Cabina.java

```
package Concurrencia;

public class Cabina {
   private String nombreCabina = "";
   private String tipo = "";
   private boolean disponible = true;

   public Cabina(String nombre, String tipo) {

      this.nombreCabina = nombre;
      this.tipo = tipo;
   }

   public String getNombreCabina() {
      return nombreCabina;
   }

   public void setNombreCabina(String nombreCabina) {
      this.nombreCabina = nombreCabina;
   }

   public String getTipo() {
      return tipo;
   }

   public void setTipo(String tipo) {
      this.tipo = tipo;
   }

   public boolean isDisponible() {
      return disponible;
   }

   public void setDisponible(boolean disponible) {
      this.disponible = disponible;
   }

}
```

CabinaTarjeta.java

```
package Concurrencia;
import ClasesAux.ListasThreadsPeaje;
import Hilos.Ambulancia;
import Hilos.Camion;
import Hilos.Coche;
import Hilos.Empleado;
import Log.FuncionesLog;
import java.util.Random;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import java.util.concurrent.locks.ReentrantReadWriteLock;
import javax.swing.JTextField;

public class CabinaPeajeTarjeta extends Cabina {
   private boolean abierta = true;
   private JTextField vehiculo;
   private ListasThreadsPeaje coches, camiones, ambulancia;
   private FuncionesLog fg;
   private boolean accesoCoche = true;
   private boolean accesoCamion = true;
   private boolean accesoAmbulancia = true;
   private int i;
   Lock lock = new ReentrantLock();


   public CabinaPeajeTarjeta(String nombre,JTextField vehiculo, FuncionesLog fg, int i) {
      super(nombre,"Tarjeta");
      this.vehiculo = vehiculo;
      this.fg = fg;
      this.i = i;
      coches = new ListasThreadsPeaje(vehiculo);
      camiones = new ListasThreadsPeaje(vehiculo);
      ambulancia = new ListasThreadsPeaje(vehiculo);
   }
```

```
 public void entradaCabinasCamiones(Camion camion){
  Random r = new Random();
   lock.lock();
   try{
      camiones.meter(camion);
      fg.writeDebugFile("El "+camion.getIdCamion()+" accede a la cabina de peaje "+i+"\n");

   }finally{
      lock.unlock();
   }

}
  public void pagoCabinasCamiones(Camion camion) throws InterruptedException{
  Random r = new Random();
   lock.lock();
   try{

      fg.writeDebugFile("El "+camion.getIdCamion()+" esta pagando en la cabina de peaje "+i+"\n");
      Thread.sleep((int)(Math.random() * 8000 + 6000));
   }finally{
      lock.unlock();
   }

}
   public void salidaCabinasCamiones(Camion camion){
  Random r = new Random();
   lock.lock();
   try{
      camiones.sacar(camion);
      fg.writeDebugFile("El "+camion.getIdCamion()+" sale de la cabina de peaje "+i+"\n");

   }finally{
      lock.unlock();
   }

}
```

```
public void entradaCabinaCoches(Coche cc){
  Random r = new Random();
   lock.lock();
   try{
      accesoCoche = false;
       coches.meter(cc);
       fg.writeDebugFile("El "+cc.getIdCoche()+" accede a la cabina de peaje "+i+"\n");
       Thread.sleep(6000+r.nextInt(8000));
       cc.getPaso().mirar();
       coches.sacar(cc);
       fg.writeDebugFile("El "+cc.getIdCoche()+" sale de la cabina de peaje de vuelta a la carretera\n");
       accesoCoche = true;


   }catch(InterruptedException e){
       System.out.println(e.toString());
   }finally{
       lock.unlock();
   }

}

  public void entradaCabinaAmbulancia(Ambulancia a){
  Random r = new Random();
   lock.lock();
   try{
      accesoAmbulancia = false;
       ambulancia.meter(a);
       fg.writeDebugFile("La "+a.getIdAmbulancia()+" accede a la cabina de peaje "+i+"\n");

       Thread.sleep(6000+r.nextInt(8000));
       ambulancia.sacar(a);
       fg.writeDebugFile("La "+a.getIdAmbulancia()+" sale de la cabina de peaje de vuelta a la carretera\n");
       accesoAmbulancia = true;
```

```
        }catch(InterruptedException e){
            System.out.println(e.toString());
        }finally{
            lock.unlock();
        }

    }



    public boolean isAbierta() {
        return abierta;
    }

    public void setAbierta(boolean abierta) {
        this.abierta = abierta;
    }
}
```

CabinaManual.java

```
package Concurrencia;
import ClasesAux.ListasThreadsPeaje;
import Hilos.Ambulancia;
import Hilos.Camion;
import Hilos.Coche;
import Hilos.Empleado;
import Log.FuncionesLog;
import java.util.ArrayList;
import java.util.Random;
import java.util.concurrent.*;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import java.util.logging.Level;
import java.util.logging.Logger;
import static javax.management.Query.or;
import javax.swing.JTextField;

public class CabinaPeajeManual extends Cabina{
    private JTextField vehiculo;
    private JTextField empleado;
    private ListasThreadsPeaje empleados, coches, camiones,ambulancia;
    private FuncionesLog fg;
    private int i;
    private boolean accesoCoche = true;
    private boolean accesoCamion = true;
    private boolean cocheEsperando = false;
    private boolean camionEsperando = false;
    private boolean ambulanciaEsperando = false;
    private boolean empleadoEsperando = false;


    //Concurrencia
    Lock lock = new ReentrantLock();
    Condition esperaEmpleado = lock.newCondition();
    Condition esperaCoche = lock.newCondition();
    Condition esperaCamion = lock.newCondition();
```

```
 Condition esperaAmbulancia = lock.newCondition();
 Condition cd = lock.newCondition();
 Condition cd2 = lock.newCondition();




 public CabinaPeajeManual(String nombre,JTextField vehiculo, JTextField empleado, FuncionesLog fg, int i) {
    super(nombre,"Manual");
    this.fg = fg;
    this.vehiculo = vehiculo;
    coches = new ListasThreadsPeaje(vehiculo);
    camiones = new ListasThreadsPeaje(vehiculo);
    ambulancia = new ListasThreadsPeaje(vehiculo);
    this.empleado = empleado;
    empleados = new ListasThreadsPeaje(empleado);
    this.i = i;
 }

 public void entradaEmpleadosCabinas(Empleado e){
lock.lock();
try{
    empleados.meter(e);
    fg.writeDebugFile("El "+e.getIdEmpleado()+" accede a la cabina de peaje "+i+"\n");
    if(!isCocheEsperando() || !isCamionEsperando() || !isAmbulanciaEsperando()){
       setEmpleadoEsperando(true);
       try{
         esperaEmpleado.await();
       }catch(InterruptedException ar){
          System.out.println(ar.toString());
       }
    }

}finally{
   lock.unlock();
}

 }
```

```
public void salidaEmpleadoDescanso(Empleado e) throws InterruptedException{

   lock.lock();
   try{
   empleados.sacar(e);
   fg.writeDebugFile("El "+e.getIdEmpleado()+" comienza su descanso en la cabina "+i+"\n");
   Thread.sleep(5000);
   fg.writeDebugFile("El "+e.getIdEmpleado()+" finaliza su descanso en la cabina "+i+"\n");
   setEmpleadoEsperando(false);
   }finally{
      lock.unlock();
   }
}
public void vueleveEmpleado(Empleado e){
   lock.lock();
   try{
      if(!isCocheEsperando() ){
         setEmpleadoEsperando(true);
         try{
            esperaEmpleado.await();
         }catch(InterruptedException a){
            System.out.println(a.toString());
         }
      }
   }finally{
      lock.unlock();
   }
}
public void entradaCabinasCoches(Coche coche){

   lock.lock();
   try{

      coches.meter(coche);
      fg.writeDebugFile("El "+coche.getIdCoche()+" accede a la cabina de peaje "+i+"\n");
      accesoCoche = false;
      if(isEmpleadoEsperando()){
         esperaEmpleado.signal();
```

```
        }else{
            setCocheEsperando(true);
        }
        try{
          esperaCoche.await();
        }catch(InterruptedException e){
            System.out.println(e.toString());
        }

    }finally{
        lock.unlock();
    }

}
public void entradaCabinasAmbulancia(Ambulancia a){
 lock.lock();
    try{

        ambulancia.meter(a);
        fg.writeDebugFile("La "+a.getIdAmbulancia()+" accede a la cabina de peaje "+i+"\n");
        if(isEmpleadoEsperando()){
            esperaEmpleado.signal();
        }else{
            setAmbulanciaEsperando(true);
        }
        try{
          esperaAmbulancia.await();
        }catch(InterruptedException e){
            System.out.println(e.toString());
        }

    }finally{
        lock.unlock();
    }

}
public void pagoAmbulancia(Empleado e) throws InterruptedException{
    lock.lock();
```

```
        try{
            fg.writeDebugFile("El empleado "+e.getIdEmpleado()+" esta cobrando a la ambulancia en la cabina "+i+"\n");
            Thread.sleep((int) (Math.random() * 8000 + 6000));
            setAmbulanciaEsperando(false);
            esperaAmbulancia.signal();
            e.setActividadesSeguidas(e.getActividadesSeguidas()+1);
        }finally{
            lock.unlock();
        }
    }
    public void realizarPagoCoches(Empleado e) throws InterruptedException{

      lock.lock();
      try{
      fg.writeDebugFile("El empleado "+e.getIdEmpleado()+" esta cobrando al coche en la cabina "+i+"\n");
      Thread.sleep((int) (Math.random() * 8000 + 6000));
       setCocheEsperando(false);
       esperaCoche.signal();
       e.setActividadesSeguidas(e.getActividadesSeguidas()+1);
    }finally{
    lock.unlock();
}


    }
     public void realizarPagoCamiones(Empleado e) throws InterruptedException{

      lock.lock();
      try{
      fg.writeDebugFile("El empleado "+e.getIdEmpleado()+" esta cobrando al camion en la cabina "+i+"\n");
      Thread.sleep((int) (Math.random() * 8000 + 6000));
       setCamionEsperando(false);
       esperaCamion.signal();
       e.setActividadesSeguidas(e.getActividadesSeguidas()+1);
    }finally{
    lock.unlock();
}
    }
    public void salidaCoche(Coche coche){
```

```
        coches.sacar(coche);
        fg.writeDebugFile("El "+coche.getIdCoche()+" sale de la cabina de peaje de vuelta a la carretera \n");
        accesoCoche = true;
    }
    public void cocheCabinaManual(Coche coche){
        lock.lock();
        try{
        entradaCabinasCoches(coche);
        coche.getPaso().mirar();
        salidaCoche(coche);
    }finally{
    lock.unlock();
}
    }

    public void entradaCabinasCamiones(Camion camion){
      lock.lock();
        try{

            camiones.meter(camion);
            fg.writeDebugFile("El "+camion.getIdCamion()+" accede a la cabina de peaje "+i+"\n");
            accesoCamion = false;
            if(isEmpleadoEsperando()){
                esperaEmpleado.signal();
            }else{
                setCamionEsperando(true);
            }
            try{
              esperaCamion.await();
            }catch(InterruptedException e){
                System.out.println(e.toString());
            }

        }finally{
            lock.unlock();
        }

    }
```

```
    public void salidaCamion(Camion camion){
        camiones.sacar(camion);
        fg.writeDebugFile("El "+camion.getIdCamion()+" sale de la cabina de peaje de vuelta a la carretera \n");
       accesoCamion = true;
    }
    public void camionCabinaManual(Camion camion){
        lock.lock();
        try{
        entradaCabinasCamiones(camion);
        camion.getPaso().mirar();
        salidaCamion(camion);
    }finally{
    lock.unlock();
}
    }


    public void salidaAmbulancia(Ambulancia am){
        ambulancia.sacar(am);
        fg.writeDebugFile("La "+am.getIdAmbulancia()+" sale de la cabina de peaje de vuelta a la carretera \n");
    }
    public void ambulanciaCabinaManual(Ambulancia am){
        lock.lock();
        try{
        entradaCabinasAmbulancia(am);
        salidaAmbulancia(am);
    }finally{
    lock.unlock();
}
    }



    public boolean isCocheEsperando() {
        return cocheEsperando;
    }

    public void setCocheEsperando(boolean cocheEsperando) {
```

```
        this.cocheEsperando = cocheEsperando;
    }

    public boolean isCamionEsperando() {
        return camionEsperando;
    }

    public void setCamionEsperando(boolean camionEsperando) {
        this.camionEsperando = camionEsperando;
    }

    public boolean isEmpleadoEsperando() {
        return empleadoEsperando;
    }

    public void setEmpleadoEsperando(boolean empleadoEsperando) {
        this.empleadoEsperando = empleadoEsperando;
    }

    public boolean isAmbulanciaEsperando() {
        return ambulanciaEsperando;
    }

    public void setAmbulanciaEsperando(boolean ambulanciaEsperando) {
        this.ambulanciaEsperando = ambulanciaEsperando;
    }

}
```

CabinaPeaje.java

```
package Concurrencia;
import ClasesAux.ListasThreadsPeaje;
import Hilos.*;
import Log.*;
import java.util.ArrayList;
import java.util.Random;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JTextField;
public class CabinaPeaje {
   //Elementos de la interfaz

   private JTextField colaEntradaPeaje;
   private JTextField EmpleadoCabinaCamiones1M;
   private JTextField EmpleadoCabinaCamiones2M;
   private JTextField EmpleadoCabinaCoches1M;
   private JTextField EmpleadoCabinaCoches2M;
   private JTextField EmpleadoCabinaCoches3M;
   private JTextField camionCabina1M;
   private JTextField camionCabina2M;
   private JTextField camionCabina3T1;
   private JTextField camionCabina4T;
   private JTextField cocheCabina1M1;
   private JTextField cocheCabina2M;
   private JTextField cocheCabina3M;
   private JTextField cocheCabina4T;
   private JTextField cocheCabina5T1;
   private JTextField cocheCabina6T;

   private String pago;
   private boolean  evacuar = false;

   //Clases concurrentes
   private CabinaPeajeManual cabinaCoches1, cabinaCoches2, cabinaCoches3, cabinaCamiones1, cabinaCamiones2;
```

```
    private CabinaPeajeTarjeta cabinaCoches4, cabinaCoches5, cabinaCoches6, cabinaCamiones3, cabinaCamiones4;

    private Paso paso;

    private boolean cabinaCocheDisponible = true, cabinaCamionDisponible = true;


    //Clases log

    private FuncionesLog fg;

    //

    private Lock lock = new ReentrantLock();

    private Condition coches = lock.newCondition();

    private Condition cochesA = lock.newCondition();

    private Condition camiones = lock.newCondition();

    private Condition ambu = lock.newCondition();

    private ArrayList<Coche> aforoCoches = new ArrayList();

    private ArrayList<Camion> aforoCamiones = new ArrayList();

    private ArrayList<Ambulancia> aforoAmbulancias = new ArrayList();

    private ListasThreadsPeaje cocheEntrada,camionEntrada, ambulancia;

    private int ambulanciasEsperando = 0;


    //Crear constructor con sus metodos

    public CabinaPeaje(JTextField colaEntrada, FuncionesLog fg, JTextField cocheCabina1M, JTextField EmpleadoCabinaCoches1M,
JTextField cocheCabina2M, JTextField EmpleadoCabinaCoches2M, JTextField cocheCabina3M, JTextField
EmpleadoCabinaCoches3M, JTextField camionCabina1M, JTextField EmpleadoCabinaCamiones1M, JTextField camionCabina2M,
JTextField EmpleadoCabinaCamiones2M, JTextField cocheCabina4T, JTextField cocheCabina5T1, JTextField cocheCabina6T,
JTextField camionCabina3T1, JTextField camionCabina4T, Paso paso) {

        this.colaEntradaPeaje = colaEntrada;

        cocheEntrada = new ListasThreadsPeaje(colaEntrada);

        camionEntrada = new ListasThreadsPeaje(colaEntrada);

        ambulancia = new ListasThreadsPeaje(colaEntrada);

        this.fg = fg;

        this.paso = paso;

        this.EmpleadoCabinaCamiones1M = EmpleadoCabinaCamiones1M;

        this.EmpleadoCabinaCamiones2M = EmpleadoCabinaCamiones2M;

        this.EmpleadoCabinaCoches1M = EmpleadoCabinaCoches1M;

        this.EmpleadoCabinaCoches2M = EmpleadoCabinaCoches2M;

        this.EmpleadoCabinaCoches3M = EmpleadoCabinaCoches3M;

        this.cocheCabina1M1 = cocheCabina1M;

        this.cocheCabina2M = cocheCabina2M;

        this.cocheCabina3M = cocheCabina3M;

        this.cocheCabina4T = cocheCabina4T;
```

```
    this.cocheCabina5T1 = cocheCabina5T1;
    this.cocheCabina6T = cocheCabina6T;
    this.camionCabina1M = camionCabina1M;
    this.camionCabina2M = camionCabina2M;
    this.camionCabina3T1 = camionCabina3T1;
    this.camionCabina4T = camionCabina4T;
    this.cabinaCoches1 = new CabinaPeajeManual("CabinaCoches1", cocheCabina1M, EmpleadoCabinaCoches1M, fg, 1);
    this.cabinaCoches2 = new CabinaPeajeManual("CabinaCoches2", cocheCabina2M, EmpleadoCabinaCoches2M, fg, 2);
    this.cabinaCoches3 = new CabinaPeajeManual("CabinaCoches3", cocheCabina3M, EmpleadoCabinaCoches3M, fg, 3);
    this.cabinaCamiones1 = new CabinaPeajeManual("CabinaCamiones1", camionCabina1M, EmpleadoCabinaCamiones1M, fg, 4);
    this.cabinaCamiones2 = new CabinaPeajeManual("CabinaCamiones2", camionCabina2M, EmpleadoCabinaCamiones2M, fg, 5);
    this.cabinaCoches4 = new CabinaPeajeTarjeta("CabinaCoches4", cocheCabina4T, fg, 1);
    this.cabinaCoches5 = new CabinaPeajeTarjeta("CabinaCoches5", cocheCabina5T1, fg, 2);
    this.cabinaCoches6 = new CabinaPeajeTarjeta("CabinaCoches6", cocheCabina6T, fg, 3);
    this.cabinaCamiones3 = new CabinaPeajeTarjeta("CabinaCamiones3", camionCabina3T1, fg, 4);
    this.cabinaCamiones4 = new CabinaPeajeTarjeta("CabinaCamiones4", camionCabina4T, fg, 5);

}

public void entradaColaPeajeCoche(Coche c) {
    Random r = new Random();
    lock.lock();
    try {
        fg.writeDebugFile("El " + c.getIdCoche() + " llega al Peaje \n");
        aforoCoches.add(c);
        if(ambulanciasEsperando > 0){
            try {
                cochesA.await();
            } catch (InterruptedException ex) {
                Logger.getLogger(CabinaPeaje.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

        if (!cabinaDisponible(c)) {
            cocheEntrada.meter(c);
            try {
                fg.writeDebugFile("El " + c.getIdCoche() + " esta en la cola de espera, al no haber cabinas disponibles \n");
                coches.await();
```

```
            } catch (InterruptedException e) {
                System.out.println(e.toString());
            }
            cocheEntrada.sacar(c);
        }
        c.setCab(cabinaManualCocheDisponible());

        if (c.getCab() == null) {
            c.setCpt(cabinaTarjetaCocheDisponible());

        }

    } finally {
        lock.unlock();
    }
}

public void entradaColaPeajeCamion(Camion c) {
    Random r = new Random();
    lock.lock();
    try {
        fg.writeDebugFile("El " + c.getIdCamion() + " llega al Peaje \n");
        aforoCamiones.add(c);

        if (!cabinaDisponibleCamion(c)) {
            camionEntrada.meter(c);
            try {
                fg.writeDebugFile("El " + c.getIdCamion() + " esta en la cola de espera, al no haber cabinas disponibles \n");
                camiones.await();

            } catch (InterruptedException e) {
                System.out.println(e.toString());
            }
            camionEntrada.sacar(c);
        }
        c.setCpm(cabinaManualCamionDisponible());
```

```
        if (c.getCpm() == null) {
            c.setCpt(cabinaTarjetaCamionDisponible());

        }

        camionEntrada.sacar(c);
    } finally {
        lock.unlock();
    }
}

public void entradaColaPeajeAmbulancia(Ambulancia a) {
    lock.lock();
    try {
        ambulanciasEsperando++;
        fg.writeDebugFile("La " + a.getIdAmbulancia()+ " llega al Peaje \n");
        aforoAmbulancias.add(a);

        if (!cabinaDisponibleAmbulancia(a)) {
            ambulancia.meter(a);
            try {
                fg.writeDebugFile("La " + a.getIdAmbulancia()+ " esta en la cola de espera, al no haber cabinas disponibles \n");
                ambu.await();

            } catch (InterruptedException e) {
                System.out.println(e.toString());
            }
            ambulancia.sacar(a);
        }
        a.setCpm(cabinaManualCocheDisponible());

        if (a.getCpm()== null) {
            a.setCpt(cabinaTarjetaCocheDisponible());

        }
        ambulanciasEsperando--;
        cochesA.signalAll();
```

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

public void salidaPeajeAmbulancia(Ambulancia a) {
 lock.lock();
    try {
        if (a.getCpm()== null) {
            a.getCpt().setDisponible(true);
            if (a.getCpt().isAbierta()) {
                ambu.signal();
                aforoAmbulancias.remove(a);
            }
        } else {
            a.getCpm().setDisponible(true);
            ambu.signal();
             aforoAmbulancias.remove(a);
        }
        fg.writeDebugFile("La " + a.getIdAmbulancia()+ " ha abandonado el peaje\n");
    } finally {
        lock.unlock();
    }
}

public void salidaPeajeCoche(Coche c) {
    lock.lock();
    try {
        if (c.getCab() == null) {
            c.getCpt().setDisponible(true);
            if (c.getCpt().isAbierta()) {
                aforoCoches.remove(c);
                if (getAmbulanciasEsperando() > 0) {
                    ambu.signal();
                } else {
                    coches.signal();
                }
            }
```

```
        } else {
            c.getCab().setDisponible(true);
            aforoCoches.remove(c);
            if (getAmbulanciasEsperando() > 0) {
                ambu.signal();
            } else {
                coches.signal();
            }
        }
        fg.writeDebugFile("El " + c.getIdCoche() + " ha abandonado el peaje\n");
    } finally {
        lock.unlock();
    }
}

public void salidaPeajeCamiones(Camion c) {
    lock.lock();
    try {
        if (c.getCpm() == null) {
            c.getCpt().setDisponible(true);
            if (c.getCpt().isAbierta()) {
                camiones.signal();
                aforoCamiones.remove(c);
            }
        } else {
            c.getCpm().setDisponible(true);
            camiones.signal();
            aforoCamiones.remove(c);
        }
        fg.writeDebugFile("El " + c.getIdCamion() + " ha abandonado el peaje\n");
    } finally {
        lock.unlock();
    }
}

private CabinaPeajeManual cabinaManualCocheDisponible() {
    CabinaPeajeManual cabinaDisponible = null;
    if (cabinaCoches1.isDisponible()) {
```

```
        cabinaCoches1.setDisponible(false);
        cabinaDisponible = cabinaCoches1;
    } else if (cabinaCoches2.isDisponible()) {
        cabinaCoches2.setDisponible(false);
        cabinaDisponible = cabinaCoches2;
    } else if (cabinaCoches3.isDisponible()) {
        cabinaCoches3.setDisponible(false);
        cabinaDisponible = cabinaCoches3;
    }
    return cabinaDisponible;
}

private CabinaPeajeTarjeta cabinaTarjetaCocheDisponible() {
    CabinaPeajeTarjeta cabinaDisponible = null;
    if (cabinaCoches4.isDisponible()) {
        cabinaCoches4.setDisponible(false);
        if (cabinaCoches4.isAbierta()) {
            cabinaDisponible = cabinaCoches4;
        }
    } else if (cabinaCoches5.isDisponible()) {
        cabinaCoches5.setDisponible(false);
        if (cabinaCoches5.isAbierta()) {
            cabinaDisponible = cabinaCoches5;
        }
    } else if (cabinaCoches6.isDisponible()) {
        cabinaCoches6.setDisponible(false);
        if (cabinaCoches6.isAbierta()) {
            cabinaDisponible = cabinaCoches6;
        }
    }
    return cabinaDisponible;
}

private CabinaPeajeManual cabinaManualCamionDisponible() {
    CabinaPeajeManual cabinaDisponible = null;
    if (cabinaCamiones1.isDisponible()) {
        cabinaCamiones1.setDisponible(false);
        cabinaDisponible = cabinaCamiones1;
```

```
    } else if (cabinaCamiones2.isDisponible()) {
        cabinaCamiones2.setDisponible(false);
        cabinaDisponible = cabinaCamiones2;
    }
    return cabinaDisponible;
  }

  //Método que asigna una cabina automática a un camion
  private CabinaPeajeTarjeta cabinaTarjetaCamionDisponible() {
    CabinaPeajeTarjeta cabinaDisponible = null;
    if (cabinaCamiones3.isDisponible()) {
        cabinaCamiones3.setDisponible(false);
        if (cabinaCamiones3.isAbierta()) {
            cabinaDisponible = cabinaCamiones3;
        }
    } else if (cabinaCamiones4.isDisponible()) {
        cabinaCamiones4.setDisponible(false);
        if (cabinaCamiones4.isAbierta()) {
            cabinaDisponible = cabinaCamiones4;
        }
    }
    return cabinaDisponible;
  }

  private boolean cabinaDisponible(Coche c) {

    if ((cabinaCoches1.isDisponible() || cabinaCoches2.isDisponible()) || cabinaCoches3.isDisponible()) {

        return true;
    } else {
        return ((cabinaCoches4.isAbierta() && cabinaCoches4.isDisponible()) || (cabinaCoches5.isAbierta() &&
cabinaCoches5.isDisponible()) || (cabinaCoches6.isAbierta() && cabinaCoches6.isDisponible()));
    }

  }
  private boolean cabinaDisponibleAmbulancia(Ambulancia a) {

    if ((cabinaCoches1.isDisponible() || cabinaCoches2.isDisponible()) || cabinaCoches3.isDisponible()) {
```

```
            return true;
        } else {
            return ((cabinaCoches4.isAbierta() && cabinaCoches4.isDisponible()) || (cabinaCoches5.isAbierta() &&
cabinaCoches5.isDisponible()) || (cabinaCoches6.isAbierta() && cabinaCoches6.isDisponible()));
        }

    }

    private boolean cabinaDisponibleCamion(Camion c) {

        if (cabinaCamiones1.isDisponible() || cabinaCamiones2.isDisponible()) {

            return true;
        } else {
            return ((cabinaCamiones3.isAbierta() && cabinaCamiones3.isDisponible()) || (cabinaCamiones4.isAbierta() &&
cabinaCamiones4.isDisponible()));
        }

    }
    public void abrirCabinaCamiones4(){
        lock.lock();
        try{
            cabinaCamiones4.setAbierta(true);
                if (cabinaCamiones4.isDisponible()) {
                    camiones.signal();
                }
        }finally{
            lock.unlock();
        }
    }
    //No se por que pero esta cabina no se abre, en la parte de distribuida y la 4 si , cuando hago excatamento lo mismo
     public void abrirCabinaCamiones3(){
        lock.lock();
        try{
            cabinaCamiones3.setAbierta(true);
                if (cabinaCamiones3.isDisponible()) {
                    camiones.signal();
```

```
            }
    }finally{
        lock.unlock();
    }
}
public void cierraCabinas(String cabina){
    try{
        lock.lock();
        switch (cabina) {
            case "CabinaCoches4":
                cabinaCoches4.setAbierta(false);
                break;
            case "CabinaCoches5":
                cabinaCoches5.setAbierta(false);
                break;
            case "CabinaCoches6":
                cabinaCoches6.setAbierta(false);
                break;
            case "CabinaCamiones3":
                cabinaCamiones3.setAbierta(false);
                break;
            case "CabinaCamiones4":
                cabinaCamiones4.setAbierta(false);
                break;
        }
    }finally{
        lock.unlock();
    }
}
public void abreCabinas(String cabina) {
    //Dependiendo del string, despertaremos a los coches de una cabina u otra

    lock.lock();
    try {
        switch (cabina) {
            case "CabinaCoches4":
                cabinaCoches4.setAbierta(true);
                if (cabinaCoches4.isDisponible()) {
```

```
        if (getAmbulanciasEsperando() > 0) {
            ambu.signal();
        }
        else{
            coches.signal();
        }
    }  break;
case "CabinaCoches5":
    cabinaCoches5.setAbierta(true);
    if (cabinaCoches5.isDisponible()) {
        if (getAmbulanciasEsperando() > 0) {
            ambu.signal();
        }
        else{
            coches.signal();
        }
    }  break;
case "CabinaCoches6":
    cabinaCoches6.setAbierta(true);
    if (cabinaCoches6.isDisponible()) {
        if (getAmbulanciasEsperando() > 0) {
            ambu.signal();
        }
        else{
            coches.signal();
        }
    }  break;
case "CabinaCamiones3":
    cabinaCamiones3.setAbierta(true);
    if (cabinaCamiones3.isDisponible()) {
        camiones.signal();
    }
    break;
case "CabinaCamiones4":
    cabinaCamiones4.setAbierta(true);
    if (cabinaCamiones4.isDisponible()){
        camiones.signal();
    }
```

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

public boolean isEvacuar() {
    return evacuar;
}

public void setEvacuar(boolean evacuar) {
    lock.lock();
    try{
    this.evacuar = evacuar;
    }finally{
        lock.unlock();
    }
}



public CabinaPeajeManual getCabinaCoches1() {
    return cabinaCoches1;
}



public void setCabinaCoches1(CabinaPeajeManual cabinaCoches1) {
    this.cabinaCoches1 = cabinaCoches1;
}

public CabinaPeajeManual getCabinaCoches2() {
    return cabinaCoches2;
}

public void setCabinaCoches2(CabinaPeajeManual cabinaCoches2) {
    this.cabinaCoches2 = cabinaCoches2;
```

```
}

public CabinaPeajeManual getCabinaCoches3() {
   return cabinaCoches3;
}

public void setCabinaCoches3(CabinaPeajeManual cabinaCoches3) {
   this.cabinaCoches3 = cabinaCoches3;
}

public CabinaPeajeManual getCabinaCamiones1() {
   return cabinaCamiones1;
}

public void setCabinaCamiones1(CabinaPeajeManual cabinaCamiones1) {
   this.cabinaCamiones1 = cabinaCamiones1;
}

public CabinaPeajeManual getCabinaCamiones2() {
   return cabinaCamiones2;
}

public void setCabinaCamiones2(CabinaPeajeManual cabinaCamiones2) {
   this.cabinaCamiones2 = cabinaCamiones2;
}

public CabinaPeajeTarjeta getCabinaCoches4() {
   return cabinaCoches4;
}

public void setCabinaCoches4(CabinaPeajeTarjeta cabinaCoches4) {
   this.cabinaCoches4 = cabinaCoches4;
}

public CabinaPeajeTarjeta getCabinaCoches5() {
   return cabinaCoches5;
}
```

```
public void setCabinaCoches5(CabinaPeajeTarjeta cabinaCoches5) {
   this.cabinaCoches5 = cabinaCoches5;
}

public CabinaPeajeTarjeta getCabinaCoches6() {
   return cabinaCoches6;
}

public void setCabinaCoches6(CabinaPeajeTarjeta cabinaCoches6) {
   this.cabinaCoches6 = cabinaCoches6;
}

public CabinaPeajeTarjeta getCabinaCamiones3() {
   return cabinaCamiones3;
}

public void setCabinaCamiones3(CabinaPeajeTarjeta cabinaCamiones3) {
   this.cabinaCamiones3 = cabinaCamiones3;
}

public CabinaPeajeTarjeta getCabinaCamiones4() {
   return cabinaCamiones4;
}

public void setCabinaCamiones4(CabinaPeajeTarjeta cabinaCamiones4) {
   this.cabinaCamiones4 = cabinaCamiones4;
}

public Paso getPaso() {
   return paso;
}

public boolean isCabinaCocheDisponible() {
   return cabinaCocheDisponible;
}

public void setCabinaCocheDisponible(boolean cabinaCocheDisponible) {
   this.cabinaCocheDisponible = cabinaCocheDisponible;
```

```
}

public boolean isCabinaCamionDisponible() {
   return cabinaCamionDisponible;
}

public void setCabinaCamionDisponible(boolean cabinaCamionDisponible) {
   this.cabinaCamionDisponible = cabinaCamionDisponible;
}

public int getAmbulanciasEsperando() {
   return ambulanciasEsperando;
}

public void setAmbulanciasEsperando(int ambulanciasEsperando) {
   this.ambulanciasEsperando = ambulanciasEsperando;
}

public JTextField getColaEntradaPeaje() {
   return colaEntradaPeaje;
}

public JTextField getEmpleadoCabinaCamiones2M() {
   return EmpleadoCabinaCamiones2M;
}

public JTextField getEmpleadoCabinaCamiones1M() {
   return EmpleadoCabinaCamiones1M;
}

public JTextField getEmpleadoCabinaCoches1M() {
   return EmpleadoCabinaCoches1M;
}

public JTextField getEmpleadoCabinaCoches2M() {
   return EmpleadoCabinaCoches2M;
}
```

```
public JTextField getEmpleadoCabinaCoches3M() {
    return EmpleadoCabinaCoches3M;
}

public JTextField getCamionCabina1M() {
    return camionCabina1M;
}

public JTextField getCamionCabina2M() {
    return camionCabina2M;
}

public JTextField getCamionCabina3T1() {
    return camionCabina3T1;
}

public JTextField getCamionCabina4T() {
    return camionCabina4T;
}

public JTextField getCocheCabina1M1() {
    return cocheCabina1M1;
}

public JTextField getCocheCabina2M() {
    return cocheCabina2M;
}

public JTextField getCocheCabina3M() {
    return cocheCabina3M;
}

public JTextField getCocheCabina4T() {
    return cocheCabina4T;
}

public JTextField getCocheCabina5T1() {
    return cocheCabina5T1;
```

```
    }

    public JTextField getCocheCabina6T() {
        return cocheCabina6T;
    }

    public FuncionesLog getFg() {
        return fg;
    }

    public ArrayList<Coche> getAforoCoches() {
        return aforoCoches;
    }

    public ArrayList<Camion> getAforoCamiones() {
        return aforoCamiones;
    }

    public ArrayList<Ambulancia> getAforoAmbulancias() {
        return aforoAmbulancias;
    }

}
```

Coche.java

```
package Hilos;

import Concurrencia.CabinaPeaje;
import Concurrencia.CabinaPeajeManual;
import Concurrencia.CabinaPeajeTarjeta;
import Concurrencia.Paso;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Coche extends Thread{

   private String id;
   private CabinaPeaje cabinapeaje;
   private CabinaPeajeManual cab = null;
   private CabinaPeajeTarjeta cpt =  null;

   private Paso paso;

   public Coche(int id, CabinaPeaje cb,Paso paso){
      this.id = "Coche"+id;
      this.cabinapeaje = cb;

      this.paso = paso;
   }



   @Override
   public void run(){
      cabinapeaje.getPaso().mirar();
      cabinapeaje.entradaColaPeajeCoche(this);
      if(!cabinapeaje.isEvacuar()){
      if(this.getCab()!= null){
         //Comprobamos que ha entrado a una cabina manual
         cabinapeaje.getPaso().mirar();
         getCab().cocheCabinaManual(this);
      }
      else{
```

```
        //Comprobamos que ha entrado a una cabina automática
        cabinapeaje.getPaso().mirar();
        getCpt().entradaCabinaCoches(this);


    }
    //Una vez que hemos salido de la cabina, nos saldremos del peaje
    cabinapeaje.getPaso().mirar();
    cabinapeaje.salidaPeajeCoche(this);

}
    System.out.println("Evacuando "+ getIdCoche());
    cabinapeaje.salidaPeajeCoche(this);
}




public CabinaPeajeManual getCab() {
    return cab;
}

public void setCpt(CabinaPeajeTarjeta cpt) {
    this.cpt = cpt;
}

public void setCab(CabinaPeajeManual cab) {
    this.cab = cab;
}

public Paso getPaso() {
    return paso;
}

public void setPaso(Paso paso) {
    this.paso = paso;
}

public CabinaPeajeTarjeta getCpt() {
```

```
        return cpt;
    }

    public String getIdCoche() {
        return id;
    }

    public CabinaPeaje getCabinapeaje() {
        return cabinapeaje;
    }

}
```

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

import Concurrencia.CabinaPeaje;
import Concurrencia.CabinaPeajeManual;
import Concurrencia.Paso;
import Distribuida.Gestor;
import Hilos.Ambulancia;
import Hilos.Camion;
import Hilos.Coche;
import Hilos.CrearCamiones;
import Hilos.CrearCoches;
import Hilos.Empleado;
import Log.FuncionesLog;
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import javax.swing.JButton;

/**
 *
 * @author david
 */
public class main extends javax.swing.JFrame {

    private boolean botonPulsado = false;
    private CabinaPeaje cb;
    private final FuncionesLog fg;
    private final boolean debug = true;
    private Paso paso;

    public main() {
        initComponents();
        fg = new FuncionesLog(debug);
```

```
        paso = new Paso();

        cb = new CabinaPeaje(colaEntradaPeaje, fg,cocheCabina1M1, EmpleadoCabinaCoches1M, cocheCabina2M,
EmpleadoCabinaCoches2M, cocheCabina3M, EmpleadoCabinaCoches3M, camionCabina1M, EmpleadoCabinaCamiones1M,
camionCabina2M, EmpleadoCabinaCamiones2M,cocheCabina4T, cocheCabina5T1, cocheCabina6T, camionCabina3T1,
camionCabina4T,paso);

        Empleado e1 = new Empleado(1, cb.getCabinaCoches1(),cb.getPaso(),cb);

        Empleado e2 = new Empleado(2, cb.getCabinaCoches2(),cb.getPaso(),cb);

        Empleado e3 = new Empleado(3, cb.getCabinaCoches3(),cb.getPaso(),cb);

        Empleado e4 = new Empleado(4, cb.getCabinaCamiones1(),cb.getPaso(),cb);

        Empleado e5 = new Empleado(5, cb.getCabinaCamiones2(),cb.getPaso(),cb);

        e1.start();

        e2.start();

        e3.start();

        e4.start();

        e5.start();

        CrearCoches co = new CrearCoches(cb, fg);


        co.start();

        CrearCamiones cc = new CrearCamiones(cb, fg);

        cc.start();


         //Inicializamos la conexión del RMI

        try{

           Gestor obj = new Gestor(cb);

           Registry registry = LocateRegistry.createRegistry(1099);

           Naming.rebind("//localhost/ObjetoCabinaPeaje",obj);

           System.out.println("El Objeto Peaje ha quedado registrado");

        }

        catch(Exception e) {

           System.out.println("Error: " + e.getMessage());

           e.printStackTrace();

        }



    }


    @SuppressWarnings("unchecked")

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

    private void initComponents() {
```

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



private void botonPararActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(!isBotonPulsado()) //Si no se ha pulsado
    {
        setBotonPulsado(true); //Lo cambiamos a pulsado
        getBotonParar().setText("Reanudar"); //Cambiamos el texto
        paso.cerrar();//Cerramos el paso
    }
    else //Si ya se había pulsado
    {
        setBotonPulsado(false); //Lo cambiamos
        getBotonParar().setText("Parar"); //Cambiamos el texto
        paso.abrir();//Abrimos el paso
    }

}



private void botonAmbulanciaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Ambulancia a = new Ambulancia(1, cb,paso);
    a.start();
}

private void evacuarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
   cb.setEvacuar(true);
}

/**
 * @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(main.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(main.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(main.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(main.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() {
            new main().setVisible(true);
        }
    });
}

public boolean isBotonPulsado() {
    return botonPulsado;
}

public void setBotonPulsado(boolean botonPulsado) {
    this.botonPulsado = botonPulsado;
```

```
}



public JButton getBotonParar() {
    return botonParar;
}


// Variables declaration - do not modify
private javax.swing.JTextField EmpleadoCabinaCamiones1M;
private javax.swing.JTextField EmpleadoCabinaCamiones2M;
private javax.swing.JTextField EmpleadoCabinaCoches1M;
private javax.swing.JTextField EmpleadoCabinaCoches2M;
private javax.swing.JTextField EmpleadoCabinaCoches3M;
private javax.swing.JButton botonAmbulancia;
private javax.swing.JButton botonParar;
private javax.swing.JTextField camionCabina1M;
private javax.swing.JTextField camionCabina2M;
private javax.swing.JTextField camionCabina3T1;
private javax.swing.JTextField camionCabina4T;
private javax.swing.JTextField cocheCabina1M1;
private javax.swing.JTextField cocheCabina2M;
private javax.swing.JTextField cocheCabina3M;
private javax.swing.JTextField cocheCabina4T;
private javax.swing.JTextField cocheCabina5T1;
private javax.swing.JTextField cocheCabina6T;
private javax.swing.JTextField colaEntradaPeaje;
private javax.swing.JButton evacuar;
// End of variables declaration

public Paso getPaso() {
    return paso;
}

public void setPaso(Paso paso) {
    this.paso = paso;
}
```

```
}
```