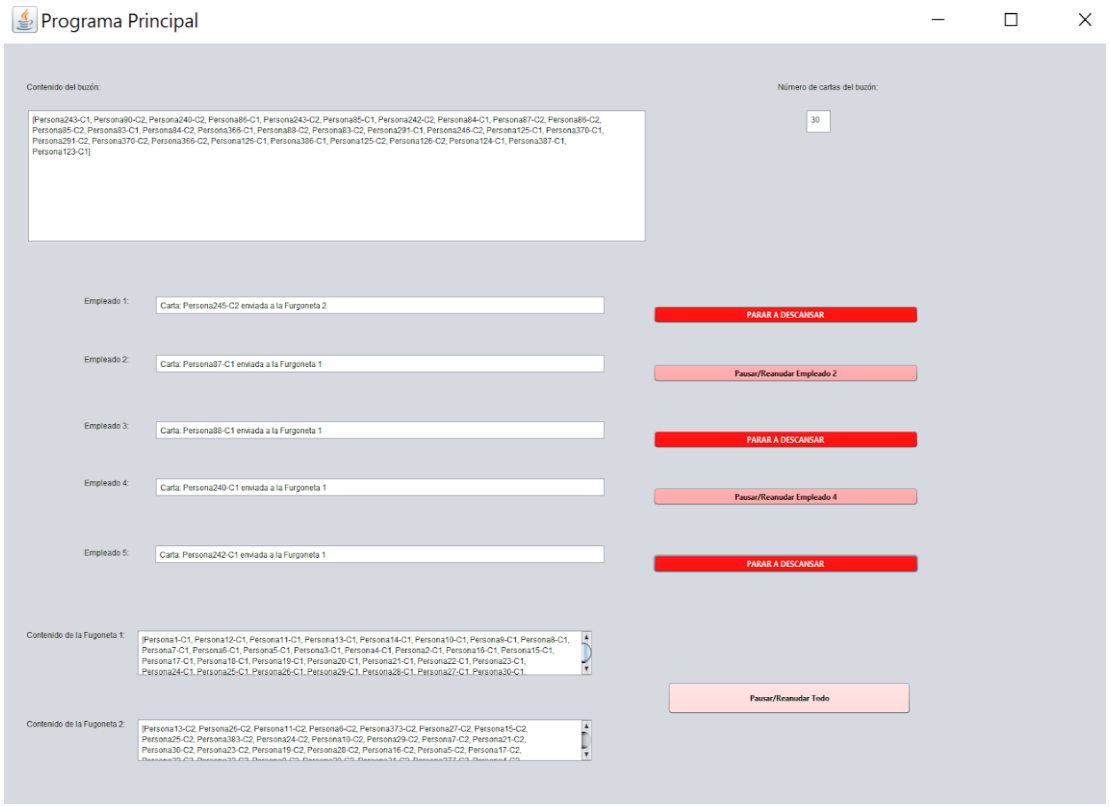
En primer lugar, se va a mostrar y explicar el diseño del sistema, haciendo hincapié en aquello que consideramos más relevante para el funcionamiento del programa. Y, en segundo lugar, se explicarán las herramientas de sincronización utilizadas:

En primer lugar, la clase **Interfaz**, a continuación, se mostrará lo más relevante de esta clase:



Esta imagen muestra al programa en plena ejecución. En la parte derecha se encuentran los **botones, en un tono rojo**, que se utilizan para **pausar** a cada empleado o el programa en su totalidad si así se desea. Dicho lo cual, a continuación, vamos a ver cómo cambia la interfaz al interactuar con estos botones:

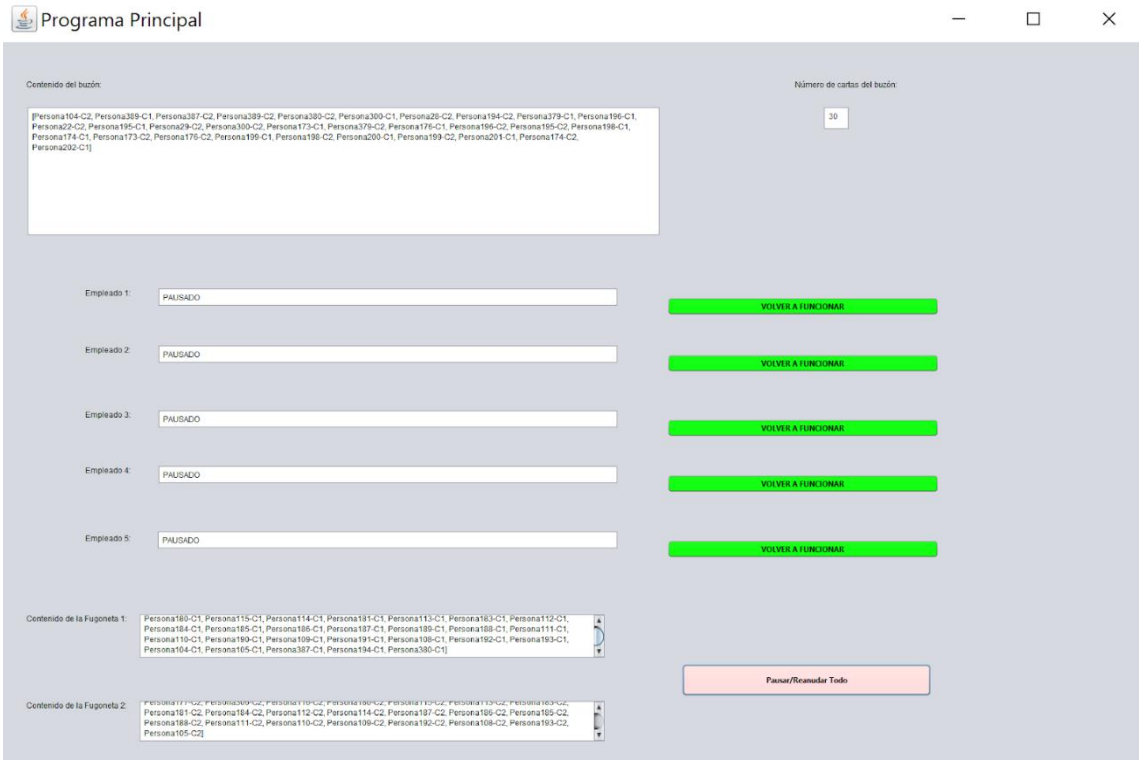
En esta imagen se muestra lo que sucede al pulsar los botones para detener la ejecución del empleado1, empleado3 y empleado5. Como se observa, los botones de estos empleados **cambian su color a verde**, y el título del botón a “Volver a funcionar”. Además, el contenido de los JTextPane referentes a estos empleados muestra que está “Pausado”.



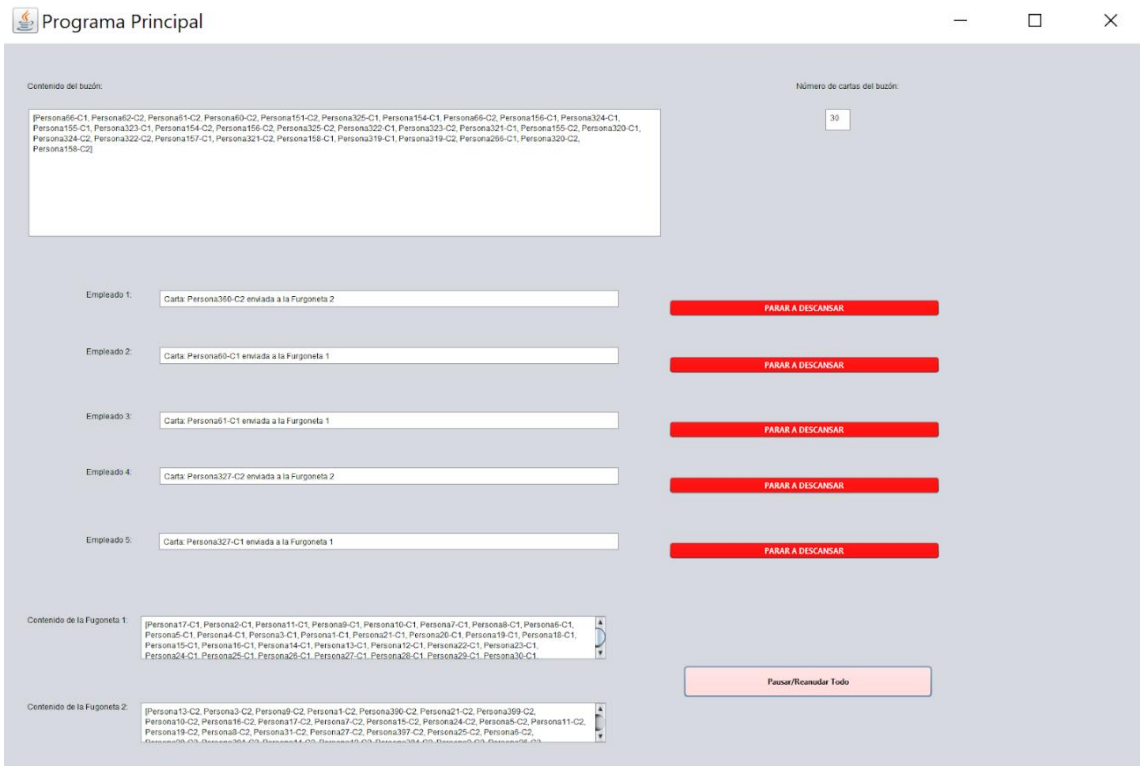
Si se vuelve a pulsar sobre estos tres botones, los empleados reanudarán su ejecución normal, el color de los botones volverá a ser rojo y el título de éstos

será “Parar a descansar”. Además, los JTextPane de estos empleados vuelven a la normalidad.

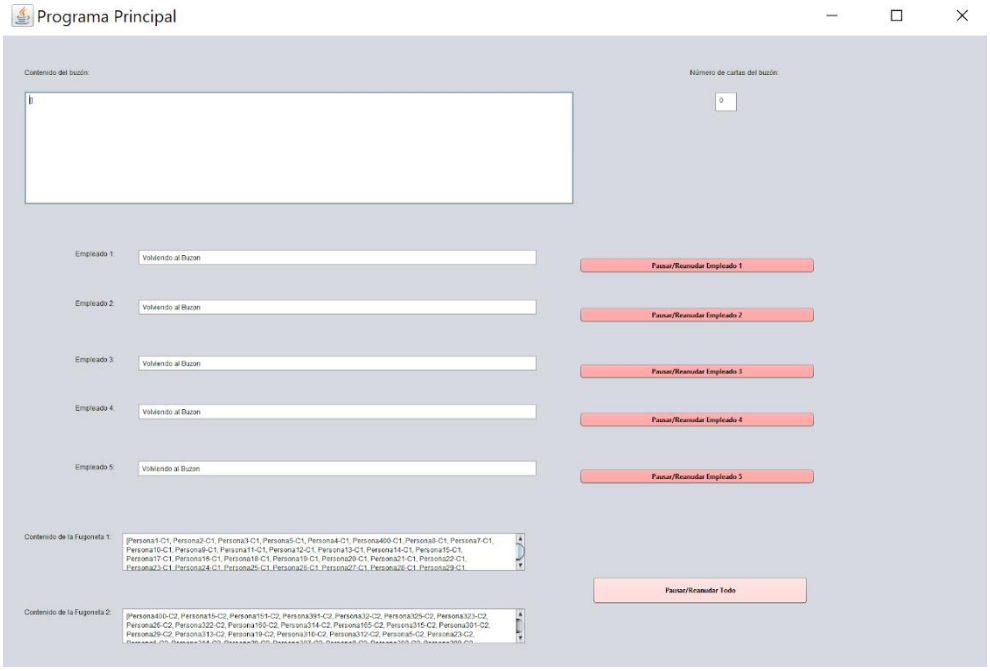
Una vez visto el funcionamiento de los botones de pausa y reanudación de los empleados, es momento del botón utilizado para **pausar y reanudar el programa en su totalidad**. Al pulsar este botón sucede lo siguiente: la ejecución de las personas y empleados se detiene, así que el contenido actual del buzón no se verá alterado, ya que las personas dejan de depositar cartas y los empleados dejan de recogerlas. Además, al detenerse los empleados se muestran los cambios que se producían en sus JTextPane y botones de pausa/reanudación que se comentó anteriormente. Por último, las furgonetas dejan de almacenar nuevas cartas ya que los empleados han dejado de cogerlas del buzón y depositarlas en cada furgoneta.
Esto se muestra en la siguiente imagen:



Si se quiere volver a reanudar la ejecución de todo el programa se debe volver a pulsar el mismo botón. Una vez hecho esto, las personas y empleados volverán a su ejecución normal, haciendo que el contenido del buzón, los JTextPane de cada empleado y el contenido de cada furgoneta se reanude. Además, se observa como los botones de pausa/reanudación de cada empleado han cambiado de color nuevamente, indicando que están listos para detenerse en caso de que así se desee.
Esto se muestra en la siguiente imagen:

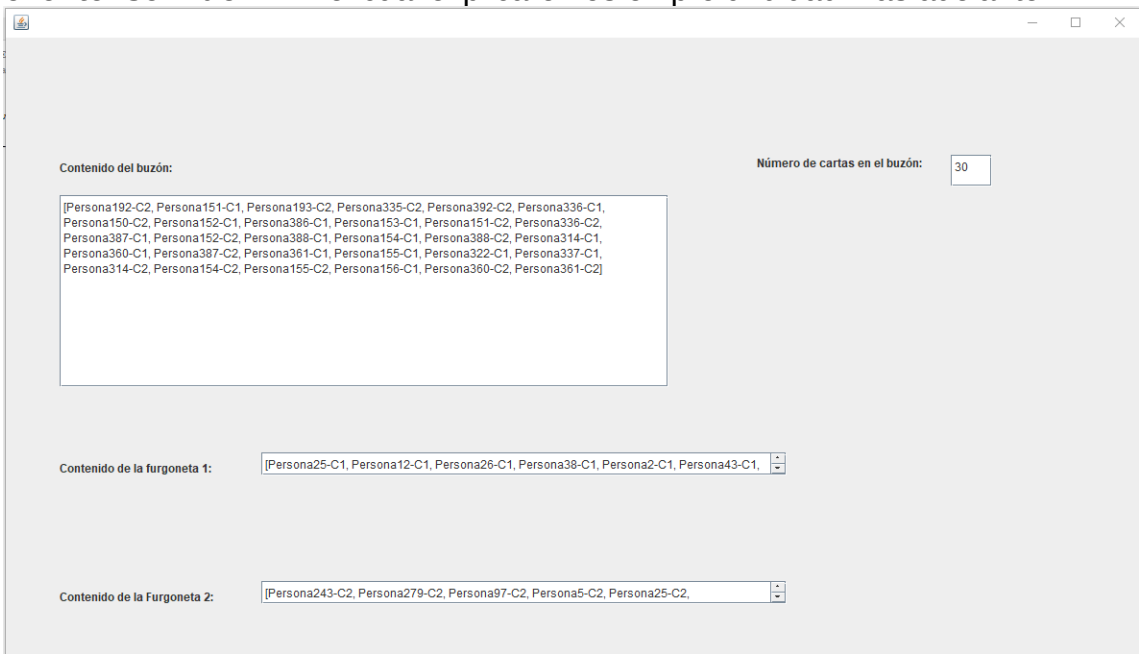


Para concluir con interfaz gráfica, se ha ejecutado el programa siendo **400 personas las que dejan 2 cartas en el buzón**, por tanto, una vez estas cartas han sido recogidas por los empleados y llevadas a las furgonetas en su totalidad, se muestra el contenido del buzón vacío y el "Número de cartas del buzón" es cero, como se muestra a continuación:

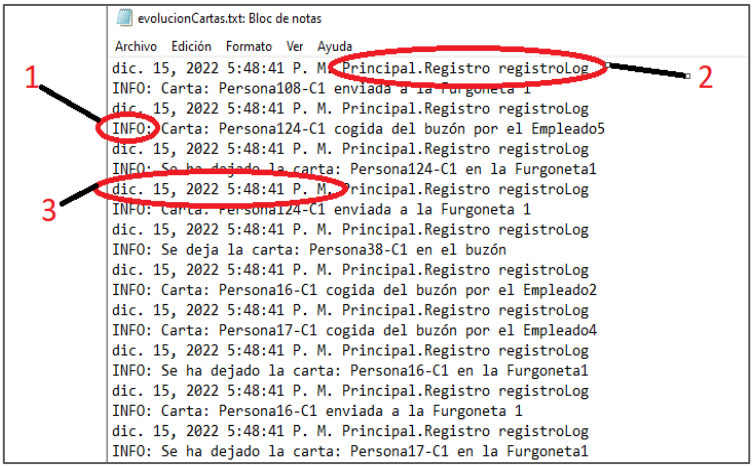


En cuanto a la **interfaz gráfica distribuida** va a tener un mecanismo muy simple. Su interfaz va a ser muy simple. Contendrá las cartas que tendremos en el buzón, las cartas que se guarden en las dos furgonetas y el número de cartas que hay en el buzón. Esta información y cartas es la misma que hay en la interfaz gráfica principal (concurrente), solamente cogemos esos datos y se los enviamos

a la interfaz gráfica distribuida mediante un mecanismo de envío de información **cliente- servidor RMI** el cual explicaremos en profundidad más adelante.



También, un elemento visual que nos gustaría reflejar en la memoria es como se quedan registrados los registros de la clase Logger al .txt que nosotros creamos.



1. Aquí podemos apreciar como dentro de la clase Logger tenemos diferentes opciones a la hora de reflejar un mensaje. En este caso, creímos el más adecuado que fuera el tipo “INFO”, el cual desea reflejar plasmar un texto informativo acerca de un evento de la aplicación. Por tanto, informaremos en el txt de lo que está ocurriendo en la aplicación.

2. Esta será la ruta de clases donde se ejecuta la acción que plasma el mensaje en el .txt. Entra dentro del paquete “Principal”, después a la clase “Registro” y por último se ejecuta el método “registroLog”.

3. Se puede ver que la clase Logger accede sin nosotros pedírselo a la fecha y hora de nuestra máquina para registrar el mensaje en el momento que está ocurriendo.

Por último, como se puede apreciar en la imagen, se va registrando las acciones que van ocurriendo, ya sea que Empleado está dejando una carta en cual Furgoneta o que Persona está dejando la carta en el buzón. Estando siempre la carta registrada y controlada en todo momento.

En segundo lugar, se va a explicar el funcionamiento de los mecanismos utilizados para la sincronización, éstos han sido utilizados en tres clases:

- **Clase Buzón**: actúa como clase compartida entre la clase persona y empleado.
  La condición para las personas es que solo pueden dejar cartas cuando haya hueco disponible en el buzón, sabiendo esto se ha decidido usar **monitores** para el cumplimiento de esta condición. El método **dejarCarta()**, utilizado por las personas, comprueba si la cantidad de cartas que contiene el arrayList es igual a la capacidad máxima del buzón(30 cartas), en caso afirmativo el proceso queda bloqueado mediante un wait() hasta que otro proceso le saque de esta situación, mientras que en caso de no estar lleno se añade la nueva carta al arrayList y se realiza un notifyAll() con el que despierto al resto de procesos bloqueados porque no hubieran cartas en el buzón.
  Por otro lado, los empleados deben esperar a que hayan cartas disponibles en el buzón, para ello usan el método **cogerCarta()**, en el cual mientras el arrayList esté vacío, el proceso tendrá que esperar. Pero en caso de haber cartas cogerá la carta de la posición [0], esta es la carta que más tiempo lleva en el buzón y realiza un notifyAll() para despertar al resto de procesos e informar que ahora hay hueco disponible para dejar cartas en el buzón.

- **Clase Furgoneta**: la condición para el acceso a la furgoneta es que solo un empleado puede entrar a depositar una carta a la vez en una determinada furgoneta, por tanto, ante esta condición hemos decidido utilizar **Locks** como mecanismo que garantice la exclusión mutua. Los métodos **dejarF1()** y **dejarF2()** son utilizados por los empleados para depositar cartas en la furgoneta 1 o en la furgoneta 2, en ambos métodos se produce lo siguiente: un empleado accede a una determinada furgoneta en exclusividad y puede depositar una carta, una vez hecho esto se desbloquea el cerrojo, permitiendo a otros empleados (procesos) acceder a la furgoneta en exclusividad.

- **Clase Pausar**: clase utilizada para detener a los empleados y el programa desde la interfaz. Contiene un boolean “cerrado” que empieza siendo false, y hay un cerrojo con una condition “parar” asociada a él. Con el método **mirar()**, las personas y empleados ven el estado de “cerrado”

antes de seguir con su ejecución, en caso de ser true deberá detenerse. El método **abrir()**, pone a false la variable “cerrado” y realiza un signalAll() para despertar a los procesos dormidos por la condition. El método **cerrar()**, pone a true la variable “cerrado”.
En conclusión, para pausar en la interfaz se usa el método cerrar() y para reanudar se usa el método abrir().

3. Clases principales que intervienen con su descripción (atributos y métodos)

En primer lugar, encontramos la parte **Concurrente,** ésta se encuentra en el paquete llamado Principal, en el cual encontramos clases con diferentes usos y objetivos para la realización del programa.

3.1. Clase Persona:

Esta clase ha sido modelado como hilo mediante la herencia de la clase **Thread**, además, esta clase recoge, en su método **run()**, todas las acciones que realizarán las personas durante su ejecución.

Atributos:

- **private Buzon buzon;** //Se llama a la clase buzón
- **private String identificador;** //La identificación de cada carta
- **private String carta1;**
- **private String carta2;**
- **private Registro log;** //Servirá para registrar todo lo ocurrido en la clase
- **private Pausar pausa** //Para pausar cuando se necesite
- **carta1 = identificador + "-C1";**
- **carta2 = identificador + "-C2";**
- **char caracter1 = carta1.charAt(carta1.length() - 1);**//Ultimo caracter de la carta 1 (1)
- **char caracter2 = carta2.charAt(carta2.length() - 1);**//Ultimo caracter de la carta 2 (2)

Métodos:

- ***public void run()***
  Este es el método principal de esta clase, llama a los métodos de buzón para dejar la primera y la segunda carta en el buzón, dejando entre una y otra un tiempo aleatorio entre 400 y 800 ms como especifica el enunciado.
  Queremos destacar y explicar cómo diferenciamos entre si es la primera carta o la segunda carta de la persona (ya que se envían 2 por persona).
  Esto lo hacemos generando 2 cartas, como se indica en los requisitos, una **acabada en C1 y otra en C2**. Entonces cogemos el último carácter del nombre de la carta (Ejemplo: Persona1-C1) que puede ser o ‘1’ o ‘2’ y dependiendo de cuál sea el carácter se realizará una acción u otra. Por ejemplo, en la clase Empleado,

usando este mecanismo, se enviará una carta a la furgoneta1 o a la furgoneta2 dependiendo del último carácter de la práctica.

3.2. Clase Empleado:

Esta clase también ha sido modelada como hilo mediante la herencia de la clase **Thread**, además, esta clase recoge, en su método **run()**, todas las acciones que realizarán los empleados durante su ejecución.

Atributos:

- **private Buzon buzon;** //Se llama a la clase buzón
- **private Furgoneta furgoneta;** //Se llama a la clase furgoneta
- **private String identificador;** /La identificación de cada carta
- **private String carta;** //La carta recogida de buzón
- **private JTextPane texto;** //El texto de la interfaz donde se mostrará la acción de cada empleado
- **private Registro log;** //Servirá para registrar todo lo ocurrido en la clase
- **private Pausar pausa;**//Para pausar cuando se necesite

Métodos:

- ***public void run()***
  Este es el método principal de esta clase, llama a métodos de la clase buzón para que el empleado recoja las cartas que éste contiene, y también llama a métodos de la clase furgoneta para que los empleados depositen las cartas en la furgoneta correspondiente, y a destacar también el uso del método registroLog(), con este método se guardará en el archivo log las acciones que vaya realizando cada empleado. Como hemos explicado en la clase persona, diferenciaremos qué carta va a cada furgoneta dependiendo del último carácter del nombre de la carta. A la furgoneta 1 irán todas las Cartas1 y a la furgoneta2 irán todas las Cartas2.

  char caracter = cartaEmpleado.charAt(cartaEmpleado.length() - 1);

- ***public void textoEmpleado(JTextPane texto,String mensaje)***
  Método utilizado para establecer en el JTextPane del empleado la acción que ha realizado de depositar cartas en la furgoneta 1 o en la furgoneta 2.

- ***public void irBuzon(JTextPane texto, String mensaje)***
  Método utilizado para establecer en el JTextPane del empleado que éste ha cogido una carta del buzón.

- ***public void volverBuzon(JTextPane texto)***
  Método utilizado para establecer en el JTextPane del empleado que éste está volviendo al buzón tras haber dejado las cartas en las furgonetas.

3.3. Clase Buzón: esta clase actúa como una clase compartida (buffer compartido) entre las clases persona y empleado. Ambas clases cuentan con diferentes condiciones para acceder al buzón, por tanto, deberá contar con mecanismos que garanticen su funcionamiento. Como ya se ha explicado anteriormente, el mecanismo utilizado son **monitores**.
Cabe destacar, que en esta clase se almacenan en un arrayList las cartas depositadas por las personas y que los empleados deben llevar a las furgonetas, definiendo métodos para que puedan realizar estas acciones.

Atributos:

- **ArrayList<String> cartas = new ArrayList<>();** //Buffer del buzón donde se guardan y se sacan las cartas
- **private int capacidad;** // Capacidad máxima (30)
- **private JTextPane nCartas;** //Se mostrará el nº de cartas en el momento
- **private JTextPane bufferbuzon;** //Se mostrará que cartas hay en el buffer

Métodos:

- ***public synchronized void dejarCarta(String carta) throws InterruptedException***
  Método utilizado por las personas para depositar sus dos cartas en el buzón.

- ***public synchronized String cogerCarta(String carta) throws InterruptedException***
  Método utilizado por los empleados para coger las cartas del buzón.

- ***public void cantidadCartas()***
  Método para mostrar en la interfaz el número de cartas que hay en el buzón.

- ***public void elementosArraylist()***
  Método para que en la interfaz se muestre las cartas hay en el buzón.

- ***public String cantidadCartasDistribuida()***
  Método para mostrar, en la interfaz de la parte Distribuida, el número de cartas que hay en el buzón.

- ***public String elementosArraylistDistribuida()***
  Método para que, en la interfaz de la parte Distribuida, se muestre las cartas hay en el buzón.

3.4. Clase Furgoneta: en esta clase se almacenan las cartas que han depositado los empleados en cada furgoneta. Para cada furgoneta se ha creado un arrayList donde se guardan las cartas depositadas en ellas.
Además, esta clase cuenta con condiciones de acceso para los empleados, por ello, se recurrirá al uso de mecanismos que garanticen la exclusión mutua. Como ya se ha explicado anteriormente, el mecanismo utilizado son **Locks.**

Atributos:

- **ArrayList<String> F1 = new ArrayList<>();** //Lista donde se guardan las cartas de la furgoneta 1
- **ArrayList<String> F2 = new ArrayList<>();** //Lista donde se guardan las cartas de la furgoneta 2
- **private JTextPane textoF1;** //Mostrará el contenido de la F1
- **private JTextPane textoF2;** //Mostrará el contenido de la F2
- **private Registro log;** //Servirá para registrar todo lo ocurrido en la clase
- **private Lock cerrojo = new ReentrantLock();** //Se declarará un cerrojo para controlar que solamente actúa 1 empleado.

Métodos:

- ***public void dejarF1(String carta)***
  Método utilizado por los empleados para depositar cartas en el arrayList perteneciente a la furgoneta 1.

- ***public void dejarF2(String carta)***
  Método utilizado por los empleados para depositar cartas en el arrayList perteneciente a la furgoneta 2.

- ***public void elementosArraylistF1()***
  Método utilizado para que en la interfaz aparezcan las cartas que hay almacenadas en la furgoneta 1.

- ***public void elementosArraylistF2()***
  Método utilizado para que en la interfaz aparezcan las cartas que hay almacenadas en la furgoneta 2.

- ***public String elementosArraylistF1Distribuida()***
  Método utilizado para que en la interfaz Distribuida aparezcan las cartas que hay almacenadas en la furgoneta 1.

- ***public String elementosArraylistF2Distribuida()***
  Método utilizado para que en la interfaz Distribuida aparezcan las cartas que hay almacenadas en la furgoneta 2.

3.5. Clase Interfaz: clase donde se realiza la interfaz gráfica y donde se inicializan todos aquellos objetos que serán utilizados para el correcto funcionamiento del programa. Apartes esta clase funcionará como **Servidor** en la parte distribuida. Se encargará de coger los datos de los TextPane y enviarlos a un **Objeto Remoto** para que así les pueda llegar al **Cliente (la Interfaz Gráfica Distribuida)**. Por tanto, en el main de esta clase haremos lo siguiente para que pueda funcionar como servidor:

1 - Declarar la Clase Interfaz que va a ser enviada al objeto remoto.
2 - Declarar la Clase ObjetoRemoto donde se realizará las funciones que cogerán los datos de la Interfaz Gráfica Concurrente para ser enviados a la Interfaz Gráfica Distribuida. Por tanto, le meteremos a esta Clase ObjetoRemoto la interfaz por parámetro.
3 - El registry de la clase Registry con el valor de (1099)
4 - El Naming.rebind("/localhost/PECLIvanWalter")

Por último, vamos a explicar el funcionamiento de los botones de la interfaz. Los botones tendrán dos estados. Estos dos estados se controlan mediante un booleano llamado "**parado**" y "**reanudado**" que se inicializarán **FALSE**. Dependiendo del estado del booleano, el botón hará que un empleado pare de recoger cartas o prosiga con su tarea. Aparte, cada vez que se pulse el botón cambiará la apariencia estética del mismo.También en cada acción del botón llamaremos a la clase Pausar mediante los metodos **.cerrar()** si el empleado va a descansar o **.abrir()** si el empleado va a seguir.
El botón **"PARAR/REANUDAR TODO"** realizará este funcionamiento en todos los botones a la vez dependiendo de si queremos parar o reanudar todos los botones.

Atributos:

- **private int numero_personas;** //Las 400 personas
- **private boolean parado = false**; //Booleano que controla el funcionamiento de los botones. Hace que el empleado pare.
- **private boolean reanudado = false;**//Booleano que controla el funcionamiento de los botones. Hace que el empleado reanude.
- **private Pausar pausaGlobal = new Pausar();** //Pausa todo el proceso
- **private Pausar pausaEmpleado1 = new Pausar();** //Pausa 1 empleado, lo mismo para todas
- **private Pausar pausaEmpleado2 = new Pausar();**
- **private Pausar pausaEmpleado3 = new Pausar();**
- **private Pausar pausaEmpleado4 = new Pausar();**
- **private Pausar pausaEmpleado5 = new Pausar();**

Métodos:

- ***private void jButton_PausarRenaudarTodoMouseClicked(java.awt.event.MouseEvent evt)***

Método empleado para que el botón para pausar/reanudar todo el programa funcione correctamente.

- ***private void pausarReanudarEmpleado1MouseClicked(java.awt.event.MouseEvent evt)***
  Método empleado para que el botón de pausar/reanudar empleado1 funcione correctamente.

- ***private void pausarReanudarEmpleado2MouseClicked(java.awt.event.MouseEvent evt)***
  Método empleado para que el botón de pausar/reanudar empleado2 funcione correctamente.

- ***private void pausarReanudarEmpleado3MouseClicked(java.awt.event.MouseEvent evt)***
  Método empleado para que el botón de pausar/reanudar empleado3 funcione correctamente.

- ***private void pausarReanudarEmpleado4MouseClicked(java.awt.event.MouseEvent evt)***
  Método empleado para que el botón de pausar/reanudar empleado4 funcione correctamente.

- ***private void pausarReanudarEmpleado5MouseClicked(java.awt.event.MouseEvent evt)***
  Método empleado para que el botón de pausar/reanudar empleado5 funcione correctamente.

- ***public String buzonTexto()***
  Metodo que devolverá el contenido del JTextPane del contenido de las cartas del buzón para ser posteriormente enviado al Objeto Remoto
  String textoBuzon = jTextPane2.getText();

- ***public String cantidadBuzon()***
  Método que devolverá el contenido del JTextPane del número de cartas que hay en el buzón para ser posteriormente enviado al Objeto Remoto
  String cantidadBuzon = jTextPane1.getText();

- ***public String F1Texto()***
  Método que devolverá el contenido del JTextPane de la Furgoneta1 para ser posteriormente enviado al Objeto Remoto.
  String furgoneta1 = jTextPane8.getText();

- ***public String F2Texto()***
  Método que devolverá el contenido del JTextPane de la Furgoneta2 para ser posteriormente enviado al Objeto Remoto
  String furgoneta2 = jTextPane9.getText();

3.6. Clase Pausar: esta clase se ha realizado gracias a los apuntes subidos por los profesores, gracias a ella conseguimos el objetivo de pausar/reanudar a los empleados y la ejecución total del programa al pulsar determinados botones para ello. En apartados anteriores se ha explicado esta clase y sus métodos con mayor detalle.

Atributos:

- **private boolean cerrado=false;** // Variable boolean que va cambiando su estado dependiendo si se quiere pausar o reanudar la ejecución.
- **private Lock cerrojo = new ReentrantLock();** //Creación del cerrojo.
- **private Condition parar = cerrojo.newCondition();** //Condition asociada al cerrojo.

Métodos:

- ***public void mirar()***
  Mediante este método, los objetos que puedan ser detenidos comprobarán el estado de la variable “cerrado” para saber si pueden continuar con su ejecución o no.

- ***public void abrir()***
  Desde la interfaz gráfica, si se quiere reanudar a un empleado o la ejecución total del programa, se llamará a este método.

- ***public void cerrar()***
  Desde la interfaz gráfica, si se quiere parar a un empleado o la ejecución total del programa, se llamará a este método.

3.7. Clase Registro: Esta clase será la encargada de **registrar** todos los pasos que ocurren en la aplicación desde que las Personas dejan las cartas en el Buzón hasta que los Empleados dejan cada carta en las Furgonetas. Esto será posible utilizando la c**lase Logger de Java**, la cual nos dará la facilidad de registrar en un archivo .txt que creamos, poder dejar registrado las acciones en dicho archivo. Cuando se produzca una acción que deba ser registrada, lo haremos mediante una variable String que será enviada por parámetro a esta Clase para que se guarde en el .txt. En caso de no llegar a introducir un mensaje, saltará un mensaje de error en la consola que se registrará mediante un **log.warning**. La clase Logger cuenta con métodos como **severe, config o fine** que no usaremos en esta práctica.

Atributos:

- **private Logger logger ;** //Se utilizará la clase Logger de Java
- **private File archivo;** //Archivo (txt) donde insertaremos los logs
- **FileHandler archivoManipulable;** //Direccion donde se guardará o generará el documento de texto

Métodos:

- ***public void registroLog(String mensaje)***
  Este método se encargará de recibir por parámetro un **String** que va a ser un mensaje dependiendo de la acción y la clase donde se realice. Este String se meterá dentro de un **log.info** que posteriormente se guardará en un .**txt** y se mostrará por consola.

En segundo lugar, encontramos la **parte Distribuida**, ésta se encuentra en el paquete llamado Distribuida.

3.8. Clase Distribuida (Interface remota): simplemente se declaran los métodos que van a ser enviados entre el cliente y servidor. Imprescindible extender de la clase **Remote** y que cada clase cuente con un **RemoteException.**

Métodos:

- **String buzonTexto()** //Devuelve qué cartas hay en el buzón de la interfaz

- **String cantidadBuzon()** //Devuelve cuantas cartas hay de la interfaz

- **String F1Texto()** //Devuelve qué cartas hay en la furgoneta 1 de la interfaz

- **String F2Texto()** //Devuelve qué cartas hay en la furgoneta 2 de la interfaz
  En todos los métodos será necesario un **throws RemoteException, InterruptedException;**

3.9. Clase InterfazDistribuida: Esta Interfaz **será la interfaz cliente** de nuestra aplicación. Pedirá los datos al servidor y mediante un objeto remoto y una interface, tendremos los datos que el cliente solicite. En este caso el cliente solicitará el contenido de Buzón, Furgoneta1, Furgoneta2 y el nº de cartas restantes en el Buzón de la Interfaz Gráfica Principal. Le llegarán mediante **RMI** y los podrá introducir en su interfaz. Esto es posible gracias a que la clase Interfaz.java (servidor) crea objetos remotos y los hace visibles. Por tanto, el servidor esperará que el cliente invoque a los métodos de los objetos remotos para poder coger la información y plasmarla en esta clase InterfazDistribuida.java.
Para que esto sea posible se debe usar la **clase Naming** que llamará a la **Interface**.:
Naming.lookup("/localhost/PECLIvanWalter");

Para finalizar esta clase y la interfaz se introducirá los datos en el JTextPane mediante un. setText.
jTextPane1.setText(obj.buzonTexto());

Atributos:

- **private static Registro registro;** //Servirá para registrar todo lo ocurrido en la clase
- **private static Logger logger;** //Poder realizar warnings en las diferentes excepciones que puede dar el programa
- **private Distribuida obj;** //Para poder llamar a los métodos de la interface.

Métodos:

- ***public void introducirTexto() throws RemoteException, InterruptedException***
  Método encargado de introducir los datos en los **JTextPane** dependiendo de cual es(buzón,furgoneta1,furgoneta2,nºcartasbuzon).

- ***public static void main(String args[]) throws InterruptedException, RemoteException, NotBoundException, MalformedURLException***
  Procede a llamar al método anterior dentro de un **While (true**

- ***public void run()***
  Mantiene la interfaz visible y controla las excepciones que se puedan producir durante el proceso, mediante la clase Logger y los log.warning.

3.10. Clase ObjetoRemoto: Esta es la Clase objeto remoto del servidor y cliente, donde se retornarán en cada método los datos de la clase Interfaz para ser enviados a la Interfaz Distribuida y poder ser adjuntados ahí.

Atributos:

- **private Interfaz interfaz;** //Se llama a la interfaz principal para coger sus datos en los métodos.

Métodos:

- ***public String buzonTexto() throws RemoteException, InterruptedException***
  Método que devolverá el contenido del JTextPane del **contenido** de las cartas del **buzón** para ser posteriormente introducido en la interfaz distribuida.

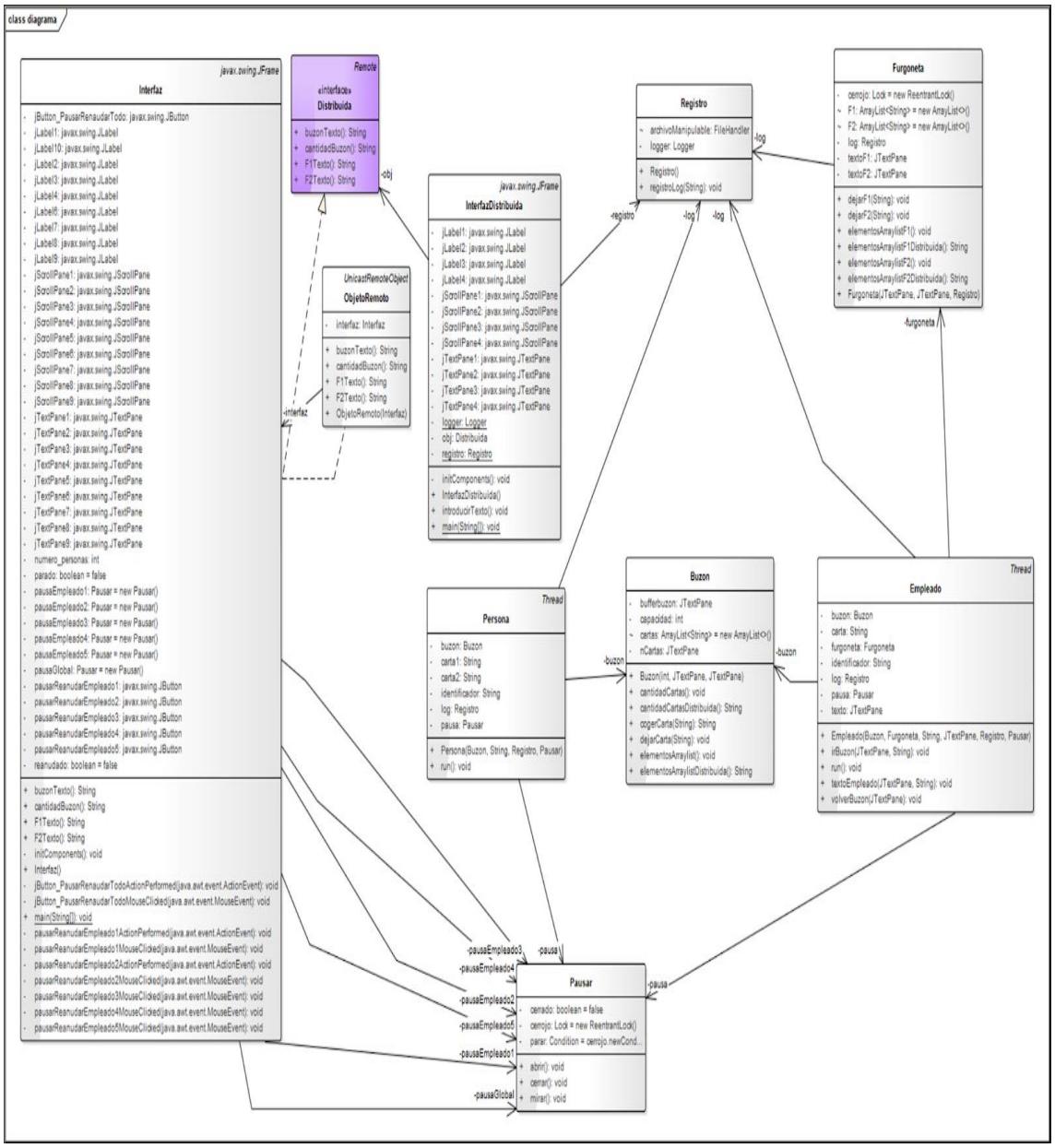
- ***public String cantidadBuzon() throws RemoteException, InterruptedException***

Método que devolverá el contenido del JTextPane de la **cantidad** de las cartas del **buzón** para ser posteriormente introducido en la interfaz distribuida.

- ***public String F1Texto() throws RemoteException, InterruptedException***
  Método que devolverá el contenido del JTextPane del contenido de las cartas en la **furgoneta 1** para ser posteriormente introducido en la interfaz distribuida.

- ***public String F2Texto() throws RemoteException, InterruptedException***
  Método que devolverá el contenido del JTextPane del contenido de las cartas en la **furgoneta 2** para ser posteriormente introducido en la interfaz distribuida.

4. Diagrama de clases



Este es el diagrama de clases resultante de la realización de este programa, de él se pueden destacar ciertas cosas. Se observan las relaciones entre las diferentes clases, destacando como la clase **buzón** actúa claramente como clase compartida entre las clases persona y empleado.
A destacar también la clase **Registro**, la cual tiene varias relaciones de asociación con las clases persona, empleado, furgoneta e interfazDistribuda, esto se debe a que esta clase sirve para almacenar en un **archivo .txt las** acciones que estas clases realizan.
Respecto a la clase **pausar**, vemos que tiene relaciones de asociación con las clases persona, empleado y la interfaz, ya que de esta clase depende que se realicen las pausas y reanudaciones al pulsar los botones de la interfaz.
Por último, cabe destacar como arriba a la izquierda tienen lugar las relaciones de las clases e interface de la parte **Distribuida** de esta práctica.

5. Anexo, código fuente.

PAQUETE PRINCIPAL

CLASE BUZÓN

```
package Principal;

import java.util.ArrayList;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import javax.swing.JTextPane;

public class Buzon {

    ArrayList<String> cartas = new ArrayList<>();
    private int capacidad;
    private JTextPane nCartas;
    private JTextPane bufferbuzon;

    public Buzon(int capacidad, JTextPane nCartas,
JTextPane bufferbuzon) {
        this.capacidad = capacidad;
        this.nCartas = nCartas;
        this.bufferbuzon = bufferbuzon;
    }

    public synchronized void dejarCarta(String carta)
throws InterruptedException {

        try {
            while (cartas.size() == capacidad) { //si el
buffer esta lleno
                wait();
            }
            cartas.add(carta);
            cantidadCartas();
            elementosArraylist();

        } finally {
            notifyAll();

        }
    }

    public synchronized String cogerCarta(String carta)
throws InterruptedException {
        try {
```

```
            while (cartas.isEmpty()) { //si el buffer esta
lleno
                wait();
            }
            String cartaCogida = cartas.get(0);
            cartas.remove(0);
            cantidadCartas();
            elementosArraylist();
            return cartaCogida;
        } finally {
            notifyAll();

        }

    }

    public void cantidadCartas() {

        int cantidadCartas = cartas.size();
        String cantidad = Integer.toString(cantidadCartas);
        nCartas.setText(cantidad);

    }

    public void elementosArraylist() {

        String devolver = (cartas.toString() + "\t");
        bufferbuzon.setText(devolver);

    }

    public String cantidadCartasDistribuida() {

        int cantidadCartas = cartas.size();
        String cantidad = Integer.toString(cantidadCartas);
        nCartas.setText(cantidad);

        return cantidad;
    }

    public String elementosArraylistDistribuida() {

        String devolver = (cartas.toString() + "\t");

        return devolver;

    }
}
```

CLASE EMPLEADO

```
package Principal;

import static java.lang.Thread.sleep;
import javax.swing.JTextPane;

public class Empleado extends Thread {

    private Buzon buzon;
    private Furgoneta furgoneta;
    private String identificador;
    private String carta;
    private JTextPane texto;
    private Registro log;
    private Pausar pausa;

    public Empleado(Buzon buzon, Furgoneta furgoneta,
String identificador, JTextPane texto, Registro log, Pausar
pausa) {
        this.buzon = buzon;
        this.furgoneta = furgoneta;
        this.identificador = identificador;
        this.texto = texto;
        this.log = log;
        this.pausa = pausa;

    }

    public void run() {
        while (true) {
            try {

                int tiempo_aleatorio = (int) (Math.random()
* (700 - 400 + 1) + 400);
                pausa.mirar();
                String cartaEmpleado =
buzon.cogerCarta(carta);
                String mensajeCogerCarta = "Carta: " +
cartaEmpleado + " cogida del buzón por el " +
identificador;
                pausa.mirar();
                irBuzon(texto, mensajeCogerCarta);
                sleep(200); //LO HACEMOS PARA QUE SE VEA EL
TEXTO DEL MENSAJE, EN CASO DE NO HACERLO EL MENSAJE NO SE
LLEGA A APRECIAR EN LA INTERFAZ
                log.registroLog(mensajeCogerCarta);

                char caracter =
cartaEmpleado.charAt(cartaEmpleado.length() - 1);

                if (caracter == '1') {
```

```
                    pausa.mirar();
                    furgoneta.dejarF1(cartaEmpleado);
                    String mensajeE1 = "Carta: " +
cartaEmpleado + " enviada a la Furgoneta 1";

                    pausa.mirar();
                    log.registroLog(mensajeE1);
                    textoEmpleado(texto, mensajeE1);
                    sleep(tiempo_aleatorio);
                    pausa.mirar();
                    volverBuzon(texto);
                    sleep(300); //LO HACEMOS PARA QUE SE
VEA EL TEXTO DEL MENSAJE, EN CASO DE NO HACERLO EL MENSAJE
NO SE LLEGA A APRECIAR EN LA INTERFAZ
                }
                if (caracter == '2') {

                    pausa.mirar();
                    furgoneta.dejarF2(cartaEmpleado);
                    String mensajeE2 = "Carta: " +
cartaEmpleado + " enviada a la Furgoneta 2";
                    pausa.mirar();
                    log.registroLog(mensajeE2);
                    textoEmpleado(texto, mensajeE2);
                    sleep(tiempo_aleatorio);
                    pausa.mirar();
                    volverBuzon(texto);
                    sleep(300); //LO HACEMOS PARA QUE SE
VEA EL TEXTO DEL MENSAJE, EN CASO DE NO HACERLO EL MENSAJE
NO SE LLEGA A APRECIAR EN LA INTERFAZ
                }

            } catch (Exception e) {
            }

        }
    }

    public void textoEmpleado(JTextPane texto, String
mensaje) {

        texto.setText(mensaje);

    }

    public void irBuzon(JTextPane texto, String mensaje) {
        texto.setText(mensaje);

    }
```

```
    public void volverBuzon(JTextPane texto) {
        texto.setText("Volviendo al Buzon");


    }
}
```

CLASE FURGONETA

```
package Principal;

import java.util.ArrayList;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import javax.swing.JTextPane;

public class Furgoneta {

    ArrayList<String> F1 = new ArrayList<>();
    ArrayList<String> F2 = new ArrayList<>();
    private JTextPane textoF1;
    private JTextPane textoF2;
    private Registro log;
    private Lock cerrojo = new ReentrantLock();

    public Furgoneta(JTextPane textoF1, JTextPane textoF2,
Registro log) {
        this.textoF1 = textoF1;
        this.textoF2 = textoF2;
        this.log = log;

    }

    public void dejarF1(String carta) {
        cerrojo.lock();
        F1.add(carta);
        elementosArraylistF1();
        String mensaje = "Se ha dejado la carta: " + carta
+ " en la Furgoneta1";
        log.registroLog(mensaje);
        cerrojo.unlock();

    }

    public void dejarF2(String carta) {
        cerrojo.lock();
        F2.add(carta);
        elementosArraylistF2();
        String mensaje = "Se ha dejado la carta: " + carta
+ " en la Furgoneta2";
```

```
        log.registroLog(mensaje);
        cerrojo.unlock();
    }

    public void elementosArraylistF1() {

        textoF1.setText(F1.toString());

    }

    public void elementosArraylistF2() {

        textoF2.setText(F2.toString());

    }

    public String elementosArraylistF1Distribuida() {

        return F1.toString();

    }

    public String elementosArraylistF2Distribuida() {

        return F2.toString();

    }
}
```

CLASE INTERFAZ

```
package Principal;

import Distribuida.ObjetoRemoto;
import java.awt.Color;
import java.awt.TextField;
import java.io.File;
import java.io.IOException;
import static java.lang.Thread.sleep;
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.logging.FileHandler;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;

public class Interfaz extends javax.swing.JFrame {

    private int numero_personas;
    private boolean parado = false;
    private boolean reanudado = false;
```

```
    private Pausar pausaGlobal = new Pausar();
    private Pausar pausaEmpleado1 = new Pausar();
    private Pausar pausaEmpleado2 = new Pausar();
    private Pausar pausaEmpleado3 = new Pausar();
    private Pausar pausaEmpleado4 = new Pausar();
    private Pausar pausaEmpleado5 = new Pausar();

    public Interfaz() {
        initComponents();
        Registro log = new Registro();
        numero_personas = 401;

        Buzon buzon = new Buzon(30, jTextPane1,
jTextPane2);
        Furgoneta furgoneta = new Furgoneta(jTextPane8,
jTextPane9, log);
        Empleado e1 = new Empleado(buzon, furgoneta,
"Empleado1", jTextPane3, log,
pausaEmpleado1);//pausaEmpleado1
        Empleado e2 = new Empleado(buzon, furgoneta,
"Empleado2", jTextPane4, log,
pausaEmpleado2);//pausaEmpleado2
        Empleado e3 = new Empleado(buzon, furgoneta,
"Empleado3", jTextPane5, log,
pausaEmpleado3);//pausa1Empleado3
        Empleado e4 = new Empleado(buzon, furgoneta,
"Empleado4", jTextPane6, log,
pausaEmpleado4);//pausa1Empleado4
        Empleado e5 = new Empleado(buzon, furgoneta,
"Empleado5", jTextPane7, log,
pausaEmpleado5);//pausaEmpleado5
        e1.start();
        e2.start();
        e3.start();
        e4.start();
        e5.start();

        for (int i = 1; i < numero_personas; i++) {
            Persona p = new Persona(buzon, "Persona" + i,
log, pausaGlobal);
            p.start();

        }

    }

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

```
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jButton_PausarRenaudarTodo = new
javax.swing.JButton();
        pausarReanudarEmpleado1 = new
javax.swing.JButton();
        pausarReanudarEmpleado2 = new
javax.swing.JButton();
        pausarReanudarEmpleado3 = new
javax.swing.JButton();
        pausarReanudarEmpleado4 = new
javax.swing.JButton();
        pausarReanudarEmpleado5 = new
javax.swing.JButton();
        jLabel10 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextPane1 = new javax.swing.JTextPane();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTextPane2 = new javax.swing.JTextPane();
        jScrollPane3 = new javax.swing.JScrollPane();
        jTextPane3 = new javax.swing.JTextPane();
        jScrollPane4 = new javax.swing.JScrollPane();
        jTextPane4 = new javax.swing.JTextPane();
        jScrollPane5 = new javax.swing.JScrollPane();
        jTextPane5 = new javax.swing.JTextPane();
        jScrollPane6 = new javax.swing.JScrollPane();
        jTextPane6 = new javax.swing.JTextPane();
        jScrollPane7 = new javax.swing.JScrollPane();
        jTextPane7 = new javax.swing.JTextPane();
        jScrollPane8 = new javax.swing.JScrollPane();
        jTextPane8 = new javax.swing.JTextPane();
        jScrollPane9 = new javax.swing.JScrollPane();
        jTextPane9 = new javax.swing.JTextPane();

        setDefaultCloseOperation(javax.swing.WindowConstant
s.EXIT_ON_CLOSE);
        setTitle("Programa Principal");
        setBackground(new java.awt.Color(102, 255, 204));

        jLabel1.setText("Contenido del buzón:");

        jLabel2.setText("Número de cartas del buzón:");

        jLabel3.setText("Empleado 1:");
```

```
        jLabel4.setText("Empleado 5:");

        jLabel6.setText("Empleado 3:");

        jLabel7.setText("Empleado 4:");

        jLabel8.setText("Contenido de la Fugoneta 1:");

        jLabel9.setText("Contenido de la Fugoneta 2:");

        jButton_PausarRenaudarTodo.setBackground(new
java.awt.Color(255, 204, 204));
        jButton_PausarRenaudarTodo.setFont(new
java.awt.Font("Segoe UI", 1, 12)); // NOI18N
        jButton_PausarRenaudarTodo.setText("Pausar/Reanudar
Todo");
        jButton_PausarRenaudarTodo.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mouseClicked(java.awt.event.MouseEvent evt) {
                jButton_PausarRenaudarTodoMouseClicked(evt)
;
            }
        });
        jButton_PausarRenaudarTodo.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton_PausarRenaudarTodoActionPerformed(e
vt);
            }
        });

        pausarReanudarEmpleado1.setBackground(new
java.awt.Color(255, 153, 153));
        pausarReanudarEmpleado1.setFont(new
java.awt.Font("Segoe UI", 1, 12)); // NOI18N
        pausarReanudarEmpleado1.setText("Pausar/Reanudar
Empleado 1");
        pausarReanudarEmpleado1.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mouseClicked(java.awt.event.MouseEvent evt) {
                pausarReanudarEmpleado1MouseClicked(evt);
            }
        });
        pausarReanudarEmpleado1.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
```

```
                pausarReanudarEmpleado1ActionPerformed(evt)
;
            }
        });

        pausarReanudarEmpleado2.setBackground(new
java.awt.Color(255, 153, 153));
        pausarReanudarEmpleado2.setFont(new
java.awt.Font("Segoe UI", 1, 12)); // NOI18N
        pausarReanudarEmpleado2.setText("Pausar/Reanudar
Empleado 2");
        pausarReanudarEmpleado2.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mouseClicked(java.awt.event.MouseEvent evt) {
                pausarReanudarEmpleado2MouseClicked(evt);
            }
        });
        pausarReanudarEmpleado2.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                pausarReanudarEmpleado2ActionPerformed(evt)
;
            }
        });

        pausarReanudarEmpleado3.setBackground(new
java.awt.Color(255, 153, 153));
        pausarReanudarEmpleado3.setFont(new
java.awt.Font("Segoe UI", 1, 12)); // NOI18N
        pausarReanudarEmpleado3.setText("Pausar/Reanudar
Empleado 3");
        pausarReanudarEmpleado3.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mouseClicked(java.awt.event.MouseEvent evt) {
                pausarReanudarEmpleado3MouseClicked(evt);
            }
        });

        pausarReanudarEmpleado4.setBackground(new
java.awt.Color(255, 153, 153));
        pausarReanudarEmpleado4.setFont(new
java.awt.Font("Segoe UI", 1, 12)); // NOI18N
        pausarReanudarEmpleado4.setText("Pausar/Reanudar
Empleado 4");
        pausarReanudarEmpleado4.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mouseClicked(java.awt.event.MouseEvent evt) {
```

```
                pausarReanudarEmpleado4MouseClicked(evt);
            }
        });

        pausarReanudarEmpleado5.setBackground(new
java.awt.Color(255, 153, 153));
        pausarReanudarEmpleado5.setFont(new
java.awt.Font("Segoe UI", 1, 12)); // NOI18N
        pausarReanudarEmpleado5.setText("Pausar/Reanudar
Empleado 5");
        pausarReanudarEmpleado5.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mouseClicked(java.awt.event.MouseEvent evt) {
                pausarReanudarEmpleado5MouseClicked(evt);
            }
        });

        jLabel10.setText("Empleado 2:");

        jTextPane1.setForeground(new java.awt.Color(51, 51,
51));
        jScrollPane1.setViewportView(jTextPane1);

        jScrollPane2.setViewportView(jTextPane2);

        jScrollPane3.setViewportView(jTextPane3);

        jScrollPane4.setViewportView(jTextPane4);

        jScrollPane5.setViewportView(jTextPane5);

        jScrollPane6.setViewportView(jTextPane6);

        jScrollPane7.setViewportView(jTextPane7);

        jScrollPane8.setViewportView(jTextPane8);

        jScrollPane9.setViewportView(jTextPane9);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(39, 39, 39)
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
```

```
                    .addGroup(layout.createSequentialGroup(
)
                        .addGroup(layout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(layout.createSequenti
alGroup()
                                .addGroup(layout.createPara
llelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jLabel9)
                                    .addComponent(jLabel8))
                                .addGap(18, 18, 18)
                                .addGroup(layout.createPara
llelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jScrollPa
ne8, javax.swing.GroupLayout.PREFERRED_SIZE, 721,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jScrollPa
ne9, javax.swing.GroupLayout.PREFERRED_SIZE, 721,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                            .addGroup(layout.createSequenti
alGroup()
                                .addGap(91, 91, 91)
                                .addGroup(layout.createPara
llelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                    .addGroup(layout.create
SequentialGroup()
                                        .addComponent(jLabe
l4)
                                        .addPreferredGap(ja
vax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(jScro
llPane7, javax.swing.GroupLayout.PREFERRED_SIZE, 713,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(javax.swing.G
roupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                        .addComponent(jLabe
l7)
                                        .addPreferredGap(ja
vax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(jScro
llPane6, javax.swing.GroupLayout.PREFERRED_SIZE, 712,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(javax.swing.G
roupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                        .addComponent(jLabe
l6)
```

```
                                        .addPreferredGap(ja
vax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(jScro
llPane5, javax.swing.GroupLayout.PREFERRED_SIZE, 712,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(javax.swing.G
roupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                        .addComponent(jLabe
l10)
                                        .addPreferredGap(ja
vax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(jScro
llPane4, javax.swing.GroupLayout.PREFERRED_SIZE, 712,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(javax.swing.G
roupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                        .addComponent(jLabe
l3)
                                        .addGap(41, 41, 41)
                                        .addComponent(jScro
llPane3, javax.swing.GroupLayout.PREFERRED_SIZE, 712,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                                .addGroup(layout.createPara
llelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addGroup(layout.create
SequentialGroup()
                                        .addGap(75, 75, 75)
                                        .addGroup(layout.cr
eateParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                                            .addComponent(p
ausarReanudarEmpleado1,
javax.swing.GroupLayout.PREFERRED_SIZE, 419,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(p
ausarReanudarEmpleado2,
javax.swing.GroupLayout.PREFERRED_SIZE, 419,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(p
ausarReanudarEmpleado3,
javax.swing.GroupLayout.PREFERRED_SIZE, 419,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(p
ausarReanudarEmpleado4,
javax.swing.GroupLayout.PREFERRED_SIZE, 419,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                                                .addGroup(layou
t.createSequentialGroup()
                                                    .addGap(23,
23, 23)
                                                    .addCompone
nt(jButton_PausarRenaudarTodo,
javax.swing.GroupLayout.PREFERRED_SIZE, 384,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                                            .addGroup(javax.swing.G
roupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                                .addPreferredGap(ja
vax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addComponent(pausa
rReanudarEmpleado5, javax.swing.GroupLayout.PREFERRED_SIZE,
419, javax.swing.GroupLayout.PREFERRED_SIZE))))
                                .addGroup(layout.createSequenti
alGroup()
                                    .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 979,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGap(163, 163, 163)
                                    .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                            .addContainerGap(322,
Short.MAX_VALUE))
                        .addGroup(layout.createSequentialGroup(
)
                            .addComponent(jLabel1)
                            .addPreferredGap(javax.swing.Layout
Style.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jLabel2)
                            .addGap(484, 484, 484))))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(57, 57, 57)
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jLabel2))
                .addGap(26, 26, 26)
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 201,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 37,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(79, 79, 79)
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel3)
                    .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup(
)
                        .addGap(15, 15, 15)
                        .addComponent(pausarReanudarEmplead
o1)))
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup(
)
                        .addGap(45, 45, 45)
                        .addGroup(layout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel10)
                            .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(layout.createSequentialGroup(
)
                        .addGap(60, 60, 60)
                        .addComponent(pausarReanudarEmplead
o2)))
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup(
)
                        .addGap(57, 57, 57)
                        .addGroup(layout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel6)
                            .addComponent(jScrollPane5,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(layout.createSequentialGroup(
)
                        .addGap(72, 72, 72)
                        .addComponent(pausarReanudarEmplead
o3)))
```

```
                .addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED, 43, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane6,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel7)
                    .addGroup(layout.createSequentialGroup(
)
                        .addGap(15, 15, 15)
                        .addComponent(pausarReanudarEmplead
o4)))
                .addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED, 58, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane7,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup(
)
                        .addGap(15, 15, 15)
                        .addComponent(pausarReanudarEmplead
o5)))
                .addGap(85, 85, 85)
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel8)
                    .addComponent(jScrollPane8,
javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(8, 8, 8)
                .addComponent(jButton_PausarRenaudarTodo,
javax.swing.GroupLayout.PREFERRED_SIZE, 49,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane9,
javax.swing.GroupLayout.PREFERRED_SIZE, 66,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel9))
                .addGap(293, 293, 293))
        );
```

```
        pack();
    }// </editor-fold>


    private void
jButton_PausarRenaudarTodoActionPerformed(java.awt.event.Ac
tionEvent evt)
{

        // TODO add your handling code here:
    }


    private void
jButton_PausarRenaudarTodoMouseClicked(java.awt.event.Mouse
Event evt)
{

        if (!parado) { //Si no lo ha pulsado entonces se
para.
            parado = true;
            reanudado = false;
            pausaGlobal.cerrar();
            //************************************

            pausaEmpleado1.cerrar();

            pausarReanudarEmpleado1.setText("VOLVER A
FUNCIONAR");
            jTextPane3.setText("PAUSADO");
            pausarReanudarEmpleado1.setBackground(Color.gre
en);
            pausarReanudarEmpleado1.setForeground(Color.bla
ck);
            //************************************

            pausaEmpleado2.cerrar();
            pausarReanudarEmpleado2.setText("VOLVER A
FUNCIONAR");
            jTextPane4.setText("PAUSADO");
            pausarReanudarEmpleado2.setBackground(Color.gre
en);
            pausarReanudarEmpleado2.setForeground(Color.bla
ck);
            //************************************

            pausaEmpleado3.cerrar();
            pausarReanudarEmpleado3.setText("VOLVER A
FUNCIONAR");
            jTextPane5.setText("PAUSADO");
```

```
            pausarReanudarEmpleado3.setBackground(Color.gre
en);
            pausarReanudarEmpleado3.setForeground(Color.bla
ck);
            //************************************

            pausaEmpleado4.cerrar();
            pausarReanudarEmpleado4.setText("VOLVER A
FUNCIONAR");
            jTextPane6.setText("PAUSADO");
            pausarReanudarEmpleado4.setBackground(Color.gre
en);
            pausarReanudarEmpleado4.setForeground(Color.bla
ck);

            //************************************
            pausaEmpleado5.cerrar();
            pausarReanudarEmpleado5.setText("VOLVER A
FUNCIONAR");
            jTextPane7.setText("PAUSADO");
            pausarReanudarEmpleado5.setBackground(Color.gre
en);
            pausarReanudarEmpleado5.setForeground(Color.bla
ck);

        } else { //Pero si ya lo ha pusaldo entocnes se
reanuda.
            reanudado = true;
            parado = false;
            pausaGlobal.abrir();

            //************************************
            pausaEmpleado1.abrir();
            pausarReanudarEmpleado1.setText("PARAR A
DESCANSAR");
            pausarReanudarEmpleado1.setBackground(Color.red
);
            pausarReanudarEmpleado1.setForeground(Color.whi
te);

            //************************************
            pausaEmpleado2.abrir();
            pausarReanudarEmpleado2.setText("PARAR A
DESCANSAR");
            pausarReanudarEmpleado2.setBackground(Color.red
);
            pausarReanudarEmpleado2.setForeground(Color.whi
te);
            //************************************

            pausaEmpleado3.abrir();
```

```
            pausarReanudarEmpleado3.setText("PARAR A
DESCANSAR");
            pausarReanudarEmpleado3.setBackground(Color.red
);
            pausarReanudarEmpleado3.setForeground(Color.whi
te);
            //************************************
            pausaEmpleado4.abrir();
            pausarReanudarEmpleado4.setText("PARAR A
DESCANSAR");
            pausarReanudarEmpleado4.setBackground(Color.red
);
            pausarReanudarEmpleado4.setForeground(Color.whi
te);
            //************************************

            pausaEmpleado5.abrir();
            pausarReanudarEmpleado5.setText("PARAR A
DESCANSAR");
            pausarReanudarEmpleado5.setBackground(Color.red
);
            pausarReanudarEmpleado5.setForeground(Color.whi
te);
        }
    }


    private void
pausarReanudarEmpleado1MouseClicked(java.awt.event.MouseEve
nt evt)
{

        if (!parado) { //Si no lo ha pulsado entonces se
para.
            parado = true;
            reanudado = false;
            pausaEmpleado1.cerrar();
            pausarReanudarEmpleado1.setText("VOLVER A
FUNCIONAR");
            jTextPane3.setText("PAUSADO");
            pausarReanudarEmpleado1.setBackground(Color.gre
en);
            pausarReanudarEmpleado1.setForeground(Color.bla
ck);

        } else { //Pero si ya lo ha pusaldo entonces se
reanuda.
            reanudado = true;
            parado = false;
            pausaEmpleado1.abrir();
```

```
            pausarReanudarEmpleado1.setText("PARAR A
DESCANSAR");
            pausarReanudarEmpleado1.setBackground(Color.red
);
            pausarReanudarEmpleado1.setForeground(Color.whi
te);

        }
    }

    private void
pausarReanudarEmpleado2MouseClicked(java.awt.event.MouseEve
nt evt)
{
        if (!parado) { //Si no lo ha pulsado entonces se
para.
            parado = true;
            reanudado = false;
            pausaEmpleado2.cerrar();
            pausarReanudarEmpleado2.setText("VOLVER A
FUNCIONAR");
            jTextPane4.setText("PAUSADO");
            pausarReanudarEmpleado2.setBackground(Color.gre
en);
            pausarReanudarEmpleado2.setForeground(Color.bla
ck);
        } else { //Pero si ya lo ha pusaldo entonces se
reanuda.
            reanudado = true;
            parado = false;
            pausaEmpleado2.abrir();
            pausarReanudarEmpleado2.setText("PARAR A
DESCANSAR");
            pausarReanudarEmpleado2.setBackground(Color.red
);
            pausarReanudarEmpleado2.setForeground(Color.whi
te);

        }
    }

    private void
pausarReanudarEmpleado3MouseClicked(java.awt.event.MouseEve
nt evt)
{

        if (!parado) { //Si no lo ha pulsado entonces se
para.
            parado = true;
            reanudado = false;
            pausaEmpleado3.cerrar();
```

```
            pausarReanudarEmpleado3.setText("VOLVER A
FUNCIONAR");
            jTextPane5.setText("PAUSADO");
            pausarReanudarEmpleado3.setBackground(Color.gre
en);
            pausarReanudarEmpleado3.setForeground(Color.bla
ck);

        } else { //Pero si ya lo ha pusaldo entonces se
reanuda.
            reanudado = true;
            parado = false;
            pausaEmpleado3.abrir();
            pausarReanudarEmpleado3.setText("PARAR A
DESCANSAR");
            pausarReanudarEmpleado3.setBackground(Color.red
);
            pausarReanudarEmpleado3.setForeground(Color.whi
te);

        }
    }

    private void
pausarReanudarEmpleado4MouseClicked(java.awt.event.MouseEve
nt evt)
{

        if (!parado) { //Si no lo ha pulsado entonces se
para.
            parado = true;
            reanudado = false;
            pausaEmpleado4.cerrar();
            pausarReanudarEmpleado4.setText("VOLVER A
FUNCIONAR");
            jTextPane6.setText("PAUSADO");
            pausarReanudarEmpleado4.setBackground(Color.gre
en);
            pausarReanudarEmpleado4.setForeground(Color.bla
ck);

        } else { //Pero si ya lo ha pusaldo entonces se
reanuda.
            reanudado = true;
            parado = false;
            pausaEmpleado4.abrir();
            pausarReanudarEmpleado4.setText("PARAR A
DESCANSAR");
            pausarReanudarEmpleado4.setBackground(Color.red
);
```

```
            pausarReanudarEmpleado4.setForeground(Color.whi
te);
        }
    }

    private void
pausarReanudarEmpleado5MouseClicked(java.awt.event.MouseEve
nt evt)
{

        if (!parado) { //Si no lo ha pulsado entonces se
para.
            parado = true;
            reanudado = false;
            pausaEmpleado5.cerrar();
            pausarReanudarEmpleado5.setText("VOLVER A
FUNCIONAR");
            jTextPane7.setText("PAUSADO");
            pausarReanudarEmpleado5.setBackground(Color.gre
en);
            pausarReanudarEmpleado5.setForeground(Color.bla
ck);

        } else { //Pero si ya lo ha pusaldo entonces se
reanuda.
            reanudado = true;
            parado = false;
            pausaEmpleado5.abrir();
            pausarReanudarEmpleado5.setText("PARAR A
DESCANSAR");
            pausarReanudarEmpleado5.setBackground(Color.red
);
            pausarReanudarEmpleado5.setForeground(Color.whi
te);

        }
    }

    private void
pausarReanudarEmpleado1ActionPerformed(java.awt.event.Actio
nEvent evt)
{
        // TODO add your handling code here:
    }


    private void
pausarReanudarEmpleado2ActionPerformed(java.awt.event.Actio
nEvent evt)
{
        // TODO add your handling code here:
```

```
    }


    public String buzonTexto() {

        String textoBuzon = jTextPane2.getText();

        return textoBuzon;
    }

    public String cantidadBuzon() {

        String cantidadBuzon = jTextPane1.getText();

        return cantidadBuzon;
    }

    public String F1Texto() {

        String furgoneta1 = jTextPane8.getText();

        return furgoneta1;
    }

    public String F2Texto() {

        String furgoneta2 = jTextPane9.getText();

        return furgoneta2;
    }

    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/lookandf
eel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info
: javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(in
fo.getClassName());
                    break;
                }
            }
```

```
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Interfaz.cla
ss.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Interfaz.cla
ss.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Interfaz.cla
ss.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch
(javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Interfaz.cla
ss.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Interfaz().setVisible(true);
                try {
                    Interfaz interfaz = new Interfaz();
                    ObjetoRemoto obj = new
ObjetoRemoto(interfaz);
                    Registry registry =
LocateRegistry.createRegistry(1099);
                    Naming.rebind("/localhost/PECLIvanWalte
r", obj);
                } catch (IOException ex) {
                }
            }
        });

    }


    // Variables declaration - do not
modify
    private javax.swing.JButton jButton_PausarRenaudarTodo;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
```

```
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JScrollPane jScrollPane5;
    private javax.swing.JScrollPane jScrollPane6;
    private javax.swing.JScrollPane jScrollPane7;
    private javax.swing.JScrollPane jScrollPane8;
    private javax.swing.JScrollPane jScrollPane9;
    private javax.swing.JTextPane jTextPane1;
    private javax.swing.JTextPane jTextPane2;
    private javax.swing.JTextPane jTextPane3;
    private javax.swing.JTextPane jTextPane4;
    private javax.swing.JTextPane jTextPane5;
    private javax.swing.JTextPane jTextPane6;
    private javax.swing.JTextPane jTextPane7;
    private javax.swing.JTextPane jTextPane8;
    private javax.swing.JTextPane jTextPane9;
    private javax.swing.JButton pausarReanudarEmpleado1;
    private javax.swing.JButton pausarReanudarEmpleado2;
    private javax.swing.JButton pausarReanudarEmpleado3;
    private javax.swing.JButton pausarReanudarEmpleado4;
    private javax.swing.JButton pausarReanudarEmpleado5;
    // End of variables declaration
}
```

CLASE PAUSAR

```
package Principal;

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

public class Pausar {

    private boolean cerrado = false;
    private Lock cerrojo = new ReentrantLock();
    private Condition parar = cerrojo.newCondition();

    public void mirar() {
        cerrojo.lock();
        try {
            while (cerrado) {
                try {
                    parar.await();
                } catch (InterruptedException ie) {
                }
            }
```

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

    /**
     * Método para abrir el cerrojo y avisar a los objetos
que estén parados
     */
    public void abrir() {
        cerrojo.lock();
        try {
            cerrado = false;
            parar.signalAll();
        } finally {
            cerrojo.unlock();
        }
    }

    /**
     * Método para cerrar el cerrojo
     */
    public void cerrar() {
        cerrojo.lock();
        try {
            cerrado = true;
        } finally {
            cerrojo.unlock();
        }
    }

}
```

CLASE PERSONA

```
package Principal;

public class Persona extends Thread {

    private Buzon buzon;
    private String identificador;
    private String carta1;
    private String carta2;
    private Registro log;
    private Pausar pausa;

    public Persona(Buzon buzon, String identificador,
Registro log, Pausar pausa) {
        this.buzon = buzon;
        this.identificador = identificador;
        this.log = log;
```

```
        this.pausa = pausa;

    }

    public void run() {
        try {

            int tiempo_aleatorio = (int) (Math.random() *
(800 - 400 + 1) + 400);
            carta1 = identificador + "-C1";
            carta2 = identificador + "-C2";
            char caracter1 = carta1.charAt(carta1.length()
- 1);
            char caracter2 = carta2.charAt(carta2.length()
- 1);

            if (caracter1 == '1') {
                pausa.mirar();
                buzon.dejarCarta(carta1);
                String mensajeC1 = "Se deja la carta: " +
carta1 + " en el buzón";
                pausa.mirar();
                log.registroLog(mensajeC1);
                sleep(tiempo_aleatorio);
            }
            if (caracter2 == '2') {
                pausa.mirar();
                buzon.dejarCarta(carta2);
                String mensajeC2 = "Se deja la carta: " +
carta2 + " en el buzón";
                pausa.mirar();
                log.registroLog(mensajeC2);
                sleep(tiempo_aleatorio);
            }

        } catch (Exception e) {
        }

    }

}
```

CLASE REGISTRO

```
package Principal;

import java.io.File;
import java.io.IOException;
import java.util.logging.FileHandler;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;
```

```
/**
 *
 * @author isr10
 */
public class Registro {

    private Logger logger;
    FileHandler archivoManipulable;

    public Registro() {
        try {
            logger = Logger.getLogger("evolucionCartas");
            archivoManipulable = new
FileHandler("C:\\Users\\isr10\\Documents\\NetBeansProjects\
\PECL1_Ivan_Walter\\src\\main\\java\\Log\\evolucionCartas.t
xt");
            logger.addHandler(archivoManipulable);
            SimpleFormatter formato = new
SimpleFormatter();
            archivoManipulable.setFormatter(formato);

        } catch (IOException ex) {

            logger.warning("ERROR:" + ex.getMessage());
        }
    }

    public void registroLog(String mensaje) {
        //TODO NOS FUNCIONA. LO MALO, NOS CREA 1 ARCHIVO
POR 1 CARTA CREADA.
        logger.info(mensaje);

    }

}
```

PAQUETE DISTRIBUIDA

INTERFACE DISTRIBUIDA

```
package Distribuida;

import java.rmi.Remote;
import java.rmi.RemoteException;


public interface Distribuida extends Remote {
```

```
    String buzonTexto() throws RemoteException,
InterruptedException;

    String cantidadBuzon() throws RemoteException,
InterruptedException;

    String F1Texto() throws RemoteException,
InterruptedException;

    String F2Texto() throws RemoteException,
InterruptedException;

}
```

CLASE INTERFAZDISTRIBUIDA

```
package Distribuida;

import Principal.Registro;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.util.logging.Logger;
import javax.swing.JTextPane;

public class InterfazDistribuida extends javax.swing.JFrame
{

    private static Registro registro;
    private static Logger logger;
    private Distribuida obj;

    public InterfazDistribuida() throws
InterruptedException, RemoteException, NotBoundException,
MalformedURLException {
        initComponents();

        obj = (Distribuida)
Naming.lookup("/localhost/PECLIvanWalter");

    }

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

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
```

```
        jLabel4 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextPane1 = new javax.swing.JTextPane();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTextPane2 = new javax.swing.JTextPane();
        jScrollPane3 = new javax.swing.JScrollPane();
        jTextPane3 = new javax.swing.JTextPane();
        jScrollPane4 = new javax.swing.JScrollPane();
        jTextPane4 = new javax.swing.JTextPane();

        setDefaultCloseOperation(javax.swing.WindowConstant
s.EXIT_ON_CLOSE);

        jLabel1.setText("Contenido del buzón:");

        jLabel2.setText("Número de cartas en el buzón:");

        jLabel3.setText("Contenido de la furgoneta 1:");

        jLabel4.setText("Contenido de la Furgoneta 2:");

        jScrollPane1.setViewportView(jTextPane1);

        jScrollPane2.setViewportView(jTextPane2);

        jScrollPane3.setViewportView(jTextPane3);

        jScrollPane4.setViewportView(jTextPane4);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(56, 56, 56)
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup(
)
                        .addComponent(jLabel1)
                        .addPreferredGap(javax.swing.Layout
Style.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jLabel2)
                        .addGap(30, 30, 30)
                        .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 43,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                    .addGroup(layout.createSequentialGroup(
)
                        .addGroup(layout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 634,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGroup(layout.createSequenti
alGroup()
                                .addGroup(layout.createPara
llelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jLabel3)
                                    .addComponent(jLabel4))
                                .addGap(50, 50, 50)
                                .addGroup(layout.createPara
llelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                    .addComponent(jScrollPa
ne3, javax.swing.GroupLayout.DEFAULT_SIZE, 547,
Short.MAX_VALUE)
                                    .addComponent(jScrollPa
ne4))))
                        .addGap(214, 214, 214)))
                .addGap(0, 156, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup(
)
                        .addGap(127, 127, 127)
                        .addComponent(jLabel1)
                        .addGap(21, 21, 21))
                    .addGroup(javax.swing.GroupLayout.Align
ment.TRAILING, layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(layout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 33,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel2))
                        .addGap(9, 9, 9)))
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup(
)
```

```
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 200,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(76, 76, 76)
                        .addComponent(jLabel3))
                    .addGroup(javax.swing.GroupLayout.Align
ment.TRAILING, layout.createSequentialGroup()
                        .addGap(268, 268, 268)
                        .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGroup(layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup(
)
                        .addGap(118, 118, 118)
                        .addComponent(jLabel4))
                    .addGroup(javax.swing.GroupLayout.Align
ment.TRAILING, layout.createSequentialGroup()
                        .addGap(110, 110, 110)
                        .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(59, Short.MAX_VALUE))
        );

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

    public void introducirTexto() throws RemoteException,
InterruptedException {

        jTextPane1.setText(obj.buzonTexto());
        jTextPane2.setText(obj.cantidadBuzon());
        jTextPane3.setText(obj.F1Texto());
        jTextPane4.setText(obj.F2Texto());

    }

    public static void main(String args[]) throws
InterruptedException, RemoteException, NotBoundException,
MalformedURLException {

        InterfazDistribuida interfaz = new
InterfazDistribuida();
        interfaz.setVisible(true);
        boolean siempre = true;
        while (siempre) {

            interfaz.introducirTexto();
```

```
        }
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info
: javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(in
fo.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(InterfazDist
ribuida.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(InterfazDist
ribuida.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(InterfazDist
ribuida.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);
        } catch
(javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(InterfazDist
ribuida.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    InterfazDistribuida interfaz = new
InterfazDistribuida();
                    new
InterfazDistribuida().setVisible(true);

                    interfaz.introducirTexto();
                } catch (InterruptedException ie) {
                    logger.warning(ie.getMessage());
                    String error = ie.getMessage();
                    registro.registroLog(error);
                } catch (RemoteException re) {
                    logger.warning(re.getMessage());
                    String error = re.getMessage();
                    registro.registroLog(error);
```

```
                } catch (NotBoundException ne) {
                    logger.warning(ne.getMessage());
                    String error = ne.getMessage();
                    registro.registroLog(error);
                } catch (MalformedURLException me) {
                    logger.warning(me.getMessage());
                    String error = me.getMessage();
                    registro.registroLog(error);
                }
            }
        });
    }

    // Variables declaration - do not
modify
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JTextPane jTextPane1;
    private javax.swing.JTextPane jTextPane2;
    private javax.swing.JTextPane jTextPane3;
    private javax.swing.JTextPane jTextPane4;
    // End of variables declaration
}
```

CLASE OBJETOREMOTO

```
package Distribuida;

import Principal.Interfaz;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class ObjetoRemoto extends UnicastRemoteObject
implements Distribuida {

    private Interfaz interfaz;

    public ObjetoRemoto(Interfaz interfaz) throws
RemoteException {

        this.interfaz = interfaz;
    }

    public String buzonTexto() throws RemoteException,
InterruptedException {
```

```
        String texto = interfaz.buzonTexto();

        return texto;
    }

    public String cantidadBuzon() throws RemoteException,
InterruptedException {
        String texto = interfaz.cantidadBuzon();

        return texto;
    }

    public String F1Texto() throws RemoteException,
InterruptedException {
        String texto = interfaz.F1Texto();

        return texto;
    }

    public String F2Texto() throws RemoteException,
InterruptedException {
        String texto = interfaz.F2Texto();

        return texto;
```