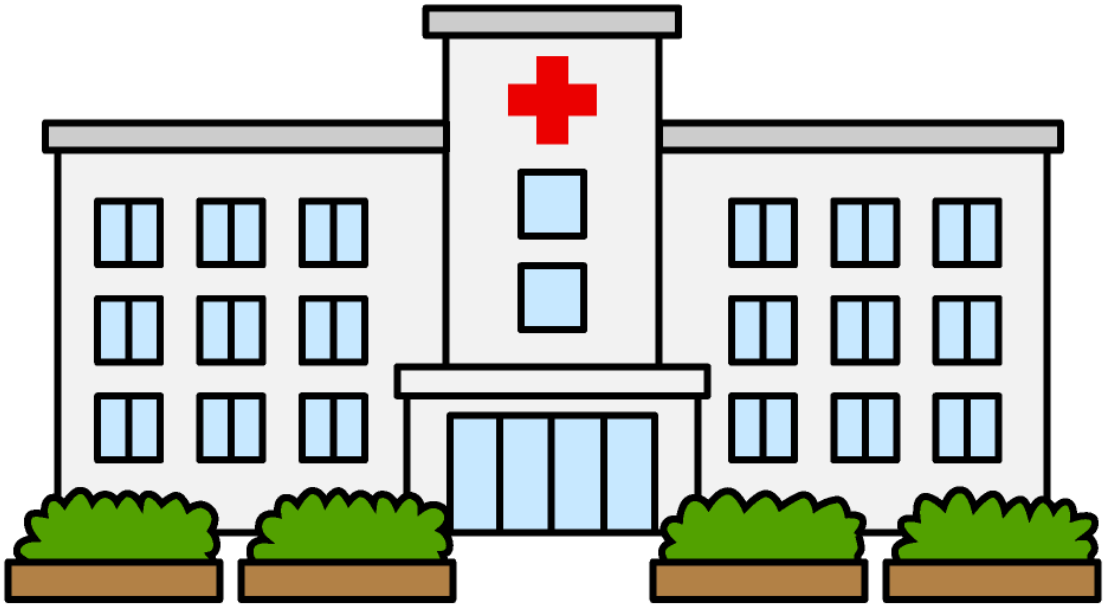
PROGRAMACIÓN AVANZADA

SIMULACIÓN DEL FUNCIONAMIENTO DE UN HOSPITAL



Grado en Ingeniería Informática

Curso 2020/2021 – Convocatoria Ordinaria

03209702Y – Calvo Herrero, Alba

Tabla de contenido

Análisis de alto nivel

La simulación consistirá en usar concurrencia en java para que el funcionamiento del hospital sea correcto, asegurando la exclusión mutua en secciones críticas, que no se produzcan deadlocks, inanición y más problemas de coordinación entre hilos.

Para ello haremos uso de Locks y Conditions, locks de escritura y monitores de mesa.

Diseño General del sistema y de las herramientas de sincronización utilizadas

Locks y Conditions

Los locks permiten acceder a un recurso compartido en una sección crítica en exclusión mutua. De manera que se han usado en las clases Recepción, DepósitoVacunas, SalaDescanso, SalaVacunación y SalaObservación en los métodos de insertar y extraer para evitar que se accedan a los recursos a la vez y se generen problemas de coordinación de los hilos.

Sin embargo, los locks sólo permiten resolver problemas de comunicación entre hilos. Necesitamos objetos condition para resolver los problemas de sincronización.

Dependiendo de la clase hemos usado más o menos Conditions. En todas se ha hecho uso de lleno y vacío para que cuando no haya elementos en una cola se haga vacio.await(), entonces el hilo espera a que haya y pueda proceder con el método de extraer, y lo mismo al revés, para que cuando el número de elementos sea máximo, es decir, que la cola ya esté llena y no haya hueco para insertar, el hilo espera hasta que se extraiga un elemento, entonces se hará lleno.signal() y podrá proceder.

Locks de escritura

Es necesario asegurar que no se escriba a la vez en el log porque se puede perder información. Para ello, en la clase de EvolucionHospital, se crea un writeLock que aseguran la exclusión mutua del fichero evolucionHospital.txt.

Monitores de mesa

En la clase Paciente queremos que el paciente espere a ser vacunado y entonces entre en la sala de observación. Para que espere comprueba que está vacunado con isVacunado(). Este método usa monitores, ya que tiene un wait() y el método setVacunado() tiene un notify() que hará que el hilo continue. Por tanto, el hilo espera hasta que el sanitario haga setVacunado(true) y entonces puede entrar en la sala de observación.

Una vez esté en la sala de observación tendrá que esperar, si tiene reacción, a que un sanitario le atienda. Esto se hará de la misma manera con monitores con los métodos hasReaccion() y setReaccion().

Clases Principales

Hospital

Es la clase main del proyecto. Se encarga de crear los hilos (2000 pacientes, los dos auxiliares y los 10 sanitarios), el fichero log, las salas de vacunación, observación, la recepción, el depósito de vacunas y la recepción.

Además, para la parte de programación distribuida, se crea un gestor que contendrá los datos del hospital. La interfaz local podrá hacer uso del gestor por lo que se le pasará por parámetro, mientras que la interfaz remota no lo tendrá, sino que tendrá que hacer uso de la interfaz de gestor en la red.

Paciente

El paciente entra entre 1 y 3 segundos a la recepción, donde el auxiliar le pasará a la sala de vacunación si está citado y sino, le saca del hospital.

```
sleep(1000 + (int) (2000 * Math.random())); // Entra al hospital entre
1 y 3 segundos
r.insertar(this);
```

Después tenemos este trozo de programa:

```
if (isVacunado()) {
      //Entran a la sala de observacion y esperan 10 s
      reaccion = Math.random() < 0.05; // A un 5% le da reaccion
      pos = so.insertarPaciente(this);
      sleep(10000);
      //Si le da reaccion solicitará a un sanitario disponible
      if (reaccion) {
           so.solicitarSanitario(this);
      }
      //Espera a que no haya reacción o ya ha sido tratada y sale del hos
pital
       if(!hasReaccion()) {
            so.extraerPaciente(pos);
       }
}
```

Espera a que haya sido vacunado con el método synchronized isVacunado() que hará un wait(). Cuando el sanitario haya vacunado al paciente hará un setVacunado(true). Este método hará notify al hilo por lo que puede continuar su ejecución.

```
    public synchronized boolean isVacunado(){
        try{
            wait();
        } catch (InterruptedException ie) {}
        return vacunado;
    }
```

```
    public synchronized void setVacunado(boolean vac) {
        vacunado = vac;
        notify();
    }
```

Por tanto, ya puede pasar a la sala de observación donde espera 10 segundos a que le de reacción. Si la variable reacción es true entonces solicita un sanitario que esté disponible. Para que no salga del hospital antes de que el sanitario acuda a tratarle y le trate, hemos hecho uso de otro monitor.

Comprueba hasReaccion es false (esto se cumplirá tanto como si tenía reacción y ya ha sido tratada, como si no tenía) y sale de la sala de observación. El método hasReaccion tiene un wait y setReaccion un notify, por lo que esperará a que el sanitario acuda, le trate y entonces haga un setReaccion(false), de manera que ya puede salir del hospital.

```
public synchronized boolean hasReaccion() {
        try {
            if (reaccion) {
                wait();
            }
        } catch (InterruptedException ie) {}
        return reaccion;
    }

    public synchronized void setReaccion(boolean reac) {
        reaccion = reac;
        notify();
    }
```

Sanitario

El sanitario empieza en la sala de descanso (sd) donde se cambia en 5 a 8 segundos y escribe en el log cuando entra y cuando sale junto con la hora con LocalDateTime.now.

```
//Entran a la sala de decanso a cambiarse
sd.insertar(this);
eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Sanitario " + id +
" entra al hospital y se cambia en la sala de descanso.");
sleep(5000 + (int) (3000 * Math.random()));
sd.extraer(this);
eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Sanitario " + id +
" sale de la sala de descanso despues de cambiarse.");
```

A partir de aquí, hacemos un while (true) para el resto del comportamiento del hilo, ya que entrará a la sala de vacunación y solo entrará a la sala de descanso cuando haya vacunado a 15 pacientes.

Lo primero que hace es después de haber descansado, comprobar que hay un paciente en la sala de observación (so) al que le ha dado reacción para ir a tratarle.

```
while (true) {

//Comprueba que hay pacientes con reaccion
paciente = so.tratarPaciente();

//Si hay un paciente, lo trata
if(paciente != null) {
puesto = so.insertarSanitario(this, paciente);

 //Sanitario trata al paciente con reaccion (2-5 s)
 eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Paciente "

      + paciente.getIdPaciente()+ " sufre una reaccion y es atendido por"
 + id);
sleep(5000 + (int) Math.random() * 3000);
so.extraerSanitario(puesto);
paciente.setReaccion(false);
paciente = null;
}
```

Tanto si había un paciente al que tratar como si no, entra en la sala de vacunación

```
//Va a la sala de descanso cada 15 pacientes vacunados
for (int vacunados = 0; vacunados < 15; vacunados++) {

    if (sv.puestoListo(puestoVacunacion)) {

        //Coge una vacuna si hay disponibles
        dv.extraer();

        //Vacuna al paciente (3-5 s) y lo lleva a la sala de observacion
        sleep(3000 + (int) (2000 * Math.random()));
        p = sv.extraerPaciente(puestoVacunacion);
        p.setVacunado(true);
        a1.registroVacunacion(this, p);
    }
}
    //Cierra su puesto y sale de la sala de vacunación
    sv.extraerSanitario(this);

    //Descansa de 5 a 8 segundos
    sd.insertar(this);
    eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Sanitario " +
        id + " entra en la sala de descanso.");
    sleep(5000 + (int) (3000 * Math.random()));
    sd.extraer(this);
    setDisponible(true);

    eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Sanitario " +
        id + " sale de la sala de descanso.");
```

Descansa cada 15 pacientes vacunados. Coge una vacuna del depósito de vacunas dv, vacuna al paciente (3-5segundos), lo extrae de la sala de vacunación y le pone setVacunado(true) lo que le hará notify para que vaya a la sala de observación. Le manda el registro de vacunación al auxiliar.

Cuando acaba el for, ya ha vacunado a 15 pacientes, así que sale de la sala de vacunación y entra la sala de descanso sd . Descansa de 5 a 8 segundos, sale del descanso y entonces se ha cabado la iteración del while por lo que vuelve a empezar a comprobar si hay pacientes con reacción y a entrar en la sala de vacunación.

Auxiliar

Si el id es A1 tendrá un comportamiento distinto que si es A2.

Si es el A1 tiene que ir a la recepción y descansar cada 10 pacientes que ha llevado a la sala de vacunación o no.

```
descanso = false;
if (id == "A1") {

    //Descansa cada 10 pacientes que lleva o no a la sala de vacunación
    for (int numPacientes = 0; numPacientes < 10; numPacientes++) {
        sleep(500 + (int) (500 * Math.random())); //tarda entre 0.5 - 1 segundos
        p = r.extraer();
        this.setPaciente(p);

        //Si está citado, entra en la sala de vacunación
        if (p.isCitado()) {
            sv.insertarPaciente(p);

        } else {
            String frase = "Paciente " + p.getIdPaciente() + " ha acudido sin cita.";
            System.out.println(frase);
            eh.escribir(dtf.format(LocalDateTime.now()) + "  " + frase);
        }

    }
    //Descansa tras 10 pacientes
    descanso = true;
    sd.insertar(this); //Entra en la sala de descanso
    eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Auxiliar " + id +
        " entra la sala de descanso.");
    sleep(3000 + (int) (2000 * Math.random()));  //Descansa entre 3 y 5 segundos
    sd.extraer(this); //Sale de la sala de descanso
    eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Auxiliar " + id +
        " sale de la sala de descanso.");

}
```

Si está citado, lo extrae de la recepción y lo lleva a la sala de vacunación y sino imprime por pantalla que no está citado y lo escribe en el log. Cuando descansa se hace descanso = true, esto nos servirá para la interfaz. Descansa entre 3 y 5 segundos, y como esta en un while(true) vuelve a empezar.

Si es el A2 se encarga de crear objetos vacunas e insertarlas en el depósito de vacunación y descansa cada 20 hechas.

```
else { //A2

    //Descansa cada 20 vacunas
    for (int i=0; i<20; i++){
        //Crea una vacuna y la inserta en el deposito
        Object vacuna = new Object();
        sleep(500 + (int) (500 * Math.random()));
        dv.insertar(vacuna);
    }

    descanso = true;
    sd.insertar(this);
    eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Auxiliar " + id +
    " entra la sala de descanso.");
    sleep(1000 + (int) (3000 * Math.random())); // Descansa entre 1 y 4 segundos
    sd.extraer(this);
    eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Auxiliar " + id +
    " sale de la sala de descanso.");
}
```

Recepción

La recepción tiene una cola de pacientes de tipo Array de Paciente, los conditions vacio y lleno, el lock control y los enteros `int in=0, out=0, numElem=0,maximo=0.`

Que nos servirán para asegurar la exclusión mutua.

Tiene tres métodos: insertar, extraer y getColaPacientes.

Insertar adquiere el lock para que nadie más entre. Inserta el paciente en la primera posición disponible es decir, cuando colaPacientes == null, y hasta que eso no suceda la posición in aumentará. Cuando se haya encontrado lo introduce en esa posición y hace vacio.signal.

```
public void insertar(Paciente p) throws InterruptedException, IOException {
    control.lock();
    try{
        in = 0;
        while(colaPacientes[in] != null) {
            in++;
        }
        colaPacientes[in] = p;
        numElem++;
        in = (in+1) % maximo;
        vacio.signal();
        eh.escribir(dtf.format(LocalDateTime.now()) + "  " +"El paciente "
                + p.getIdPaciente() + " ha entrado en la recepcion.");
    } finally { control.unlock();}
}
```

En extraer, si no hay elementos que extraer esperará a que los haya haciendo vacio.await que despertará cuando se inserte un paciente haciendo vacio.signal. Cuando se haya extraído resta un elemento.

```
public Paciente extraer() throws InterruptedException, IOException {
    control.lock();
    while(numElem == 0){
        vacio.await();
    }
    try{
        Paciente p;
        p = colaPacientes[out];
        colaPacientes [out] = null;
        out = (out + 1) % maximo;
        numElem = numElem - 1;
        lleno.signal();
        eh.escribir(dtf.format(LocalDateTime.now()) + "   " +"El paciente "
                + p.getIdPaciente() + " ha salido de la recepcion.");
        return p;

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

SalaVacunación

Tiene un comportamiento similar a recepción, solo que tiene métodos insertar y extraer tanto para pacientes como para sanitarios. Estos tienen diferentes conditions porque hay dos colas en vez de una.

```
private Paciente[] colaPacientes;
private Sanitario[] colaSanitarios;
private int in=0, i = 0, numPacientes=0, numSanitarios=0, maximo=0;
private Lock control = new ReentrantLock();

private Condition lleno = control.newCondition();
private Condition vacio = control.newCondition();
private Condition llenoP = control.newCondition();
private Condition vacioP = control.newCondition();
private Condition listo = control.newCondition();
private Condition abierto = control.newCondition();
```

insertarSanitario inserta en la primera posición disponible, mientras que insertarPaciente lo inserta en la primera posición disponible que tenga un sanitario en ese puesto.

```
public void insertarPaciente(Paciente obj) throws InterruptedException, IOException {
    control.lock();
    while (numPacientes == maximo){
        llenoP.await();
    }
    while (numSanitarios == 0) {
        abierto.await();
    }
    try{
        in = 0;
        while(colaPacientes[in] != null) {
            in++;
        }
        while (colaSanitarios[in] == null) {
            abierto.await();
        }
        colaPacientes[in] = obj;
        //El sanitario tiene un paciente en su puesto por tanto no está disponible
        colaSanitarios[in].setDisponible(false);
        numPacientes++;
        eh.escribir(dtf.format(LocalDateTime.now()) + "  " +"El paciente "
                + obj.getIdPaciente() + " ha entrado en la sala de vacunacion.");
        vacioP.signal();
        listo.signal();

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

Para que el sanitario vacune deberá tener un paciente en el mismo puesto. Esto lo comprueba con puestoListo:

```
public boolean puestoListo(int pos) throws InterruptedException {
    control.lock();
    boolean l = false;
    while(colaPacientes[pos] == null){
        listo.await();
    }
    try {
        if (colaPacientes[pos] != null) {
            l = true;
        }
        return l;
    } finally { control.unlock(); }
}
```

Además tiene un método para que cuando un paciente tenga reacción busque un sanitario disponible. Está disponible cuando el puesto sea distinto de null y además el puesto no esté listo, es decir, que no haya un paciente aún al que vacunar. Entonces notificará al sanitario de que es necesitado por un paciente, y le indicará qué paciente es.

```java
public void buscarSanitario(Paciente paciente) throws InterruptedException {
    control.lock();
    int puesto = 0;
    while(numSanitarios == 0) {
        vacio.await();
    }
    try{

        while(colaSanitarios[puesto] == null && puestoListo(puesto)) {
            puesto++;
        }

        colaSanitarios[puesto].setNecesitaSanitario(true);
        colaSanitarios[puesto].setPaciente(paciente);

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

SalaDescanso

Comportamiento similar al de recepción se inserta y se extraen los objetos con exclusión mutua con conditions y un lock.

SalaObservación

```
public void insertar(Object obj) throws InterruptedException {
    control.lock();

    try{
        in = 0;
        while(cola[in] != null) {
            in++;
        }
        cola[in] = obj;
        numElem++;
        vacio.signal();

    } finally { control.unlock();}
}

public void extraer(Object obj) throws InterruptedException {
    control.lock();
    while(numElem == 0){
        vacio.await();
    }
    try{
        int pos;
        pos = java.util.Arrays.asList(cola).indexOf(obj);
        cola [pos] = null;

        numElem = numElem - 1;
    }
    finally{
        control.unlock();
    }
}
```

DepositoVacunas

Mismo comportamiento que depósitoVacunas. Sólo que getNumElem() devuelve el número de vacunas disponibles.

EvolucionHospital

Es una clase para escribir en el log asegurando la exclusión mutua con un writeLock.

```
private ReadWriteLock lock = new ReentrantReadWriteLock();
private Lock w = lock.writeLock();

public EvolucionHospital (FileWriter fichero) {
    this.fichero = fichero;
}

public void escribir(String frase) throws IOException {
    w.lock();
    try
    {
        fichero = new FileWriter("evolucionHospital.txt", true);
        fichero.write("\n" + frase);

    } finally {
       try {

           if (null != fichero) {
               fichero.close();
           }
       } catch (Exception e2) {
         e2.printStackTrace();
       }
       w.unlock();
    }

}
```

Gestor

Obtiene los datos de todo el hospital. Se usa para gestionar los datos que necesita la interfaz

HiloPintor

Se encarga de recibir los datos del gestor directamente (si es local) o de la interfaz gestor (si es remoto). Puede recibir el gestor o no. Si no lo recibe será remoto y por tanto tendrá disponible la opción de cerrar los puesto de vacunación.

```
if (remoto) {
    gestor = (InterfazGestor) Naming.lookup("//localhost/ObjetoGestor");
    puestosPorCerrar = ventana.getPuestosPorCerrar();
    for (int i=0; i < puestosPorCerrar.length; i++) {
        //Si se ha pedido que se cierre el puesto de vacunacion
        if(puestosPorCerrar[i]) {
            gestor.cerrarPuestoVacunacion(i);
            ventana.setPuestosCerrados();
        }
    }
}
```

Tanto si es remoto como si es local le pasará el resto de datos a Ventana con los siguientes métodos:

```
//Recepcion
ventana.mostrarRecepcion(gestor.getColaRecepcion(),
        gestor.getPacienteRecepcion(), gestor.getAuxiliarR());

//Sala de vacunacion
ventana.mostrarVacunas(gestor.numVacunas(), gestor.getAuxiliarV() );
ventana.mostrarPV1(gestor.getSanitarioVacunacion(0), gestor.getPacienteVacunacion(0));
ventana.mostrarPV2(gestor.getSanitarioVacunacion(1), gestor.getPacienteVacunacion(1));
ventana.mostrarPV3(gestor.getSanitarioVacunacion(2), gestor.getPacienteVacunacion(2));
ventana.mostrarPV4(gestor.getSanitarioVacunacion(3), gestor.getPacienteVacunacion(3));
ventana.mostrarPV5(gestor.getSanitarioVacunacion(4), gestor.getPacienteVacunacion(4));
ventana.mostrarPV6(gestor.getSanitarioVacunacion(5), gestor.getPacienteVacunacion(5));
ventana.mostrarPV7(gestor.getSanitarioVacunacion(6), gestor.getPacienteVacunacion(6));
ventana.mostrarPV8(gestor.getSanitarioVacunacion(7), gestor.getPacienteVacunacion(7));
ventana.mostrarPV9(gestor.getSanitarioVacunacion(8), gestor.getPacienteVacunacion(8));
ventana.mostrarPV10(gestor.getSanitarioVacunacion(9), gestor.getPacienteVacunacion(9));

//Sala descanso
ventana.mostrarDescanso(gestor.getColaDescanso());

//Sala observacion
ventana.mostrarPO1(gestor.getPacienteO(0));
ventana.mostrarPO2(gestor.getPacienteO(1));
ventana.mostrarPO3(gestor.getPacienteO(2));
ventana.mostrarPO4(gestor.getPacienteO(3));
ventana.mostrarPO5(gestor.getPacienteO(4));
ventana.mostrarPO6(gestor.getPacienteO(5));
ventana.mostrarPO7(gestor.getPacienteO(6));
ventana.mostrarPO8(gestor.getPacienteO(7));
ventana.mostrarPO9(gestor.getPacienteO(8));
ventana.mostrarPO10(gestor.getPacienteO(9));
ventana.mostrarPO11(gestor.getPacienteO(10));
ventana.mostrarPO12(gestor.getPacienteO(11));
ventana.mostrarPO13(gestor.getPacienteO(12));
ventana.mostrarPO14(gestor.getPacienteO(13));
ventana.mostrarPO15(gestor.getPacienteO(14));
ventana.mostrarPO16(gestor.getPacienteO(15));
ventana.mostrarPO17(gestor.getPacienteO(16));
ventana.mostrarPO18(gestor.getPacienteO(17));
ventana.mostrarPO19(gestor.getPacienteO(18));
ventana.mostrarPO20(gestor.getPacienteO(19));

//Si es remoto puede cerrar puestos de vacunacion
```

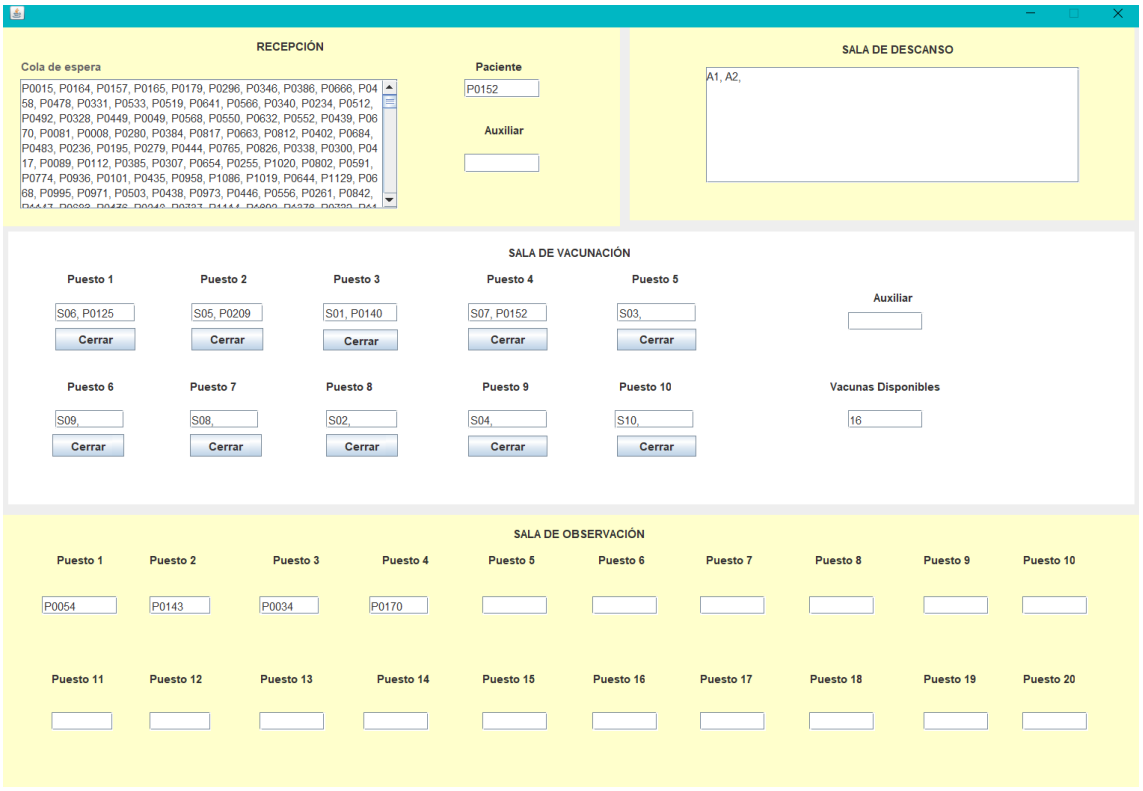
Ventana

Habrá dos ventanas, una local con gestor que no podrá cerrar puestos y una remota sin gestor.

```java
public Ventana() {
    initComponents();
    this.setVisible(true);
    HiloPintor hp = new HiloPintor(this);
    hp.start();
}
public Ventana(Gestor gestor) {
    initComponents();
    this.setVisible(true);
    this.jButtonCerrar1.setEnabled(false);
    this.jButtonCerrar2.setEnabled(false);
    this.jButtonCerrar3.setEnabled(false);
    this.jButtonCerrar4.setEnabled(false);
    this.jButtonCerrar5.setEnabled(false);
    this.jButtonCerrar6.setEnabled(false);
    this.jButtonCerrar7.setEnabled(false);
    this.jButtonCerrar8.setEnabled(false);
    this.jButtonCerrar9.setEnabled(false);
    this.jButtonCerrar10.setEnabled(false);


    HiloPintor hp = new HiloPintor(gestor, this);
    hp.start();
}
```

Tendrá la siguiente pinta:



InterfazGestor

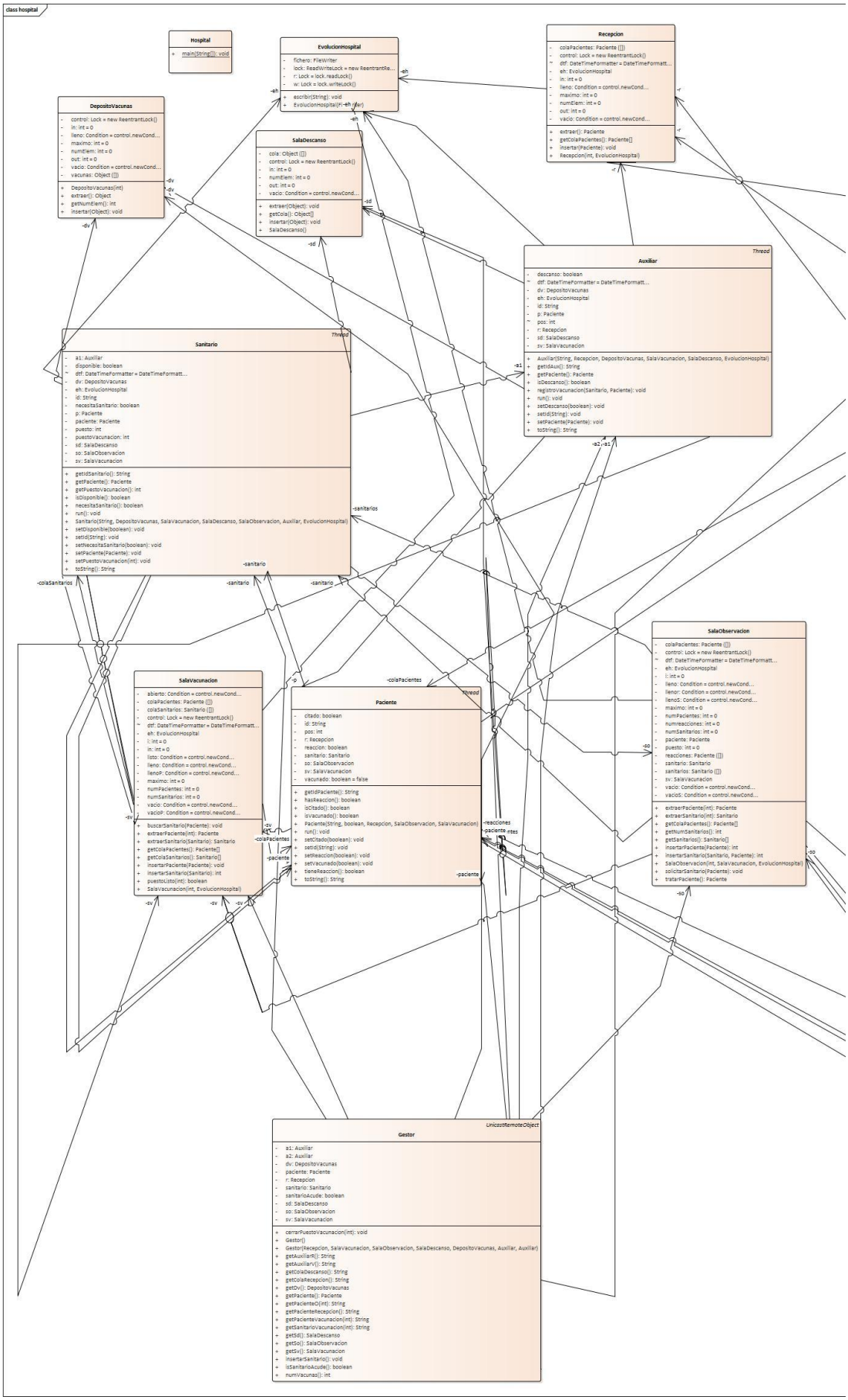
Usa RMI para conectarse con gestor y que este le de la información a través de sus métodos.

Sólo tiene los métodos declarados sin lo que hace el código, esto es lo que verá la clase hilo pintor.

```
public interface InterfazGestor extends Remote {

    public String getAuxiliarV() throws RemoteException;
    public String getPacienteO(int pos) throws RemoteException;
    public String getColaRecepcion() throws RemoteException;
    public String getColaDescanso() throws RemoteException;
    public String getSanitarioVacunacion(int pos) throws RemoteException;
    public String getPacienteVacunacion(int pos) throws RemoteException;
    public String getPacienteRecepcion() throws RemoteException;
    public String getAuxiliarR() throws RemoteException;
    public int numVacunas() throws RemoteException;
    public void cerrarPuestoVacunacion(int puesto) throws RemoteException;
    public void insertarSanitario() throws RemoteException;
}
```

Diagrama de clases



```
package hospital;



import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Auxiliar extends Thread {

    private String id;
    private Recepcion r;
    private SalaVacunacion sv;
    private SalaDescanso sd;
    private Paciente p;
    private EvolucionHospital eh;
    private DepositoVacunas dv;
    private boolean descanso;
    int pos;
    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

    public Auxiliar(String id, Recepcion r, DepositoVacunas dv, SalaVacunacion sv, SalaDescanso
sd, EvolucionHospital eh) {
        this.id = id;
        this.r = r;
        this.dv = dv;
        this.sv = sv;
        this.sd = sd;
        this.eh = eh;
```

```
    this.p = p;
    this.descanso = descanso;
}

public void run(){
    try {
        while (true) {
            descanso = false;
            if (id == "A1") {

                //Descansa cada 10 pacientes que lleva o no a la sala de vacunación
                for (int numPacientes = 0; numPacientes < 10; numPacientes++) {
                    sleep(500 + (int) (500 * Math.random())); //tarda entre 0.5 - 1 segundos
                    p = r.extraer();
                    this.setPaciente(p);

                    //Si está citado, entra en la sala de vacunación
                    if (p.isCitado()) {
                        sv.insertarPaciente(p);

                    } else {
                        String frase = "Paciente " + p.getIdPaciente() + " ha acudido sin cita.";
                        System.out.println(frase);
                        eh.escribir(dtf.format(LocalDateTime.now()) + "  " + frase);
                    }

                }
                //Descansa tras 10 pacientes
                descanso = true;
                sd.insertar(this); //Entra en la sala de descanso
```

```
            eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Auxiliar " + id + " entra la sala de descanso.");
            sleep(3000 + (int) (2000 * Math.random()));  //Descansa entre 3 y 5 segundos
            sd.extraer(this); //Sale de la sala de descanso
            eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Auxiliar " + id + " sale de la sala de descanso.");

          }
          else { //A2

            //Descansa cada 20 vacunas
            for (int i=0; i<20; i++){
              //Crea una vacuna y la inserta en el deposito
              Object vacuna = new Object();
              sleep(500 + (int) (500 * Math.random()));
              dv.insertar(vacuna);
            }

            descanso = true;
            sd.insertar(this);
            eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Auxiliar " + id + " entra la sala de descanso.");
            sleep(1000 + (int) (3000 * Math.random())); // Descansa entre 1 y 4 segundos
            sd.extraer(this);
            eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Auxiliar " + id + " sale de la sala de descanso.");
          }
        }
      } catch (InterruptedException e) {} catch (IOException ex) {
        Logger.getLogger(Auxiliar.class.getName()).log(Level.SEVERE, null, ex);
      }
    }
```

```
public void registroVacunacion(Sanitario s, Paciente p) throws IOException{
   // Lleva el registro de vacunación, lo muestra por pantalla y lo escribe en el log
   String frase = "Paciente " + p.getIdPaciente() + " vacunado en el puesto " +
      (s.getPuestoVacunacion()+1) + " por " + s.getIdSanitario();
   System.out.println(frase);
   eh.escribir(dtf.format(LocalDateTime.now()) + "  " + frase);
}

public String getIdAux() {
   return id;
}

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

public Paciente getPaciente() {
   return p;
}

public void setPaciente(Paciente p) {
   this.p = p;
}

public boolean isDescanso() {
   return descanso;
}

public void setDescanso(boolean descanso) {
   this.descanso = descanso;
```

```
    }


    @Override
    public String toString() {
        return id;
    }
}

package hospital;

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

public class DepositoVacunas {
    private Object[] vacunas;
    private int in=0, out=0, numElem=0,maximo=0;
    private Lock control = new ReentrantLock();
    private Condition lleno = control.newCondition();
    private Condition vacio = control.newCondition();

    public DepositoVacunas(int max){
        this.maximo = max;
        this.numElem = numElem;
        vacunas = new Object[max];
    }

    public void insertar(Object obj) throws InterruptedException {
        control.lock();
```

```
    while (numElem == maximo){
        lleno.await();
    }
    try{
        vacunas[in] = obj;
        numElem++;
        in = (in+1) % maximo;
        vacio.signal();
        System.out.println("vacunas: " + numElem);
    } finally { control.unlock();}
}

public Object extraer() throws InterruptedException {
    control.lock();
    while(numElem == 0){
        vacio.await();
    }
    try{
        Object obj;
        obj = vacunas[out];
        vacunas [out] = null;
        out = (out + 1) % maximo;
        numElem = numElem - 1;
        lleno.signal();
        System.out.println("vacunas: " + numElem);
        return obj;
    }
    finally{
        control.unlock();
    }
}
```

```
    public int getNumElem() {
        return numElem;
    }

}

package hospital;

import java.io.FileWriter;
import java.io.IOException;
import java.util.concurrent.locks.*;

public class EvolucionHospital {
    private FileWriter fichero;
    private ReadWriteLock lock = new ReentrantReadWriteLock();
    private Lock w = lock.writeLock();

    public EvolucionHospital (FileWriter fichero) {
        this.fichero = fichero;
    }

    public void escribir(String frase) throws IOException {
        w.lock();
        try
        {
            fichero = new FileWriter("evolucionHospital.txt", true);
            fichero.write("\n" + frase);

        } finally {
          try {
```

```
            if (null != fichero) {
               fichero.close();
            }
         } catch (Exception e2) {
           e2.printStackTrace();
         }
         w.unlock();
       }

   }

}

package hospital;

import interfaz.InterfazGestor;
import java.io.IOException;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Gestor extends UnicastRemoteObject implements InterfazGestor {

   public Gestor() throws RemoteException{ }

   private Recepcion r;
   private SalaVacunacion sv;
   private SalaObservacion so;
   private SalaDescanso sd;
```

```
    private DepositoVacunas dv;
    private Auxiliar a1;
    private Auxiliar a2;
    private Sanitario sanitario;
    private boolean sanitarioAcude;
    private Paciente paciente;


    public Gestor(Recepcion r, SalaVacunacion sv, SalaObservacion so, SalaDescanso sd,
DepositoVacunas dv, Auxiliar a1, Auxiliar a2) throws RemoteException{
        this.r = r;
        this.sv = sv;
        this.so = so;
        this.sd = sd;
        this.dv = dv;
        this.a1 = a1;
        this.a2 = a2;
    }


    public String getAuxiliarV() {
        String id = "";
        if (!a2.isDescanso()) {
            id = a2.getIdAux();}
        return id;
    }

    public String getPacienteO(int pos) {
        sanitarioAcude = false;
        String paciente = "";
        if(so.getColaPacientes()[pos] != null) {
```

```
            paciente = so.getColaPacientes()[pos].toString();
            if(so.getColaPacientes()[pos].tieneReaccion()) {
                sanitarioAcude = true;
                if (so.getSanitarios()[pos] != null) {
                    paciente += ", " + so.getSanitarios()[pos].getIdSanitario();
                }
            }
        }
        return paciente;
    }

    public boolean isSanitarioAcude() {
        return sanitarioAcude;
    }

    public Paciente getPaciente() {
        return paciente;
    }

    public void cerrarPuestoVacunacion(int puesto) {
        try {

            sanitario = sv.extraerSanitario(sv.getColaSanitarios()[puesto]);
            sanitario.setDisponible(false);
            sd.insertar(sanitario);

        } catch (InterruptedException ie) {} catch (IOException ex) {
            Logger.getLogger(Gestor.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    public void insertarSanitario() {
```

```
    try {
        sv.insertarSanitario(sanitario);
    } catch(InterruptedException ie) {} catch (IOException ex) {
        Logger.getLogger(Gestor.class.getName()).log(Level.SEVERE, null, ex);
    }
}
public String getColaRecepcion() {
    String colaRecepcion = "";
    Paciente[] colaPacientes = r.getColaPacientes();
    for (int i=0; i < colaPacientes.length ; i++) {
        if (colaPacientes[i] != null) {
            colaRecepcion += colaPacientes[i].getIdPaciente() + ", ";
        }
    }
    return colaRecepcion;
}
public String getColaDescanso() {
    String colaDescanso = "";
    Object[] cola = sd.getCola();
    for (int i=0; i < cola.length ; i++) {
        if (cola[i] != null) {
            colaDescanso += cola[i].toString() + ", ";
        }
    }
    return colaDescanso;
}
public String getSanitarioVacunacion(int pos){
    String sanitario = "";
    if(sv.getColaSanitarios()[pos] != null) {
        sanitario = sv.getColaSanitarios()[pos].toString();
    }
```

```
    return sanitario;
}

public String getPacienteVacunacion(int pos){
    String paciente = "";
    if(sv.getColaPacientes()[pos] != null) {
        paciente = sv.getColaPacientes()[pos].toString();
    }
    return paciente;
}

public String getPacienteRecepcion() {
    String id = "";
    if(a1.getPaciente() != null) {
        id = a1.getPaciente().getIdPaciente();
    }
    return id;
}


public String getAuxiliarR() {
    String id = "";
    if (!a1.isDescanso()) {
        id = a1.getIdAux();}
    return id;
}

public int numVacunas() {
    return dv.getNumElem();
}
```

```
    public SalaVacunacion getSv() {
        return sv;
    }

    public SalaObservacion getSo() {
        return so;
    }

    public SalaDescanso getSd() {
        return sd;
    }

    public DepositoVacunas getDv() {
        return dv;
    }

}

package hospital;

import interfaz.Ventana;
import java.io.FileWriter;
import java.io.IOException;
import static java.lang.Thread.sleep;
import java.rmi.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Hospital {

    public static void main(String[] args) throws InterruptedException, IOException {
```

```
// Fichero log
FileWriter fichero = null;
fichero = new FileWriter("evolucionHospital.txt");
EvolucionHospital eh = new EvolucionHospital(fichero);

// Creación sala vacunación, sala observación  y recepción
Recepcion r = new Recepcion(2000, eh);
SalaVacunacion sv = new SalaVacunacion(10, eh);
SalaObservacion so = new SalaObservacion(20, sv, eh);
SalaDescanso sd = new SalaDescanso();
DepositoVacunas dv = new DepositoVacunas(2000);


// Creación de los dos auxiliares
Auxiliar a1 = new Auxiliar("A1", r, dv, sv, sd, eh);
Auxiliar a2 = new Auxiliar("A2", r, dv, sv, sd, eh);
a1.start();
a2.start();

// Creación de los 10 sanitarios
for (int j=1; j<11; j++) {
   String id = String.format("%02d", j);
   Sanitario s = new Sanitario(id, dv, sv, sd, so, a1, eh);
   s.start();
}

// Creación de 2000 pacientes
boolean citado = true;
int noCitados = 0;
for (int i=1; i<2001; i++) {
```

```
            if (noCitados < 20) { //1% de 2000 pacientes, son 20 pacientes no citados
               citado = Math.random() < 0.5;
               if (!citado) noCitados++;
            }
            else {
               citado = true;
            }
            String id = String.format("%04d", i);
            Paciente p = new Paciente(id, citado, r, so, sv);
            p.start();
         }

         // Creación del gestor y de las interfaces
         try {
            Gestor gestor = new Gestor(r, sv, so, sd, dv, a1, a2);
            Registry registry = LocateRegistry.createRegistry(1099);
            Naming.rebind("//localhost/ObjetoGestor", gestor);
            Ventana v = new Ventana(gestor); // Ventana local
            Ventana v2 = new Ventana();      // Ventana remota
         } catch (Exception e) {
            System.out.println(" Error: " +  e.getMessage());
            e.printStackTrace();
         }
      }
   }
```

```
package hospital;

import java.io.IOException;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.management.monitor.Monitor;

public class Paciente extends Thread {

    private String id ;
    private boolean citado;
    private Recepcion r;
    private SalaObservacion so;
    private SalaVacunacion sv;
    private int pos;
    private boolean reaccion;
    private boolean vacunado = false;



    public Paciente(String id, boolean citado, Recepcion r, SalaObservacion so, SalaVacunacion
sv) throws InterruptedException {
        this.id = "P" + id;
        this.citado = citado;
        this.r = r;
        this.so = so;
        this.sv = sv;
        this.vacunado = false;
        this.reaccion = false;
    }

    public void run(){

        try {
            sleep(1000 + (int) (2000 * Math.random())); // Entra al hospital entre 1 y 3 segundos
```

```
        r.insertar(this);

        if (isVacunado()) {
            //Entran a la sala de observacion y esperan 10 s
            reaccion = Math.random() < 0.05; // A un 5% le da reaccion
            pos = so.insertarPaciente(this);
            sleep(10000);
            //Si le da reaccion solicitará a un sanitario disponible
            if (reaccion) {
                so.solicitarSanitario(this);
            }
            //Espera a que no haya reacción o ya ha sido tratada y sale del hospital
            if(!hasReaccion()) {
                so.extraerPaciente(pos);
            }
        }

    } catch (InterruptedException e) {} catch (IOException ex) {
        Logger.getLogger(Paciente.class.getName()).log(Level.SEVERE, null, ex);
    }

}


//Uso de monitores

public synchronized boolean hasReaccion() {
    try {
        if (reaccion) {
            wait();
        }
```

```
    } catch (InterruptedException ie) {}
    return reaccion;
}

public synchronized void setReaccion(boolean reac) {
    reaccion = reac;
    notify();
}

public synchronized boolean isVacunado(){
    try{
        wait();
    } catch (InterruptedException ie) {}
    return vacunado;
}

public synchronized void setVacunado(boolean vac) {
    vacunado = vac;
    notify();
}

// Getters y setters
public boolean tieneReaccion() {
    return reaccion;
}

public String getIdPaciente() {
    return id;
}

public void setId(String id) {
```

```
        this.id = id;
    }

    public boolean isCitado() {
        return citado;
    }

    public void setCitado(boolean citado) {
        this.citado = citado;
    }

    @Override
    public String toString() {
        return id;
    }

}

package hospital;

import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Arrays;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class Recepcion {

    private Paciente[] colaPacientes;
```

```
private int in=0, out=0, numElem=0,maximo=0;
private Lock control = new ReentrantLock();
private Condition vacio = control.newCondition();
private Condition lleno = control.newCondition();
private EvolucionHospital eh;
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

public Recepcion(int max, EvolucionHospital eh){
   this.maximo = max;
   this.eh = eh;
   this.colaPacientes = new Paciente[max];
}

public void insertar(Paciente p) throws InterruptedException, IOException {
   control.lock();
   try{
      in = 0;
      while(colaPacientes[in] != null) {
         in++;
      }
      colaPacientes[in] = p;
      numElem++;
      in = (in+1) % maximo;
      vacio.signal();
      eh.escribir(dtf.format(LocalDateTime.now()) + "  " +"El paciente "
            + p.getIdPaciente() + " ha entrado en la recepcion.");
   } finally { control.unlock();}
}

public Paciente extraer() throws InterruptedException, IOException {
   control.lock();
```

```
        while(numElem == 0){
            vacio.await();
        }
        try{
            Paciente p;
            p = colaPacientes[out];
            colaPacientes [out] = null;
            out = (out + 1) % maximo;
            numElem = numElem - 1;
            lleno.signal();
            eh.escribir(dtf.format(LocalDateTime.now()) + "  " +"El paciente " + p.getIdPaciente() + "
ha salido de la recepcion.");
            return p;

        } finally{
            control.unlock();
        }
    }

    public Paciente[] getColaPacientes() {
        return this.colaPacientes;
    }

}

package hospital;

import static java.lang.Thread.sleep;
import java.util.Arrays;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
```

```
import java.util.concurrent.locks.ReentrantLock;

public class SalaDescanso {
   private Object[] cola;
   private int in=0, out=0, numElem=0;
   private Lock control = new ReentrantLock();
   private Condition vacio = control.newCondition();

   public SalaDescanso(){
      cola = new Object[2000];
      this.cola = cola;
   }

   public void insertar(Object obj) throws InterruptedException {
      control.lock();

      try{
         in = 0;
         while(cola[in] != null) {
            in++;
         }
         cola[in] = obj;
         numElem++;
         vacio.signal();

      } finally { control.unlock();}
   }

   public void extraer(Object obj) throws InterruptedException {
      control.lock();
      while(numElem == 0){
```

```
            vacio.await();
        }
        try{
            int pos;
            pos = java.util.Arrays.asList(cola).indexOf(obj);
            cola [pos] = null;

            numElem = numElem - 1;
        }
        finally{
            control.unlock();
        }
    }

    public Object[] getCola() {
        return cola;
    }

}

package hospital;

import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Arrays;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class SalaObservacion {
```

```
private Paciente[] colaPacientes;
private Paciente[] reacciones;
private Sanitario[] sanitarios;
private Sanitario sanitario;
private Paciente paciente;
private int i=0, puesto=0, numPacientes=0,maximo=0, numreacciones= 0, numSanitarios=0;
private Lock control = new ReentrantLock();
private Condition lleno = control.newCondition();
private Condition vacio = control.newCondition();
private Condition llenoS = control.newCondition();
private Condition vacioS = control.newCondition();
private Condition llenor = control.newCondition();
private SalaVacunacion sv;
private EvolucionHospital eh;
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");


public SalaObservacion(int max, SalaVacunacion sv, EvolucionHospital eh){
   this.maximo = max;
   colaPacientes = new Paciente[max];
   sanitarios = new Sanitario[max];
   reacciones = new Paciente[max];
   this.sv = sv;
   this.sanitario = sanitario;
   this.eh = eh;
   this.colaPacientes = colaPacientes;

}

public int insertarPaciente(Paciente obj) throws InterruptedException, IOException {
```

```
    control.lock();
    while (numPacientes == maximo){
        lleno.await();
    }
    try{
        i = 0;
        while(colaPacientes[i] != null) {
            i++;
        }
        colaPacientes[i] = obj;
        numPacientes++;
        vacio.signal();
        eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Paciente " + obj.getIdPaciente() +
                " entra en la sala de observacion en el puesto " + (i+1));
        return i;
    } finally { control.unlock();}
}

public Paciente extraerPaciente(int pos) throws InterruptedException, IOException {
    control.lock();
    while(numPacientes == 0){
        vacio.await();
    }
    try{
        Paciente obj;
        obj = colaPacientes[pos];
        colaPacientes[pos] = null;
        numPacientes = numPacientes - 1;
        lleno.signal();

        eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Paciente " + obj.getIdPaciente() +
```

```
                " sale de la sala de observacion en el puesto " + (pos+1));
            return obj;
        }
        finally{
            control.unlock();
        }
    }

    public int insertarSanitario(Sanitario obj, Paciente p) throws InterruptedException,
IOException {
        control.lock();
        while (numSanitarios == maximo){
            llenoS.await();
        }
        try{
            i = 0;
            while(colaPacientes[i]== null && !colaPacientes[i].equals(p)) {
                i++;
            }
            sanitarios[i] = obj;
            numSanitarios++;
            vacioS.signal();
            return i;
        } finally { control.unlock();}
    }

    public Sanitario extraerSanitario(int pos) throws InterruptedException, IOException {
        control.lock();
        while(numSanitarios == 0){
            vacioS.await();
        }
```

```
    try{
        Sanitario obj;
        obj = sanitarios[pos];
        sanitarios[pos] = null;
        numSanitarios = numSanitarios - 1;
        llenoS.signal();
        return obj;
    }
    finally{
        control.unlock();
    }
}

public void solicitarSanitario(Paciente p) throws InterruptedException {
    control.lock();
    while(numreacciones == maximo) {
        llenor.await();
    }
    try {
        i = 0;
        while(reacciones[i] != null) {
            i++;
        }
        reacciones[i] = p;
        numreacciones++;
    }
    finally {
        control.unlock();
    }
}
```

```
public Paciente tratarPaciente() throws InterruptedException  {
   control.lock();

   try {
      i = 0;
      while(i<reacciones.length && reacciones[i] == null) {
         i++;
      }
      if(i!= reacciones.length) {
         paciente = reacciones[i];
         reacciones[i] = null;
         numreacciones--;
         llenor.signal();
      } else {
         paciente = null;
      }
      return paciente;
   }
   finally {
      control.unlock();
   }
}

public int getNumSanitarios() {
   return numSanitarios;
}

public Paciente[] getColaPacientes() {
   return colaPacientes;
}
```

```
    public Sanitario[] getSanitarios() {
        return sanitarios;
    }


}

package hospital;

import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Arrays;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class SalaVacunacion {

    private Paciente[] colaPacientes;
    private Sanitario[] colaSanitarios;
    private int in=0, i = 0, numPacientes=0, numSanitarios=0, maximo=0;
    private Lock control = new ReentrantLock();

    private Condition lleno = control.newCondition();
    private Condition vacio = control.newCondition();
    private Condition llenoP = control.newCondition();
    private Condition vacioP = control.newCondition();
    private Condition listo = control.newCondition();
    private Condition abierto = control.newCondition();
```

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
private EvolucionHospital eh;

public SalaVacunacion(int max, EvolucionHospital eh){
   this.maximo = max;
   this.eh = eh;
   colaPacientes = new Paciente[max];
   colaSanitarios = new Sanitario[max];
}

public int insertarSanitario(Sanitario obj) throws InterruptedException, IOException {
   control.lock();
   while (numSanitarios == maximo) {
      lleno.await();
   }
   try {
      i = 0;
      while(colaSanitarios[i] != null) {
         i++;
      }
      colaSanitarios[i] = obj;
      int pos = i;
      numSanitarios++;
      eh.escribir(dtf.format(LocalDateTime.now()) + "  " +"El paciente "
            + obj.getIdSanitario() + " ha entrado en la sala de vacunacion.");
      vacio.signal();
      abierto.signal();
      return pos;
   } finally { control.unlock();}
}
```

```
public void insertarPaciente(Paciente obj) throws InterruptedException, IOException {
   control.lock();
   while (numPacientes == maximo){
      llenoP.await();
   }
   while (numSanitarios == 0) {
      abierto.await();
   }
   try{
      in = 0;
      while(colaPacientes[in] != null) {
         in++;
      }
      while (colaSanitarios[in] == null) {
         abierto.await();
      }
      colaPacientes[in] = obj;
      //El sanitario tiene un paciente en su puesto por tanto no está disponible
      colaSanitarios[in].setDisponible(false);
      numPacientes++;
      eh.escribir(dtf.format(LocalDateTime.now()) + "  " +"El paciente "
            + obj.getIdPaciente() + " ha entrado en la sala de vacunacion.");
      vacioP.signal();
      listo.signal();

   } finally { control.unlock();}
}

public Paciente extraerPaciente(int pos) throws InterruptedException, IOException {
   control.lock();
   while(numPacientes == 0){
```

```
        vacioP.await();
    }
    try{
        Paciente obj;
        obj = colaPacientes[pos];
        colaPacientes[pos] = null;
        numPacientes = numPacientes - 1;
        llenoP.signal();
        eh.escribir(dtf.format(LocalDateTime.now()) + "  " +"El paciente "
                + obj.getIdPaciente() + " ha salido de la sala de vacunacion.");
        return obj;
    }
    finally{
        control.unlock();
    }
}

public Sanitario extraerSanitario(Sanitario obj) throws InterruptedException, IOException {
    control.lock();
    while(numSanitarios == 0){
        vacio.await();
    }
    try{

        int i;
        i = java.util.Arrays.asList(colaSanitarios).indexOf(obj);
        colaSanitarios[i] = null;
        numSanitarios = numSanitarios - 1;
        obj.setDisponible(true);
        eh.escribir(dtf.format(LocalDateTime.now()) + "  " +"El paciente "
                + obj.getIdSanitario() + " ha salido de la sala de vacunacion.");
```

```
            lleno.signal();
            return obj;
        }
        finally{
            control.unlock();
        }
    }

    public boolean puestoListo(int pos) throws InterruptedException {
        control.lock();
        boolean l = false;
        while(colaPacientes[pos] == null){
            listo.await();
        }
        try {
            if (colaPacientes[pos] != null) {
                l = true;
            }
            return l;
        } finally { control.unlock(); }
    }

    public void buscarSanitario(Paciente paciente) throws InterruptedException {
        control.lock();
        int puesto = 0;
        while(numSanitarios == 0) {
            vacio.await();
        }
        try{

            while(colaSanitarios[puesto] == null && puestoListo(puesto)) {
```

```
            puesto++;
          }

          colaSanitarios[puesto].setNecesitaSanitario(true);
          colaSanitarios[puesto].setPaciente(paciente);

      } finally { control.unlock();}
   }

   public Paciente[] getColaPacientes() {
      return colaPacientes;
   }

   public Sanitario[] getColaSanitarios() {
      return colaSanitarios;
   }

}
```

```
package hospital;

import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Sanitario extends Thread {

   private String id;
   private SalaVacunacion sv;
```

```
    private SalaObservacion so;
    private SalaDescanso sd;
    private Paciente p;
    private DepositoVacunas dv;
    private int puestoVacunacion;
    private Auxiliar a1;
    private boolean disponible;
    private boolean necesitaSanitario;
    private Paciente paciente;
    private EvolucionHospital eh;
    private int puesto;
    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

    public Sanitario(String id, DepositoVacunas dv, SalaVacunacion sv, SalaDescanso sd,
SalaObservacion so, Auxiliar a1, EvolucionHospital eh) throws InterruptedException {
        this.id = "S" + id;
        this.dv = dv;
        this.sv = sv;
        this.sd = sd;
        this.so = so;
        this.puestoVacunacion = puestoVacunacion;
        this.a1 = a1;
        this.p = p;
        this.disponible = disponible;
        this.necesitaSanitario = false;
        this.paciente = paciente;
        this.eh = eh;
    }

    public void run() {
        try {
```

```
        //Entran a la sala de decanso a cambiarse
        sd.insertar(this);
        eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Sanitario " + id + " entra al hospital y se cambia en la sala de descanso.");
        sleep(5000 + (int) (3000 * Math.random()));
        sd.extraer(this);
        eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Sanitario " + id + " sale de la sala de descanso despues de cambiarse.");

        while (true) {

          //Comprueba que hay pacientes con reaccion
          paciente = so.tratarPaciente();

          //Si hay un paciente, lo trata
          if(paciente != null) {
            puesto = so.insertarSanitario(this, paciente);

            //Sanitario trata al paciente con reaccion (2-5 s)
            eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Paciente "
                  + paciente.getIdPaciente()+ " sufre una reaccion y es atendido por" + id);
            sleep(5000 + (int) Math.random() * 3000);
            so.extraerSanitario(puesto);
            paciente.setReaccion(false);
            paciente = null;
          }
          //Entra en la sala de vacunación
          puestoVacunacion = sv.insertarSanitario(this);

          //Va a la sala de descanso cada 15 pacientes vacunados
          for (int vacunados = 0; vacunados < 15; vacunados++) {
```

```
            if (sv.puestoListo(puestoVacunacion)) {

               //Coge una vacuna si hay disponibles
               dv.extraer();

               //Vacuna al paciente (3-5 s) y lo lleva a la sala de observacion
               sleep(3000 + (int) (2000 * Math.random()));
               p = sv.extraerPaciente(puestoVacunacion);
               p.setVacunado(true);
               a1.registroVacunacion(this, p);
            }
         }
            //Cierra su puesto y sale de la sala de vacunación
            sv.extraerSanitario(this);

            //Descansa de 5 a 8 segundos
            sd.insertar(this);
            eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Sanitario " + id + " entra en la sala de descanso.");
            sleep(5000 + (int) (3000 * Math.random()));
            sd.extraer(this);
            setDisponible(true);

            eh.escribir(dtf.format(LocalDateTime.now()) + "  " + "Sanitario " + id + " sale de la sala de descanso.");

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

```
}

public Paciente getPaciente() {
   return paciente;
}

public void setPaciente(Paciente paciente) {
   this.paciente = paciente;
}

public String getIdSanitario() {
   return id;
}

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

public boolean necesitaSanitario() {
   return necesitaSanitario;
}

public void setNecesitaSanitario(boolean necesitaSanitario) {
   this.necesitaSanitario = necesitaSanitario;
}

public int getPuestoVacunacion() {
   return puestoVacunacion;
}

public void setPuestoVacunacion(int puestoVacunacion) {
```

```
        this.puestoVacunacion = puestoVacunacion;
    }

    public boolean isDisponible() {
        return disponible;
    }

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

    @Override
    public String toString() {
        return id;
    }

}

package interfaz;

import hospital.*;
import java.net.MalformedURLException;
import java.rmi.*;
import java.util.logging.Level;
import java.util.logging.Logger;

public class HiloPintor extends Thread {

    private InterfazGestor gestor;
    private Ventana ventana;
    Paciente [] colaPacientes;
```

```
private boolean[] puestosPorCerrar = new boolean[10];
private boolean remoto = false;

public HiloPintor(Ventana ventana) {
   this.ventana = ventana;
}
public HiloPintor(Gestor gestor, Ventana ventana) {
   this.gestor = gestor;
   this.ventana = ventana;
}

public void run() {
   if (gestor == null) {
      remoto = true;
   }
   while (true) {

      try {
         if (remoto) {
            gestor = (InterfazGestor) Naming.lookup("//localhost/ObjetoGestor");
            puestosPorCerrar = ventana.getPuestosPorCerrar();
            for (int i=0; i < puestosPorCerrar.length; i++) {
               //Si se ha pedido que se cierre el puesto de vacunacion
               if(puestosPorCerrar[i]) {
                  gestor.cerrarPuestoVacunacion(i);
                  ventana.setPuestosCerrados();
               }
            }
         }
```

```
        //Recepcion
        ventana.mostrarRecepcion(gestor.getColaRecepcion(),
            gestor.getPacienteRecepcion(), gestor.getAuxiliarR());

        //Sala de vacunacion
        ventana.mostrarVacunas(gestor.numVacunas(), gestor.getAuxiliarV() );
        ventana.mostrarPV1(gestor.getSanitarioVacunacion(0),
gestor.getPacienteVacunacion(0));
        ventana.mostrarPV2(gestor.getSanitarioVacunacion(1),
gestor.getPacienteVacunacion(1));
        ventana.mostrarPV3(gestor.getSanitarioVacunacion(2),
gestor.getPacienteVacunacion(2));
        ventana.mostrarPV4(gestor.getSanitarioVacunacion(3),
gestor.getPacienteVacunacion(3));
        ventana.mostrarPV5(gestor.getSanitarioVacunacion(4),
gestor.getPacienteVacunacion(4));
        ventana.mostrarPV6(gestor.getSanitarioVacunacion(5),
gestor.getPacienteVacunacion(5));
        ventana.mostrarPV7(gestor.getSanitarioVacunacion(6),
gestor.getPacienteVacunacion(6));
        ventana.mostrarPV8(gestor.getSanitarioVacunacion(7),
gestor.getPacienteVacunacion(7));
        ventana.mostrarPV9(gestor.getSanitarioVacunacion(8),
gestor.getPacienteVacunacion(8));
        ventana.mostrarPV10(gestor.getSanitarioVacunacion(9),
gestor.getPacienteVacunacion(9));

        //Sala descanso
        ventana.mostrarDescanso(gestor.getColaDescanso());

        //Sala observacion
        ventana.mostrarPO1(gestor.getPacienteO(0));
        ventana.mostrarPO2(gestor.getPacienteO(1));
        ventana.mostrarPO3(gestor.getPacienteO(2));
        ventana.mostrarPO4(gestor.getPacienteO(3));
```

```
            ventana.mostrarPO5(gestor.getPacienteO(4));
            ventana.mostrarPO6(gestor.getPacienteO(5));
            ventana.mostrarPO7(gestor.getPacienteO(6));
            ventana.mostrarPO8(gestor.getPacienteO(7));
            ventana.mostrarPO9(gestor.getPacienteO(8));
            ventana.mostrarPO10(gestor.getPacienteO(9));
            ventana.mostrarPO11(gestor.getPacienteO(10));
            ventana.mostrarPO12(gestor.getPacienteO(11));
            ventana.mostrarPO13(gestor.getPacienteO(12));
            ventana.mostrarPO14(gestor.getPacienteO(13));
            ventana.mostrarPO15(gestor.getPacienteO(14));
            ventana.mostrarPO16(gestor.getPacienteO(15));
            ventana.mostrarPO17(gestor.getPacienteO(16));
            ventana.mostrarPO18(gestor.getPacienteO(17));
            ventana.mostrarPO19(gestor.getPacienteO(18));
            ventana.mostrarPO20(gestor.getPacienteO(19));

            //Si es remoto puede cerrar puestos de vacunacion


            sleep(1000);
        } catch (InterruptedException ex) { } catch (NotBoundException ex) {
            Logger.getLogger(HiloPintor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (MalformedURLException ex) {
            Logger.getLogger(HiloPintor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (RemoteException ex) {
            Logger.getLogger(HiloPintor.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
  }
}
```

```
package interfaz;

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

public interface InterfazGestor extends Remote {

    public String getAuxiliarV() throws RemoteException;
    public String getPacienteO(int pos) throws RemoteException;
    public String getColaRecepcion() throws RemoteException;
    public String getColaDescanso() throws RemoteException;
    public String getSanitarioVacunacion(int pos) throws RemoteException;
    public String getPacienteVacunacion(int pos) throws RemoteException;
    public String getPacienteRecepcion() throws RemoteException;
    public String getAuxiliarR() throws RemoteException;
    public int numVacunas() throws RemoteException;
    public void cerrarPuestoVacunacion(int puesto) throws RemoteException;
    public void insertarSanitario() throws RemoteException;
}
```

```
package interfaz;

import hospital.*;

public class Ventana extends javax.swing.JFrame {

    private boolean[] puestosPorCerrar = new boolean[10];

    public Ventana() {
        initComponents();
```

```
    this.setVisible(true);
    HiloPintor hp = new HiloPintor(this);
    hp.start();
}
public Ventana(Gestor gestor) {
    initComponents();
    this.setVisible(true);
    this.jButtonCerrar1.setEnabled(false);
    this.jButtonCerrar2.setEnabled(false);
    this.jButtonCerrar3.setEnabled(false);
    this.jButtonCerrar4.setEnabled(false);
    this.jButtonCerrar5.setEnabled(false);
    this.jButtonCerrar6.setEnabled(false);
    this.jButtonCerrar7.setEnabled(false);
    this.jButtonCerrar8.setEnabled(false);
    this.jButtonCerrar9.setEnabled(false);
    this.jButtonCerrar10.setEnabled(false);


    HiloPintor hp = new HiloPintor(gestor, this);
    hp.start();
}

public boolean[] getPuestosPorCerrar() {
    return puestosPorCerrar;
}
public void setPuestosCerrados() {
    for (int i=0; i < puestosPorCerrar.length; i++) {
        puestosPorCerrar[i] = false;
    }
}
```

```
public void mostrarRecepcion(String colaRecepcion, String paciente, String auxiliarR) {
    this.jTextAreaRecepcion.setText(colaRecepcion);
    this.jTextPacienteR.setText(paciente);
    this.jTextAuxiliarR.setText(auxiliarR);
}
public void mostrarVacunas(int numVacunas, String auxiliar) {
    this.jTextVacunas.setText(""+ numVacunas);
    this.jTextAuxiliarV.setText(auxiliar);
}

public void mostrarPV1(String sanitario, String paciente) {
    this.jTextPV1.setText(sanitario + ", " + paciente);
}
public void mostrarPV2(String sanitario, String paciente) {
    this.jTextPV2.setText(sanitario + ", " + paciente);
}
public void mostrarPV3(String sanitario, String paciente) {
    this.jTextPV3.setText(sanitario + ", " + paciente);
}
public void mostrarPV4(String sanitario, String paciente) {
    this.jTextPV4.setText(sanitario + ", " + paciente);
}
public void mostrarPV5(String sanitario, String paciente) {
    this.jTextPV5.setText(sanitario + ", " + paciente);
}
public void mostrarPV6(String sanitario, String paciente) {
    this.jTextPV6.setText(sanitario + ", " + paciente);
}
public void mostrarPV7(String sanitario, String paciente) {
    this.jTextPV7.setText(sanitario + ", " + paciente);
}
```

```
public void mostrarPV8(String sanitario, String paciente) {
   this.jTextPV8.setText(sanitario + ", " + paciente);
}
public void mostrarPV9(String sanitario, String paciente) {
   this.jTextPV9.setText(sanitario + ", " + paciente);
}
public void mostrarPV10(String sanitario, String paciente) {
   this.jTextPV10.setText(sanitario + ", " + paciente);
}

public void mostrarPO1(String paciente) {
   this.jTextPO1.setText(paciente);
}
public void mostrarPO2(String paciente) {
   this.jTextPO2.setText(paciente);
}
public void mostrarPO3(String paciente) {
   this.jTextPO3.setText(paciente);
}
public void mostrarPO4(String paciente) {
   this.jTextPO4.setText(paciente);
}
public void mostrarPO5(String paciente) {
   this.jTextPO5.setText(paciente);
}
public void mostrarPO6(String paciente) {
   this.jTextPO6.setText(paciente);
}
public void mostrarPO7(String paciente) {
   this.jTextPO7.setText(paciente);
}
```

```
public void mostrarPO8(String paciente) {
   this.jTextPO8.setText(paciente);
}
public void mostrarPO9(String paciente) {
   this.jTextPO9.setText(paciente);
}
public void mostrarPO10(String paciente) {
   this.jTextPO10.setText(paciente);
}
public void mostrarPO11(String paciente) {
   this.jTextPO11.setText(paciente);
}
public void mostrarPO12(String paciente) {
   this.jTextPO12.setText(paciente);
}
public void mostrarPO13(String paciente) {
   this.jTextPO13.setText(paciente);
}
public void mostrarPO14(String paciente) {
   this.jTextPO14.setText(paciente);
}
public void mostrarPO15(String paciente) {
   this.jTextPO15.setText(paciente);
}
public void mostrarPO16(String paciente) {
   this.jTextPO16.setText(paciente);
}
public void mostrarPO17(String paciente) {
   this.jTextPO17.setText(paciente);
}
public void mostrarPO18(String paciente) {
```

```
    this.jTextPO18.setText(paciente);
}
public void mostrarPO19(String paciente) {
    this.jTextPO19.setText(paciente);
}
public void mostrarPO20(String paciente) {
    this.jTextPO20.setText(paciente);
}

public void mostrarDescanso(String colaDescanso) {
    this.jTextAreaDescanso.setText(colaDescanso);
}

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

    jPanelRecepcion = new javax.swing.JPanel();
    jLabel3 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTextAreaRecepcion = new javax.swing.JTextArea();
    jLabel18 = new javax.swing.JLabel();
    jLabel19 = new javax.swing.JLabel();
    jTextPacienteR = new javax.swing.JTextField();
    jLabel16 = new javax.swing.JLabel();
    jTextAuxiliarR = new javax.swing.JTextField();
    jPanelSDescanso = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jScrollPane2 = new javax.swing.JScrollPane();
    jTextAreaDescanso = new javax.swing.JTextArea();
```

```
jPanelSVacunacion = new javax.swing.JPanel();
jLabel4 = new javax.swing.JLabel();
jLabel6 = new javax.swing.JLabel();
jLabel7 = new javax.swing.JLabel();
jLabel8 = new javax.swing.JLabel();
jLabel9 = new javax.swing.JLabel();
jLabel10 = new javax.swing.JLabel();
jLabel11 = new javax.swing.JLabel();
jLabel12 = new javax.swing.JLabel();
jLabel13 = new javax.swing.JLabel();
jLabel14 = new javax.swing.JLabel();
jLabel15 = new javax.swing.JLabel();
jTextPV1 = new javax.swing.JTextField();
jTextPV2 = new javax.swing.JTextField();
jTextPV3 = new javax.swing.JTextField();
jTextPV6 = new javax.swing.JTextField();
jTextPV4 = new javax.swing.JTextField();
jTextPV5 = new javax.swing.JTextField();
jTextPV8 = new javax.swing.JTextField();
jTextPV7 = new javax.swing.JTextField();
jTextPV9 = new javax.swing.JTextField();
jTextPV10 = new javax.swing.JTextField();
jButtonCerrar5 = new javax.swing.JButton();
jButtonCerrar1 = new javax.swing.JButton();
jButtonCerrar9 = new javax.swing.JButton();
jButtonCerrar4 = new javax.swing.JButton();
jButtonCerrar8 = new javax.swing.JButton();
jButtonCerrar3 = new javax.swing.JButton();
jButtonCerrar2 = new javax.swing.JButton();
jButtonCerrar7 = new javax.swing.JButton();
jButtonCerrar6 = new javax.swing.JButton();
```

```
jButtonCerrar10 = new javax.swing.JButton();
jTextAuxiliarV = new javax.swing.JTextField();
jLabel17 = new javax.swing.JLabel();
jTextVacunas = new javax.swing.JTextField();
jLabel20 = new javax.swing.JLabel();
jPanelSObservacion = new javax.swing.JPanel();
jLabel5 = new javax.swing.JLabel();
jLabel21 = new javax.swing.JLabel();
jLabel22 = new javax.swing.JLabel();
jLabel23 = new javax.swing.JLabel();
jLabel24 = new javax.swing.JLabel();
jLabel25 = new javax.swing.JLabel();
jLabel26 = new javax.swing.JLabel();
jLabel27 = new javax.swing.JLabel();
jLabel28 = new javax.swing.JLabel();
jLabel29 = new javax.swing.JLabel();
jLabel30 = new javax.swing.JLabel();
jLabel31 = new javax.swing.JLabel();
jLabel32 = new javax.swing.JLabel();
jLabel33 = new javax.swing.JLabel();
jLabel34 = new javax.swing.JLabel();
jLabel35 = new javax.swing.JLabel();
jLabel36 = new javax.swing.JLabel();
jLabel37 = new javax.swing.JLabel();
jLabel38 = new javax.swing.JLabel();
jLabel39 = new javax.swing.JLabel();
jLabel40 = new javax.swing.JLabel();
jTextPO1 = new javax.swing.JTextField();
jTextPO5 = new javax.swing.JTextField();
jTextPO14 = new javax.swing.JTextField();
jTextPO12 = new javax.swing.JTextField();
```

```
jTextPO11 = new javax.swing.JTextField();
jTextPO4 = new javax.swing.JTextField();
jTextPO2 = new javax.swing.JTextField();
jTextPO3 = new javax.swing.JTextField();
jTextPO13 = new javax.swing.JTextField();
jTextPO15 = new javax.swing.JTextField();
jTextPO16 = new javax.swing.JTextField();
jTextPO18 = new javax.swing.JTextField();
jTextPO6 = new javax.swing.JTextField();
jTextPO8 = new javax.swing.JTextField();
jTextPO17 = new javax.swing.JTextField();
jTextPO7 = new javax.swing.JTextField();
jTextPO20 = new javax.swing.JTextField();
jTextPO10 = new javax.swing.JTextField();
jTextPO19 = new javax.swing.JTextField();
jTextPO9 = new javax.swing.JTextField();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setBackground(new java.awt.Color(51, 51, 51));
setForeground(new java.awt.Color(51, 51, 51));
setResizable(false);
setType(java.awt.Window.Type.POPUP);
getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

jPanelRecepcion.setBackground(new java.awt.Color(255, 255, 204));

jLabel3.setForeground(new java.awt.Color(51, 51, 51));
jLabel3.setText("RECEPCIÓN");

jTextAreaRecepcion.setColumns(20);
jTextAreaRecepcion.setLineWrap(true);
```

```
        jTextAreaRecepcion.setRows(5);
        jScrollPane1.setViewportView(jTextAreaRecepcion);

        jLabel18.setForeground(new java.awt.Color(102, 102, 102));
        jLabel18.setText("Cola de espera");

        jLabel19.setForeground(new java.awt.Color(51, 51, 51));
        jLabel19.setText("Paciente");

        jLabel16.setForeground(new java.awt.Color(51, 51, 51));
        jLabel16.setText("Auxiliar");

        javax.swing.GroupLayout jPanelRecepcionLayout = new
javax.swing.GroupLayout(jPanelRecepcion);
        jPanelRecepcion.setLayout(jPanelRecepcionLayout);
        jPanelRecepcionLayout.setHorizontalGroup(

jPanelRecepcionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanelRecepcionLayout.createSequentialGroup()
                .addGap(19, 19, 19)

.addGroup(jPanelRecepcionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.T
RAILING)
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanelRecepcionLayout.createSequentialGroup()
                        .addGap(252, 252, 252)
                        .addComponent(jLabel3)
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addGroup(jPanelRecepcionLayout.createSequentialGroup()
                        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 404,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 71,
Short.MAX_VALUE)
```

```
.addGroup(jPanelRecepcionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jTextPacienteR, javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jTextAuxiliarR, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, 81, Short.MAX_VALUE)
                    .addComponent(jLabel16, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(85, 85, 85))
            .addGroup(jPanelRecepcionLayout.createSequentialGroup()
                .addComponent(jLabel18, javax.swing.GroupLayout.PREFERRED_SIZE, 94, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel19)
                .addGap(104, 104, 104))))
    );
    jPanelRecepcionLayout.setVerticalGroup(
        jPanelRecepcionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanelRecepcionLayout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jLabel3)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanelRecepcionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel18)
                .addComponent(jLabel19))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanelRecepcionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 139, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addGroup(jPanelRecepcionLayout.createSequentialGroup()
                    .addComponent(jTextPacienteR, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(26, 26, 26)
                    .addComponent(jLabel16)
                    .addGap(18, 18, 18)
                    .addComponent(jTextAuxiliarR, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap(18, Short.MAX_VALUE))
        );

        getContentPane().add(jPanelRecepcion, new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 660, -1));

        jPanelSDescanso.setBackground(new java.awt.Color(255, 255, 204));

        jLabel2.setForeground(new java.awt.Color(51, 51, 51));
        jLabel2.setText("SALA DE DESCANSO");

        jTextAreaDescanso.setColumns(20);
        jTextAreaDescanso.setLineWrap(true);
        jTextAreaDescanso.setRows(5);
        jScrollPane2.setViewportView(jTextAreaDescanso);

        javax.swing.GroupLayout jPanelSDescansoLayout = new
javax.swing.GroupLayout(jPanelSDescanso);
        jPanelSDescanso.setLayout(jPanelSDescansoLayout);
        jPanelSDescansoLayout.setHorizontalGroup(

jPanelSDescansoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanelSDescansoLayout.createSequentialGroup()
                .addGap(226, 226, 226)
                .addComponent(jLabel2)
```

```
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabel1)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanelSDescansoLayout.createSequentialGroup()
            .addContainerGap(81, Short.MAX_VALUE)
            .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 400,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(59, 59, 59))
    );
    jPanelSDescansoLayout.setVerticalGroup(

jPanelSDescansoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanelSDescansoLayout.createSequentialGroup()
            .addGap(15, 15, 15)

.addGroup(jPanelSDescansoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                .addComponent(jLabel2)
                .addComponent(jLabel1))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 124,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(40, Short.MAX_VALUE))
    );

    getContentPane().add(jPanelSDescanso, new
org.netbeans.lib.awtextra.AbsoluteConstraints(671, 0, 540, 207));

    jPanelSVacunacion.setBackground(new java.awt.Color(255, 255, 255));

    jLabel4.setForeground(new java.awt.Color(51, 51, 51));
    jLabel4.setText("SALA DE VACUNACIÓN");
```

```
jLabel6.setForeground(new java.awt.Color(51, 51, 51));
jLabel6.setText("Puesto 2");

jLabel7.setForeground(new java.awt.Color(51, 51, 51));
jLabel7.setText("Puesto 1");

jLabel8.setForeground(new java.awt.Color(51, 51, 51));
jLabel8.setText("Puesto 3");

jLabel9.setForeground(new java.awt.Color(51, 51, 51));
jLabel9.setText("Puesto 5");

jLabel10.setForeground(new java.awt.Color(51, 51, 51));
jLabel10.setText("Puesto 4");

jLabel11.setForeground(new java.awt.Color(51, 51, 51));
jLabel11.setText("Puesto 10");

jLabel12.setForeground(new java.awt.Color(51, 51, 51));
jLabel12.setText("Puesto 9");

jLabel13.setForeground(new java.awt.Color(51, 51, 51));
jLabel13.setText("Puesto 8");

jLabel14.setForeground(new java.awt.Color(51, 51, 51));
jLabel14.setText("Puesto 7");

jLabel15.setForeground(new java.awt.Color(51, 51, 51));
jLabel15.setText("Puesto 6");

jTextPV1.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextPV1ActionPerformed(evt);
    }
});
jTextPV1.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
    public void propertyChange(java.beans.PropertyChangeEvent evt) {
        jTextPV1PropertyChange(evt);
    }
});

jTextPV2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextPV2ActionPerformed(evt);
    }
});

jTextPV3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextPV3ActionPerformed(evt);
    }
});

jTextPV6.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextPV6ActionPerformed(evt);
    }
});

jTextPV4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextPV4ActionPerformed(evt);
```

```
    }
});

jTextPV5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextPV5ActionPerformed(evt);
    }
});

jTextPV8.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextPV8ActionPerformed(evt);
    }
});

jTextPV7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextPV7ActionPerformed(evt);
    }
});

jTextPV9.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextPV9ActionPerformed(evt);
    }
});

jTextPV10.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextPV10ActionPerformed(evt);
    }
```

```
});

jButtonCerrar5.setText("Cerrar");
jButtonCerrar5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonCerrar5ActionPerformed(evt);
    }
});

jButtonCerrar1.setText("Cerrar");
jButtonCerrar1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonCerrar1ActionPerformed(evt);
    }
});

jButtonCerrar9.setText("Cerrar");
jButtonCerrar9.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonCerrar9ActionPerformed(evt);
    }
});

jButtonCerrar4.setText("Cerrar");
jButtonCerrar4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonCerrar4ActionPerformed(evt);
    }
});

jButtonCerrar8.setText("Cerrar");
```

```
jButtonCerrar8.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonCerrar8ActionPerformed(evt);
    }
});

jButtonCerrar3.setText("Cerrar");
jButtonCerrar3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonCerrar3ActionPerformed(evt);
    }
});

jButtonCerrar2.setText("Cerrar");
jButtonCerrar2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonCerrar2ActionPerformed(evt);
    }
});

jButtonCerrar7.setText("Cerrar");
jButtonCerrar7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonCerrar7ActionPerformed(evt);
    }
});

jButtonCerrar6.setText("Cerrar");
jButtonCerrar6.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonCerrar6ActionPerformed(evt);
```

```
        }
    });

    jButtonCerrar10.setText("Cerrar");
    jButtonCerrar10.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButtonCerrar10ActionPerformed(evt);
        }
    });

    jLabel17.setForeground(new java.awt.Color(51, 51, 51));
    jLabel17.setText("Vacunas Disponibles");

    jLabel20.setForeground(new java.awt.Color(51, 51, 51));
    jLabel20.setText("Auxiliar");

    javax.swing.GroupLayout jPanelSVacunacionLayout = new
javax.swing.GroupLayout(jPanelSVacunacion);
    jPanelSVacunacion.setLayout(jPanelSVacunacionLayout);
    jPanelSVacunacionLayout.setHorizontalGroup(

jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanelSVacunacionLayout.createSequentialGroup()
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jLabel4)
            .addGap(540, 540, 540))
        .addGroup(jPanelSVacunacionLayout.createSequentialGroup()

.addGroup(jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                .addGroup(jPanelSVacunacionLayout.createSequentialGroup()
                    .addGap(47, 47, 47)
```

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

                .addGroup(jPanelSVacunacionLayout.createSequentialGroup()

                  .addGap(3, 3, 3)


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

                    .addComponent(jTextPV6, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 74, javax.swing.GroupLayout.PREFERRED_SIZE)

                    .addGroup(jPanelSVacunacionLayout.createSequentialGroup()


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

                        .addComponent(jTextPV1, javax.swing.GroupLayout.PREFERRED_SIZE,
86, javax.swing.GroupLayout.PREFERRED_SIZE)

                        .addComponent(jButtonCerrar1,
javax.swing.GroupLayout.PREFERRED_SIZE, 86, javax.swing.GroupLayout.PREFERRED_SIZE))

                    .addGap(60, 60, 60)


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

                        .addComponent(jButtonCerrar2,
javax.swing.GroupLayout.PREFERRED_SIZE, 77, javax.swing.GroupLayout.PREFERRED_SIZE)

                        .addComponent(jTextPV2,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 77,
javax.swing.GroupLayout.PREFERRED_SIZE)))))

                  .addComponent(jButtonCerrar6, javax.swing.GroupLayout.PREFERRED_SIZE,
77, javax.swing.GroupLayout.PREFERRED_SIZE))

              .addGap(219, 219, 219)

              .addComponent(jTextPV9, javax.swing.GroupLayout.PREFERRED_SIZE, 85,
javax.swing.GroupLayout.PREFERRED_SIZE)

              .addGap(72, 72, 72)

              .addComponent(jTextPV10, javax.swing.GroupLayout.PREFERRED_SIZE, 85,
javax.swing.GroupLayout.PREFERRED_SIZE))

            .addGroup(jPanelSVacunacionLayout.createSequentialGroup()

              .addGap(63, 63, 63)
```

```
.addGroup(jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                .addGroup(jPanelSVacunacionLayout.createSequentialGroup()

.addGroup(jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                    .addGroup(jPanelSVacunacionLayout.createSequentialGroup()
                      .addComponent(jLabel15)
                      .addGap(81, 81, 81)
                      .addComponent(jLabel14))
                    .addGroup(jPanelSVacunacionLayout.createSequentialGroup()
                      .addComponent(jLabel7)
                      .addGap(92, 92, 92)
                      .addComponent(jLabel6)))

.addGroup(jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                    .addGroup(jPanelSVacunacionLayout.createSequentialGroup()
                      .addGap(93, 93, 93)
                      .addComponent(jLabel8))
                    .addGroup(jPanelSVacunacionLayout.createSequentialGroup()
                      .addGap(85, 85, 85)

.addGroup(jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false)
                        .addComponent(jLabel13)
                        .addComponent(jButtonCerrar8,
javax.swing.GroupLayout.PREFERRED_SIZE, 77, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jTextPV8)))))
                .addGroup(jPanelSVacunacionLayout.createSequentialGroup()
                  .addGap(131, 131, 131)

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

```
                    .addComponent(jButtonCerrar7,
javax.swing.GroupLayout.PREFERRED_SIZE, 77, javax.swing.GroupLayout.PREFERRED_SIZE)

                    .addComponent(jTextPV7, javax.swing.GroupLayout.PREFERRED_SIZE, 74,
javax.swing.GroupLayout.PREFERRED_SIZE)))

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

                    .addGap(274, 274, 274)


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

                    .addComponent(jTextPV3, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 80, javax.swing.GroupLayout.PREFERRED_SIZE)

                    .addComponent(jButtonCerrar3,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 80,
javax.swing.GroupLayout.PREFERRED_SIZE))))


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

                .addGroup(jPanelSVacunacionLayout.createSequentialGroup()

                    .addGap(75, 75, 75)


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

                    .addComponent(jButtonCerrar9,
javax.swing.GroupLayout.PREFERRED_SIZE, 85, javax.swing.GroupLayout.PREFERRED_SIZE)

                    .addComponent(jButtonCerrar4,
javax.swing.GroupLayout.PREFERRED_SIZE, 85, javax.swing.GroupLayout.PREFERRED_SIZE)

                    .addComponent(jTextPV4, javax.swing.GroupLayout.PREFERRED_SIZE, 85,
javax.swing.GroupLayout.PREFERRED_SIZE))

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


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

                    .addComponent(jTextPV5, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 85, javax.swing.GroupLayout.PREFERRED_SIZE)

                    .addComponent(jButtonCerrar5,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 85,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                        .addComponent(jButtonCerrar10,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 85,
javax.swing.GroupLayout.PREFERRED_SIZE)))

                    .addGroup(jPanelSVacunacionLayout.createSequentialGroup()

                        .addGap(95, 95, 95)

                        .addComponent(jLabel10)

                        .addGap(105, 105, 105)

                        .addComponent(jLabel9))

                    .addGroup(jPanelSVacunacionLayout.createSequentialGroup()

                        .addGap(90, 90, 90)

                        .addComponent(jLabel12)

                        .addGap(96, 96, 96)

                        .addComponent(jLabel11)))))

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

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

                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 162,
Short.MAX_VALUE)


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

                        .addComponent(jTextVacunas, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 81, javax.swing.GroupLayout.PREFERRED_SIZE)

                        .addComponent(jTextAuxiliarV, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 81, javax.swing.GroupLayout.PREFERRED_SIZE))

                    .addGap(226, 226, 226))

                .addGroup(jPanelSVacunacionLayout.createSequentialGroup()

                    .addGap(143, 143, 143)

                    .addComponent(jLabel17)

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


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

```
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanelSVacunacionLayout.createSequentialGroup()
                .addContainerGap(925, Short.MAX_VALUE)
                .addComponent(jLabel20, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(232, 232, 232)))
        );
        jPanelSVacunacionLayout.setVerticalGroup(

jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanelSVacunacionLayout.createSequentialGroup()
                .addGap(14, 14, 14)
                .addComponent(jLabel4)

.addGroup(jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                .addGroup(jPanelSVacunacionLayout.createSequentialGroup()
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                        .addComponent(jLabel6)
                        .addComponent(jLabel7)
                        .addComponent(jLabel8)
                        .addComponent(jLabel10)
                        .addComponent(jLabel9))
                    .addGap(19, 19, 19)

.addGroup(jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                        .addComponent(jTextPV1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jTextPV2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addComponent(jTextPV3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jTextPV5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jTextPV4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

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

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

                .addComponent(jButtonCerrar3,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE)

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

                .addComponent(jButtonCerrar5, javax.swing.GroupLayout.PREFERRED_SIZE,
23, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jButtonCerrar2, javax.swing.GroupLayout.PREFERRED_SIZE,
24, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jButtonCerrar1, javax.swing.GroupLayout.PREFERRED_SIZE,
24, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jButtonCerrar4, javax.swing.GroupLayout.PREFERRED_SIZE,
24, javax.swing.GroupLayout.PREFERRED_SIZE))))

            .addGroup(jPanelSVacunacionLayout.createSequentialGroup()

                .addGap(56, 56, 56)

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

        .addGap(30, 30, 30)

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

            .addComponent(jLabel11)

            .addComponent(jLabel12)

            .addComponent(jLabel13)

            .addComponent(jLabel14)

            .addComponent(jLabel15)
```

```
                .addComponent(jLabel17))
            .addGap(18, 18, 18)

.addGroup(jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(jTextPV6, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPV7, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPV8, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPV9, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPV10, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextVacunas, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(jButtonCerrar6, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButtonCerrar7, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButtonCerrar8, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButtonCerrar9, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButtonCerrar10, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap(43, Short.MAX_VALUE))

.addGroup(jPanelSVacunacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addGroup(jPanelSVacunacionLayout.createSequentialGroup()
                .addGap(62, 62, 62)
```

```
                .addComponent(jLabel20)
                .addContainerGap(214, Short.MAX_VALUE)))
        );

        getContentPane().add(jPanelSVacunacion, new
org.netbeans.lib.awtextra.AbsoluteConstraints(6, 219, 1205, -1));

        jPanelSObservacion.setBackground(new java.awt.Color(255, 255, 204));

        jLabel5.setForeground(new java.awt.Color(51, 51, 51));
        jLabel5.setText("SALA DE OBSERVACIÓN");

        jLabel21.setForeground(new java.awt.Color(51, 51, 51));
        jLabel21.setText("Puesto 1");

        jLabel22.setForeground(new java.awt.Color(51, 51, 51));
        jLabel22.setText("Puesto 2");

        jLabel23.setForeground(new java.awt.Color(51, 51, 51));
        jLabel23.setText("Puesto 3");

        jLabel24.setForeground(new java.awt.Color(51, 51, 51));
        jLabel24.setText("Puesto 4");

        jLabel25.setForeground(new java.awt.Color(51, 51, 51));
        jLabel25.setText("Puesto 5");

        jLabel26.setForeground(new java.awt.Color(51, 51, 51));
        jLabel26.setText("Puesto 6");

        jLabel27.setForeground(new java.awt.Color(51, 51, 51));
```

```
jLabel27.setText("Puesto 7");

jLabel28.setForeground(new java.awt.Color(51, 51, 51));
jLabel28.setText("Puesto 8");

jLabel29.setForeground(new java.awt.Color(51, 51, 51));
jLabel29.setText("Puesto 9");

jLabel30.setForeground(new java.awt.Color(51, 51, 51));
jLabel30.setText("Puesto 10");

jLabel31.setForeground(new java.awt.Color(51, 51, 51));
jLabel31.setText("Puesto 17");

jLabel32.setForeground(new java.awt.Color(51, 51, 51));
jLabel32.setText("Puesto 16");

jLabel33.setForeground(new java.awt.Color(51, 51, 51));
jLabel33.setText("Puesto 15");

jLabel34.setForeground(new java.awt.Color(51, 51, 51));
jLabel34.setText("Puesto 14");

jLabel35.setForeground(new java.awt.Color(51, 51, 51));
jLabel35.setText("Puesto 13");

jLabel36.setForeground(new java.awt.Color(51, 51, 51));
jLabel36.setText("Puesto 11");

jLabel37.setForeground(new java.awt.Color(51, 51, 51));
jLabel37.setText("Puesto 12");
```

```
        jLabel38.setForeground(new java.awt.Color(51, 51, 51));
        jLabel38.setText("Puesto 18");

        jLabel39.setForeground(new java.awt.Color(51, 51, 51));
        jLabel39.setText("Puesto 19");

        jLabel40.setForeground(new java.awt.Color(51, 51, 51));
        jLabel40.setText("Puesto 20");

        jTextPO4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextPO4ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanelSObservacionLayout = new
javax.swing.GroupLayout(jPanelSObservacion);
        jPanelSObservacion.setLayout(jPanelSObservacionLayout);
        jPanelSObservacionLayout.setHorizontalGroup(

jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanelSObservacionLayout.createSequentialGroup()

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.TRAILING)
                    .addGroup(jPanelSObservacionLayout.createSequentialGroup()
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jLabel5)
                        .addGap(14, 14, 14))
                    .addGroup(jPanelSObservacionLayout.createSequentialGroup()
```

```
.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanelSObservacionLayout.createSequentialGroup()
                        .addGap(42, 42, 42)
                        .addComponent(jTextPO1, javax.swing.GroupLayout.PREFERRED_SIZE, 81, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanelSObservacionLayout.createSequentialGroup()
                        .addGap(52, 52, 52)

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jTextPO11, javax.swing.GroupLayout.PREFERRED_SIZE, 66, javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                .addComponent(jLabel36)
                                .addComponent(jLabel21)))))
                .addGap(34, 34, 34)

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jLabel22)
                    .addComponent(jTextPO2)
                    .addComponent(jTextPO12)
                    .addComponent(jLabel37, javax.swing.GroupLayout.DEFAULT_SIZE, 66, Short.MAX_VALUE))
                .addGap(52, 52, 52)

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

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jTextPO4, javax.swing.GroupLayout.PREFERRED_SIZE, 64, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanelSObservacionLayout.createSequentialGroup()

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.TRAILING)

                        .addComponent(jTextPO13,
javax.swing.GroupLayout.PREFERRED_SIZE, 70, javax.swing.GroupLayout.PREFERRED_SIZE)

                        .addGroup(jPanelSObservacionLayout.createSequentialGroup()

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.TRAILING)

                            .addComponent(jLabel23)

                            .addComponent(jTextPO3,
javax.swing.GroupLayout.PREFERRED_SIZE, 64, javax.swing.GroupLayout.PREFERRED_SIZE))

                        .addGap(6, 6, 6)))

                    .addComponent(jLabel35, javax.swing.GroupLayout.PREFERRED_SIZE,
70, javax.swing.GroupLayout.PREFERRED_SIZE))

                .addGap(55, 55, 55)

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.TRAILING)

                    .addComponent(jLabel24)

                    .addComponent(jLabel34))))

                .addComponent(jTextPO14, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)

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

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

                    .addComponent(jLabel33, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
javax.swing.GroupLayout.PREFERRED_SIZE))

                .addGroup(jPanelSObservacionLayout.createSequentialGroup()
```

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

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jTextPO15, javax.swing.GroupLayout.PREFERRED_SIZE, 70, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(jPanelSObservacionLayout.createSequentialGroup()
                            .addGap(6, 6, 6)
                            .addComponent(jLabel25))
                        .addComponent(jTextPO5, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 70, javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addGap(48, 48, 48)

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

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                        .addComponent(jLabel26)
                        .addComponent(jLabel32))
                    .addComponent(jTextPO16, javax.swing.GroupLayout.PREFERRED_SIZE, 70, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextPO6, javax.swing.GroupLayout.PREFERRED_SIZE, 70, javax.swing.GroupLayout.PREFERRED_SIZE))))
            .addGap(45, 45, 45)

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jTextPO7, javax.swing.GroupLayout.PREFERRED_SIZE, 70, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(jPanelSObservacionLayout.createSequentialGroup()

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

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

```
                    .addComponent(jLabel27)

                    .addComponent(jLabel31))

                .addComponent(jTextPO17, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
javax.swing.GroupLayout.PREFERRED_SIZE))

            .addGap(47, 47, 47)


.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)

                .addComponent(jTextPO18, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
javax.swing.GroupLayout.PREFERRED_SIZE)


.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.TRAILING)

                    .addGroup(jPanelSObservacionLayout.createSequentialGroup()


.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)

                        .addGroup(jPanelSObservacionLayout.createSequentialGroup()

                            .addComponent(jTextPO8, javax.swing.GroupLayout.PREFERRED_SIZE,
70, javax.swing.GroupLayout.PREFERRED_SIZE)


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

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


.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)

                            .addComponent(jTextPO19,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
javax.swing.GroupLayout.PREFERRED_SIZE)

                            .addComponent(jTextPO9,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
javax.swing.GroupLayout.PREFERRED_SIZE))

                        .addGap(36, 36, 36)))

                .addComponent(jTextPO10, javax.swing.GroupLayout.PREFERRED_SIZE,
70, javax.swing.GroupLayout.PREFERRED_SIZE))

                    .addGroup(jPanelSObservacionLayout.createSequentialGroup()
```

```
.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel28)
                    .addComponent(jLabel38))
                .addGap(66, 66, 66)

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

.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addGroup(jPanelSObservacionLayout.createSequentialGroup()
                        .addComponent(jLabel29)
                        .addGap(56, 56, 56)
                        .addComponent(jLabel30))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanelSObservacionLayout.createSequentialGroup()
                        .addComponent(jLabel39, javax.swing.GroupLayout.PREFERRED_SIZE, 70, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jLabel40)))
                .addGroup(jPanelSObservacionLayout.createSequentialGroup()
                    .addGap(106, 106, 106)
                    .addComponent(jTextPO20, javax.swing.GroupLayout.PREFERRED_SIZE, 70, javax.swing.GroupLayout.PREFERRED_SIZE)))))))
            .addGap(70, 70, 70))
    );
    jPanelSObservacionLayout.setVerticalGroup(

jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanelSObservacionLayout.createSequentialGroup()
            .addContainerGap()
```

```
            .addComponent(jLabel5)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel21)
                .addComponent(jLabel22)
                .addComponent(jLabel23)
                .addComponent(jLabel24)
                .addComponent(jLabel25)
                .addComponent(jLabel26)
                .addComponent(jLabel27)
                .addComponent(jLabel28)
                .addComponent(jLabel29)
                .addComponent(jLabel30))
            .addGap(31, 31, 31)
.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jTextPO1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPO2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPO3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPO4, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPO5, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPO6, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPO7, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPO8, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addComponent(jTextPO9, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jTextPO10, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 60,
Short.MAX_VALUE)


.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.BASELINE)

                .addComponent(jLabel36)

                .addComponent(jLabel37)

                .addComponent(jLabel35)

                .addComponent(jLabel34)

                .addComponent(jLabel33)

                .addComponent(jLabel32)

                .addComponent(jLabel31)

                .addComponent(jLabel38)

                .addComponent(jLabel39)

                .addComponent(jLabel40))

            .addGap(28, 28, 28)


.addGroup(jPanelSObservacionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.BASELINE)

                .addComponent(jTextPO11, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jTextPO12, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jTextPO14, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jTextPO13, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jTextPO15, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jTextPO16, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addComponent(jTextPO17, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPO18, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPO19, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextPO20, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(61, 61, 61))
        );

        getContentPane().add(jPanelSObservacion, new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 522, 1217, -1));

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

    private void jTextPV1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextPV2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextPV3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextPV6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
```

```
private void jTextPV4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jTextPV5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jTextPV8ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jTextPV7ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jTextPV9ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jTextPV10ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jTextPO4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jTextPV1PropertyChange(java.beans.PropertyChangeEvent evt) {
    // TODO add your handling code here:
}
```

```
private void jButtonCerrar1ActionPerformed(java.awt.event.ActionEvent evt) {
    puestosPorCerrar[0] = true;
}

private void jButtonCerrar2ActionPerformed(java.awt.event.ActionEvent evt) {
    puestosPorCerrar[1] = true;
}

private void jButtonCerrar3ActionPerformed(java.awt.event.ActionEvent evt) {
    puestosPorCerrar[2] = true;
}

private void jButtonCerrar4ActionPerformed(java.awt.event.ActionEvent evt) {
    puestosPorCerrar[3] = true;
}

private void jButtonCerrar5ActionPerformed(java.awt.event.ActionEvent evt) {
    puestosPorCerrar[4] = true;
}

private void jButtonCerrar6ActionPerformed(java.awt.event.ActionEvent evt) {
    puestosPorCerrar[5] = true;
}

private void jButtonCerrar7ActionPerformed(java.awt.event.ActionEvent evt) {
    puestosPorCerrar[6] = true;
}

private void jButtonCerrar8ActionPerformed(java.awt.event.ActionEvent evt) {
    puestosPorCerrar[7] = true;
```

```
}

private void jButtonCerrar9ActionPerformed(java.awt.event.ActionEvent evt) {
   puestosPorCerrar[8] = true;
}

private void jButtonCerrar10ActionPerformed(java.awt.event.ActionEvent evt) {
   puestosPorCerrar[9] = true;
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
   /* Create and display the form */
   java.awt.EventQueue.invokeLater(new Runnable() {
      public void run() {
         new Ventana().setVisible(true);
      }
   });
}

// Variables declaration - do not modify
private javax.swing.JButton jButtonCerrar1;
private javax.swing.JButton jButtonCerrar10;
private javax.swing.JButton jButtonCerrar2;
private javax.swing.JButton jButtonCerrar3;
private javax.swing.JButton jButtonCerrar4;
private javax.swing.JButton jButtonCerrar5;
private javax.swing.JButton jButtonCerrar6;
private javax.swing.JButton jButtonCerrar7;
```

```
private javax.swing.JButton jButtonCerrar8;
private javax.swing.JButton jButtonCerrar9;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel17;
private javax.swing.JLabel jLabel18;
private javax.swing.JLabel jLabel19;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel20;
private javax.swing.JLabel jLabel21;
private javax.swing.JLabel jLabel22;
private javax.swing.JLabel jLabel23;
private javax.swing.JLabel jLabel24;
private javax.swing.JLabel jLabel25;
private javax.swing.JLabel jLabel26;
private javax.swing.JLabel jLabel27;
private javax.swing.JLabel jLabel28;
private javax.swing.JLabel jLabel29;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel30;
private javax.swing.JLabel jLabel31;
private javax.swing.JLabel jLabel32;
private javax.swing.JLabel jLabel33;
private javax.swing.JLabel jLabel34;
private javax.swing.JLabel jLabel35;
```

```
private javax.swing.JLabel jLabel36;
private javax.swing.JLabel jLabel37;
private javax.swing.JLabel jLabel38;
private javax.swing.JLabel jLabel39;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel40;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JPanel jPanelRecepcion;
private javax.swing.JPanel jPanelSDescanso;
private javax.swing.JPanel jPanelSObservacion;
private javax.swing.JPanel jPanelSVacunacion;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JTextArea jTextAreaDescanso;
private javax.swing.JTextArea jTextAreaRecepcion;
private javax.swing.JTextField jTextAuxiliarR;
private javax.swing.JTextField jTextAuxiliarV;
private javax.swing.JTextField jTextPO1;
private javax.swing.JTextField jTextPO10;
private javax.swing.JTextField jTextPO11;
private javax.swing.JTextField jTextPO12;
private javax.swing.JTextField jTextPO13;
private javax.swing.JTextField jTextPO14;
private javax.swing.JTextField jTextPO15;
private javax.swing.JTextField jTextPO16;
private javax.swing.JTextField jTextPO17;
private javax.swing.JTextField jTextPO18;
```

```
    private javax.swing.JTextField jTextPO19;
    private javax.swing.JTextField jTextPO2;
    private javax.swing.JTextField jTextPO20;
    private javax.swing.JTextField jTextPO3;
    private javax.swing.JTextField jTextPO4;
    private javax.swing.JTextField jTextPO5;
    private javax.swing.JTextField jTextPO6;
    private javax.swing.JTextField jTextPO7;
    private javax.swing.JTextField jTextPO8;
    private javax.swing.JTextField jTextPO9;
    private javax.swing.JTextField jTextPV1;
    private javax.swing.JTextField jTextPV10;
    private javax.swing.JTextField jTextPV2;
    private javax.swing.JTextField jTextPV3;
    private javax.swing.JTextField jTextPV4;
    private javax.swing.JTextField jTextPV5;
    private javax.swing.JTextField jTextPV6;
    private javax.swing.JTextField jTextPV7;
    private javax.swing.JTextField jTextPV8;
    private javax.swing.JTextField jTextPV9;
    private javax.swing.JTextField jTextPacienteR;
    private javax.swing.JTextField jTextVacunas;
    // End of variables declaration
}
```