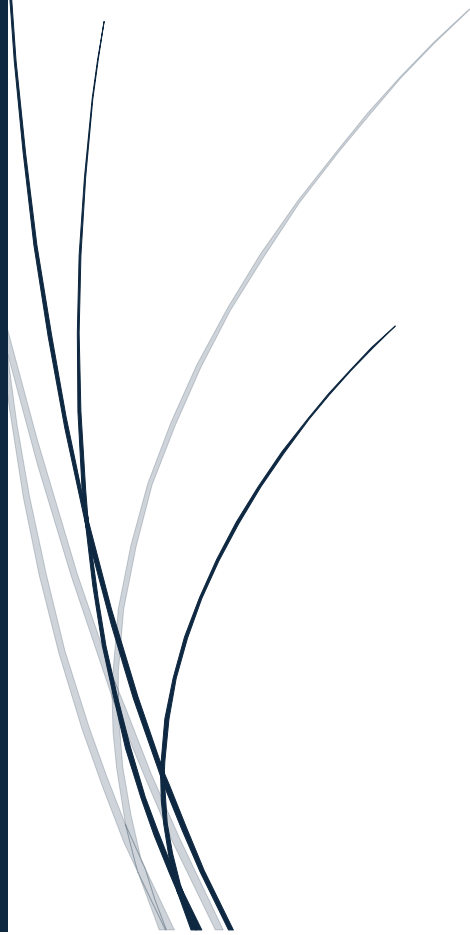




6-5-2024

# PECL1 Programación Avanzada

Grado en Ingeniería Informática  
Curso 2023/2024-Convocatoria  
Ordinaria



## Tabla de contenido

<b>1.- Análisis de alto nivel:</b>	2
<b>Requisitos Funcionales:</b>	2
• Modelado de Aeropuertos y Aerovías:	2
• Vehículos y Personas:	2
• Funcionamiento de los Autobuses:	2
• Funcionamiento de los Aviones:	2
<b>Funcionalidades específicas:</b>	2
<b>2.- Diseño general del sistema:</b>	3
<b>3.- Clases principales:</b>	4
<b>Clase Aeropuerto:</b>	4
<b>Clase GenHilloBus y GenHiloAvion:</b>	10
<b>Clase Avión:</b>	11
<b>Clase Autobus:</b>	12
<b>Clase Cliente:</b>	13
<b>Clase Servidor:</b>	14
<b>Interfaz UIUpdate:</b>	15
<b>Clase Pausa:</b>	15
<b>4.- Diagrama de clases:</b>	17
<b>5.- Código Fuente:</b>	20

## 1.- Análisis de alto nivel:

La práctica busca simular el funcionamiento de los aeropuertos de Madrid y Barcelona, utilizando un enfoque de programación concurrente y distribuida.

### Requisitos Funcionales:

- Modelado de Aeropuertos y Aerovías:
  - Dos aeropuertos: Madrid y Barcelona.
  - Cada aeropuerto cuenta con un hangar (con capacidad infinita de aviones), un taller (con capacidad para 20 aviones) con una puerta estrecha por la que solo puede pasar un avión al mismo tiempo, cuatro puertas de embarque/ desembarque, una puerta de embarque y otra de desembarque (con capacidad para 1 avión cada una), cuatro pistas de despegue/aterrizaje (con capacidad para 1 avión cada una), un área de estacionamiento (con capacidad infinita de aviones), y un área de rodaje (con capacidad infinita de aviones).
  - Dos aerovías que conectan ambos aeropuertos, cada una en un sentido con capacidad infinita de aviones.
  - En la parte dos, se debe poder abrir o cerrar las pistas de despegue/aterrizaje de un aeropuerto.
- Vehículos y Personas:
  - Aviones y autobuses son modelados como hilos.
  - Personas no son modeladas como hilos.
  - 8,000 aviones y 4,000 autobuses generados, con creación escalonada para evitar congestión.
- Funcionamiento de los Autobuses:
  - Ciclo repetitivo: transporte de pasajeros entre ciudad y aeropuerto, y viceversa.
  - Cada autobús tiene una etiqueta del tipo "B-XXXX" donde "XXXX" es un número agenciado de manera secuencial. Si ese número es impar está ubicado en Barcelona y si es Par, está ubicado en Madrid.
  - Capacidad máxima del autobús de 50 pasajeros.
  - Tiempos aleatorios para la espera y el traslado de pasajeros.
- Funcionamiento de los Aviones:
  - Ciclo repetitivo: traslado de pasajeros entre los dos aeropuertos.
  - Uso de puertas de embarque y pistas de aterrizaje/despegue, con tiempos aleatorios para procesos.
  - Cada avión se identificará como "YY-XXXX", donde XXXX es un número agenciado de manera secuencial, e YY son 2 caracteres alfabéticos al azar. Si ese número es impar está ubicado en Barcelona y si es Par, está ubicado en Madrid
  - Capacidad máxima de entre 100 y 300 pasajeros.
  - Mantenimiento periódico en el taller después de cada vuelo que dura entre 1-5 segundos. Si hace 15 vuelos, este mantenimiento es mas largo (5-10 segundos).

### Funcionalidades específicas:

- El sistema ha de tener una interfaz gráfica fácil de usar e intuitiva tanto para el servidor como para el cliente.
- El sistema ha de habilitar el uso de consultas remotas sobre el estado actual del sistema usando técnicas de RMI o sockets
- Todo el comportamiento del sistema se registra en un archivo de log.
- El servidor ha de ser capaz de pausar/reanudar el sistema.

- Utilización de técnicas de sincronización para manejar la concurrencia y evitar condiciones de carrera, especialmente en áreas críticas como el acceso a pistas y puertas de embarque
- El sistema debe ser realizado en lenguaje Java, concretamente montado sobre netbeans.

## 2.- Diseño general del sistema:

La práctica consta de dos paquetes: uno para la interfaz gráfica y otro para la lógica del proyecto. La parte gráfica contiene una primera interfaz para el lado servidor y otra para el lado cliente. Estas se forman de un JFrame cada una, y a su vez incluyen un JPanel asociado a cada aeropuerto con la información relevante. Ambos programas (cliente y servidor) se lanzan desde la interfaz gráfica.

En la interfaz de la “pantallaPrincipal” (Servidor) se inicializan los aeropuertos y aerovías (listas comunes de aviones entre ambos aeropuertos), así como todos los objetos necesarios para el funcionamiento del proyecto. Para la creación simultánea de autobuses y aviones, hemos creado una clase (hilo) que genera y comienza la ejecución de los hilos de autobuses y otra clase para los hilos de aviones. Estos hilos generadores también se lanzan desde la interfaz gráfica.

En cuanto a los aviones y autobuses, estos tienen como recursos compartidos los aeropuertos y harán llamadas a sus métodos, de tal forma que serán necesarios mecanismos de comunicación y sincronización de hilos para evitar condiciones de carrera y producir un correcto funcionamiento del programa. Esto se detallará en profundidad más adelante.

También hemos creado una clase Pausa cuya instancia es compartida por todos los hilos. Esto nos permitirá pausar y reanudar la ejecución del programa en cualquier instante desde la interfaz gráfica del lado servidor. Para implementar los métodos incluidos en esta clase, hemos empleado monitores por su sencillez para implementación. De esta forma, sus métodos están controlados mediante la cláusula synchronized y los métodos wait() (pausar) y notifyAll() (reanudar).

Adicionalmente, para la segunda parte del proyecto empleamos RMI, permitiendo el acceso remoto al módulo implementado en la primera parte. De esta forma, en el programa principal se declaran los aeropuertos y los registramos como objetos compartidos, mientras que en el lado cliente se buscan los objetos remotos y se hace uso de ellos.

En la interfaz del cliente se inician los componentes gráficos, se buscan los objetos remotos (ambos aeropuertos) y se inicia el hilo que ejecuta el ciclo del programa cliente. Es decir, hay una clase “ClienteHilo” que a través de objetos remotos (aeropuertos), pasados como parámetros, hace llamadas a sus métodos (declarados en una interfaz remota) y mostrará en la interfaz gráfica los datos obtenidos. Además, incluye la funcionalidad de cerrar y abrir las pistas de los aeropuertos también mediante métodos remotos.

Por otro lado, para evitar condiciones de carrera al acceder a variables compartidas (de la clase aeropuerto) por varios hilos hemos optado por emplear cerrojos (locks) o incluso semáforos binarios, que permite el acceso en exclusión mutua a un recurso. Ha resultado útil para arrays de aviones como la puerta del hangar, puerta del área de rodaje, puerta del área de estacionamiento, aerovías, puerta del taller o modificar las puertas de embarque/desembarque. Para controlar el número de aviones dentro de una zona empleamos semáforos de n permisos. Se ha utilizado para determinar seis puertas de embarque/desembarque, limitar un máximo de cinco puertas de embarque y un máximo de cinco para desembarque, restringir el aforo del taller a veinte aviones y el número de pistas a cuatro.

Adicionalmente, queríamos tener control sobre las pistas de despegue/aterrizaje abiertas/cerradas, por lo que creamos un array de booleanos que indica el estado de las pistas. Su acceso está restringido mediante un lock y un condition que bloquea el hilo si todas las pistas están cerradas para evitar la espera activa.

Por último, hemos creado una interfaz (UIUpdater) con los métodos necesarios para actualizar los campos de la interfaz gráfica del programa servidor. Haciendo llamadas a los métodos de esta interfaz después de modificar unas pistas, puertas de embarque o por ejemplo el hangar, se consigue mantener actualizada la interfaz gráfica del servidor ante cualquier cambio del sistema.

### 3.- Clases principales:

#### Clase Aeropuerto:

Esta clase será el objeto compartido por los hilos y donde se ha implementado la gran mayoría de la lógica del programa. Aquí se incluye como atributos arrays de aviones que simulan el hangar, área de rodaje, área de estacionamiento, pistas, puertas de embarque/desembarque, así como un array de booleanos para simular las pistas abiertas y cerradas. También se incluye como atributos los cerrojos, conditions, semáforos mencionados anteriormente. Además, hay variables enteras que llevan la cuenta de los aviones que hay en cada parte, pues serán necesarios para la interfaz del lado cliente.

Por otro lado, para simular las aerovías, hemos creado un array de aviones de salida y otro de entrada al aeropuerto. Estos arrays se pasan como parámetros al constructor de cada aeropuerto. La aerovía de salida de un aeropuerto se corresponde con la de llegada del otro aeropuerto y viceversa. También se le pasa por parámetro el cerrojo que controla el acceso en exclusión mutua a las aerovías, la instancia de la clase pausa común a todos los hilos para poder reanudar y pausar el programa y la pantalla de la interfaz gráfica para actualizar los campos de texto en base a los valores del aeropuerto. En la siguiente foto se muestra parte de lo comentado, para verlo completo consultar anexo de código fuente:

```
public class Aeropuerto extends UnicastRemoteObject implements InterfazRemota, Serializable{
    //atributos
    private int pasajeros, nAvHangar, nAvTaller, nAvEstacionamiento, nAvRodaje;
    private ArrayList<Avion> hangar, areaDeRodaje, areaDeEstacionamiento, aeroviaSalida, aeroviaLlegada, taller;
    private Lock hangarLock, pasajerosLock, pistasAbiertasLock, areaDeRodajeLock, areaDeEstacionamientoLock, aeroViasLock;
    private Semaphore pistasSem, pistasSemBin, tallerSe, puertaTaller, control6Puertas, control5Embarque, control5Desembarque, arrayEmbarque;
    private String nombre;
    private panelAeropuerto uiUpdater;
    private pantallaPrincipal pP;
    private Avion[] puertasDeEmbarque, pistasDeDespeque;
    private Pausa pausa;
    private boolean[] pistasAbiertas = {true, true, true, true};
    private Condition pistasCerradas;
    //constructor
    public Aeropuerto(ArrayList<Avion> aeroviaSalida, ArrayList<Avion> aeroviaLlegada, String nombre, Lock aeroViasLock, panelAeropuerto u,
        pantallaPrincipal p, Pausa pausa) throws RemoteException {

        this.aeroviaSalida = aeroviaSalida;
        this.aeroviaLlegada = aeroviaLlegada;
        this.hangar = new ArrayList<>();
        this.pasajeros = 0;
        this.nAvHangar = 0;
        this.nAvTaller = 0;
        this.nAvEstacionamiento = 0;
        this.nAvRodaje = 0;
        this.areaDeEstacionamiento = new ArrayList<>();
        this.areaDeRodaje = new ArrayList<>();
        this.pistasSemBin = new Semaphore(1, true);
        this.tallerSe = new Semaphore(20, true);
        this.puertaTaller = new Semaphore(1);
```

A continuación, se describirán los diferentes métodos implementados en la clase aeropuerto y empleados por la clase autobus:

**lleganPasajerosDeAutobus():** Este método descarga los pasajeros que viajan en el autobús para sumarlos al sistema del aeropuerto. Primero comprueba si el hilo debe pausar su ejecución o no mediante pausa.mirar(); y después intenta coger el cerrojo para modificar la variable pasajeros. Una vez modificados los pasajeros del aeropuerto y reseteados los del autobús, se actualiza ese valor en la interfaz gráfica del servidor y se libera el cerrojo.

```

public void lleganPasajerosDeAutobus (Autobus autobus) {
    pausa.mirar();
    try {
        pasajerosLock.lock();
        this.pasajeros += autobus.getPasajeros();
        uiUpdater.updatePasajeros(pasajeros);
        uiUpdater.updatebus1(autobus);
        autobus.setPasajeros(0);
    }
    finally{
        pasajerosLock.unlock();
    }
}

```

```

public void subenPasajerosAlBus (Autobus autobus) {
    pausa.mirar();
    if (pasajeros > 0) {
        try {
            pasajerosLock.lock();
            Random rand = new Random();
            int pasajerosQueSalen;

            if (pasajeros < 50){
                pasajerosQueSalen = rand.nextInt(pasajeros);
            }
            else{
                pasajerosQueSalen = rand.nextInt(50);
            }

            this.pasajeros -= pasajerosQueSalen;
            uiUpdater.updatePasajeros(pasajeros);
            autobus.setPasajeros(pasajerosQueSalen);
            uiUpdater.updatebus2(autobus);
        }
        finally{
            pasajerosLock.unlock();
        }
    }
}

```

**subenPasajerosAlBus():** este método monta pasajeros del aeropuerto al autobús. Siempre mira si debe pausar su ejecución el hilo y coge el cerrojo para modificar los pasajeros del aeropuerto. Sube una cantidad aleatoria de pasajeros que estén en el aeropuerto, actualiza la interfaz del servidor y libera el cerrojo.

A continuación, se describirán los métodos de la clase aeropuerto empleados por los aviones:

**entrarAlHangar():** este método permite al avión entrar al hangar. Para ello, coge el cerrojo que controla el array de aviones en el hangar, añade el avión, aumenta en uno el su contador, actualiza la interfaz y libera de nuevo el cerrojo.

**SaleDelHangar():** es igual que el anterior, pero para salir del hangar. Además, comprueba si se debe pausar el hilo o no mediante pausa.mirar(). Esto solo se comprueba en uno de los métodos (entrar o salir), pues si se hace en ambos puede dar inconsistencias y desapariciones de los aviones, pues si un avión sale de una zona y se para su ejecución antes de entrar en la siguiente, el avión queda en tierra de nadie y no se muestra en el sistema. Esto se aplica al resto de métodos del aeropuerto.

```

public void entrarAlHangar(Avion avion) {
    //pausa.mirar();
    try {
        hangarLock.lock();
        if (!hangar.contains(avion)) {
            hangar.add(avion);
            nAvHangar++;
            uiUpdater.updateHangar(hangar);
            System.out.println("El avion " + avion.getIdAv()+
        }
    }
    finally {
        hangarLock.unlock();
    }
}

```

```

public void SaleDelHangar(Avion avion) {
    pausa.mirar();
    try {
        hangarLock.lock();
        if (hangar.contains(avion)) {
            hangar.remove(avion);
            nAvHangar--;
            uiUpdater.updateHangar(hangar);
            System.out.println("El avion " + avion.getIdAv()+
        }
    }
    finally {
        hangarLock.unlock();
    }
}

```

**esperarAPuertaDeEmbarque():** permite acceder a la primera puerta de embarque libre que hay. Primero comprobamos si el hilo debe parar su ejecución. Después limita el número de puertas de embarque a cinco mediante un semáforo de cinco permisos. El siguiente semáforo limita el número de puertas de embarque/desembarque y por último un semáforo binario controla el acceso en exclusión mutua a las puertas de embarque/desembarque (array de aviones). Por último, libera el semáforo binario del array de puertas de embarque (arrayEmbarque). El resto de los semáforos se liberan cuando se dejen libres las puertas de embarque con otro método.

**esperarAPuertaDeDesembarque():** similar al anterior pero cambia el semáforo de cinco puertas de embarque (“control5Embarque”) por el de cinco puertas de desembarque (“control5Desembarque”). Así se controla que como mucho hay cinco puertas desembarcando.

```
public void esperarAPuertaDeEmbarque (Avion a) {
    pausa.mirar();
    try{
        control5Embarque.acquire(); //controla maximo hay 5 puertas de embarque
        control6Puertas.acquire(); // controla maximo 6 puertas de ambos tipos
        arrayEmbarque.acquire();//controla array de aviones en puertas de embarque/desembarque
        salirDelAreaDeEstacionamiento (a);
        System.out.println("El avion "+ a.getIdAv()+ " ha salido del area de estacionamiento de");
        int pos = uiUpdater.primerHuecoEnArray(puertasDeEmbarque);
        puertasDeEmbarque[pos] = a;
        uiUpdater.updatePuertasDeEmbarque(a,puertasDeEmbarque);
        System.out.println("El avion "+ a.getIdAv()+ " ha entrado al gate de embarque de "+ thi
    }
    catch(Exception E) {
        E.printStackTrace();
    }
    finally{
        arrayEmbarque.release();
    }
}
```

**salirDePuertaDeEmbarque():** permite liberar las puertas de embarque cuando se despegue o aterrice. Primero comprueba si debe para la ejecución, coge el semáforo binario para modificar el array de puertas de embarque/desembarque, busca la posición del avión en el array de puertas de embarque para eliminarlo, actualiza la interfaz del servidor y libera el semáforo binario, libera el que controla el número de puertas de embarque (5 permits) y el semáforo que controla el número de puertas de embarque/desembarque (6 permits). De esta forma se liberan las puertas para próximos aviones.

**salirDePuertaDeDesembarque():** similar al anterior pero libera un semáforo de puertas de desembarque de cinco permits (“control5Desembarque”) en vez de el de puertas de embarque (“control5Embarque”).

```
public void salirDePuertaDeEmbarque (Avion avion) {
    pausa.mirar();
    try {
        arrayEmbarque.acquire();
        int pos = uiUpdater.buscarAvion(avion, puertasDeEmbarque);
        puertasDeEmbarque[pos]=null;
        uiUpdater.updateSalidaPuertasDeEmbarque(avion);
        System.out.println("El avion "+ avion.getIdAv()+ " ha salido del gate de e
    }

    catch(Exception E) {
        E.printStackTrace();
    }
    finally{
        arrayEmbarque.release();
        control6Puertas.release();
        control5Embarque.release();
    }
}
```

**embarcarPasajeros():** este método sube pasajeros del aeropuerto a un avión en puerta de embarque. Siempre intentarán llenar el avión con la máxima cantidad posible (limitado por la capacidad del avión). Primero comprueba si el hilo debe continuar su ejecución. Al modificar los pasajeros del aeropuerto debe coger el cerrojo y más tarde liberarlo.

**DesembarcarPasajeros():** similar al anterior pero establece a cero los pasajeros del avión y los añade al aeropuerto.

```
public void embarcarPasajeros (Avion avion) throws InterruptedException {
    pausa.mirar();
    try{
        pasajerosLock.lock();
        int capacidad_restante = avion.getCapacidad() - avion.getPasajeros();
        if(pasajeros > capacidad_restante){
            avion.setPasajeros(capacidad_restante); //Completo avion con lo que falte
            this.pasajeros -= capacidad_restante;
            uiUpdater.updatePasajeros(pasajeros);
        }
        else{
            avion.setPasajeros(pasajeros);
            this.pasajeros = 0; //No quedaran pasajeros en el aeropuerto
            uiUpdater.updatePasajeros(pasajeros);
        }
    }
    finally{
        pasajerosLock.unlock();
        Random random = new Random();
        Thread.sleep(1000 + random.nextInt(2000));
        System.out.println("El avion "+ avion.getIdAv()+ " ha embarcado a pasajeros");
    }
}
```

**accederAlAreaDeRodaje():** antes de despegar y después de aterrizar, un avión debe acceder al área de rodaje. Este método añade el avión en dicha zona (array de aviones). Está controlado por un cerrojo y modifica el array y aumenta en uno su contador. Por último, libera el cerrojo.

**salidaDelAreaDeRodaje():** similar al anterior pero antes de nada ejecuta pausa.mirar() para ver si debe para su ejecución. Elimina el avión del área de rodaje y disminuye en uno su contador. Además, actualiza la interfaz del servidor.

```
public void accesoAlAreaDeRodaje (Avion avion) {
    //pausa.mirar();
    try {
        areaDeRodajeLock.lock();
        areaDeRodaje.add(avion);
        nAvRodaje++;
        uiUpdater.updateAreaDeRodaje(areaDeRodaje);
        System.out.println("El avion "+ avion.getIdAv()+ " ha entrar
    }
    finally {
        areaDeRodajeLock.unlock();
    }
}
```

**accedeAPistaDespegue():** permite acceder a una pista para despegar. Comprueba que hay pistas de despegue libres con un semáforo de 4 permits (pistasSem). Asegura el acceso al array de pistas en exclusión mutua con semáforo binario (pistasSembin). Además de tener pistas libres, deben estar abiertas, por lo que se lee el array de booleanos de pistas abiertas en exclusión mutua con cerrojo (pistasAbiertasLock). Este cerrojo es que se utilizará para cerrar y abrir las pistas desde la interfaz del cliente. Mientras no haya pistas abiertas, se debe esperar (pistasCerradas.await()). Saldremos del área de rodaje y buscaremos una pista libre y abierta. Se añade al array de pistas y liberamos el cerrojo de booleanos y semáforo para dar acceso a los booleanos y al array de pistas a otros. “pistasSem” se libera cuando dejemos la pista libre.



```

public void AccedeAPistaDespegue (Avion avion) {
    pausa.mirar();
    try{
        pistasSem.acquire(); //Comprueba que hay pistas libres
        pistasSemBin.acquire(); //unico modificando array de pistas
        System.out.println(avion.getIdAv() + " tiene cerrojo de pistas");
        pistasAbiertasLock.lock();//unico accediendo al array de booleanos pistas abiertas
        System.out.println(avion.getIdAv() + " tiene cerrojo de pistas abiertas");
        while(!pistasAbiertas()){ //comprobamos que hay pistas abiertas, de lo contrario esperamos
            pistasCerradas.await();
        }
        System.out.println(avion.getIdAv() + "hay pistas abiertas");
        salidaDelAreaDeRodaje(avion);
        int pos = uiUpdater.primerHuecoEnArray(pistasDeDespegue);
        while(!pistasAbiertas[pos]){ //mientras la pista libre esta cerrada, buscamos otra
            pos = (pos +1)%4;
        }
        pistasDeDespegue[pos]=avion;
        uiUpdater.updatePistasDespegue(pistasDeDespegue, avion);
        System.out.println("El avion "+ avion.getIdAv()+ " ha sido asignado a una pista de despegue e
    }catch(InterruptedException e){
        e.printStackTrace();
    }
    finally {
        pistasAbiertasLock.unlock();
        pistasSemBin.release();
    }
}

```

**solicitaPistaAterrizaje():** permite acceder a una pista de aterrizaje. Intentará coger el semáforo de pistas libres cada 1-4 segundos (tryAcquire()). Una vez conseguido, accede al array de pistas y de booleanos en exclusión mutua con “pistasSemBin” y “pistasAbiertasLock”. Mientras no haya pistas abiertas espera (pistasCerradas.await()). Buscará una pistas libre y abierta, añadirá el avión a la pista, actualiza la interfaz del servidor y libera el cerrojo y semáforo.

```

public void SolicitaPistaAterrizaje(Avion avion) {
    pausa.mirar();
    Random rand = new Random ();
    while(!pistasSem.tryAcquire()){ //comprueba que hay pistas libres
        try{
            Thread.sleep (1000 + rand.nextInt(4000));
        }catch(InterruptedException e){
            e.printStackTrace();
        }
    }

    try{
        pistasSemBin.acquire(); //unico accediendo a las pistas
        pistasAbiertasLock.lock();//unico accediendo al array de booleanos pistas abi
        while(!pistasAbiertas()){ //comprobamos que hay pistas abiertas, de lo contra
            pistasCerradas.await();
        }
        int pos = uiUpdater.primerHuecoEnArray(pistasDeDespegue);
        while(!pistasAbiertas[pos]){ //mientras la pista libre esta cerrada, buscamos
            pos = (pos+1)%4;
        }
        pistasDeDespegue[pos]=avion;
        uiUpdater.updatePistasAterrizaje(pistasDeDespegue, avion);
        System.out.println("El avion "+ avion.getIdAv()+ " ha sido asigando a una pis
        avion.annadirVuelo();
    }catch(Exception e){
        e.printStackTrace();
    }
    finally {
        pistasAbiertasLock.unlock();
        pistasSemBin.release();
    }
}

```

**despegaAvion():** se encarga de sacar el avión de la pista de aterrizaje para después meterlo en la aerovía de salida del aeropuerto. Para ello, coge el semáforo binario “pistasSemBin” para modificar el array de pistas en

exclusión mutua, actualiza la interfaz del servidor y libera el semáforo binario y el semáforo “pistasSem” indicando que se deja libre una pista.

```
//despegar
public void despegarAvion (Avion avion) throws InterruptedException {
    pausa.mirar();
    Random random = new Random();
    Thread.sleep(1000 + random.nextInt(4000));
    pistasSemBin.acquire();
    try {
        int pos = uiUpdater.buscarAvion(avion, pistasDeDespegue);
        pistasDeDespegue[pos]=null;
        uiUpdater.updateSalirDePista(avion);
        System.out.println("El avion " + avion.getIdAv() + " ha despegado");
    }
    finally {
        pistasSemBin.release();
        pistasSem.release();
    }
}
```

**avionEnElAire():** mete el avión despegado en la aerovía de salida del aeropuerto. Para garantizar la exclusión mutua al modificar las aerovías, se emplea un cerrojo (aeroviasLock).

**avionEnTierra():** simula el aterrizaje del avión dejando la aerovía tras solicitar una pista de aterrizaje. Modifica las aerovías en exclusión mutua con un cerrojo, actualiza la interfaz gráfica del servidor y libera de nuevo el cerrojo.

```
public void avionEnElAire (Avion a) {
    //pausa.mirar();
    aeroviasLock.lock();
    try{
        aeroviaSalida.add(a);
        if (this.nombre=="Adolfo Suarez") {
            pP.updateAerovia1(aeroviaSalida);
            System.out.println("El avion "+ a.getIdAv()+ " ha despegado");
        }
        else{
            pP.updateAerovia2(aeroviaSalida);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally{
        aeroviasLock.unlock();
    }
}
```

```
public void avionEnTierra (Avion a) {
    //pausa.mirar();
    aeroviasLock.lock();
    try{
        aeroviaLlegada.remove(a);
        if (this.nombre=="Adolfo Suarez") {
            pP.updateAerovia1(aeroviaSalida);
            System.out.println("El avion "+ a.getIdAv()+ " ha aterrizado");
        }
        else{
            pP.updateAerovia2(aeroviaSalida);
        }
        System.out.println("El avion "+ a.getIdAv()+ " ha aterrizado");
    } catch (Exception e) {
        e.printStackTrace();
    } finally{
        aeroviasLock.unlock();
    }
}
```

**vaAlTaller():** después de cada vuelo, el avión debe pasar por el taller. Este tiene una capacidad máxima de veinte aviones, controlado por un semáforo de veinte permits(“tallerSem”), y su acceso de uno en uno se hace mediante un semáforo binario (“puertaTaller”).

```

public void vaAlTaller (Avion avion){
    pausa.mirar();
    Random rand = new Random ();
    try{
        tallerSe.acquire();
        try {
            puertaTaller.acquire();
            salirDelAreaDeEstacionamiento(avion);
            Thread.sleep(1000); //Entramos al taller
            taller.add(avion);
            nAvTaller++;
            uiUpdater.updateTaller(taller);
        }catch(Exception e){
            e.printStackTrace();
        }
        finally{
            puertaTaller.release(); //dejamos la puerta li
            System.out.println("El avion "+ avion.getIdAv()
        }

        if (avion.getNumeroDeVuelos()==15) {
            System.out.println("El avion "+ avion.getIdAv()
            Thread.sleep (5000 + rand.nextInt(5000));
            avion.setNumeroDeVuelos(0);
        }
        else {
            System.out.println("El avion "+ avion.getIdAv()
            Thread.sleep (1000 + rand.nextInt(4000));
        }
        System.out.println("El avion "+ avion.getIdAv()+ "
        try{
            puertaTaller.acquire();
            Thread.sleep(1000); //salimos del taller
            taller.remove(avion);

```

Al implementar la parte dos del proyecto, las instancias de la clase aeropuerto serán objetos remotos a los cuales se les harán llamadas a algunos de sus métodos. Estos se declaran en una interfaz remota accesible por el lado cliente y se implementan en la clase aeropuerto. Estos son los siguientes:

```

public interface InterfazRemota extends Remote{
    //incluir metodos del obj compartido (aeropuerto)
    public int getPasajeros()throws RemoteException;
    public int getAvionesEnHangar()throws RemoteException;
    public int getAvionesEnTaller()throws RemoteException;
    public int getAvionesEnAreaEstacionamiento()throws RemoteException;
    public int getAvionesEnAreaRodaje()throws RemoteException;
    public String getAerovia1()throws RemoteException;
    public String getAerovia2()throws RemoteException;
    public void cerrarPista(int nPista) throws RemoteException;
    public void abrirPista(int nPista) throws RemoteException;
    public String getNombre() throws RemoteException;
}

```

### Clase GenHilloBus y GenHiloAvion:

para generar los autobuses y aviones de forma simultánea, se ha creado una clase hilo para generar los dos tipos de hilos (aviones y autobuses). Para construir estos dos hilos generadores se pasa como parámetros los dos aeropuertos la cantidad de hilos a crear de cada uno y una instancia de la clase Pausa compartida por todos los hilos.

```

public class GenHilosBus extends Thread{
    //Atributos
    private Aeropuerto aeropuerto1;
    private Aeropuerto aeropuerto2;
    private int nAutobuses;
    private Pausa pausa;
    //Constructor
    public GenHilosBus(Aeropuerto a1, Aeropuerto a2, int nAutobuses, Pausa pausa){
        this.aeropuerto1 = a1;
        this.aeropuerto2 = a2;
        this.nAutobuses = nAutobuses;
        this.pausa = pausa;
    }
    //metodos
    public void run(){
        Random rand = new Random();
        for (int i = 0; i<nAutobuses; i++){
            pausa.mirar();
            String str = "B-";
            if (i<1000){
                str+="0";
                if (i<100){
                    str+="0";
                    if(i<10){
                        str+="0";
                    }
                }
            }
            if(i%2==0){
                Autobus ab = new Autobus (str+i, 50, aeropuerto1);
                ab.start();
            }
            else{
                Autobus ab = new Autobus (str+i, 50, aeropuerto2);
                ab.start();
            }

            try{
                Thread.sleep(500+rand.nextInt(500));
            }catch (InterruptedException e){
                e.printStackTrace();
            }
        }
    }
}

```

```

public class GenHilosAvion extends Thread{
    //Atributos
    private Aeropuerto a1, a2;
    private int nAviones;
    private Pausa pausa;
    //Constructor
    public GenHilosAvion(Aeropuerto a1, Aeropuerto a2, int nAviones, Pausa pausa){
        this.a1 = a1;
        this.a2 = a2;
        this.nAviones = nAviones;
        this.pausa = pausa;
    }
    //Metodos
    public void run(){
        Random rand = new Random();
        for (int i = 0; i<nAviones; i++){
            String str = generarCaracteres();
            pausa.mirar();
            if (i<1000){
                str+="0";
                if (i<100){
                    str+="0";
                    if(i<10){
                        str+="0";
                    }
                }
            }
            if(i%2==0){
                Avion av = new Avion (str+i, 100 + rand.nextInt(200), a1, a2);
                av.start();
            }
            else{
                Avion av = new Avion (str+i, 100 + rand.nextInt(200), a2, a1);
                av.start();
            }

            try{
                Thread.sleep(1000+rand.nextInt(2000));
            }catch (InterruptedException e){
                e.printStackTrace();
            }
        }
    }
}

```

## Clase Avión:

esta clase hilo se encarga de simular los movimientos de un avión haciendo llamadas a los correspondientes métodos de la clase aeropuerto. Tiene atributos como capacidad, pasajeros, aeropuerto de origen y destino y su id. Al ser un hilo su ciclo de ejecución es el siguiente:

```

public class Avion extends Thread implements Serializable{
    private String id;
    private int capacidad;
    private int pasajeros;
    private int numeroDeVuelos;
    private Aeropuerto aeropuertoLinkeado;
    private Aeropuerto aeropuertoSecundario;

    public Avion (String id, int capacidad, Aeropuerto aOrigen, Aeropuerto aDestino)
    {
        this.id = id;
        this.capacidad = capacidad;
        this.pasajeros = 0;
        this.aeropuertoLinkeado=aOrigen;
        this.aeropuertoSecundario=aDestino;
        this.numeroDeVuelos=0;
    }
}

```

```

public void run() {
    while(true){
        try {
            Random rand = new Random ();
            aeropuertoLinkeado.entrarAlHangar(this);
            Thread.sleep (1000 + rand.nextInt(2000));
            aeropuertoLinkeado.SaleDelHangar(this);
            aeropuertoLinkeado.entrarAlAreaDeEstacionamiento(this);
            Thread.sleep (1000 + rand.nextInt(2000));
            aeropuertoLinkeado.esperarAPuertaDeEmbarque(this);
            aeropuertoLinkeado.embarcarPasajeros(this);
            int i = 0;
            while ((i<3)&&(this.pasajeros < this.capacidad)){
                System.out.println("El avion " + this.id + " no se ha llenado, espera a ll
                aeropuertoLinkeado.embarcarPasajeros(this);
                Thread.sleep (1000 + rand.nextInt(2000));
                Thread.sleep (1000 + rand.nextInt(4000));
                i++;
            }
            aeropuertoLinkeado.salirDePuertaDeEmbarque(this);
            aeropuertoLinkeado.accesoAlAreaDeRodaje(this);
            Thread.sleep (1000 + rand.nextInt(4000));
            aeropuertoLinkeado.AccedeAPistaDespegue(this);
            Thread.sleep (1000 + rand.nextInt(2000));
            aeropuertoLinkeado.despegaAvion(this);
            aeropuertoLinkeado.avionEnElAire(this);
            Thread.sleep (15000 + rand.nextInt(15000));
            cambiarAeropuerto();
            aeropuertoLinkeado.SolicitaPistaAterrizaje(this); //Mete al avion en la pista
            Thread.sleep (1000 + rand.nextInt(4000)); //Aterrizo
            aeropuertoLinkeado.avionEnTierra(this);
            aeropuertoLinkeado.salirDePista(this);
            aeropuertoLinkeado.accesoAlAreaDeRodaje(this);
            Thread.sleep (3000 + rand.nextInt(2000)); //camino entre pista
            aeropuertoLinkeado.esperarAPuertaDeDesembarque(this);
            Thread.sleep (1000 + rand.nextInt(4000)); //Desembarca pasajeros
            aeropuertoLinkeado.desembarcarPasajeros(this);
            aeropuertoLinkeado.salirDePuertaDeDesembarque(this);
            aeropuertoLinkeado.entrarAlAreaDeEstacionamiento(this);
            Thread.sleep (1000 + rand.nextInt(4000)); //piloto hace comprobaciones
            aeropuertoLinkeado.vaAlTaller(this);
            int randomNumber = rand.nextInt(1);
            if (randomNumber == 1) {
                aeropuertoLinkeado.entrarAlHangar(this);
                Thread.sleep (15000 + rand.nextInt(15000));
            }
        }
    }
}

```

## Clase Autobús:

esta clase hilo se encarga de simular los movimientos de un autobús que trasporta pasajeros entre el centro de la ciudad y un aeropuerto asignado. Esto lo hace durante el ciclo de ejecución llamando a métodos implementados en la clase aeropuerto. Los atributos necesarios para el autobús son el aeropuerto asignado, la capacidad, pasajeros y el id.

//Constructor y ciclo de ejecución:

```

public class Autobus extends Thread{

    private String id;
    private int capacidad;
    private int pasajeros;
    private Aeropuerto aeropuerto;

    // Constructor
    public Autobus (String id, int capacidad, Aeropuerto aeropuerto) {
        this.capacidad=capacidad;
        this.id= id;
        this.pasajeros=0;
        this.aeropuerto = aeropuerto;
    }
}

```

```

public void run() {
    while (true) {
        try{
            Random rand = new Random();
            int numAlea = rand.nextInt(3000);
            Thread.sleep(2000+numAlea); // Espera en el centro de la ciudad
            System.out.println("El autobus "+this.id+ " esta esperando en el centro de la ciudad");
            int pasCiudad = rand.nextInt(capacidad);
            pasajeros+=pasCiudad;
            System.out.println("El autobus "+this.id+ " está de camino al aeropuerto "+aer);
            Thread.sleep(rand.nextInt(5000)+5000); // De camino al aeropuerto
            aeropuerto.lleganPasajerosDeAutobus(this); // Llega al aeropuerto
            System.out.println("El autobus "+this.id+ " ha llegado al aeropuerto");
            Thread.sleep(rand.nextInt(2000)+3000); // Espera mientras bajan los pasajeros
            System.out.println("El autobus "+this.id+ " ha descargado los pasajeros");
            aeropuerto.subenPasajerosAlBus(this);
            System.out.println("El autobus "+this.id+ " ha cogido pasajeros y va a la ciudad");
            Thread.sleep(rand.nextInt(5000)+5000);
            System.out.println("El autobus "+this.id+ " ha descargado los pasajeros");
            this.pasajeros=0;
        }
        catch(Exception e) {}
    }
}

```

### Clase Cliente:

el lado del cliente se inicia desde la interfaz gráfica, pero se ejecuta en un hilo aparte (HiloCliente).

En la parte gráfica se buscan los objetos remotos (aeropuertos) y crea el hilo cliente pasando los objetos remotos como parámetros de su constructor.

```

public pantallaCliente() {

    try{
        this.reg = LocateRegistry.getRegistry("127.0.0.1",5000);
        try{
            this.i1 = (InterfazRemota) reg.lookup("aeropuerto1");
            this.i2 = (InterfazRemota) reg.lookup("aeropuerto2");
            initComponents();
            panelCliente3.setJPanelNombreAeropuerto(i1.getNombre());
            panelCliente3.setAeropuerto(i1);
            panelCliente4.setJPanelNombreAeropuerto(i2.getNombre());
            panelCliente4.setAeropuerto(i2);
        }
    }
}

```

```

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            pantallaCliente pCliente = new pantallaCliente();
            pCliente.setVisible(true);
            ClienteHilo cH = new ClienteHilo(pCliente, pCliente.getI1(),pCliente.getI2());
            cH.start();
        }
    });
}

```

Una vez iniciado el hilo cliente comienza su ciclo de ejecución:

```

public class ClienteHilo extends Thread{
    pantallaCliente pC;
    InterfazRemota i1;
    InterfazRemota i2;
    public ClienteHilo (pantallaCliente pC, InterfazRemota i1, InterfazRemota i2) {
        this.pC=pC;
        this.i1=i1;
        this.i2=i2;
    }

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

                int pasajerosA1 = i1.getPasajeros();
                int contHangarA1 = i1.getAvionesEnHangar();
                int contTallerA1 = i1.getAvionesEnTaller();
                int contEstacionamientoA1 = i1.getAvionesEnAreaEstacionamiento();
                int contRodajeA1 = i1.getAvionesEnAreaRodaje();
                panelCliente panelA1 = pC.getPanelCliente3();
                panelA1.setPasajerosText(""+pasajerosA1);
                panelA1.setHangarText(""+contHangarA1);
                panelA1.setTallerText(""+contTallerA1);
                panelA1.setEstacionamientoText(""+contEstacionamientoA1);
                panelA1.setRodajeText(""+contRodajeA1);

                int pasajerosA2 = i2.getPasajeros();
                int contHangarA2 = i2.getAvionesEnHangar();
                int contTallerA2 = i2.getAvionesEnTaller();
                int contEstacionamientoA2 = i2.getAvionesEnAreaEstacionamiento();
                int contRodajeA2 = i2.getAvionesEnAreaRodaje();

                panelCliente panelA2 = pC.getPanelCliente4();
                panelA2.setPasajerosText(""+pasajerosA2);
                panelA2.setHangarText(""+contHangarA2);
                panelA2.setTallerText(""+contTallerA2);
                panelA2.setEstacionamientoText(""+contEstacionamientoA2);
                panelA2.setRodajeText(""+contRodajeA2);

                String aerovial = i1.getAerovial();
                String aerovia2 = i1.getAerovia2();
                pC.setTextAerovial(aerovial);
                pC.setTextAerovia2(aerovia2);
            }
        }
    }
}

```

Continuamente hace consultas al servidor para actualizar los valores de la interfaz cliente.

### Clase Servidor:

el lado servidor se inicia desde la interfaz gráfica donde se crean los aeropuertos, aerovías (compartidas por ambos), se registran los aeropuertos como objetos remotos (RMI), se inician los hilos generadores de autobuses y aviones, se instancia un objeto de la clase Pausa (común a todos los hilos del proyecto) y se llama al constructor de la interfaz gráfica. A esta se le pasa como parámetros las aerovías (para poder mostrar su contenido) y el objeto Pausa. Además, en esta parte se implementan los métodos de la interfaz “UIUpdate”, que permite actualizar los campos de la interfaz gráfica cada vez que hay un cambio en el sistema.

```

public class pantallaPrincipal extends javax.swing.JFrame {

    ArrayList<Avion> aerovial;
    ArrayList<Avion> aerovia2;
    Pausa pausa;

    /**
     * Creates new form pantallaPrincipal
     * @param p1
     */
    public pantallaPrincipal(ArrayList<Avion> aerovial, ArrayList<Avion> aerovia2, Pausa pausa) {
        this.aerovial = aerovial;
        this.aerovia2 = aerovia2;
        this.pausa = pausa;
        initComponents();
    }
}

```



```

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            int nAutobuses = 4000;
            int nAviones = 8000;
            Pausa pausa = new Pausa();
            Lock aeroViasLock = new ReentrantLock();
            ArrayList<Avion> MadridBarcelona = new ArrayList<>();
            ArrayList<Avion> BarcelonaMadrid = new ArrayList<>();
            pantallaPrincipal p = new pantallaPrincipal(MadridBarcelona,BarcelonaMadrid, pausa);
            claseEscritor clEs = new claseEscritor ("log.txt");
            try{
                Aeropuerto Adolfo_Suarez = new Aeropuerto (MadridBarcelona,BarcelonaMadrid, "Adolfo Suarez", aeroViasLock,p.getPanelAeropuerto1(),p, pausa,clEs);
                Aeropuerto El_Prat = new Aeropuerto (BarcelonaMadrid,MadridBarcelona,"El prat", aeroViasLock,p.getPanelAeropuerto2(),p, pausa,clEs);
                Registry reg = LocateRegistry.createRegistry(5000);
                reg.rebind("aeropuerto1", Adolfo_Suarez);
                reg.rebind("aeropuerto2", El_Prat);
                GenHilosBus buses = new GenHilosBus(Adolfo_Suarez, El_Prat, nAutobuses, pausa);
                GenHilosAvion aviones = new GenHilosAvion(Adolfo_Suarez, El_Prat, nAviones, pausa);
                buses.start();
                aviones.start();
            }catch(Exception e){
                e.printStackTrace();
            }
            p.setVisible(true);
        }
    });
}

```

## Interfaz UIUpdate:

esta interfaz declara los siguientes métodos:

```

public interface UIUpdater {
    public void updateHangar (ArrayList<Avion> hangar);
    public void updatebus1 (Autobus bus);
    public void updatebus2 (Autobus bus);
    public void updateTaller (ArrayList<Avion> taller);
    public void updateAreaDeEstacionamiento (ArrayList<Avion> areaEstacionamiento);
    public void updatePuertasDeEmbarque (Avion[] puertasDeEmbarque);
    public void updatePuertasDeDesembarque (Avion[] puertasDeEmbarque);
    public void updateSalidaPuertasDeEmbarque (Avion a);
    public void updateAreaDeRodaje (ArrayList<Avion> areaDeRodaje);
    public void updatePistasAterrizaje (Avion[] pistas);
    public void updatePistasDespegue (Avion[] pistas);
    public void updateSalirDePista (Avion avion);
    public void setAeropuerto (Aeropuerto a);
    public int buscarAvion(Avion a, Avion[] array);
    public int primerHuecoEnArray (Avion[] array);
}

```

## Clase Pausa:

la clase pausa nos permitirá pausar y reanudar la ejecución de un programa de forma sencilla. Las clases aeropuerto y generadores de hilos tendrán la misma instancia de esta clase. Contiene tres métodos restringidos con la cláusula synchronized y un atributo booleano (parado) inicializado a false por defecto. El método mirar() comprueba el estado de la variable parado y en función de su valor duerme el hilo que llama a esta función con wait(). Este método (mirar()) se llama desde un hilo avión, autobús, o generadores de hilos durante su ciclo de ejecución. El método pausar() establece el valor de la variable parado a true. El método pausar() establece el valor parado a false y despierta aquellos hilos dormidos mediante notifyAll(). Estos métodos se ejecutan al pulsar el botón pausar/reanudar de la interfaz gráfica.



```

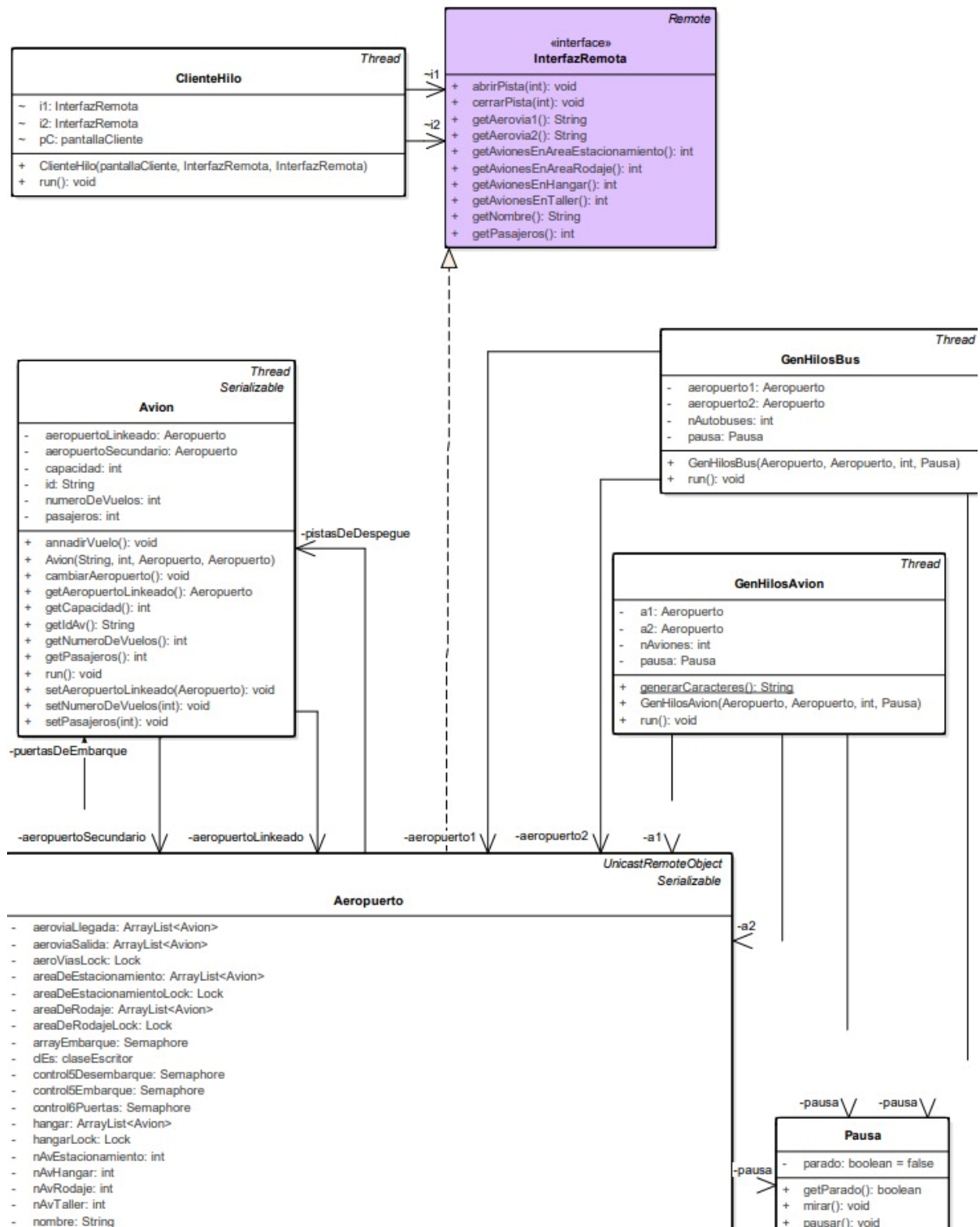
public class Pausa {
    //Atributos
    private boolean parado = false;
    //Metodos
    public synchronized void mirar() {
        try{
            while(parado) {
                wait();
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        } finally{
        }
    }

    public synchronized void pausar() {
        try{
            parado = true;
        } catch (Exception e) {
            e.printStackTrace();
        }
        finally{
        }
    }

    public synchronized void reanudar() {
        try{
            parado = false;
            notifyAll();
        } catch (Exception e) {
            e.printStackTrace();
        }
        finally{
        }
    }
}

```

## 4.- Diagrama de clases:



```

- pasajeros: int
- pasajerosLock: Lock
- pausa: Pausa
- pistasAbiertas: boolean ([]) = {true, true, tr...
- pistasAbiertasLock: Lock
- pistasCerradas: Condition
- pistasDeDespegue: Avion ([])
- pistasSem: Semaphore
- pistasSemBin: Semaphore
- pP: pantallaPrincipal
- puertasDeEmbarque: Avion ([])
- puertaTaller: Semaphore
- taller: ArrayList<Avion>
- tallerSe: Semaphore
- uiUpdater: panelAeropuerto

```

```

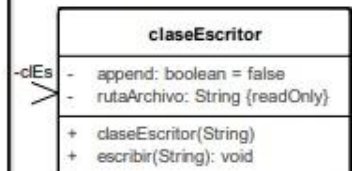
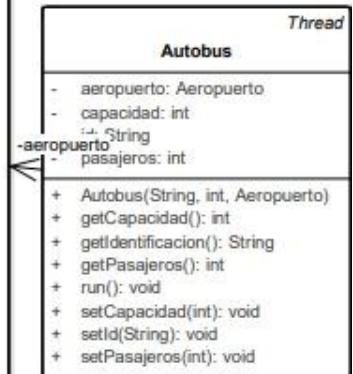
+ abrirPista(int): void
+ AccedeAPistaDespegue(Avion): void
+ accesoAlAreaDeRodaje(Avion): void
+ Aeropuerto(ArrayList<Avion>, ArrayList<Avion>, String, Lock, panelAeropuerto, pantallaPrincipal, Pausa, claseEscritor)
+ avionEnElAire(Avion): void
+ avionEnTierra(Avion): void
+ cerrarPista(int): void
+ desembarcarPasajeros(Avion): void
+ despegarAvion(Avion): void
+ embarcarPasajeros(Avion): void
+ entrarAlAreaDeEstacionamiento(Avion): void
+ entrarAlHangar(Avion): void
+ esperarAPuertaDeDesembarque(Avion): void
+ esperarAPuertaDeEmbarque(Avion): void
+ getAerovia1(): String
+ getAerovia2(): String
+ getAeroviaLlegada(): ArrayList<Avion>
+ getAeroviaSalida(): ArrayList<Avion>
+ getAeroViasLock(): Lock
+ getAreaDeEstacionamiento(): ArrayList<Avion>
+ getAreaDeEstacionamientoLock(): Lock
+ getAreaDeRodaje(): ArrayList<Avion>
+ getAreaDeRodajeLock(): Lock
+ getArrayEmbarque(): Semaphore
+ getAvionesEnAreaEstacionamiento(): int
+ getAvionesEnAreaRodaje(): int
+ getAvionesEnHangar(): int
+ getAvionesEnTaller(): int
+ getControl5Desembarque(): Semaphore
+ getControl5Embarque(): Semaphore
+ getControl6Puertas(): Semaphore
+ getHangar(): ArrayList<Avion>
+ getHangarLock(): Lock
+ getNombre(): String
+ getPasajeros(): int
+ getPasajerosLock(): Lock
+ getPistasDeDespegue(): Avion[]
+ getPistasSem(): Semaphore
+ getPuertasDeEmbarque(): Avion[]
+ getPuertaTaller(): Semaphore
+ getTaller(): ArrayList<Avion>
+ getTallerSe(): Semaphore
+ lleganPasajerosDeAutobus(Autobus): void
+ pistasAbiertas(): boolean
+ SaleDelHangar(Avion): void
+ salidaDelAreaDeRodaje(Avion): void
+ salirAreaDeRodaje(Avion): void
+ salirDelAreaDeEstacionamiento(Avion): void
+ salirDePista(Avion): void
+ salirDePuertaDeDesembarque(Avion): void
+ salirDePuertaDeEmbarque(Avion): void
+ setPasajeros(int): void
+ SolicitaPistaAterrizaje(Avion): void
+ subenPasajerosAlBus(Autobus): void
+ vaAlTaller(Avion): void

```

```

+ reanudar(): void

```





## 5.- Código Fuente:

### Aeropuerto.java:

```
package vidri.vidri.pecl;

import java.io.Serializable;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import java.util.Random;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.Semaphore;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import vidri.vidri.interfaz.panelAeropuerto;
import vidri.vidri.interfaz.pantallaPrincipal;

/**
 *
 * @author gonza
 */
public class Aeropuerto extends UnicastRemoteObject implements InterfazRemota, Serializable{

    private int pasajeros, nAvHangar, nAvTaller, nAvEstacionamiento, nAvRodaje;

    private ArrayList<Avion> hangar, areaDeRodaje, areaDeEstacionamiento, aeroviaSalida, aeroviaLlegada,taller;

    private Lock hangarLock, pasajerosLock, pistasAbiertasLock, areaDeRodajeLock,
areaDeEstacionamientoLock, aeroViasLock;

    private Semaphore pistasSem, pistasSemBin, tallerSe, puertaTaller,
control6Puertas,control5Embarque,control5Desembarque, arrayEmbarque;

    private String nombre;

    private panelAeropuerto uiUpdater;

    private pantallaPrincipal pP;

    private Avion[] puertasDeEmbarque, pistasDeDespegue;

    private Pausa pausa;

    private boolean[] pistasAbiertas = {true, true, true, true};
```



```
private Condition pistasCerradas;
```

```
private claseEscritor clEs;
```

```
public Aeropuerto(ArrayList<Avion> aeroviaSalida, ArrayList<Avion> aeroviaLlegada, String nombre, Lock  
aeroViasLock, panelAeropuerto u, pantallaPrincipal p, Pausa pausa, claseEscritor clEs) throws RemoteException  
{
```

```
    this.aeroviaSalida = aeroviaSalida;
```

```
    this.aeroviaLlegada = aeroviaLlegada;
```

```
    this.hangar = new ArrayList<>();
```

```
    this.pasajeros = 0;
```

```
    this.nAvHangar = 0;
```

```
    this.nAvTaller = 0;
```

```
    this.nAvEstacionamiento = 0;
```

```
    this.nAvRodaje = 0;
```

```
    this.areaDeEstacionamiento = new ArrayList<>();
```

```
    this.areaDeRodaje = new ArrayList<>();
```

```
    this.pistasSemBin = new Semaphore(1, true);
```

```
    this.tallerSe = new Semaphore(20, true);
```

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

```
    this.pistasSem = new Semaphore(4);
```

```
    //this.pistasAbiertas = new boolean[4];
```

```
    this.puertasDeEmbarque = new Avion[6];
```

```
    this.pistasDeDespegue = new Avion[4];
```

```
    this.pistasAbiertasLock = new ReentrantLock();
```

```
    this.pistasCerradas = pistasAbiertasLock.newCondition();
```

```
    this.pasajerosLock = new ReentrantLock();
```

```
    this.hangarLock = new ReentrantLock();
```

```
    this.areaDeRodajeLock = new ReentrantLock();
```

```
    this.areaDeEstacionamientoLock = new ReentrantLock();
```

```
    this.arrayEmbarque = new Semaphore(1, true);
```

```
    this.control6Puertas = new Semaphore(6, true);
```

```
    this.control5Embarque = new Semaphore(5, true);
```

```
    this.control5Desembarque = new Semaphore(5, true);
```

```
    this.aeroViasLock = aeroViasLock;
```

```
    this.taller = new ArrayList<>();
```

```

this.nombre=nombre;

this.uiUpdater = u;

u.setearNombre(nombre);

this.pP = p;

this.pausa = pausa;

this.clEs=clEs;

}

```

```

public void despegarAvion (Avion avion) throws InterruptedException {

    pausa.mirar();

    Random random = new Random();

    Thread.sleep(1000 + random.nextInt(4000));

    pistasSemBin.acquire();

    try {

        int pos = uiUpdater.buscarAvion(avion,pistasDeDespegue);

        pistasDeDespegue[pos]=null;

        uiUpdater.updateSalirDePista(avion);

        System.out.println("El avion " + avion.getIdAv() + " ha despegado con " + avion.getPasajeros() + " pasajeros");

        clEs.escribir("El avion " + avion.getIdAv() + " ha despegado con " + avion.getPasajeros() + " pasajeros");

    }

    finally {

        pistasSemBin.release();

        pistasSem.release();

    }

}

```

```

public void vaAlTaller (Avion avion){

    pausa.mirar();

    Random rand = new Random ();

    try{

        tallerSe.acquire();

        try {

```

```

    puertaTaller.acquire();

    salirDelAreaDeEstacionamiento(avion);

    Thread.sleep(1000); //Entramos al taller

    taller.add(avion);

    nAvTaller++;

    uiUpdater.updateTaller(taller);

} catch (Exception e) {

    e.printStackTrace();

}

finally {

    puertaTaller.release(); //dejamos la puerta libre

    System.out.println("El avion " + avion.getIdAv() + " ha entrado al taller de " + this.nombre);

    clEs.escribir("El avion " + avion.getIdAv() + " ha entrado al taller de " + this.nombre);

}

if (avion.getNumeroDeVuelos() == 15) {

    System.out.println("El avion " + avion.getIdAv() + " tiene que someterse a un mantenimiento duro en " +
this.nombre);

    clEs.escribir("El avion " + avion.getIdAv() + " tiene que someterse a un mantenimiento duro en " +
this.nombre);

    Thread.sleep(5000 + rand.nextInt(5000));

    avion.setNumeroDeVuelos(0);

}

else {

    System.out.println("El avion " + avion.getIdAv() + " se sometera a una revision rapida en " + this.nombre);

    clEs.escribir("El avion " + avion.getIdAv() + " se sometera a una revision rapida en " + this.nombre);

    Thread.sleep(1000 + rand.nextInt(4000));

}

System.out.println("El avion " + avion.getIdAv() + " ha terminado su mantenimiento en " + this.nombre);

clEs.escribir("El avion " + avion.getIdAv() + " ha terminado su mantenimiento en " + this.nombre);

try {

    puertaTaller.acquire();

    Thread.sleep(1000); //salimos del taller

    taller.remove(avion);

    nAvTaller--;

```



```

        uiUpdater.updateTaller(taller);
    }catch(Exception e){
        e.printStackTrace();
    }finally{
        puertaTaller.release();//dejamos la puerta libre
    }
}
catch (InterruptedException e){
    e.printStackTrace();
}
finally {
    tallerSe.release(); //dejamos hueco en el taller
}
System.out.println("El avion "+ avion.getIdAv()+ " ha salido del taller de "+ this.nombre);
clEs.escribir("El avion "+ avion.getIdAv()+ " ha salido del taller de "+ this.nombre);
}

```

```

public void avionEnElAire (Avion a) {
    //pausa.mirar();
    aeroViasLock.lock();
    try{
        aeroviaSalida.add(a);
        if (this.nombre=="Adolfo Suarez") {
            pP.updateAerovia1(aeroviaSalida);
            System.out.println("El avion "+ a.getIdAv()+ " ha despegado desde "+ this.nombre);
        }
        else{
            pP.updateAerovia2(aeroviaSalida);
        }
        clEs.escribir("El avion "+ a.getIdAv()+ " ha despegado desde "+ this.nombre);
    } catch (Exception e){
        e.printStackTrace();
    }finally{
        aeroViasLock.unlock();
    }
}

```

```
}  
}
```

```
public void avionEnTierra (Avion a) {  
    //pausa.mirar();  
    aeroViasLock.lock();  
    try{  
        aeroviaLlegada.remove(a);  
        if (this.nombre=="Adolfo Suarez") {  
            pP.updateAerovia1(aeroviaSalida);  
        }  
        else{  
            pP.updateAerovia2(aeroviaSalida);  
        }  
        System.out.println("El avion " + a.getIdAv() + " ha aterrizado en " + this.nombre);  
        cEs.escribir("El avion " + a.getIdAv() + " ha aterrizado en " + this.nombre);  
    }catch(Exception e){  
        e.printStackTrace();  
    }finally{  
        aeroViasLock.unlock();  
    }  
}  
  
public void salirDePista(Avion a){  
    pausa.mirar();  
  
    try{  
        pistasSemBin.acquire();  
        int pos = uiUpdater.buscarAvion(a, pistasDeDespegue);  
        pistasDeDespegue[pos]=null;  
        uiUpdater.updateSalirDePista(a);  
        System.out.println("El avion " + a.getIdAv() + " abandona la pista");  
        cEs.escribir("El avion " + a.getIdAv() + " abandona la pista");  
        pistasSem.release();  
    }catch(Exception e){
```

```

        e.printStackTrace();
    }finally{
        pistasSemBin.release();
    }
}

```

```

public void AccedeAPistaDespegue (Avion avion) {
    pausa.mirar();
    try{
        pistasSem.acquire(); //Comprueba que hay pistas libres
        pistasSemBin.acquire(); //unico modificando array de pistas
        System.out.println(avion.getIdAv() + " tiene cerrojo de pistas");
        pistasAbiertasLock.lock();//unico accediendo al array de booleanos pistas abiertas
        System.out.println(avion.getIdAv() + " tiene cerrojo de pistas abiertas");
        while(!pistasAbiertas()){ //comprobamos que hay pistas abiertas, de lo contrario esperamos
            pistasCerradas.await();
        }
        System.out.println(avion.getIdAv() + "hay pistas abiertas");
        cEs.escribir(avion.getIdAv() + "hay pistas abiertas");
        salidaDelAreaDeRodaje(avion);
        int pos = uiUpdater.primerHuecoEnArray(pistasDeDespegue);
        while(!pistasAbiertas[pos]){ //mientras la pista libre esta cerrada, buscamos otra
            pos = (pos +1)%4;
        }
        pistasDeDespegue[pos]=avion;
        uiUpdater.updatePistasDespegue(pistasDeDespegue, avion);
        System.out.println("El avion "+ avion.getIdAv()+ " ha sido asignado a una pista de despegue en "+
this.nombre);
        cEs.escribir("El avion "+ avion.getIdAv()+ " ha sido asignado a una pista de despegue en "+ this.nombre);
    }catch(InterruptedException e){
        e.printStackTrace();
    }
    finally {
        pistasAbiertasLock.unlock();
    }
}

```

```

        pistasSemBin.release();
    }
}

```

```

public void SolicitaPistaAterrizaje(Avion avion) {
    pausa.mirar();
    Random rand = new Random ();
    while(!pistasSem.tryAcquire()){ //comprueba que hay pistas libres
        try{
            Thread.sleep (1000 + rand.nextInt(4000));
        }catch(InterruptedException e){
            e.printStackTrace();
        }
    }

    try{
        pistasSemBin.acquire(); //unico accediendo a las pistas
        pistasAbiertasLock.lock();//unico accediendo al array de booleanos pistas abiertas
        while(!pistasAbiertas()){ //comprobamos que hay pistas abiertas, de lo contrario esperamos
            pistasCerradas.await();
        }
        int pos = uiUpdater.primerHuecoEnArray(pistasDeDespegue);
        while(!pistasAbiertas[pos]){ //mientras la pista libre esta cerrada, buscamos otra
            pos = (pos+1)%4;
        }
        pistasDeDespegue[pos]=avion;
        uiUpdater.updatePistasAterrizaje(pistasDeDespegue, avion);
        System.out.println("El avion "+ avion.getIdAv()+ " ha sido asigando a una pista de aterrizaje en "+this.nombre);
        clEs.escribir("El avion "+ avion.getIdAv()+ " ha sido asigando a una pista de aterrizaje en "+this.nombre);
        avion.annadirVuelo();
    }catch(Exception e){
        e.printStackTrace();
    }
}

```

```

    }
    finally {
        pistasAbiertasLock.unlock();
        pistasSemBin.release();
    }
}

public void accesoAlAreaDeRodaje (Avion avion) {
    //pausa.mirar();

    try {
        areaDeRodajeLock.lock();
        areaDeRodaje.add(avion);
        nAvRodaje++;
        uiUpdater.updateAreaDeRodaje(areaDeRodaje);
        System.out.println("El avion "+ avion.getIdAv()+ " ha entardo al area de rodaje de "+ this.nombre);
        cEs.escribir("El avion "+ avion.getIdAv()+ " ha entardo al area de rodaje de "+ this.nombre);
    }
    finally {
        areaDeRodajeLock.unlock();
    }
}

```

```

public void salidaDelAreaDeRodaje (Avion avion) {
    pausa.mirar();

    try {
        areaDeRodajeLock.lock();
        areaDeRodaje.remove(avion);
        nAvRodaje--;
        uiUpdater.updateAreaDeRodaje(areaDeRodaje);
        System.out.println("El avion "+ avion.getIdAv()+ " ha salido del area de rodaje de "+ this.nombre);
        cEs.escribir("El avion "+ avion.getIdAv()+ " ha salido del area de rodaje de "+ this.nombre);
    }
    finally {
        areaDeRodajeLock.unlock();
    }
}

```

```

    }
}

public void embarcarPasajeros (Avion avion) throws InterruptedException {
    pausa.mirar();
    try{
        pasajerosLock.lock();
        int capacidad_restante = avion.getCapacidad() - avion.getPasajeros();
        if(pasajeros > capacidad_restante){
            avion.setPasajeros(capacidad_restante); //Completo avion con lo que falte
            this.pasajeros -= capacidad_restante;
            uiUpdater.updatePasajeros(pasajeros);
        }
        else{
            avion.setPasajeros(pasajeros);
            this.pasajeros = 0; //No quedaran pasajeros en el aeropuerto
            uiUpdater.updatePasajeros(pasajeros);
        }
    }
}

finally{
    pasajerosLock.unlock();
    Random random = new Random();
    Thread.sleep(1000 + random.nextInt(2000));
    System.out.println("El avion "+ avion.getIdAv()+ " ha embarcado a pasajeros");
    cEs.escribir("El avion "+ avion.getIdAv()+ " ha embarcado a pasajeros");
}
}

public void entrarAlAreaDeEstacionamiento (Avion avion) {
    //pausa.mirar();
    try {
        areaDeEstacionamientoLock.lock();
        if (areaDeEstacionamiento.add(avion)){
            nAvEstacionamiento++;
            uiUpdater.updateAreaDeEstacionamiento(areaDeEstacionamiento);
        }
    }
}

```

```

        System.out.println("El avion "+ avion.getIdAv() + " ha entrado al area de estacionamiento de "+this.nombre);
        clEs.escribir("El avion "+ avion.getIdAv() + " ha entrado al area de estacionamiento de "+this.nombre);
    }
    else{
        System.out.println("ERROR al meter al avion" + avion.getIdAv() + " al area de estacionamiento");
    }
}catch(Exception e){
    e.printStackTrace();
}
finally {
    areaDeEstacionamientoLock.unlock();
}
}

public void salirDelAreaDeEstacionamiento (Avion a) {
    pausa.mirar();
    try {
        areaDeEstacionamientoLock.lock();
        if (areaDeEstacionamiento.isEmpty()){
            System.out.println ("El area de Estacionamiento está vacía");
        }
        else{
            areaDeEstacionamiento.remove(a);
            nAvEstacionamiento--;
            System.out.println("El avion "+ a.getIdAv() + " ha salido del area de estacionamiento de "+this.nombre);
            clEs.escribir("El avion "+ a.getIdAv() + " ha salido del area de estacionamiento de "+this.nombre);
            uiUpdater.updateAreaDeEstacionamiento(areaDeEstacionamiento);
        }
    }
    finally {
        areaDeEstacionamientoLock.unlock();
    }
}
}

```

```

public void esperarAPuertaDeEmbarque (Avion a) {
    pausa.mirar();
    try{
        control5Embarque.acquire(); //controla maximo hay 5 puertas de embarque
        control6Puertas.acquire(); // controla maximo 6 puertas de ambos tipos
        arrayEmbarque.acquire();//controla array de aviones en puertas de embarque/desembarque
        salirDelAreaDeEstacionamiento (a);
        System.out.println("El avion "+ a.getIdAv()+ " ha salido del area de estacionamiento de " + this.nombre);
        clEs.escribir("El avion "+ a.getIdAv()+ " ha salido del area de estacionamiento de " + this.nombre);
        int pos = uiUpdater.primerHuecoEnArray(puertasDeEmbarque);
        puertasDeEmbarque[pos] = a;
        uiUpdater.updatePuertasDeEmbarque(a,puertasDeEmbarque);
        System.out.println("El avion "+ a.getIdAv()+ " ha entrado al gate de embarque de "+ this.nombre + " " + pos);
        clEs.escribir("El avion "+ a.getIdAv()+ " ha entrado al gate de embarque de "+ this.nombre + " " + pos);
    }
    catch(Exception E) {
        E.printStackTrace();
    }
    finally{
        arrayEmbarque.release();
    }
}

```

```

public void esperarAPuertaDeDesembarque (Avion a) {
    pausa.mirar();
    try{
        control5Desembarque.acquire(); //controla maximo hay 5 puertas de desembarque
        control6Puertas.acquire(); // controla maximo 6 puertas de ambos tipos
        arrayEmbarque.acquire();//controla array de aviones en puertas de embarque/desembarque
        salirAreaDeRodaje(a);
        System.out.println("El avion "+ a.getIdAv()+ " ha salido del area de rodaje de " + this.nombre);
        clEs.escribir("El avion "+ a.getIdAv()+ " ha salido del area de rodaje de " + this.nombre);
        int pos = uiUpdater.primerHuecoEnArray(puertasDeEmbarque);
        puertasDeEmbarque[pos] = a;
    }
}

```



```

        uiUpdater.updatePuertasDeDesembarque(puertasDeEmbarque,a );

        System.out.println("El avion "+ a.getIdAv()+ " ha entrado al gate de embarque de "+ this.nombre);

        clEs.escribir("El avion "+ a.getIdAv()+ " ha entrado al gate de embarque de "+ this.nombre);
    }
    catch(Exception E) {
        E.printStackTrace();
    }
    finally{
        arrayEmbarque.release();
    }
}

public void salirDePuertaDeEmbarque (Avion avion) {
    pausa.mirar();
    try {
        arrayEmbarque.acquire();

        int pos = uiUpdater.buscarAvion(avion, puertasDeEmbarque);

        puertasDeEmbarque[pos]=null;

        uiUpdater.updateSalidaPuertasDeEmbarque(avion);

        System.out.println("El avion "+ avion.getIdAv()+ " ha salido del gate de embarque de "+ this.nombre);

        clEs.escribir("El avion "+ avion.getIdAv()+ " ha salido del gate de embarque de "+ this.nombre);

    }

    catch(Exception E) {
        E.printStackTrace();
    }
    finally{
        arrayEmbarque.release();

        control6Puertas.release();

        control5Embarque.release();
    }
}

public void salirDePuertaDeDesembarque (Avion avion) {

```

```

    pausa.mirar();

    try {
        arrayEmbarque.acquire();

        int pos = uiUpdater.buscarAvion(avion, puertasDeEmbarque);

        puertasDeEmbarque[pos]=null;

        uiUpdater.updateSalidaPuertasDeEmbarque(avion);

        System.out.println("El avion "+ avion.getIdAv()+ " ha salido del gate de desembarque de "+ this.nombre);

        clEs.escribir("El avion "+ avion.getIdAv()+ " ha salido del gate de desembarque de "+ this.nombre);

    }


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

    finally{
        arrayEmbarque.release();
        control6Puertas.release();
        control5Desembarque.release();
    }
}

public void SaleDelHangar(Avion avion) {
    pausa.mirar();

    try {
        hangarLock.lock();

        if (hangar.contains(avion)) {
            hangar.remove(avion);

            nAvHangar--;

            uiUpdater.updateHangar(hangar);

            System.out.println("El avion "+ avion.getIdAv()+ " ha salido del hangar de "+ this.nombre);

            clEs.escribir("El avion "+ avion.getIdAv()+ " ha salido del hangar de "+ this.nombre);

        }

    }

    finally {
        hangarLock.unlock();
    }
}

```

```
}  
}
```

```
public void entrarAlHangar(Avion avion) {  
    //pausa.mirar();  
    try {  
  
        hangarLock.lock();  
        if (!hangar.contains(avion)) {  
            hangar.add(avion);  
            nAvHangar++;  
            uiUpdater.updateHangar(hangar);  
            System.out.println("El avion "+ avion.getIdAv()+ " ha entrado al hangar de "+ this.nombre);  
            clEs.escribir("El avion "+ avion.getIdAv()+ " ha entrado al hangar de "+ this.nombre);  
        }  
    }  
    finally {  
        hangarLock.unlock();  
    }  
}
```

```
public void salirAreaDeRodaje(Avion a){  
    pausa.mirar();  
    areaDeRodajeLock.lock();  
    try{  
        areaDeRodaje.remove(a);  
        nAvRodaje--;  
        clEs.escribir("El avion "+ a.getIdAv()+ " ha salido del area de rodaje de "+ this.nombre);  
    }catch(Exception e){  
        e.printStackTrace();  
    }finally{  
        areaDeRodajeLock.unlock();  
    }  
}
```

```

}

public void lleganPasajerosDeAutobus (Autobus autobus) {

    pausa.mirar();

    try {

        pasajerosLock.lock();

        this.pasajeros += autobus.getPasajeros();

        clEs.escribir("El autobus "+ autobus.getIdentificacion()+" ha dejado "+autobus.getPasajeros()+ " en
"+this.nombre);

        uiUpdater.updatePasajeros(pasajeros);

        uiUpdater.updatebus1(autobus);

        autobus.setPasajeros(0);


    }

    finally{

        pasajerosLock.unlock();

    }

}

```

```

public void desembarcarPasajeros (Avion a) {

    pausa.mirar();

    pasajerosLock.lock();

    try{

        pasajeros+=a.getPasajeros();

        uiUpdater.updatePasajeros(pasajeros);

        a.setPasajeros(0);

        System.out.println("Los pasajeros del avion "+ a.getIdAv()+ " han desembarcado");

        clEs.escribir("Los pasajeros del avion "+ a.getIdAv()+ " han desembarcado");

    }

    finally{

        pasajerosLock.unlock();

    }

}

```

```

public void subenPasajerosAlBus (Autobus autobus) {

```

```

    pausa.mirar();
    if (pasajeros > 0) {
        try {
            pasajerosLock.lock();

            Random rand = new Random();

            int pasajerosQueSalen;

            if (pasajeros < 50){
                pasajerosQueSalen = rand.nextInt(pasajeros);
            }
            else{
                pasajerosQueSalen = rand.nextInt(50);
            }

            this.pasajeros -= pasajerosQueSalen;
            uiUpdater.updatePasajeros(pasajeros);
            autobus.setPasajeros(pasajerosQueSalen);
            clEs.escribir("Han subido "+ pasajerosQueSalen + " al autobus " +autobus.getIdentificacion());
            uiUpdater.updatebus2(autobus);
        }
        finally{
            pasajerosLock.unlock();
        }
    }
}

```

@Override

```

public void cerrarPista(int nPista)throws RemoteException{
    try{
        pistasAbiertasLock.lock();//unico que accede all array de booleanos pistas abiertas
        pistasAbiertas[nPista-1] = false;
    }catch(Exception e){
        e.printStackTrace();
    }finally{

```

```

        pistasAbiertasLock.unlock();
    }
}

```

@Override

```

public void abrirPista(int nPista)throws RemoteException{
    try{
        pistasAbiertasLock.lock();//unico que accede all array de booleanos pistas abiertas
        pistasAbiertas[nPista-1] = true;
    }catch(Exception e){
        e.printStackTrace();
    }finally{
        pistasCerradas.signalAll(); //ya hay pistas abiertas
        pistasAbiertasLock.unlock();
    }
}

```

```

public boolean pistasAbiertas(){ //comprueba si hay pista abiertas
    boolean abiertas = false;
    int cont = 0;
    while((!abiertas) && cont<4){
        if(pistasAbiertas[cont]){
            abiertas = true;
        }
        cont++;
    }
    return abiertas;
}

```

/\*\*

\* Get the value of pasajeros

\*

\* @return the value of pasajeros

\*/

```
public int getPasajeros() throws RemoteException{  
    return pasajeros;  
}
```

```
public int getAvionesEnHangar()throws RemoteException{  
    return nAvHangar;  
}
```

```
public int getAvionesEnTaller()throws RemoteException{  
    return nAvTaller;  
}
```

```
public int getAvionesEnAreaEstacionamiento()throws RemoteException{  
    return nAvEstacionamiento;  
}
```

```
public int getAvionesEnAreaRodaje()throws RemoteException{  
    return nAvRodaje;  
}
```

```
public String getAerovia1()throws RemoteException{  
    String cadena1 = "";  
    for (int i = 0; i<aeroviaSalida.size(); i++) {  
        cadena1 += aeroviaSalida.get(i).getIdAv() + ", ";  
    }  
    return cadena1;  
}
```

```
public String getAerovia2()throws RemoteException{  
    String cadena2 = "";  
    for (int i = 0; i<aeroviaLlegada.size(); i++) {  
        cadena2 += aeroviaLlegada.get(i).getIdAv() + ", ";  
    }  
}
```

```
}  
    return cadena2;  
}
```

```
/**  
 * Set the value of pasajeros  
 *  
 * @param pasajeros new value of pasajeros  
 */  
public void setPasajeros(int pasajeros) {  
    this.pasajeros = pasajeros;  
}
```

```
public ArrayList<Avion> getHangar() {  
    return hangar;  
}
```

```
public ArrayList<Avion> getAreaDeRodaje() {  
    return areaDeRodaje;  
}
```

```
public ArrayList<Avion> getAreaDeEstacionamiento() {  
    return areaDeEstacionamiento;  
}
```

```
public Avion[] getPuertasDeEmbarque() {  
    return puertasDeEmbarque;  
}
```

```
public Avion[] getPistasDeDespegue() {  
    return pistasDeDespegue;  
}
```



```
public ArrayList<Avion> getAeroviaSalida() {  
    return aeroviaSalida;  
}
```

```
public ArrayList<Avion> getAeroviaLlegada() {  
    return aeroviaLlegada;  
}
```

```
public ArrayList<Avion> getTaller() {  
    return taller;  
}
```

```
public Lock getHangarLock() {  
    return hangarLock;  
}
```

```
public Lock getPasajerosLock() {  
    return pasajerosLock;  
}
```

```
public Lock getAreaDeRodajeLock() {  
    return areaDeRodajeLock;  
}
```

```
public Lock getAreaDeEstacionamientoLock() {  
    return areaDeEstacionamientoLock;  
}
```

```
/*
```

```
public Lock getPistasLock() {  
    return pistasLock;  
}
```

```
*/
```

```
public Lock getAeroViasLock() {  
    return aeroViasLock;  
}
```

```
}

public Semaphore getPistasSem() {
    return pistasSem;
}

public Semaphore getTallerSe() {
    return tallerSe;
}

public Semaphore getPuertaTaller() {
    return puertaTaller;
}

public Semaphore getControl6Puertas() {
    return control6Puertas;
}

public Semaphore getControl5Embarque() {
    return control5Embarque;
}

public Semaphore getControl5Desembarque() {
    return control5Desembarque;
}

public Semaphore getArrayEmbarque() {
    return arrayEmbarque;
}

public String getNombre() {
    return nombre;
}
}
```

### **Autobús.java:**

```
package vidrio.vidri.pecl;

import java.util.Random;

/**
 *
 * @author gonza
 */
public class Autobus extends Thread{

    private String id;
    private int capacidad;
    private int pasajeros;
    private Aeropuerto aeropuerto;

    // Constructor
    public Autobus (String id, int capacidad, Aeropuerto aeropuerto) {

        this.capacidad=capacidad;
        this.id= id;
        this.pasajeros=0;
        this.aeropuerto = aeropuerto;
    }

    public void run() {
        while (true) {
            try{
                Random rand = new Random();
                int numAlea = rand.nextInt(3000);
                Thread.sleep(2000+numAlea); // Espera en el centro de la ciudad
                System.out.println("El autobus "+this.id+ " esta esperando en el centro de la ciudad para ir a " +
aeropuerto.getNombre());
                int pasCiudad = rand.nextInt(capacidad);
                pasajeros+=pasCiudad;
                System.out.println("El autobus "+this.id+ " está de camino al aeropuerto "+aeropuerto.getNombre());
```

```

        Thread.sleep(rand.nextInt(5000)+5000); // De camino al aeropuerto
        aeropuerto.lleganPasajerosDeAutobus(this); // LLega al aeropuerto
        System.out.println("El autobus "+this.id+ " ha llegado al aeropuerto");
        Thread.sleep(rand.nextInt(2000)+3000); // Espera mientras bajan los pasajeros
        System.out.println("El autobus "+this.id+ " ha descargado los pasajeros");
        aeropuerto.subenPasajerosAlBus(this);
        System.out.println("El autobus "+this.id+ " ha cogido pasajeros y va a la ciudad");
        Thread.sleep(rand.nextInt(5000)+5000);
        System.out.println("El autobus "+this.id+ " ha descargado los pasajeros");
        this.pasajeros=0;
    }
    catch(Exception e) {}
}

public String getIdentificacion() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public int getCapacidad() {
    return capacidad;
}

public void setCapacidad(int capacidad) {
    this.capacidad = capacidad;
}

public int getPasajeros() {
    return pasajeros;
}

```

```
public void setPasajeros(int pasajeros) {  
    this.pasajeros = pasajeros;  
}  
}
```

### **Avión.java:**

```
package vidrio.vidri.pecl;  
  
import java.io.Serializable;  
import java.util.Random;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
/**  
 *  
 * @author gonza  
 */  
  
public class Avion extends Thread implements Serializable{  
    private String id;  
    private int capacidad;  
    private int pasajeros;  
    private int numeroDeVuelos;  
    private Aeropuerto aeropuertoLinkeado;  
    private Aeropuerto aeropuertoSecundario;  
  
    public Avion (String id, int capacidad, Aeropuerto aOrigen, Aeropuerto aDestino) {  
        this.id = id;  
        this.capacidad = capacidad;  
        this.pasajeros = 0;  
        this.aeropuertoLinkeado=aOrigen;  
        this.aeropuertoSecundario=aDestino;  
        this.numeroDeVuelos=0;  
    }  
}
```

```

public void run() {
    while(true){
        try {
            Random rand = new Random ();
            aeropuertoLinkeado.entrarAlHangar(this);
            Thread.sleep (1000 + rand.nextInt(2000));
            aeropuertoLinkeado.SaleDelHangar(this);
            aeropuertoLinkeado.entrarAlAreaDeEstacionamiento(this);
            Thread.sleep (1000 + rand.nextInt(2000));
            aeropuertoLinkeado.esperarAPuertaDeEmbarque(this);
            aeropuertoLinkeado.embarcarPasajeros(this);
            int i = 0;
            while ((i<3)&&(this.pasajeros < this.capacidad)){
                System.out.println("El avion " + this.id + " no se ha llenado, espera a llenar mas");
                aeropuertoLinkeado.embarcarPasajeros(this);
                Thread.sleep (1000 + rand.nextInt(2000));
                Thread.sleep (1000 + rand.nextInt(4000));
                i++;
            }
            aeropuertoLinkeado.salirDePuertaDeEmbarque(this);
            aeropuertoLinkeado.accesoAlAreaDeRodaje(this);
            Thread.sleep (1000 + rand.nextInt(4000));
            aeropuertoLinkeado.AccedeAPistaDespegue(this);
            Thread.sleep (1000 + rand.nextInt(2000));
            aeropuertoLinkeado.despegaAvion(this);
            aeropuertoLinkeado.avionEnElAire(this);
            Thread.sleep (15000 + rand.nextInt(15000));
            cambiarAeropuerto();
            aeropuertoLinkeado.SolicitaPistaAterrizaje(this); //Mete al avion en la pista
            Thread.sleep (1000 + rand.nextInt(4000)); //Aterrizaje
            aeropuertoLinkeado.avionEnTierra(this);
            aeropuertoLinkeado.salirDePista(this);
            aeropuertoLinkeado.accesoAlAreaDeRodaje(this);
            Thread.sleep (3000 + rand.nextInt(2000)); //camino entre pista

```

```

aeropuertoLinkeado.esperarAPuertaDeDesembarque(this);
Thread.sleep (1000 + rand.nextInt(4000)); //Desembarca pasajeros
aeropuertoLinkeado.desembarcarPasajeros(this);
aeropuertoLinkeado.salirDePuertaDeDesembarque(this);
aeropuertoLinkeado.entrarAlAreaDeEstacionamiento(this);
Thread.sleep (1000 + rand.nextInt(4000)); //piloto hace comprobaciones
aeropuertoLinkeado.vaAlTaller(this);
int randomNumber = rand.nextInt(1);
if (randomNumber == 1) {
    aeropuertoLinkeado.entrarAlHangar(this);
    Thread.sleep (15000 + rand.nextInt(15000));
}
} catch (InterruptedException ex) {
    Logger.getLogger(Avion.class.getName()).log(Level.SEVERE, null, ex);
}
}
}

public void cambiarAeropuerto () {
    Aeropuerto oldSecundario = this.aeropuertoSecundario;
    this.aeropuertoSecundario =aeropuertoLinkeado;
    this.aeropuertoLinkeado = oldSecundario;

}

public Aeropuerto getAeropuertoLinkeado() {
    return aeropuertoLinkeado;
}

public void setAeropuertoLinkeado(Aeropuerto aeropuertoLinkeado) {
    this.aeropuertoLinkeado = aeropuertoLinkeado;
}

public int getNumeroDeVuelos() {
    return numeroDeVuelos;
}

```

```

    }

    public void annadirVuelo() {
        this.numeroDeVuelos +=1;
    }


    public void setNumeroDeVuelos(int numeroDeVuelos) {
        this.numeroDeVuelos = numeroDeVuelos;
    }


    public int getPasajeros() {
        return pasajeros;
    }


    public void setPasajeros(int pasajeros) {
        this.pasajeros = pasajeros;
    }


    public int getCapacidad() {
        return capacidad;
    }


    public String getIdAv() {
        return id;
    }
}

```

#### **clienteHilo.java:**

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package vidri.vidri.pecl;


import java.rmi.AccessException;

```



```

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.ArrayList;

import javax.swing.JOptionPane;

import vidrio.vidri.interfaz.panelCliente;

import vidrio.vidri.interfaz.pantallaCliente;


/**
 *
 * @author gonza
 */
public class ClienteHilo extends Thread{

    pantallaCliente pC;

    InterfazRemota i1;

    InterfazRemota i2;

    public ClienteHilo (pantallaCliente pC, InterfazRemota i1, InterfazRemota i2) {

        this.pC=pC;

        this.i1=i1;

        this.i2=i2;

    }


    public void run() {

        try{

            while(true){

                int pasajerosA1 = i1.getPasajeros();

                int contHangarA1 = i1.getAvionesEnHangar();

                int contTallerA1 = i1.getAvionesEnTaller();

                int contEstacionamientoA1 = i1.getAvionesEnAreaEstacionamiento();

                int contRodajeA1 = i1.getAvionesEnAreaRodaje();

                panelCliente panelA1 = pC.getPanelCliente3();

                panelA1.setPasajerosText(""+pasajerosA1);

```

```
panelA1.setHangarText(""+contHangarA1);  
panelA1.setTallerText(""+contTallerA1);  
panelA1.setEstacionamientoText(""+contEstacionamientoA1);  
panelA1.setRodajeText(""+contRodajeA1);
```

```
int pasajerosA2 = i2.getPasajeros();  
int contHangarA2 = i2.getAvionesEnHangar();  
int contTallerA2 = i2.getAvionesEnTaller();  
int contEstacionamientoA2 = i2.getAvionesEnAreaEstacionamiento();  
int contRodajeA2 = i2.getAvionesEnAreaRodaje();
```

```
panelCliente panelA2 = pC.getPanelCliente4();  
panelA2.setPasajerosText(""+pasajerosA2);  
panelA2.setHangarText(""+contHangarA2);  
panelA2.setTallerText(""+contTallerA2);  
panelA2.setEstacionamientoText(""+contEstacionamientoA2);  
panelA2.setRodajeText(""+contRodajeA2);
```

```
String aerovia1 = i1.getAerovia1();  
String aerovia2 = i1.getAerovia2();  
pC.setTextAerovia1(aerovia1);  
pC.setTextAerovia2(aerovia2);
```

```
//Thread.sleep(1000); //hace una consulta cada 1 segundo
```

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

### **genHilosAvion.java:**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package vidri.vidri.pecl;

import java.util.ArrayList;
import java.util.Random;

/**
 *
 * @author enriq
 */
public class GenHilosAvion extends Thread{

    //Atributos

    private Aeropuerto a1, a2;

    private int nAviones;

    private Pausa pausa;

    //Constructor

    public GenHilosAvion(Aeropuerto a1, Aeropuerto a2, int nAviones, Pausa pausa){

        this.a1 = a1;

        this.a2 = a2;

        this.nAviones = nAviones;

        this.pausa = pausa;

    }

    //Metodos

    public void run(){

        Random rand = new Random();

        for (int i = 0; i<nAviones; i++){

            String str = generarCaracteres();

            pausa.mirar();

            if (i<1000){
```

```

        str+="0";
    if (i<100){
        str+="0";
        if(i<10){
            str+="0";
        }
    }
}

if(i%2==0){

    Avion av = new Avion (str+i, 100 + rand.nextInt(200), a1, a2);
    av.start();
}
else{
    Avion av = new Avion (str+i, 100 + rand.nextInt(200), a2, a1);
    av.start();
}
try{
    Thread.sleep(1000+rand.nextInt(2000));
}catch(InterruptedException e){
    e.printStackTrace();
}
}
}

public static String generarCaracteres() {
    char[] alfabeto = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".toCharArray();
    Random random = new Random();
    int indice1 = random.nextInt(26); // Para la primera letra
    int indice2 = random.nextInt(26); // Para la segunda letra
    char letra1 = alfabeto[indice1];
    char letra2 = alfabeto[indice2];
    return "" + letra1 + letra2 + "-";
}
}

```

## **genHilosBus.java:**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package vidri.vidri.pecl;

import java.util.ArrayList;
import java.util.Random;

/**
 *
 * @author enriq
 */
public class GenHilosBus extends Thread{
    //Atributos
    private Aeropuerto aeropuerto1;
    private Aeropuerto aeropuerto2;
    private int nAutobuses;
    private Pausa pausa;
    //Constructor
    public GenHilosBus(Aeropuerto a1, Aeropuerto a2, int nAutobuses, Pausa pausa){
        this.aeropuerto1 = a1;
        this.aeropuerto2 = a2;
        this.nAutobuses = nAutobuses;
        this.pausa = pausa;
    }
    //metodos
    public void run(){
        Random rand = new Random();
        for (int i = 0; i<nAutobuses; i++){
            pausa.mirar();
            String str = "B-";
            if (i<1000){
```

```

        str+="0";
    if (i<100){
        str+="0";
        if(i<10){
            str+="0";
        }
    }
}

if(i%2==0){
    Autobus ab = new Autobus (str+i, 50, aeropuerto1);
    ab.start();
}
else{
    Autobus ab = new Autobus (str+i, 50, aeropuerto2);
    ab.start();
}

try{
    Thread.sleep(500+rand.nextInt(500));
}catch(InterruptedException e){
    e.printStackTrace();
}

}

}

}

```

### **InterfazRemota.java:**

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to edit this template
 */
package vidrio.vidri.pecl;

```

```

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.ArrayList;

/**
 *
 * @author enriq
 */
public interface InterfazRemota extends Remote{

    //incluir metodos del obj compartido (aeropuerto)

    public int getPasajeros()throws RemoteException;

    public int getAvionesEnHangar()throws RemoteException;

    public int getAvionesEnTaller()throws RemoteException;

    public int getAvionesEnAreaEstacionamiento()throws RemoteException;

    public int getAvionesEnAreaRodaje()throws RemoteException;

    public String getAerovia1()throws RemoteException;

    public String getAerovia2()throws RemoteException;

    public void cerrarPista(int nPista) throws RemoteException;

    public void abrirPista(int nPista) throws RemoteException;

    public String getNombre() throws RemoteException;

```

### **pausa.java:**

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package vidri.vidri.pecl;

/**
 *
 * @author enriq
 */
public class Pausa {

    //Atributos

```

```
private boolean parado = false;

//Metodos

public synchronized void mirar(){

    try{

        while(parado){

            wait();

        }

    }catch(InterruptedException e){

        e.printStackTrace();

    }finally{

    }

}
```

```
public synchronized void pausar(){

    try{

        parado = true;

    }catch(Exception e){

        e.printStackTrace();

    }

    finally{

    }

}
```

```
public synchronized void reanudar(){

    try{

        parado = false;

        notifyAll();

    }catch(Exception e){

        e.printStackTrace();

    }

    finally{

    }

}
```



```

    }
}

public boolean getParado(){
    return parado;
}
}

```

### **UIUpdate.java:**

```

package vidrio.vidri.pecl;

import java.util.ArrayList;

/**
 *
 * @author gonza
 */
public interface UIUpdater {

    public void updateHangar (ArrayList<Avion> hangar);
    public void updatebus1 (Autobus bus);
    public void updatebus2 (Autobus bus);
    public void updateTaller (ArrayList<Avion> taller);
    public void updateAreaDeEstacionamiento (ArrayList<Avion> areaEstacionamiento);
    public void updatePuertasDeEmbarque (Avion[] puertasDeEmbarque);
    public void updatePuertasDeDesembarque (Avion[] puertasDeEmbarque);
    public void updateSalidaPuertasDeEmbarque (Avion a);
    public void updateAreaDeRodaje (ArrayList<Avion> areaDeRodaje);
    public void updatePistasAterrizaje (Avion[] pistas);
    public void updatePistasDespegue (Avion[] pistas);
    public void updateSalirDePista (Avion avion);
    public void setAeropuerto (Aeropuerto a);
    public int buscarAvion(Avion a, Avion[] array);
    public int primerHuecoEnArray (Avion[] array);
}

```

### **claseEscritor.java:**

```
package vidrio.vidri.pecl;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

/**
 *
 * @author gonza
 */
public class claseEscritor {
    private final String rutaArchivo;
    private boolean append = false;

    public claseEscritor(String rutaArchivo) {
        this.rutaArchivo = rutaArchivo;
    }

    public synchronized void escribir(String texto) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(rutaArchivo, append))) {
            LocalDateTime now = LocalDateTime.now();
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
            String formattedDateTime = now.format(formatter);
            writer.write(formattedDateTime + texto);
            writer.newLine(); // Añade una nueva línea después de cada entrada
            append=true;
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

## **pantallaPrincipal.java:**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */
package vidri.vidri.interfaz;

import java.rmi.Naming;
import java.rmi.registry LocateRegistry;
import java.rmi.registry.Registry;
import java.util.ArrayList;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import vidri.vidri.interfaz.panelAeropuerto;
import vidri.vidri.pecl.Aeropuerto;
import vidri.vidri.pecl.Avion;
import vidri.vidri.pecl.GenHilosAvion;
import vidri.vidri.pecl.GenHilosBus;
import vidri.vidri.pecl.Pausa;
import vidri.vidri.pecl.claseEscritor;

/**
 *
 * @author gonza
 */
public class pantallaPrincipal extends javax.swing.JFrame {

    ArrayList<Avion> aerovia1;
    ArrayList<Avion> aerovia2;
    Pausa pausa;

    /**
     * Creates new form pantallaPrincipal
     * @param p1
     */
    public pantallaPrincipal(ArrayList<Avion> aerovia1, ArrayList<Avion> aerovia2, Pausa pausa) {
```

```
this.aerovia1 =aerovia1;

this.aerovia2 = aerovia2;

this.pausa = pausa;

initComponents();
}
```

```
public panelAeropuerto getPanelAeropuerto1() {
    return panelAeropuerto1;
}
```

```
public panelAeropuerto getPanelAeropuerto2() {
    return panelAeropuerto2;
}
```

```
public void updateAerovia1 (ArrayList<Avion> aerovia){
    String str = "";
    for (int i = 0; i<aerovia.size(); i++) {
        str += aerovia.get(i).getIdAv() + " , ";
    }
    aerovia1Text.setText(str);
}
```

```
public void updateAerovia2 (ArrayList<Avion> aerovia){
    String str = "";
    for (int i = 0; i<aerovia.size(); i++) {
        str += aerovia.get(i).getIdAv() + " , ";
    }
    aerovia2Text.setText(str);
}
```

```
/**
```

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

```
*/
```

```
@SuppressWarnings("unchecked")
```

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">
```

```
private void initComponents() {
```

```
    panelAeropuerto1 = new vidrio.vidri.interfaz.panelAeropuerto();
```

```
    panelAeropuerto2 = new vidrio.vidri.interfaz.panelAeropuerto();
```

```
    aerovia1Text = new javax.swing.JTextField();
```

```
    aerovia2Text = new javax.swing.JTextField();
```

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

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

```
    pausarButton = new javax.swing.JButton();
```

```
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
    setTitle("AeroJava");
```

```
    aerovia1Text.setEditable(false);
```

```
    aerovia1Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
    aerovia2Text.setEditable(false);
```

```
    aerovia2Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
    jLabel1.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
    jLabel1.setText("Madrid-Barcelona:");
```

```
    jLabel2.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
    jLabel2.setText("Barcelona-Madrid:");
```

```
    pausarButton.setText("Pausar/Reanudar");
```

```
    pausarButton.addActionListener(new java.awt.event.ActionListener() {
```

```
        public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            pausarButtonActionPerformed(evt);
```

```
        }
```

```
    });
```

```

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(panelAeropuerto1, javax.swing.GroupLayout.PREFERRED_SIZE, 618,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(panelAeropuerto2, javax.swing.GroupLayout.PREFERRED_SIZE, 617,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(layout.createSequentialGroup()
                    .addGap(21, 21, 21)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 121,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 121,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(57, 57, 57)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(aerovia1Text, javax.swing.GroupLayout.DEFAULT_SIZE, 1095, Short.MAX_VALUE)
                        .addComponent(aerovia2Text)))
                .addGroup(layout.createSequentialGroup()
                    .addGap(584, 584, 584)
                    .addComponent(pausarButton)))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(10, 10, 10)
            .addComponent(pausarButton)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

```

```

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

            .addComponent(panelAeropuerto2,                                javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(panelAeropuerto1,                                javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

        .addGap(18, 18, 18)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

            .addComponent(aerovia1Text, javax.swing.GroupLayout.DEFAULT_SIZE, 30, Short.MAX_VALUE)

            .addComponent(jLabel1,                                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(aerovia2Text,                                javax.swing.GroupLayout.PREFERRED_SIZE,                                30,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jLabel2,                                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

    );

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

```

```

private void pausarButtonActionPerformed(java.awt.event.ActionEvent evt) {
    if(pausa.getParado()){
        pausa.reanudar();
    }
    else{
        pausa.pausar();
    }
}

```

```

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */

```

```

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
 * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
 */

try {
    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (ClassNotFoundException ex) {
    java.util.logging.Logger.getLogger(pantallaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
} catch (InstantiationException ex) {
    java.util.logging.Logger.getLogger(pantallaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
} catch (IllegalAccessException ex) {
    java.util.logging.Logger.getLogger(pantallaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
    java.util.logging.Logger.getLogger(pantallaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
}

//</editor-fold>

```

```

/* Create and display the form */

```

```

java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        int nAutobuses = 4000;
        int nAviones = 8000;
        Pausa pausa = new Pausa();
        Lock aeroViasLock = new ReentrantLock();
        ArrayList<Avion> MadridBarcelona = new ArrayList<>();
        ArrayList<Avion> BarcelonaMadrid = new ArrayList<>();
        pantallaPrincipal p = new pantallaPrincipal(MadridBarcelona,BarcelonaMadrid, pausa);
    }
}

```



```

        claseEscritor clEs = new claseEscritor ("log.txt");

        try{

            Aeropuerto Adolfo_Suarez = new Aeropuerto (MadridBarcelona,BarcelonaMadrid, "Adolfo Suarez",
aeroViasLock,p.getPanelAeropuerto1()),p, pausa,clEs);

            Aeropuerto El_Pratt = new Aeropuerto (BarcelonaMadrid,MadridBarcelona,"El Pratt",
aeroViasLock,p.getPanelAeropuerto2()),p, pausa,clEs);

            Registry reg = LocateRegistry.createRegistry(5000);

            reg.rebind("aeropuerto1", Adolfo_Suarez);

            reg.rebind("aeropuerto2", El_Pratt);

            GenHilosBus buses = new GenHilosBus(Adolfo_Suarez, El_Pratt, nAutobuses, pausa);

            GenHilosAvion aviones = new GenHilosAvion(Adolfo_Suarez, El_Pratt, nAviones, pausa);

            buses.start();

            aviones.start();

        }catch(Exception e){

            e.printStackTrace();

        }

        p.setVisible(true);

    }

});

}

```

```

// Variables declaration - do not modify

private javax.swing.JTextField aerovia1Text;

private javax.swing.JTextField aerovia2Text;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private vidrio.vidri.interfaz.panelAeropuerto panelAeropuerto1;

private vidrio.vidri.interfaz.panelAeropuerto panelAeropuerto2;

private javax.swing.JButton pausarButton;

// End of variables declaration

}

```

## **pantallaCliente.java:**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */
package vidri.vidri.interfaz;

import java.rmi.AccessException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.Registry;
import java.util.ArrayList;
import vidri.vidri.pecl.Avion;
import vidri.vidri.pecl.InterfazRemota;
import java.rmi.registry.LocateRegistry;
import javax.swing.JOptionPane;
import vidri.vidri.pecl.ClienteHilo;

/**
 *
 * @author gonza
 */
public class pantallaCliente extends javax.swing.JFrame {

    /**
     * Creates new form pantallaCliente
     */
    Registry reg;
    InterfazRemota i1;
    InterfazRemota i2;

    public pantallaCliente() {
```

```

try{
    this.reg = LocateRegistry.getRegistry("127.0.0.1",5000);
    try{
        this.i1 = (InterfazRemota) reg.lookup("aeropuerto1");
        this.i2= (InterfazRemota) reg.lookup("aeropuerto2");
        initComponents();
        panelCliente3.setjPanelNombreAeropuerto(i1.getNombre());
        panelCliente3.setAeropuerto(i1);
        panelCliente4.setjPanelNombreAeropuerto(i2.getNombre());
        panelCliente4.setAeropuerto(i2);
    }
    catch (NotBoundException ex){
        error();
    }
    catch(AccessException ex) {
        error();
    }
}
catch (RemoteException e) {
    error();
}
}

```

```

public Registry getReg() {
    return reg;
}

```

```

public InterfazRemota getI1() {
    return i1;
}

```

```

public InterfazRemota getI2() {
    return i2;
}

```

```
public panelCliente getPanelCliente3() {  
    return panelCliente3;  
}
```

```
public panelCliente getPanelCliente4() {  
    return panelCliente4;  
}
```

```
public void setTextAerovia1(String str){  
    aerovia1Text.setText(str);  
}
```

```
public void setTextAerovia2(String str){  
    aerovia2Text.setText(str);  
}
```

```
public void error() {  
    JOptionPane.showMessageDialog(this, "El servidor no está disponible!");  
    System.exit(0);  
}
```

```
/**
```

```
 * This method is called from within the constructor to initialize the form.
```

```
 * WARNING: Do NOT modify this code. The content of this method is always
```

```
 * regenerated by the Form Editor.
```

```
 */
```

```
@SuppressWarnings("unchecked")
```

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">
```

```
private void initComponents() {
```

```
    panelCliente3 = new vidrio.vidri.interfaz.panelCliente();
```

```
    panelCliente4 = new vidrio.vidri.interfaz.panelCliente();
```

```
    aerovia1Text = new javax.swing.JTextField();
```

```
    aerovia2Text = new javax.swing.JTextField();
```

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

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

```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
aerovia1Text.setEditable(false);
```

```
aerovia1Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
aerovia2Text.setEditable(false);
```

```
aerovia2Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
jLabel1.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
jLabel1.setText("Madrid-Barcelona:");
```

```
jLabel2.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
jLabel2.setText("Barcelona-Madrid:");
```

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

```
getContentPane().setLayout(layout);
```

```
layout.setHorizontalGroup(
```

```
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
        .addGroup(layout.createSequentialGroup()
```

```
            .addGap(57, 57, 57)
```

```
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
                .addGroup(layout.createSequentialGroup()
```

```
                    .addComponent(panelCliente3, javax.swing.GroupLayout.PREFERRED_SIZE,  
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
                    .addComponent(panelCliente4, javax.swing.GroupLayout.PREFERRED_SIZE,  
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addGroup(layout.createSequentialGroup()
```

```
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 121,  
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 121,  
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addGap(57, 57, 57)
```

```

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addComponent(aerovia1Text, javax.swing.GroupLayout.DEFAULT_SIZE, 820, Short.MAX_VALUE)
            .addComponent(aerovia2Text))))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);

layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(panelCliente4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(panelCliente3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addComponent(aerovia1Text)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(aerovia2Text, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addContainerGap())
        );

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

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */

```

```

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
 * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
 */
try {
    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (ClassNotFoundException ex) {
    java.util.logging.Logger.getLogger(pantallaCliente.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
} catch (InstantiationException ex) {
    java.util.logging.Logger.getLogger(pantallaCliente.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
} catch (IllegalAccessException ex) {
    java.util.logging.Logger.getLogger(pantallaCliente.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
    java.util.logging.Logger.getLogger(pantallaCliente.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
}

//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        pantallaCliente pCliente = new pantallaCliente();
        pCliente.setVisible(true);
        ClienteHilo cH = new ClienteHilo(pCliente, pCliente.getI1(), pCliente.getI2());
        cH.start();
    }
});

```

```

}

// Variables declaration - do not modify
private javax.swing.JTextField aerovia1Text;
private javax.swing.JTextField aerovia2Text;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private vidri.vidri.interfaz.panelCliente panelCliente3;
private vidri.vidri.interfaz.panelCliente panelCliente4;

// End of variables declaration
}

```

### **panelCliente.java:**

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JPanel.java to edit this template
 */
package vidri.vidri.interfaz;

import javax.swing.JTextField;
import vidri.vidri.pecl.InterfazRemota;

/**
 *
 * @author gonza
 */
public class panelCliente extends javax.swing.JPanel {

    InterfazRemota i;

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

```



```

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

    jPanelNombreAeropuerto = new javax.swing.JTextField();
    hangarLabel = new javax.swing.JLabel();
    pasajerosText = new javax.swing.JTextField();
    tallerLabel = new javax.swing.JLabel();
    tallerText = new javax.swing.JTextField();
    estacionamientoLabel = new javax.swing.JLabel();
    estacionamientoText = new javax.swing.JTextField();
    rodajeLabel = new javax.swing.JLabel();
    rodajeText = new javax.swing.JTextField();
    pista1Label = new javax.swing.JLabel();
    pista2Label = new javax.swing.JLabel();
    pista3Label = new javax.swing.JLabel();
    pista4Label = new javax.swing.JLabel();
    jButtonPista1 = new javax.swing.JToggleButton();
    jButtonPista2 = new javax.swing.JToggleButton();
    jButtonPista3 = new javax.swing.JToggleButton();
    jButtonPista4 = new javax.swing.JToggleButton();
    pasajerosLabel = new javax.swing.JLabel();
    hangarText1 = new javax.swing.JTextField();

    jPanelNombreAeropuerto.setEditable(false);
    jPanelNombreAeropuerto.setFont(new java.awt.Font("Arial", 1, 24)); // NOI18N
    jPanelNombreAeropuerto.setHorizontalAlignment(javax.swing.JTextField.CENTER);
    jPanelNombreAeropuerto.addActionListener(new java.awt.event.ActionListener() {

```

```
public void actionPerformed(java.awt.event.ActionEvent evt) {  
    jPanelNombreAeropuertoActionPerformed(evt);  
}  
});  
  
hangarLabel.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
hangarLabel.setText("Nº Aviones en Hangar:");  
  
pasajerosText.setEditable(false);  
pasajerosText.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
  
tallerLabel.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
tallerLabel.setText("Nº Aviones en Taller:");  
  
tallerText.setEditable(false);  
tallerText.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
  
estacionamientoLabel.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
estacionamientoLabel.setText("Nº Aviones en Área de Estacionamiento:");  
  
estacionamientoText.setEditable(false);  
estacionamientoText.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
  
rodajeLabel.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
rodajeLabel.setText("Nº Aviones en Área de rodaje:");  
  
rodajeText.setEditable(false);  
  
pista1Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
pista1Label.setText("Pista 1: ");  
  
pista2Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
pista2Label.setText("Pista 2:");
```

```
pista3Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
pista3Label.setText("Pista 3:");
```

```
pista4Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
pista4Label.setText("Pista 4:");
```

```
jButtonPista1.setText("Cerrar");
jButtonPista1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonPista1ActionPerformed(evt);
    }
});
```

```
jButtonPista2.setText("Cerrar");
jButtonPista2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonPista2ActionPerformed(evt);
    }
});
```

```
jButtonPista3.setText("Cerrar");
jButtonPista3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonPista3ActionPerformed(evt);
    }
});
```

```
jButtonPista4.setText("Cerrar");
jButtonPista4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonPista4ActionPerformed(evt);
    }
});
```

```

pasajerosLabel.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
pasajerosLabel.setText("Nº Pasajeros en Aeropuerto:");

hangarText1.setEditable(false);
hangarText1.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
this.setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(10, 10, 10)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jPanelNombreAeropuerto)
                .addGap(10, 10, 10)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(pista2Label, javax.swing.GroupLayout.PREFERRED_SIZE, 51, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(10, 10, 10)
                        .addComponent(pista1Label))
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(hangarLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(rodajeLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(tallerLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(estacionamientoLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(10, 10, 10)
                .addComponent(pasajerosLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(10, 10, 10))
        .addGroup(layout.createSequentialGroup()
            .addGap(10, 10, 10)
            .addComponent(hangarText1, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(rodajeText1, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(tallerText1, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(estacionamientoText1, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(pista2Text1, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(pista1Text1, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(pasajerosText1, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10))
    )
    .addGap(10, 10, 10)
    .addComponent(botonSalir, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
    .addGap(10, 10, 10));

```

```

        .addComponent(jButtonPista1)

        .addComponent(jButtonPista2))

        .addGap(0, 0, Short.MAX_VALUE)))

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(layout.createSequentialGroup())

            .addGap(45, 45, 45)

            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())

                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                        .addComponent(pasajerosText,
                            javax.swing.GroupLayout.Alignment.TRAILING,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 150, javax.swing.GroupLayout.PREFERRED_SIZE)

                        .addComponent(tallerText,
                            javax.swing.GroupLayout.Alignment.TRAILING,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 150, javax.swing.GroupLayout.PREFERRED_SIZE)

                        .addComponent(estacionamientoText,
                            javax.swing.GroupLayout.Alignment.TRAILING,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 150, javax.swing.GroupLayout.PREFERRED_SIZE)

                        .addComponent(rodajeText,
                            javax.swing.GroupLayout.Alignment.TRAILING,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 150, javax.swing.GroupLayout.PREFERRED_SIZE))

                    .addContainerGap())

                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())

                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

                        .addGroup(layout.createSequentialGroup())

                            .addComponent(pista4Label)

                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

                            .addComponent(jButtonPista4))

                        .addGroup(layout.createSequentialGroup())

                            .addComponent(pista3Label)

                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

                            .addComponent(jButtonPista3)))

                    .addGap(37, 37, 37)))

            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())

                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

                .addComponent(hangarText1,
                    javax.swing.GroupLayout.PREFERRED_SIZE, 150,
                    javax.swing.GroupLayout.PREFERRED_SIZE)

                .addContainerGap()))))

    );

    layout.setVerticalGroup(

```

```

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup())

.addContainerGap()

.addComponent(iPanelNombreAeropuerto,      javax.swing.GroupLayout.PREFERRED_SIZE,      27,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(pasajerosLabel,      javax.swing.GroupLayout.PREFERRED_SIZE,      25,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(pasajerosText,      javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(hangarLabel,      javax.swing.GroupLayout.PREFERRED_SIZE,      25,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(hangarText1,      javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(tallerLabel,      javax.swing.GroupLayout.PREFERRED_SIZE,      25,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(tallerText,      javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(estacionamientoLabel,      javax.swing.GroupLayout.PREFERRED_SIZE,      25,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(estacionamientoText,      javax.swing.GroupLayout.PREFERRED_SIZE,      25,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(rodajeLabel,      javax.swing.GroupLayout.PREFERRED_SIZE,      26,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(rodajeText))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

```

```

        .addComponent(pista3Label,                                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(pista1Label,                                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(jButtonPista1)

        .addComponent(jButtonPista3))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(pista2Label,                                javax.swing.GroupLayout.PREFERRED_SIZE,                25,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(pista4Label)

            .addComponent(jButtonPista2)

            .addComponent(jButtonPista4))

        .addGap(27, 27, 27))

    );
}
// </editor-fold>

```

```

private void jPanelNombreAeropuertoActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

```

```

public void setEstacionamientoText(String numero) {
    estacionamientoText.setText (numero);
}

```

```

public void setHangarText(String numero) {
    hangarText1.setText(numero);
}

```

```

public void setRodajeText(String numero) {
    rodajeText.setText(numero);
}

```

```

public void setTallerText(String numero) {
    tallerText.setText(numero);
}

```

```
}
```

```
public void setjPanelNombreAeropuerto(String nombre) {  
    jPanelNombreAeropuerto.setText(nombre);  
}
```

```
public void setPasajerosText(String text) {  
    pasajerosText.setText(text);  
}
```

```
public void setAeropuerto(InterfazRemota i){  
    this.i = i;  
}
```

```
private void jButtonPista1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    if (!jButtonPista1.isSelected()) {  
        jButtonPista1.setText("Cerrar");  
        try{  
            i.abrirPista(1);  
        }catch(Exception e){  
            e.printStackTrace();  
        }  
    }  
    else{  
        jButtonPista1.setText("Abrir");  
        try{  
            i.cerrarPista(1);  
        }catch(Exception e){  
            e.printStackTrace();  
        }  
    }  
}
```



```
}
```

```
private void jButtonPista3ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    if (!jButtonPista3.isSelected()) {  
        jButtonPista3.setText("Cerrar");  
        try{  
            i.abrirPista(3);  
        }catch(Exception e){  
            e.printStackTrace();  
        }  
    }  
    else{  
        jButtonPista3.setText("Abrir");  
        try{  
            i.cerrarPista(3);  
        }catch(Exception e){  
            e.printStackTrace();  
        }  
    }  
}
```

```
private void jButtonPista2ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    if (!jButtonPista2.isSelected()) {  
        jButtonPista2.setText("Cerrar");  
        try{  
            i.abrirPista(2);  
        }catch(Exception e){  
            e.printStackTrace();  
        }  
    }  
    else{  
        jButtonPista2.setText("Abrir");
```

```

try{
    i.cerrarPista(2);
}catch(Exception e){
    e.printStackTrace();
}
}
}

```

```

private void jButtonPista4ActionPerformed(java.awt.event.ActionEvent evt) {

```

```

    // TODO add your handling code here:

```

```

    if (!jButtonPista4.isSelected()) {
        jButtonPista4.setText("Cerrar");

```

```

        try{
            i.abrirPista(4);
        }catch(Exception e){
            e.printStackTrace();
        }
    }

```

```

    else{

```

```

        jButtonPista4.setText("Abrir");
        try{
            i.cerrarPista(4);
        }catch(Exception e){
            e.printStackTrace();
        }
    }

```

```

}

```

```

// Variables declaration - do not modify

```

```

private javax.swing.JLabel estacionamientoLabel;
private javax.swing.JTextField estacionamientoText;
private javax.swing.JLabel hangarLabel;
private javax.swing.JTextField hangarText1;

```

```

private javax.swing.JToggleButton jButtonPista1;
private javax.swing.JToggleButton jButtonPista2;
private javax.swing.JToggleButton jButtonPista3;
private javax.swing.JToggleButton jButtonPista4;
private javax.swing.JTextField jPanelNombreAeropuerto;
private javax.swing.JLabel pasajerosLabel;
private javax.swing.JTextField pasajerosText;
private javax.swing.JLabel pista1Label;
private javax.swing.JLabel pista2Label;
private javax.swing.JLabel pista3Label;
private javax.swing.JLabel pista4Label;
private javax.swing.JLabel rodajeLabel;
private javax.swing.JTextField rodajeText;
private javax.swing.JLabel tallerLabel;
private javax.swing.JTextField tallerText;
// End of variables declaration
}

```

### **panelAeropuerto.java:**

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JPanel.java to edit this template
 */
package vidri.vidri.interfaz;

import java.util.ArrayList;

import vidri.vidri.pecl.Autobus;
import vidri.vidri.pecl.Avion;

/**
 *
 * @author gonza
 */

```

```

public class panelAeropuerto extends javax.swing.JPanel {

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

    //Crea una clase aparte con la interfaz

    public void setearNombre (String nombre) {
        jPanelNombreAeropuerto.setText(nombre);
    }

    public void updatePistasAterrizaje (Avion[] pistas,Avion a) {
        int i = buscarAvion(a, pistas);
        switch (i){
            case 0:
                pista1Text.setText("Aterrizaje: " + a.getIdAv()+ " (" +a.getPasajeros()+ ")");
                break;
            case 1:
                pista2Text.setText("Aterrizaje: " + a.getIdAv()+ " (" +a.getPasajeros()+ ")");
                break;
            case 2:
                pista3Text.setText("Aterrizaje: " + a.getIdAv()+ " (" +a.getPasajeros()+ ")");
                break;
            case 3:
                pista4Text.setText("Aterrizaje: " + a.getIdAv()+ " (" +a.getPasajeros()+ ")");
                break;
        }
    }

    public void updateSalirDePista (Avion avion) {
        String str = "Aterrizaje: " +avion.getIdAv()+ " (" +avion.getPasajeros()+ ")";
    }
}

```

```

String str2 = "Despegue: " + avion.getIdAv() + " (" + avion.getPasajeros() + ")";
if (str.equals(pista1Text.getText()) || str2.equals(pista1Text.getText())) {
    pista1Text.setText("");
}
else if (str.equals(pista2Text.getText()) || str2.equals(pista2Text.getText())) {
    pista2Text.setText("");
}
else if (str.equals(pista3Text.getText()) || str2.equals(pista3Text.getText())) {
    pista3Text.setText("");
}
else if (str.equals(pista4Text.getText()) || str2.equals(pista4Text.getText())) {
    pista4Text.setText("");
}
else {
    System.out.println("No se han encontrado coincidencias al intentar sacar de pista");
}
}

```

```

public void updatePistasDespegue (Avion[] pistas, Avion a) {
    int i = buscarAvion(a, pistas);
    switch (i){
        case 0:
            pista1Text.setText("Despegue: " + a.getIdAv() + " (" + a.getPasajeros() + ")");
            break;
        case 1:
            pista2Text.setText("Despegue: " + a.getIdAv() + " (" + a.getPasajeros() + ")");
            break;
        case 2:
            pista3Text.setText("Despegue: " + a.getIdAv() + " (" + a.getPasajeros() + ")");
            break;
        case 3:
            pista4Text.setText("Despegue: " + a.getIdAv() + " (" + a.getPasajeros() + ")");
            break;
    }
}

```

```
}
```

```
public void updatePuertasDeDesembarque (Avion[] puertasDeEmbarque, Avion a) {
```

```
    int i = buscarAvion(a, puertasDeEmbarque);
```

```
    switch (i){
```

```
        case 0:
```

```
            gate1Text.setText("Desembarque: " +a.getIdAv());
```

```
            break;
```

```
        case 1:
```

```
            gate2Text.setText("Desembarque: " +a.getIdAv());
```

```
            break;
```

```
        case 2:
```

```
            gate3Text.setText("Desembarque: " +a.getIdAv());
```

```
            break;
```

```
        case 3:
```

```
            gate4Text.setText("Desembarque: " +a.getIdAv());
```

```
            break;
```

```
        case 4:
```

```
            gate5Text.setText("Desembarque: " +a.getIdAv());
```

```
            break;
```

```
        case 5:
```

```
            gate6Text.setText("Desembarque: " +a.getIdAv());
```

```
            break;
```

```
    }
```

```
}
```

```
public void updateSalidaPuertasDeEmbarque (Avion a) {
```

```
    String str = "Desembarque: " +a.getIdAv();
```

```
    String str2 = "Embarque: " +a.getIdAv();
```

```
    if (str.equals(gate1Text.getText()) || str2.equals(gate1Text.getText())) {
```

```
        gate1Text.setText("");
```

```
    }
```

```
    else if (str.equals(gate2Text.getText()) || str2.equals(gate2Text.getText())) {
```

```

        gate2Text.setText("");
    }
    else if (str.equals(gate3Text.getText()) || str2.equals(gate3Text.getText())) {
        gate3Text.setText("");
    }
    else if (str.equals(gate4Text.getText()) || str2.equals(gate4Text.getText())) {
        gate4Text.setText("");
    }
    else if (str.equals(gate5Text.getText()) || str2.equals(gate5Text.getText())) {
        gate5Text.setText("");
    }
    else if (str.equals(gate6Text.getText()) || str2.equals(gate6Text.getText())) {
        gate6Text.setText("");
    }
}

```

```

public void updateAreaDeRodaje (ArrayList<Avion> areaDeRodaje){
    String str = "";
    for (int i = 0; i<areaDeRodaje.size(); i++) {
        str += areaDeRodaje.get(i).getIdAv() + ", ";
    }
    rodajeText.setText(str);
}

```

```

public void updatePuertasDeEmbarque (Avion a,Avion[] puertasDeEmbarque) {
    int i = buscarAvion(a, puertasDeEmbarque);
    switch (i){
        case 0:
            gate1Text.setText("Embarque: " +a.getIdAv());
            break;
        case 1:
            gate2Text.setText("Embarque: " +a.getIdAv());
            break;
        case 2:

```

```

        gate3Text.setText("Embarque: " + a.getIdAv());
        break;
    case 3:
        gate4Text.setText("Embarque: " + a.getIdAv());
        break;
    case 4:
        gate5Text.setText("Embarque: " + a.getIdAv());
        break;
    case 5:
        gate6Text.setText("Embarque: " + a.getIdAv());
        break;
    }
}

public void updatePasajeros (int pasajeros) {
    nPasajerosText.setText(""+pasajeros);
}

public void updateAreaDeEstacionamiento (ArrayList<Avion> areaEstacionamiento) {
    String str = "";
    for (int i = 0; i<areaEstacionamiento.size(); i++) {
        str += areaEstacionamiento.get(i).getIdAv() + ", ";
    }
    estacionamientoText.setText(str);
}

public void updateHangar (ArrayList<Avion> hangar) {
    String str = "";
    for (int i = 0; i<hangar.size(); i++) {
        str += hangar.get(i).getIdAv() + ", ";
    }
    hangarText.setText(str);
}

```



```

public void updateTaller (ArrayList<Avion> taller) {
    String str = "";
    for (int i = 0; i<taller.size(); i++) {
        str += taller.get(i).getIdAv() + ", ";
    }
    tallerText.setText(str);
}

```

```

public void updatebus1 (Autobus bus){
    bus1Text.setText(bus.getIdentificacion() + "("+bus.getPasajeros()+")");
}

```

```

public void updatebus2 (Autobus bus) {
    bus2Text.setText(bus.getIdentificacion() + "("+bus.getPasajeros()+")");
}

```

```

public int buscarAvion(Avion a, Avion[] array){
    int pos = 0;
    while(pos<array.length && array[pos]!=a){
        pos++;
    }
    return pos;
}

```

```

public int primerHuecoEnArray (Avion[] array){
    int pos = 0;
    while(pos<array.length&& array[pos]!=null) {
        pos++;
    }
    return pos;
}

```

```

/**

```

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jPanelNombreAeropuerto = new javax.swing.JTextField();

bus1Label = new javax.swing.JLabel();

bus1Text = new javax.swing.JTextField();

bus2Label = new javax.swing.JLabel();

bus2Text = new javax.swing.JTextField();

nPasajerosLabel = new javax.swing.JLabel();

nPasajerosText = new javax.swing.JTextField();

hangarLabel = new javax.swing.JLabel();

hangarText = new javax.swing.JTextField();

tallerLabel = new javax.swing.JLabel();

tallerText = new javax.swing.JTextField();

estacionamientoLabel = new javax.swing.JLabel();

estacionamientoText = new javax.swing.JTextField();

gate1Label = new javax.swing.JLabel();

gate1Text = new javax.swing.JTextField();

gate2Label = new javax.swing.JLabel();

gate2Text = new javax.swing.JTextField();

gate3Label = new javax.swing.JLabel();

gate3Text = new javax.swing.JTextField();

gate4Label = new javax.swing.JLabel();

gate4Text = new javax.swing.JTextField();

gate5Label = new javax.swing.JLabel();

gate5Text = new javax.swing.JTextField();

gate6Label = new javax.swing.JLabel();

gate6Text = new javax.swing.JTextField();

rodajeLabel = new javax.swing.JLabel();

rodajeText = new javax.swing.JTextField();

```
pista1Label = new javax.swing.JLabel();
pista1Text = new javax.swing.JTextField();
pista2Label = new javax.swing.JLabel();
pista2Text = new javax.swing.JTextField();
pista3Label = new javax.swing.JLabel();
pista3Text = new javax.swing.JTextField();
pista4Label = new javax.swing.JLabel();
pista4Text = new javax.swing.JTextField();
```

```
setPreferredSize(new java.awt.Dimension(600, 500));
```

```
jPanelNombreAeropuerto.setEditable(false);
jPanelNombreAeropuerto.setFont(new java.awt.Font("Arial", 1, 24)); // NOI18N
jPanelNombreAeropuerto.setHorizontalAlignment(javax.swing.JTextField.CENTER);
jPanelNombreAeropuerto.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jPanelNombreAeropuertoActionPerformed(evt);
    }
});
```

```
bus1Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
bus1Label.setText("Bus que llega al aeropuerto:");
```

```
bus1Text.setEditable(false);
bus1Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
bus2Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
bus2Label.setText("Bus que sale del aeropuerto:");
```

```
bus2Text.setEditable(false);
bus2Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
bus2Text.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        bus2TextActionPerformed(evt);
    }
});
```

```
}  
});  
  
nPasajerosLabel.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
nPasajerosLabel.setText("Nº de Pasajeros en el Aeropuerto:");  
  
nPasajerosText.setEditable(false);  
nPasajerosText.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
  
hangarLabel.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
hangarLabel.setText("Hangar:");  
  
hangarText.setEditable(false);  
hangarText.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
  
tallerLabel.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
tallerLabel.setText("Taller:");  
  
tallerText.setEditable(false);  
tallerText.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
  
estacionamientoLabel.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
estacionamientoLabel.setText("Área de Estacionamiento:");  
  
estacionamientoText.setEditable(false);  
estacionamientoText.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
  
gate1Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
gate1Label.setText("Gate 1:");  
  
gate1Text.setEditable(false);  
gate1Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N  
  
gate2Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
gate2Label.setText("Gate 2:");
```

```
gate2Text.setEditable(false);
```

```
gate2Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
gate3Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
gate3Label.setText("Gate 3:");
```

```
gate3Text.setEditable(false);
```

```
gate3Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
gate4Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
gate4Label.setText("Gate 4:");
```

```
gate4Text.setEditable(false);
```

```
gate4Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
gate5Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
gate5Label.setText("Gate 5:");
```

```
gate5Text.setEditable(false);
```

```
gate5Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
gate6Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
gate6Label.setText("Gate 6:");
```

```
gate6Text.setEditable(false);
```

```
gate6Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
rodajeLabel.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
rodajeLabel.setText("Área de rodaje:");
```

```
rodajeText.setEditable(false);
```

```
pista1Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
pista1Label.setText("Pista 1: ");
```

```
pista1Text.setEditable(false);
```

```
pista1Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
pista1Text.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        pista1TextActionPerformed(evt);
```

```
    }
```

```
});
```

```
pista2Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
pista2Label.setText("Pista 2:");
```

```
pista2Text.setEditable(false);
```

```
pista2Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
pista2Text.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        pista2TextActionPerformed(evt);
```

```
    }
```

```
});
```

```
pista3Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
pista3Label.setText("Pista 3:");
```

```
pista3Text.setEditable(false);
```

```
pista3Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
pista3Text.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        pista3TextActionPerformed(evt);
```

```
    }
```

```
});
```

```
pista4Label.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
pista4Label.setText("Pista 4:");
```

```
pista4Text.setEditable(false);
```

```
pista4Text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
```

```
pista4Text.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        pista4TextActionPerformed(evt);
```

```
    }
```

```
});
```

```
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
```

```
this.setLayout(layout);
```

```
layout.setHorizontalGroup(
```

```
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
    .addGroup(layout.createSequentialGroup()
```

```
        .addGap(10, 10, 10)
```

```
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addComponent(jPanelNombreAeropuerto)
```

```
            .addGroup(layout.createSequentialGroup()
```

```
                .addComponent(hangarLabel)
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                .addComponent(hangarText))
```

```
            .addGroup(layout.createSequentialGroup()
```

```
                .addComponent(tallerLabel)
```

```
                .addGap(20, 20, 20)
```

```
                .addComponent(tallerText))
```

```
            .addGroup(layout.createSequentialGroup()
```

```
                .addComponent(estacionamientoLabel)
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
                .addComponent(estacionamientoText))
```

```
            .addGroup(layout.createSequentialGroup()
```

```
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
```

```
                    .addGroup(layout.createSequentialGroup()
```

```

        .addComponent(gate1Label,          javax.swing.GroupLayout.PREFERRED_SIZE,      47,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(gate1Text,          javax.swing.GroupLayout.PREFERRED_SIZE,      162,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addGap(148, 148, 148)

        .addComponent(gate4Label,          javax.swing.GroupLayout.PREFERRED_SIZE,      47,
javax.swing.GroupLayout.PREFERRED_SIZE))

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

            .addComponent(gate6Label,          javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

            .addComponent(gate5Label,          javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(gate4Text))

        .addGroup(layout.createSequentialGroup())

        .addComponent(gate3Label,          javax.swing.GroupLayout.PREFERRED_SIZE,      47,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(gate3Text,          javax.swing.GroupLayout.PREFERRED_SIZE,      162,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(gate6Text,          javax.swing.GroupLayout.PREFERRED_SIZE,      166,
javax.swing.GroupLayout.PREFERRED_SIZE))

        .addGroup(layout.createSequentialGroup())

        .addComponent(rodajeLabel)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(rodajeText))

        .addGroup(layout.createSequentialGroup())

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addComponent(pista1Label)

            .addComponent(pista2Label,          javax.swing.GroupLayout.PREFERRED_SIZE,      51,
javax.swing.GroupLayout.PREFERRED_SIZE))

        .addGap(4, 4, 4)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

            .addComponent(pista1Text)

            .addComponent(pista2Text, javax.swing.GroupLayout.DEFAULT_SIZE, 164, Short.MAX_VALUE))

```



```

.addGap(148, 148, 148)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

    .addComponent(pista3Label)

    .addComponent(pista4Label))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

    .addComponent(pista3Text)

    .addComponent(pista4Text)))

.addGroup(layout.createSequentialGroup())

    .addComponent(gate2Label,          javax.swing.GroupLayout.PREFERRED_SIZE,          47,
javax.swing.GroupLayout.PREFERRED_SIZE)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addComponent(gate2Text,          javax.swing.GroupLayout.PREFERRED_SIZE,          162,
javax.swing.GroupLayout.PREFERRED_SIZE)

    .addGap(206, 206, 206)

    .addComponent(gate5Text,          javax.swing.GroupLayout.PREFERRED_SIZE,          166,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(layout.createSequentialGroup())

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

    .addGroup(layout.createSequentialGroup())

        .addComponent(nPasajerosLabel)

        .addGap(28, 28, 28)

        .addComponent(nPasajerosText,    javax.swing.GroupLayout.PREFERRED_SIZE,    131,
javax.swing.GroupLayout.PREFERRED_SIZE))

    .addGroup(layout.createSequentialGroup())

        .addComponent(bus1Label)

        .addGap(4, 4, 4)

        .addComponent(bus1Text,          javax.swing.GroupLayout.PREFERRED_SIZE,          102,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addGap(33, 33, 33)

        .addComponent(bus2Label)))

.addGap(1, 1, 1)

    .addComponent(bus2Text,          javax.swing.GroupLayout.PREFERRED_SIZE,          91,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addContainerGap())

);

```

```

layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jPanelNombreAeropuerto, javax.swing.GroupLayout.PREFERRED_SIZE, 27,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(bus1Label, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(bus1Text, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(bus2Label, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(bus2Text, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(nPasajerosLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 37,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(nPasajerosText, javax.swing.GroupLayout.PREFERRED_SIZE, 37,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(hangarLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 25,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(hangarText, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(tallerText, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(tallerLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 25,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addComponent(estacionamientoLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

```

```

        .addComponent(estacionamientoText, javax.swing.GroupLayout.DEFAULT_SIZE, 25,
Short.MAX_VALUE))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

            .addComponent(gate1Label, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                .addComponent(gate1Text)

                .addComponent(gate4Label, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addComponent(gate4Text, javax.swing.GroupLayout.DEFAULT_SIZE, 25, Short.MAX_VALUE)))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(gate2Label, javax.swing.GroupLayout.DEFAULT_SIZE, 25, Short.MAX_VALUE)

            .addComponent(gate2Text, javax.swing.GroupLayout.DEFAULT_SIZE, 25, Short.MAX_VALUE)

            .addComponent(gate5Label, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

            .addComponent(gate5Text, javax.swing.GroupLayout.DEFAULT_SIZE, 25, Short.MAX_VALUE))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(gate3Label, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

            .addComponent(gate3Text)

            .addComponent(gate6Label, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(gate6Text))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(rodajeLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(rodajeText, javax.swing.GroupLayout.DEFAULT_SIZE, 26, Short.MAX_VALUE))

        .addGap(19, 19, 19)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(pista1Text, javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(pista3Label, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

```

```

        .addComponent(pista3Text)

        .addComponent(pista1Label,                                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

        .addComponent(pista2Label,                                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(pista2Text, javax.swing.GroupLayout.DEFAULT_SIZE, 25, Short.MAX_VALUE)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

        .addComponent(pista4Label,                                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(pista4Text)))

        .addContainerGap())

    );
}
// </editor-fold>

```

```

private void jPanelNombreAeropuertoActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

}

```

```

private void bus2TextActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

}

```

```

private void pista1TextActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

}

```

```

private void pista2TextActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

}

```

```

private void pista3TextActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

}

```

```
private void pista4TextActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}
```

```
// Variables declaration - do not modify  
private javax.swing.JLabel bus1Label;  
private javax.swing.JTextField bus1Text;  
private javax.swing.JLabel bus2Label;  
private javax.swing.JTextField bus2Text;  
private javax.swing.JLabel estacionamientoLabel;  
private javax.swing.JTextField estacionamientoText;  
private javax.swing.JLabel gate1Label;  
private javax.swing.JTextField gate1Text;  
private javax.swing.JLabel gate2Label;  
private javax.swing.JTextField gate2Text;  
private javax.swing.JLabel gate3Label;  
private javax.swing.JTextField gate3Text;  
private javax.swing.JLabel gate4Label;  
private javax.swing.JTextField gate4Text;  
private javax.swing.JLabel gate5Label;  
private javax.swing.JTextField gate5Text;  
private javax.swing.JLabel gate6Label;  
private javax.swing.JTextField gate6Text;  
private javax.swing.JLabel hangarLabel;  
private javax.swing.JTextField hangarText;  
private javax.swing.JTextField jPanelNombreAeropuerto;  
private javax.swing.JLabel nPasajerosLabel;  
private javax.swing.JTextField nPasajerosText;  
private javax.swing.JLabel pista1Label;  
private javax.swing.JTextField pista1Text;  
private javax.swing.JLabel pista2Label;  
private javax.swing.JTextField pista2Text;
```

```
private javax.swing.JLabel pista3Label;  
private javax.swing.JTextField pista3Text;  
private javax.swing.JLabel pista4Label;  
private javax.swing.JTextField pista4Text;  
private javax.swing.JLabel rodajeLabel;  
private javax.swing.JTextField rodajeText;  
private javax.swing.JLabel tallerLabel;  
private javax.swing.JTextField tallerText;  
  
// End of variables declaration  
  
}
```