PEC-LABORATORIO PROGRAMACIÓN AVANZADA

Grado en ingeniería [Informática]
Curso 2020/2021 - Convocatoria Ordinaria
09100862S - Rodríguez Pérez Lucía
03207068V - Mario Villa Labarra

09100862S - Rodríguez Pérez Lucía
03207068V - Mario Villa Labarra

Índice

1. Análisis de Alto Nivel:

El proyecto desarrollado consiste en la simulación de un hospital donde se lleva a cabo una vacunación masiva mediante un circuito en el que los pacientes deben pasar por diferentes salas hasta ser vacunados y poder marcharse a casa.

Para esta simulación son necesarios tres tipos de usuarios: Auxiliar, Sanitario, Paciente. Todos ellos se gestionan como hilos que hay que coordinar y sincronizar para garantizar un correcto funcionamiento durante su ciclo de vida.

El nexo de unión entre todos ellos será el hospital, que funciona como variable compartida donde se utilizan diferentes mecanismos para garantizar que la concurrencia se produzca de forma segura. El hospital cuenta con cuatro salas diferentes: recepción, sala de vacunación, sala de descanso del personal sanitario y la sala de observación. Las únicas que tienen aforo limitado son la sala de vacunación (10 pacientes) y la de observación (20 pacientes).

A continuación, detallamos la actividad de cada usuario y los problemas de sincronización y comunicación que hay que tener en cuenta para que no se produzcan condiciones de carrera situaciones de fallo del simulador.

Los pacientes, llegan al hospital con una cadencia entre uno y tres segundos y hacen cola hasta que llegue su turno. Uno de los dos auxiliares va registrando a los pacientes de la cola siguiendo el orden de llegada, para ello comprueba que está citado y en caso correcto, le asigna un puesto de vacunación. Este puesto debe tener un sanitario y no estar ocupado por otro paciente, en caso de que todos los puestos estén ocupados, el paciente debe esperar a que un puesto de vacunación quede libre.

El otro auxiliar se encarga de crear vacunas para que los sanitarios puedan suministrarlas, en caso de que no haya vacunas disponibles, los sanitarios deben esperar para poder vacunar a los pacientes que vayan a sus respectivos puestos.

Los sanitarios, una vez llegan al hospital, se dirigen a la sala de descanso en la que se cambian de ropa (esperan un tiempo en la sala de descanso), a continuación se les asigna un puesto libre y se dirigen a él esperando a que lleguen pacientes.

Cuando un paciente ha sido vacunado, debe ir a la sala de observación, donde el sanitario que le ha vacunado le asignará un puesto determinado y allí esperará un cierto tiempo por si la vacuna le da reacción. Si le da reacción, un sanitario procediente de la sala de descanso, irá a atenderle y si no, podrá marcharse a casa, terminando entonces su ciclo de vida.

Todo el personal del hospital, tiene descansos durante su ciclo de vida. El auxiliar del registro de pacientes, descansa cuando ha registrado a diez pacientes (correcta o incorrectamente) y el otro auxiliar, que crea vacunas, descansa cuando crea 20 vacunas. Los sanitarios descansan cuando han vacunado a quince pacientes o cuando su puesto de

vacunación se cierra para limpiarlo, en este caso, si estaban vacunando a un paciente, terminan de vacunarle y se dirigen a la sala de descanso. Todos ellos descansan en una sala conjunta y sus actividades se detienen durante el tiempo que descansan. Los tiempos de descanso se establecen aleatoriamente en un rango determinado, para los sanitarios será un tiempo entre cinco y ocho segundos una vez hayan vacunado a quince pacientes, para el auxiliar que registra a los pacientes, se le adjudica un tiempo de descanso de entre tres y cinco segundas cada vez que registre a 10 personas (correctamente o no). El auxiliar encargado de las vacunas, recibe un descanso de entre uno y cuatro segundos cada vez que suministra veinte vacunas.

Por último, para poder visualizar el progreso de la simulación, se emplea un JFrame que sirve como interfaz y muestra el identificador de cada usuario y su posición en cada instante de la ejecución. Además, el programa va escribiendo en un Log (evolucionHospital.txt) cada evento que se produce incluyendo la fecha y hora exacta en la que tiene lugar. El JFrame se conecta con el hospital mediante Sockets. Desde esta ventana, se puede pulsar un botón para cerrar cada puesto de vacunación para limpiarlo. Esta información también se envía al hospital mediante Sockets.

2. Diseño general del sistema y de las herramientas de sincronización utilizados

Tras el análisis de alto nivel, concluimos que es necesario asegurar que la concurrencia es segura en los siguientes casos: respetar los aforos, controlar las esperas cuando no hay puestos disponibles, el personal sanitario está descansando, no hay vacunas o no hay pacientes en la cola y garantizar la exclusión mutua cuando se asignan puestos a pacientes y sanitarios. Evitando todo esto, conseguimos que el sistema funcione correctamente sin que se produzcan interbloqueos, inanición o condiciones de carrera que hagan que el sistema falle o no progrese.

Como mencionamos anteriormente, un objeto de la clase Hospital, es la variable compartida por los usuarios. Los usuarios (Paciente, Auxiliar y Sanitario) se modelan como hilos y sus funciones principales están implementadas en la clase hospital. Por lo tanto, el control de la concurrencia se lleva a cabo en esta clase. Para ello, hemos utilizado diferentes herramientas.

En primer lugar, modelamos dos colas de tipo LinkedBlockingQueue. Una para almacenar a los pacientes que llegan al hospital en orden de llegada y otra para las vacunas que el Auxiliar 2 va generando. Se emplea este tipo de cola porque es bloqueante, es decir, cuando está vacía, se bloquea. De esta forma conseguimos que el primer auxiliar solo registre pacientes si hay alguno en la cola y que los sanitarios cojan vacunas solo si hay vacunas creadas. También podrían producirse bloqueos por cola llena pero como el enunciado no indica que haya límite de vacunas o de pacientes esperando, las hemos creado sin límite. Cuando alguna cola se bloquea por estar vacía, utiliza la cláusula synchronized para ponerse en espera y cuando ya tiene elementos en su interior, lo mismo para avisar a los hilos que estaban esperando.

Para las salas que tienen límite de aforo hemos utilizado semáforos. La sala de vacunación tiene un semáforo que se inicializa con diez permisos y la de observación con veinte

permisos. Los semáforos garantizan que no haya más de diez o veinte pacientes dentro de cada sala, respectivamente. Cuando un paciente entra a una de las salas, coge un permiso del semáforo (acquire) y cuando sale, lo deja. De forma que si no hay permisos disponibles, el paciente espera para entrar a que se libere alguno.
Hay un tercer semáforo que se inicializa con un solo permiso, en este caso, lo que queremos obtener con él es exclusión mutua para escribir en el Log. Si no hiciésemos esto, cuando varios hilos intentan escribir a la vez un evento, se pueden solapar las llamadas a la función de escritura y que se pierda información o los datos introducidos sean ilegibles porque se solapen.
Por último hemos empleado cerrojos. Estos cerrojos protegen los vectores que almacenan a los pacientes y sanitarios que hay en cada puesto de vacunación, observación, sala de descanso o en la recepción. Cada vez que un hilo quiere hacer algún cambio en estos vectores, el cerrojo se cierra (lock()), se hace el cambio y a continuación se abre (unlock()). Así, un solo hilo puede acceder a estos vectores a la vez para editarlos.

3. Clases principales

Sanitario
La clase Sanitario define a uno de los tres tipos de usuarios que hay en el hospital. Sus atributos son un identificador (id_sanitario) de tipo String, un contador de tipo entero que lleva la cuenta de los pacientes que ha vacunado dicho sanitario en el puesto que se le asigna, un boolean “cerrarPuesto” que se inicializa en false y cambia a true cuando el puesto que ocupa dicho sanitario en la sala de vacunación, se va a cerrar para limpiarlo. Por último el Hospital, que es la variable compartida por todos los usuarios, donde se lleva a cabo la simulación. El constructor de esta clase recibe como parámetros el identificador del sanitario y la variable compartida de tipo Hospital.

La clase Sanitario se modela como hilo, por lo tanto, extiende de Thread. El ciclo de vida del hilo se define en el método run. En primer lugar, entra en la sala de descanso durante un tiempo aleatorio entre uno y tres segundos para cambiarse de ropa, después sale y comienza un bucle infinito (while true). Dentro de este bucle, lo primero que hace al salir de la sala de descanso es comprobar si hay algún paciente en la sala de observación que haya tenido una reacción a la vacuna, y en caso de que así sea, le atiende. Después, solicita que se le asigne un puesto de vacunación y se dirige a él. A continuación se inicia otro bucle while cuyas condiciones son que el contador sea menor que quince, es decir, que haya vacunado a menos de quince pacientes en ese puesto; y que el atributo cerrarPuesto sea false. En ese caso, intenta coger una vacuna de la cola de vacunas y después espera a que llegue un paciente. Dentro del bucle comprueba de nuevo que el puesto no vaya a cerrarse por limpieza. Vacuna al paciente, lo registra en el Log, le envía a la sala de observación y cambia el atributo “vacunado” del paciente a true. Después aumenta el contador.

Cuando sale de este bucle, comprueba si ha salido porque ha vacunado a quince pacientes o porque el puesto se va a limpiar, lo registra en el Log, se va a la sala de descanso y deja el contador a cero y cerrarPuesto=false. Después del descanso sale de la sala de descanso y se inicia de nuevo su ciclo desde comprobar si hay pacientes con reacción a la vacuna en la sala de observación.

El resto de métodos de esta clase son getter de contador e id_sanitario y setter de cerrarPuesto.

Paciente

La clase Paciente sirve para modelar a los usuarios que llegan al hospital para vacunarse. Sus atributos son: un identificador (id_paciente) de tipo String, boolean vacunado que se inicializa en false hasta que ya le han puesto la vacuna, que se cambia a true, otro boolean que se inicializa en true y cambia a false si tras las comprobaciones realizadas por el auxiliar, resulta que no estaba citado para vacunarse y el Hospital, que es la variable compartida.

En su constructor, recibe como parámetros el identificador y el hospital. Como la clase extiende de Thread porque los pacientes se modelan como hilos, el comportamiento de su ciclo de vida se define en el método run. Lo primero que hace el paciente es ponerse a la cola y después, espera a que le vacunen o a que su variable registroCorrecto cambie a false. Si es vacunado la que cambia a true, espera diez segundos en observación y después, con una probabilidad del cinco por ciento tendrá reacción a la vacuna y generará una alerta para que un sanitario vaya a atenderle, o por el contrario, se irá a casa. En cualquier caso, lo deja registrado en el Log. Después, se acaba su ciclo de vida.

El resto de métodos son getter de id_paciente y setter de vacunado y registroCorrecto.

Auxiliar

La clase auxiliar sirve para modelar los hilos de los auxiliares que trabajan en el hospital. Hay dos auxiliares, uno que se encarga de registrar a los pacientes y asignarles un puesto de vacunación, y otro que crea las vacunas. Ambos, se diferencian por su identificador, el primero será “A1” y el otro, “A2”.

Los atributos de esta clase son: un String id_auxiliar, que almacena el identificador del auxiliar y la variable compartida Hospital.

El método principal de esta clase es el run, ya que la clase extiende de Thread. Lo primero que hace es definir una variable local contador que se inicializa en 0 y lleva el recuento de pacientes registrados o vacunas creadas por los auxiliares para mandarles a descansar cuando alcancen diez y veinte respectivamente. Después, se inicia un bucle infinito en el que se comprueba el identificador del auxiliar para saber cuál es su función. Si es A1, lo que hace es coger un paciente de la cola donde esperan, comprobar si está registrado y en caso correcto, escribirlo en el log y asignarle un puesto de vacunación. Después, escribe en el log los datos del paciente, el sanitario que le va a vacunar y el puesto en el que se le administra la vacuna. Si su registro es incorrecto, lo escribe en el log y cambia el atributo resgistroCorrecto del paciente a false, para que finalice su ciclo de vida.

Una vez finalizado el registro, correcta o incorrectamente, aumenta el contador y comprueba si ya ha registrado a diez pacientes. Si es así, se va a descansar a la sala de descanso durante un intervalo de entre dos y cuatro segundos y después, pone el contador a cero y vuelve a su puesto para seguir registrando pacientes de la cola.

El otro auxiliar se dedica a crear vacunas e introducirlas en la cola, cada vacuna la crea en un intervalo aleatorio entre medio segundo y un segundo y cuando ha creado veinte, se va a descansar a la sala de descanso entre uno y cuatro segundos. Cuando termina de descansar, pone el contador a cero y vuelve a su puesto para seguir creando vacunas.

El otro método de esta clase es “private boolean es_registro_correcto” una función que devuelve un booleano true si el registro del paciente es correcto y false si no, es decir, si el paciente no tenía cita para vacunarse. La elección es aleatoria, un 1% de los pacientes que llegan, no estaban citados.

Hospital

La clase Hospital es la que funciona como variable compartida, en ella es donde se han creado todos los métodos que utilizan el resto de clases y contiene las regiones críticas y los mecanismos para protegerlas. Por lo tanto es la más extensa y la que modela realmente el funcionamiento del programa.

Los atributos de esta clase son: dos LinkedBlockingQueue, una para la cola de pacientes y otra para la de vacunas; tres semáforos: uno para la sala de vacunación con diez permisos, otro para la de observación con veinte permisos, para los aforos y otro para escribir en el log con un permiso, este último es para dar exclusión mutua a la escritura en el Log.

Después hay un array de tamaño diez de Paciente para los puestos de vacunación y otro igual de Sanitario. Con ellos se modelan los puestos de vacunación. Tres arrays de String, dos de tamaño veinte para los pacientes y sanitarios en los puestos de observación y uno de tamaño diez para los sanitarios en la sala de descanso. Por último, un array de tipo boolean de tamaño veinte que sirve para activar las alertas cuando un paciente sufre una reacción a la vacuna.

Cuatro cerrojos de tipo ReentrantLock, uno para cada sala que sirven para garantizar la exclusión mutua cuando se editan los vectores de pacientes y sanitarios en cada sala. Después hay seis atributos de tipo string que almacenan lo que se debe escribir en cada cuadro de la ventana de la interfaz, dos arrays de tipo string con lo que se debe escribir en los puestos de vacunación y observación de la ventana y un atributo de tipo ListaThreads que sirve para mostrar la cola de pacientes que hay esperando en el hospital.

Por último, hay un atributo BufferedWriter y un FileWriter que se utilizan para escribir en el txt que sirve como Log.

El constructor del hospital no recibe ningún parámetro porque todos se inicializan dentro de la clase. Los puestos vacíos, las alertas en false y los String para rellenar la ventana de la interfaz, vacíos también.

A continuación, se describen todos los métodos de la clase.

HacerCola(Paciente paciente): este método recibe como parámetro un paciente. La finalidad del método es introducir al paciente en la cola de pacientes que esperan para entrar a vacunarse. En primer lugar, se activa el cerrojo de la recepción para garantizar la exclusión mutua, después, se introduce al paciente usando offer y se introduce en la

colaEsperaLista (atributo de tipo ListaThreads) para poder mostrarlo en la ventana de la interfaz. Se registra en el log y después, se avisa mediante un notifyAll a los hilos que esperan para coger un paciente de la cola porque estaba vacía. Finalmente, se libera el cerrojo.

sacarDeCola(): este método se utiliza cuando el auxiliar A1 quiere coger un paciente de la cola para registrarle. Lo primero que hace, es comprobar si hay pacientes dentro, si no, lo escribe en el log y se queda esperando mediante wait(). Cuando recibe notify de que ya hay pacientes dentro, pone el cerrojo de la sala de recepción, coge uno con el método poll() y también lo saca de colaEsperaLista para que se actualice en la ventana de la interfaz. Lo registra en el log y devuelve el paciente registrado para que el auxiliar le pueda asignar un puesto de vacunación. Por último, el cerrojo se libera.

entrarSalaDescanso(String id_sanitario): esta función es utilizada por los sanitarios para dirigirse a la sala de descanso. En primer lugar activamos el cerrojo correspondiente a la sala de descanso con Lock(). A continuación comprobamos las posiciones del array de sanitariosSalaDescanso, de manera que si hay alguna posición en “null” coloca al sanitario (identificado con id_sanitario), salimos del bucle con “Break” porque ya se ha asignado un puesto.

Además creamos un String llamado “contenido” que almacena los identificadores de todo el personal sanitario que se encuentran en la sala de descanso, de forma que recorremos todas las posiciones del array de sanitariosSalaDescanso añadiendo las posiciones diferentes de “null” a “contenido”.
Por último comprobamos si alguno de los auxiliares está de descanso y le añadimos al String “contenido” para poder enviarlo a la ventana de la interfaz y finalmente, liberamos el cerrojo con Unlock().

salirSalaDescanso(String id_sanitario): este metodo, es similar a la anterior, funcionando a la inversa. Comenzamos activando el cerrojo que corresponde a la sala de descanso con Lock() y a continuación comprobamos si alguna posición del array sanitariosSalaDescanso se encuentra rellena con (id_sanitario). Si encuentra al sanitario que busca la función, establece esa posición del array en “null” simulando así que deja la sala de descanso.

De nuevo creamos la variable String “contenido” y recorremos el array sanitarioSalaDescanso añadiendo a contenido las posiciones del array diferentes de “null”, además de añadir a los auxiliares en caso de que alguno de ellos esté descansando. Finalmente liberamos el cerrojo con Unlock().

asignarPuestoVacunacionSanitario(Sanitario id_sanitario): esta función, permite rellenar los puestos de vacunación de sanitarios devolviendo el número correspondiente del puesto. Comenzamos activando el cerrojo correspondiente a la sala de vacunación con Lock() e inicializamos una variable de tipo “int” llamada puesto en -1 (nunca se devolverá este valor porque siempre hay algún puesto vacío, ya que hay 10 sanitarios para 10 puestos de vacunacion).
A continuación recorremos el array sanitarioPuestoVacunacion, de forma que si encontramos un puesto vacío, le asignamos el sanitario, obtenemos el puesto y hacemos

“break” para salir del bucle. Por último liberamos el cerrojo con Unlock() y devolvemos el puesto.

hayPaciente(int puesto): esta función, devuelve “True” o “False” en función de si hay un paciente asignado al puesto que le pasamos al método.
Si en el array pacientePuestoVacunacion la posición del puesto está vacía, la función devolverá False, si no True.

vacunar(int puesto): función que devuelve el paciente una vez ha sido vacunado.
La función comprueba que exista un paciente en el puesto gracias al método anterior y en ese caso, la función devuelve el Paciente que estaba en ese puesto para que el Sanitario sepa a quién está vacunando y pueda continuar su ejecución.

dejarPuestoVacunacionSanitario(int puestoAsignado): método empleado para simular que los sanitarios dejan el puesto de vacunación cuando les toca descansar o cierran su puesto para limpiarlo.
Comenzamos activando ell cerrojo correspondiente a la sala de vacunación con Lock().
La función recibe como parámetro el puesto del sanitario que va a abandonar y utiliza este valor para establecer dicha posición en el array sanitarioPuestoVacunación en “null” para simular que está vacío. Además, lo cambia en el array de Strings que se utiliza para mostrar el estado del hospital en la interfaz.
Por último se libera el cerrojo con Unlock().

hayAlertas(): función utilizada para comprobar si hay algún puesto de la sala de observación en alerta y devolver el puesto para que un sanitario que termina de descansar, atienda la alerta.
Comenzamos creando una variable “int” puestoAlerta en la que vamos a almacenar el puesto que devolveremos en caso de que exista alguna alerta.
El cerrojo correspondiente con la sala de observación activa con Lock() para garantizar la exclusión mutua.
A continuación recorremos el array alertasObservacion para comprobar si hay alguna posición en “True” en ese caso, se asigna el puesto a la variable puestoAlerta y se sale del bucle con “break”. Siempre devuelve el primer puesto con alerta, independientemente de si ha sido el primer paciente en ponerse malo.
Por último liberamos el cerrojo con Unlock() y devolvemos la variable puestoAlerta.

atenderAlerta(int puestoAlerta, String id_sanitario): este método se utiliza para atender la alerta que se produce en uno de los puestos de la sala de observación obtenido con la función anterior.
Comenzamos haciendo Lock() en el cerrojo de la sala de observación. En el array llamado sanitarioPuestoObservacion colocamos en la posición del puestoAlerta el id_sanitario, de manera que juntamos al sanitario con el paciente en el puesto que tenía alerta.
A continuación ponemos la posición del puestoAlerta en "false" en el array alertasObservacion para simular que se ha solucionado la alerta. En el array de Strings puestosObservacion ponemos junto al id del paciente, -> id del sanitario que ha ido a atenderle.
Para finalizar establecemos el cerrojo en Unlock() y devolvemos el identificador del paciente en la posición puestoAlerta.

asignarPuestoVacunacionPaciente(Paciente paciente): este método, permite rellenar los puestos de vacunación de pacientes devolviendo el número correspondiente del puesto. Comenzamos haciendo acquire() sobre el semáforo salaVacunacion para que no permita entrar a la función a más de diez hilos. Esto es así porque el aforo de la sala de vacunación es de diez pacientes, por lo tanto, no podrán ocuparse más de diez puestos. Los permisos del semáforo se liberan cuando el paciente abandona la sala de vacunación. A continuación, activamos el cerrojo correspondiente a la sala de vacunación con Lock() e inicializamos una variable de tipo "int" llamada puestoAsignado en -1.
Como hemos hecho en otros métodos, buscamos un puesto que tenga asignado a un sanitario, que no esté ocupado por otro paciente y que el sanitario no haya vacunado aún a 15 pacientes, en ese caso, le tocaría descansar. Recorremos el array sanitarioPuestoVacunacion,buscando un puesto diferente de "null" y el array pacientePuestoVacunacion buscando un puesto "null".
Una vez que encontramos el puesto correcto, lo almacenamos en la variable puestoAsignado y colocamos al paciente en el puesto correspondiente. En el puesto de vacunación correspondiente, añadimos el identificador del paciente para simular que se dirige a ese puesto y poder mostrarlo en la ventana.

Por último hacemos notifyAll() para avisar a todos los hilos de sanitario que estaban esperando a que hubiese un paciente en su puesto para vacunarle. Hacemos "break" para salir del bucle, ponemos el cerrojo en Unlock y devolvemos la variable puestoAsignado.

dejarPuestoSalaVacunacionPaciente(int puesto): función empleada para simular que los pacientes dejan el puesto de vacunación para dirigirse a la sala de observación.
Comenzamos estableciendo activando el cerrojo correspondiente a la sala de vacunación con Lock().
A continuación se establece la posición recibida como parámetro del array pacientePuestoVacunacion en "null", y en la posición correspondiente del array puestosVacunacion se deja simplemente el identificador del sanitario, para que en la ventana se muestre que el paciente se ha marchado.
Como mencionamos en la función asignarPuestoVacunacionPaciente, una vez que el paciente deja la sala, se hace release al semáforo salaVacunación, para permitir la entrada de un nuevo paciente a la sala. Por último se modifica el cerrojo para ponerlo en Unlock().

asignarPuestoSalaObservacion(String paciente, int puestoVacunacion): esta función sirve para asignar un puesto en la sala de observación a cada paciente que se vacuna. Recibe como parámetros el identificador del paciente y el puesto de la sala de vacunación en el que estaba. Primero intenta hacer acquire al semáforo que controla el aforo de la sala de observación y cuando lo consigue, abandona la sala de vacunación con el método anterior "dejarPuestoSalaVacunaciónPaciente(puestoVacunacion)".
Ahora se activa el cerrojo de la sala de observación con Lock(), se busca el primer puesto vacío recorriendo el array de puestos de observación y cuando encuentra uno en null, se lo asigna al paciente y lo escribe en el array de Strings que se muestra en la ventana. Por último se libera el cerrojo con Unlock()

generarAlerta(String id_paciente): Este método sirve para avisar a los sanitarios cuando un paciente sufre una reacción a la vacuna. El paciente la llama indicando su id y la función primero activa el cerrojo de la sala de observación con Lock() y después busca el puesto que ocupa recorriendo el array de pacientePuestoObservación. Cuando le encuentra, lo escribe en el Log y pone en true su puesto en el array alertasObservacion. Finalmente, hace break para salir del bucle y libera el cerrojo de la sala de observación con Unlock().

anadirVacuna(): Este método lo utiliza el auxiliar A2 para añadir vacunas en la cola. Las vacunas se modelan con un Object y se introducen en la colaVacunas con offer(). En el String vacunasDisponibles, se actualiza su valor al nuevo tamaño de la cola. Por último hace un notifyAll por si hay algún hilo de Sanitario bloqueado esperando a que haya vacunas para poder cogerla.

cogerVacuna(): Unido al método anterior, los sanitarios llaman a este procedimiento para coger una vacuna de la cola de vacunas. En caso de que la cola esté vacía, lo escribe en el Log y espera con wait() a que haya vacunas. Cuando haya vacunas, coge una y actualiza el String con el tamaño de la cola para mostrar en la ventana el número de vacunas disponibles.

dejarPuestoSalaObservacionPaciente(String id_paciente): este método se utiliza cuando el paciente ha pasado diez segundos en observación y no ha tenido reacción a la vacuna, o bien, ha tenido reacción y ya le han atendido. Recibe como parámetro el id del paciente.
En primer lugar, se activa el cerrojo de la sala de observación con Lock(). Después, con un bucle se recorre el array de puestos de observación buscando el puesto en el que se encuentra el paciente. Cuando le encuentra, pone el puesto en null y sale del bucle. También libera el semáforo de la sala de observación, para permitir que entre otro paciente y no sobrepasar el aforo de 20 pacientes.

dejarPuestoSalaObservacionSanitario(int puestoAlerta): este método sirve para que un Sanitario deje el puesto de observación que ocupa tras atender a un paciente que había activado una alerta. Recibe como parámetro el puesto que debe abandonar.
Primero se activa el cerrojo de la sala de observación con Lock(), luego se pone el puesto de sanitarioPuestoObservación en null y el puesto que se muestra en la ventana, vacío. Por último, se libera el cerrojo con unlock().

salirAuxSalaDescanso(): este método se utiliza cuando los auxiliares terminan su descanso y vuelven a su puesto. Lo que hace es actualizar el contenido de la sala de descanso que se muestra en la ventana de la interfaz. Para ello activa el cerrojo con Lock(), recorre el array de sanitarioSalaDescanso y escribe los sanitarios que estén allí cuando él se va. Finalmente se libera el cerrojo de la sala de descanso con Unlock().

escribirEnLog(String texto, LocalDateTime fecha): Esta función recibe como parámetro un texto de tipo String y una fecha y hora. Su finalidad es escribir en el Log EvolucionHospital.txt los eventos que se producen durante la ejecución.
Para ello, se adquiere un semáforo inicializado con un solo permiso para garantizar exclusión mutua a la hora de escribir en el log. Después, se añade al String texto recibido como parámetro, la fecha y hora entre corchetes. Si existe el archivo, escribe sobre él y si no, lo crea con el método createNewFile().
Con un objeto FileWriter, conseguimos sobreescribir el archivo sin borrar lo anterior y con un objeto BufferedWriter, escribimos en el archivo. Finalmente, se cierran las instancias del FileWriter y BufferedWriter y se libera el semáforo con release().

limpiarPuesto(int puestoCerrado): Esta función recibe como parámetro la posición del puesto de vacunación que se va a limpiar para que expulse al sanitario y le envíe a descansar. Si en el momento de cerrar el puesto, hay algún paciente en él, se espera a que termine de vacunarle y después establece el atributo cerrarPuesto del Sanitario de dicho puesto en true, para que pare su ciclo de vacunación y vaya a descansar. De otro modo, si el sanitario estaba suspendido esperando a que llegara algún paciente para vacunarle, creamos un paciente ficticio (que nunca llega a ejecutar su método run) y le colocamos en la misma posición que el sanitario para que este se despierte y al ver que tiene su atributo cerrarPuesto en true, se vaya a descansar.

(getters + setters): Estos métodos sirven para poder acceder a algunos atributos privados de la clase Hospital que es necesario adquirir o editar desde otras clases.

HiloCargarDatos

Esta clase, está modelada como un hilo que se llama desde la clase Servidor y que crea todos los objetos del resto de clases y los inicia. El único atributo que tiene es el Hospital. Lo recibe en su constructor y a partir de ahí, en el método run inicializa todo lo demás.
En primer lugar, escribe una nueva línea en el Log indicando que se inicia una nueva ejecución del progama para poder diferenciarla de otras.
Lo primero que hace es crear e iniciar con start() los hilos de los sanitarios. Sus identificadores son S0, S1, … , S9. Después hace lo mismo con los auxiliares, cuyos identificadores son A1 y A2. Por último, inicializa los 2000 pacientes. En este caso, para evitar aglomeraciones, lo hace en intervalos aleatorios de entre 1 y 3 segundos entre cada paciente. Sus identificadores son: P0000, P0001, … , P1999.

ListaThreads

Esta clase es, básicamente, una reutilización de código de una de las sesiones de laboratorio del curso. Sus atributos son un ArrayList de Paciente y un String que almacena los pacientes que hay en el array en formato String.
Su constructor recibe como parámetro el String donde se guardan los identificadores de los pacientes de la cola.

Los métodos de la clase son: meter(Paciente t) y sacar(Paciente t) que emplean la cláusula synchronized para evitar condiciones de carrera. Meter utiliza el método add() para añadir pacientes al ArrayList y sacar, lo elimina con remove(). El método imprimir() recorre el ArrayList y escribe su contenido en el String cuadroColaEspera que se utilizará para poder visualizar la cola en la ventana de la interfaz. Por último, el getter del cuadroColaEspera, para poder mostrarlo.

HiloCliente

Esta clase es la más importante para poder visualizar los datos de la ejecución del programa en la ventana JFrame. Su finalidad principal es conectarse mediante un Socket a la clase Servidor que le envía lo que debe imprimir en cada uno de los JTextField de la ventana.

Sus atributos son: la ventana JFrame, objeto de la clase VentanaHospital, el Socket cliente y los canales DataInputStream entrada y DataOutputStream salida para poder enviar y recibir datos a través del socket. El constructor recibe como parámetro la ventana.

En su método run, lo que hace es cada segundo, intentar conectarse con el cliente en el puerto 5000, abrir los canales de entrada y salida y guardar todos los datos que recibe para poder mandárrselos a la ventana y que los imprima en los JTextField que le corresponda con el método rellenarVentana. Cada vez que recibe toda la información, envía también un -1, este dato no es necesario porque se usa cuando se cierran puestos para la limpieza pero lo hacemos así para utilizar el mismo servidor para todo. Tras cada envío y recepción de datos, cierra el socket para después abrir otro.

Servidor

Esta clase es de tipo main, es decir, es la que hay que ejecutar para que el programa empiece a funcionar. Ella es la encargada de crear el hospital y el hilo inicializador que carga los datos. Cuando ya ha hecho esto, lo siguiente que tiene que hacer es abrir un socket para poder enviar información a la ventana de la interfaz de lo que está sucediendo en el hospital en cada momento. También puede recibir los puertos que se deben cerrar para limpiar y tiene que notificarlo al hospital para que cierre el puesto y envíe al sanitario a descansar. En este caso, el servidor no se cierra cada vez que termina una conexión, sino que utiliza la interfaz ServerSocket para crear una conexión para cada solicitud del servidor. Desde ahí, acepta la conexión del cliente y envía todo lo que hay en los String que corresponden a las casillas de la ventana. También, recibe del cliente un entero que indica si se ha solicitado cerrar algún puesto de vacunación. Si recibe -1, quiere decir que la

conexión era simplemente para actualizar la ventana. Si recibe un valor diferente, quiere decir que se ha pulsado el botón de cerrar de algún puesto y por lo tanto, debe avisar al hospital de que cierre el puesto.
Tanto el cliente como el servidor utilizan el canal de salida para escribir con writeUTF para String y writeInt para enteros. Y el canal de entrada para leer con readUTF y readInt.
Una vez recibidos y enviados todos los datos, se cierra la conexión con el cliente pero no el servidor, el servidor sigue activo.

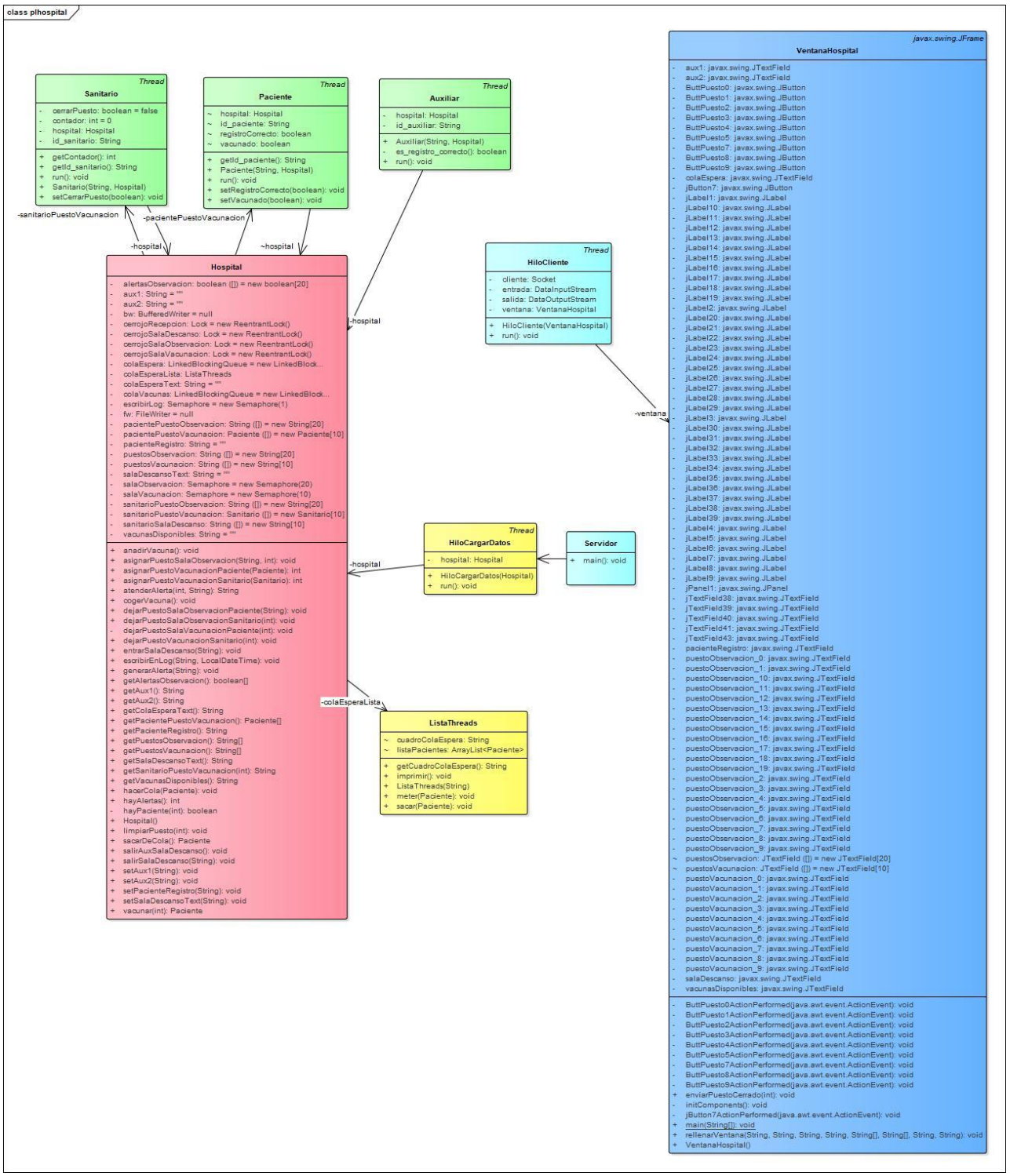
VentanaHospital

Esta clase contiene el JFrame con todos los JTextFields y los JButtons que muestran de forma gráfica el estado del hospital en todo momento. Lo que hace es crear un objeto de la clase HiloCliente e iniciarlo con start() para que se conecte con el servidor y obtenga el estado del hospital cada segundo. Desde HiloCliente se llama a la función rellenarVentana que recibe como parámetros todos los Strings que tiene que escribrir en los JTextField. Lo escribe con setText().
Su otra función sirve para enviar mediante una conexión al Socket Servidor los puestos que se quieren cerrar porque se ha pulsado alguno de sus botones de cierre. Para ello, los botones tienen asociado un evento ButtPuesto**n**ActionPerformed que llama a esta función pasándole como parámetro el número del puesto que se quiere cerrar.

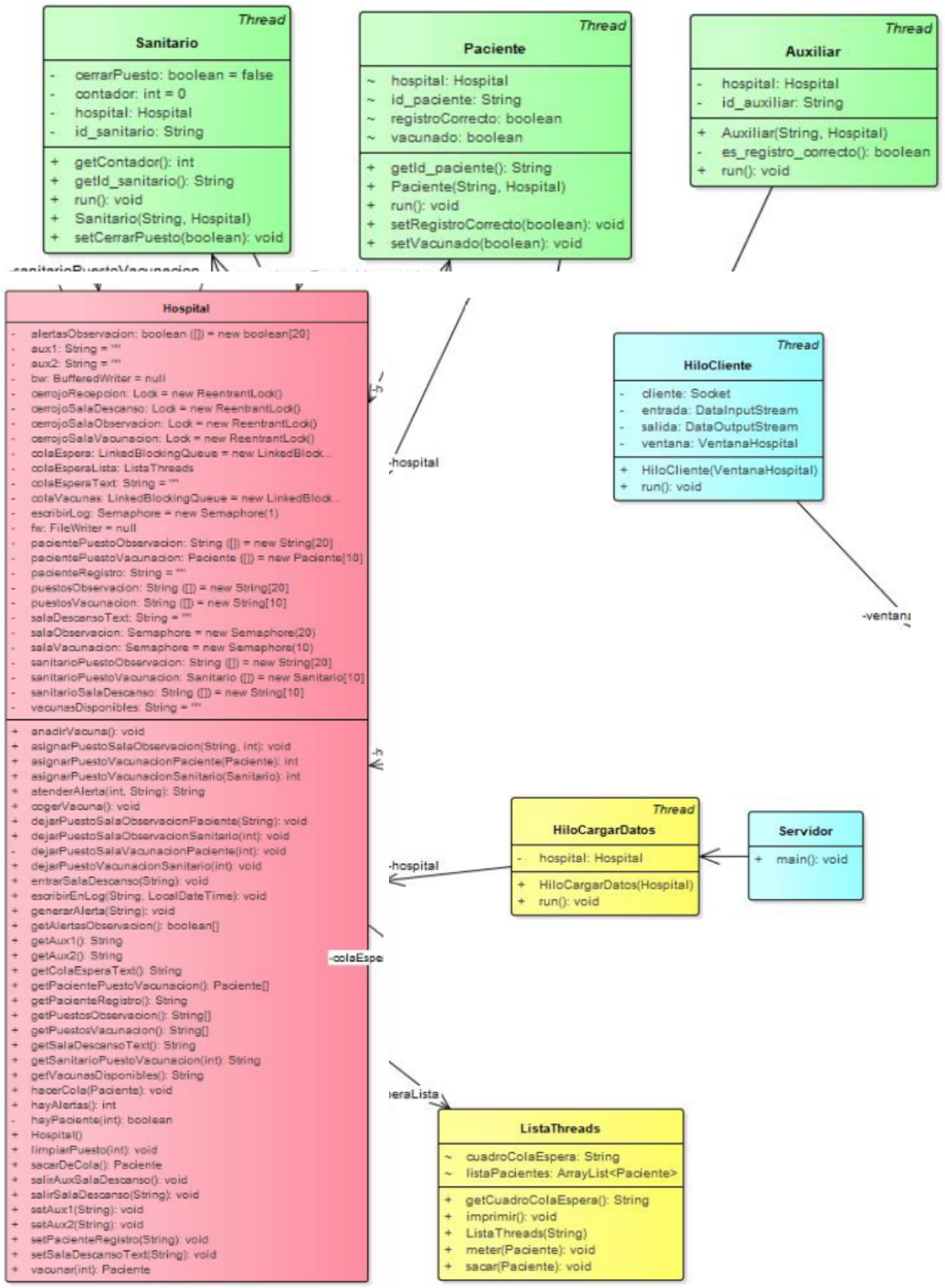
Generalidades: Todos los tiempos aleatorios se modelan mediante la función random() importada de la librería Math. Esta función devuelve un número entre 0.0 y 1.0, la multiplicamos por la diferencia entre los segundos del intervalo +1 y le sumamos el valor de inicio del intervalo. Después, pasamos este número real a entero y lo multiplicamos por 1000 para dormir el hilo con sleep durante ese tiempo, dado en milisegundos.

4. Diagrama de clases

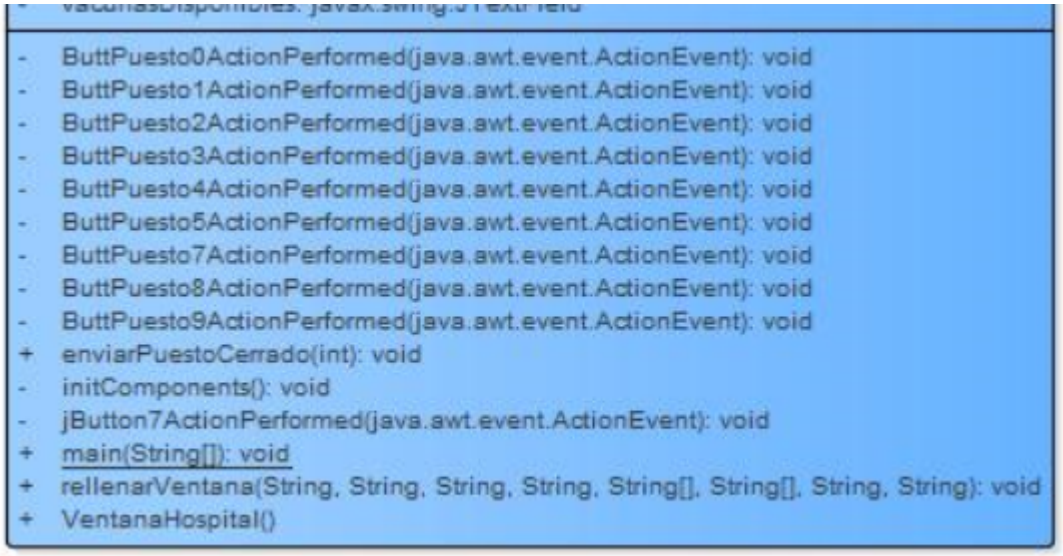
Esquema general:



Esquemas por partes para una mejor visualización



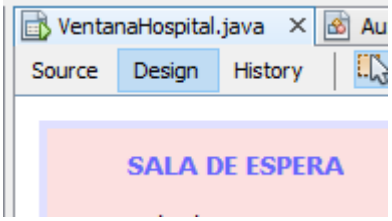
De la clase VentanaHospital destacamos las funciones de pulsar los botones de cierre de puestos y la de rellenar los JTextField. Los atributos de esta clase son todos los elementos que se visualizan en la ventana.

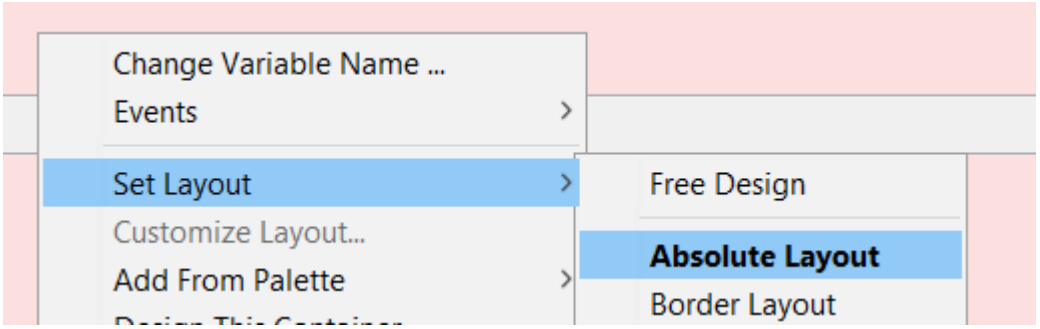


5. Anexos

a. Breve explicación para ejecutar el proyecto en NetBeans

Si en el momento de abrir el proyecto no se permite ejecutar las ventanas con método main, es necesario abrir la clase VentanaHospital, pulsar en design, botón derecho al JFrame y set LayOut → Absolute Layout.





b. Código fuente del programa

Se adjunta el proyecto NetBeans completo.

Código fuente

Auxiliar

```
package com.mycompany.plhospital;

import java.time.LocalDateTime;


public class Auxiliar extends Thread{
   private String id_auxiliar;
   private Hospital hospital;

   public Auxiliar(String id_auxiliar, Hospital hospital) {
      this.id_auxiliar = id_auxiliar;
      this.hospital = hospital;
   }

   public void run(){
      int contador=0;
      int tiempoDescanso;
      int tiempoVacuna;
      while(true){
         if(this.id_auxiliar == "A1"){
            hospital.setAux1(id_auxiliar);
            Paciente paciente = (Paciente) hospital.sacarDeCola();
            hospital.setPacienteRegistro(paciente.getId_paciente());
            if(es_registro_correcto()){
               String text= ("El paciente "+paciente.getId_paciente()+" se ha registrado "
                     + "correctamente.");
               hospital.escribirEnLog(text, LocalDateTime.now());
               int puestoAsignado=-1;
               while(puestoAsignado==-1){
                  puestoAsignado= hospital.asignarPuestoVacunacionPaciente(paciente);
               }
               String sanitario = hospital.getSanitarioPuestoVacunacion(puestoAsignado);
               text= ("El paciente "+paciente.getId_paciente()+" será vacunado por el sanitario
"+sanitario+ " en el puesto PV0"+puestoAsignado);
               hospital.escribirEnLog(text, LocalDateTime.now());
            }
            else{
               String text =("El paciente "+paciente.getId_paciente()+" se ha registrado "
                     + "incorrectamente, debe irse a casa.");
               hospital.escribirEnLog(text, LocalDateTime.now());
               paciente.setRegistroCorrecto(false);
```

```
            }
            contador++;
            if(contador==10){
                tiempoDescanso=(int)(Math.random()*3+2)*1000;
                String text= ("El auxiliar A1 ha registrado 10 pacientes,"
                        + "descansa "+tiempoDescanso+".");
                hospital.escribirEnLog(text, LocalDateTime.now());
                hospital.setAux1("");
                String salaDescanso=hospital.getSalaDescansoText()+" "+id_auxiliar;
                hospital.setSalaDescansoText(salaDescanso);
                try {
                    sleep(tiempoDescanso);
                } catch (InterruptedException ex) {}
                hospital.setAux1(id_auxiliar);
                hospital.salirAuxSalaDescanso();
                contador=0;
            }
        }
        else{
            hospital.setAux2(id_auxiliar);
            hospital.anadirVacuna();
            tiempoVacuna=(int) (Math.random()*501 + 500);
            try {
                sleep(tiempoVacuna);
            } catch (InterruptedException ex) {}
            contador++;
            if (contador==20){
                tiempoDescanso = (int) (Math.random()*4 +1)*1000;
                String text = ("El auxiliar A2 ha creado 20 vacunas,"
                        + "descansa "+tiempoDescanso+".");
                hospital.escribirEnLog(text, LocalDateTime.now());
                hospital.setAux2("");
                String salaDescanso=hospital.getSalaDescansoText()+" "+id_auxiliar;
                hospital.setSalaDescansoText(salaDescanso);
                try{
                    sleep(tiempoDescanso);
                }
                catch(InterruptedException ie){}
                hospital.setAux2(id_auxiliar);
                hospital.salirAuxSalaDescanso();
                contador=0;
            }
        }
    }

}
```

```
    private boolean es_registro_correcto(){
        boolean resultado=true;
        int tiempoComprobacion = (int) (Math.random()*501+500);
        try {
            sleep(tiempoComprobacion);
        } catch (InterruptedException ex) {}
        int aleatorio = (int) (Math.random()*100+1); //numero aleatorio entre 1 y 100
        if (aleatorio==1){ //1 % de pacientes no estaban citados.
            resultado=false;
        }
        return resultado;
    }
}
```

HiloCargarDatos

```
package com.mycompany.plhospital;

import java.time.LocalDateTime;

public class HiloCargarDatos extends Thread {
    private Hospital hospital;
    public HiloCargarDatos (Hospital hospital){
        this.hospital=hospital;
    }
    public void run(){
        hospital.escribirEnLog("\n ----------------------NUEVA EJECUCIÓN DEL HOSPITAL -->
", LocalDateTime.now());
        String id_sanitario;
        for(int i=0; i<10; i++){
            id_sanitario = "S0"+i;
            Sanitario s = new Sanitario(id_sanitario, hospital);
            s.start();
        }
        Auxiliar a1 = new Auxiliar("A1", hospital);
        Auxiliar a2 = new Auxiliar ("A2", hospital);
        a1.start();
        a2.start();
        try {
            sleep(1500);
        } catch (InterruptedException ex) {}
        String id_paciente;
        Paciente p;
        int tiempoAleatorio;
        for (int i=0; i<10; i++){
            id_paciente = "P000"+i;
```

```
            p = new Paciente(id_paciente, hospital);
            p.start();
            tiempoAleatorio=(int) (Math.random()*3+1)*1000;
            try {
                sleep(tiempoAleatorio);
            } catch (InterruptedException ex) {}
        }
        for (int i=10; i<100; i++){
            id_paciente = "P00"+i;
            p = new Paciente(id_paciente, hospital);
            p.start();
            tiempoAleatorio=(int) (Math.random()*3+1)*1000;
            try {
                sleep(tiempoAleatorio);
            } catch (InterruptedException ex) {}
        }
        for (int i=100; i<1000; i++){
            id_paciente = "P0"+i;
            p = new Paciente(id_paciente, hospital);
            p.start();
            tiempoAleatorio=(int) (Math.random()*3+1)*1000;
            try {
                sleep(tiempoAleatorio);
            } catch (InterruptedException ex) {}
        }
        for (int i=1000; i<2000; i++){
            id_paciente = "P"+i;
            p = new Paciente(id_paciente, hospital);
            p.start();
            tiempoAleatorio=(int) (Math.random()*3+1)*1000;
            try {
                sleep(tiempoAleatorio);
            } catch (InterruptedException ex) {}
        }
    }
}
```

HiloCliente

```
package com.mycompany.plhospital;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
```

```
public class HiloCliente extends Thread {
    private VentanaHospital ventana;
    private Socket cliente;
    private DataInputStream entrada;
    private DataOutputStream salida;



    public HiloCliente(VentanaHospital ventana){
        this.ventana=ventana;
    }

    public void run(){
        String aux1, aux2, vacunasDisponibles, colaEspera, salaDescanso, pacienteRegistro;
        String[] puestosVacunacion = new String[10];
        String [] puestosObservacion = new String[20];
        while(true){
            try {
                sleep(1000);
            } catch (InterruptedException ex) {}
            try {
                cliente = new Socket(InetAddress.getLocalHost(),5000);
                entrada = new DataInputStream(cliente.getInputStream());  //Creamos los canales
de entrada/salida
                salida = new DataOutputStream(cliente.getOutputStream());
                aux1 = entrada.readUTF();
                aux2 = entrada.readUTF();
                vacunasDisponibles = entrada.readUTF();
                colaEspera= entrada.readUTF();
                for (int i=0;i<10;i++){
                    puestosVacunacion[i]=entrada.readUTF();
                }
                for (int i=0;i<20;i++){
                    puestosObservacion[i]=entrada.readUTF();
                }
                salaDescanso=entrada.readUTF();
                pacienteRegistro=entrada.readUTF();
                salida.writeInt(-1);
                cliente.close();
                ventana.rellenarVentana(aux1, aux2, vacunasDisponibles, colaEspera,
puestosVacunacion, puestosObservacion, salaDescanso, pacienteRegistro);
            } catch (IOException ex) {}

        }
    }
}
```

Hospital

```
package com.mycompany.plhospital;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.util.concurrent.*;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;


/**
 *
 * @author lucyr
 */
public class Hospital {
    private LinkedBlockingQueue colaEspera = new LinkedBlockingQueue();
    private LinkedBlockingQueue colaVacunas = new LinkedBlockingQueue();
    private Semaphore salaVacunacion = new Semaphore(10);
    private Semaphore salaObservacion = new Semaphore(20);
    private Semaphore escribirLog = new Semaphore(1);
    // La recepcion no necesita semáforo porque no tiene aforo limitado y la sala de
    // descanso del personal tampoco porque solo hay 10 sanitarios, nunca va a haber más
de 10
    private Paciente[] pacientePuestoVacunacion = new Paciente[10];
    private Sanitario[] sanitarioPuestoVacunacion = new Sanitario[10];
    private String[] pacientePuestoObservacion = new String[20];
    private String[] sanitarioPuestoObservacion = new String[20];
    private String[] sanitarioSalaDescanso = new String[10];
    private boolean[] alertasObservacion = new boolean[20];
    private Lock cerrojoSalaDescanso= new ReentrantLock();
    private Lock cerrojoRecepcion = new ReentrantLock();
    private Lock cerrojoSalaVacunacion=new ReentrantLock();
    private Lock cerrojoSalaObservacion = new ReentrantLock();
    private String colaEsperaText="";
    private String salaDescansoText="";
    private ListaThreads colaEsperaLista;
    private String[] puestosVacunacion= new String[10];
    private String[] puestosObservacion =new String[20];
    private String vacunasDisponibles="";
    private String aux1="";
    private String aux2="";
    private String pacienteRegistro="";
```

```
    private BufferedWriter bw = null;
    private FileWriter fw = null;


    public Hospital() {
        for (int i=0; i<10; i++){
            pacientePuestoVacunacion[i]=null;
            sanitarioPuestoVacunacion[i]=null;
            pacientePuestoObservacion[i]=null;
            sanitarioPuestoObservacion[i]=null;
            sanitarioSalaDescanso[i]=null;
            alertasObservacion[i]=false;
            puestosVacunacion[i]="";
            puestosObservacion[i]="";
        }
        for (int i=10; i<20;i++){
            pacientePuestoObservacion[i]=null;
            sanitarioPuestoObservacion[i]=null;
            alertasObservacion[i]=false;
            puestosObservacion[i]="";
        }
        this.colaEsperaLista=new ListaThreads(this.colaEsperaText);
    }

    public void hacerCola(Paciente paciente) {
        cerrojoRecepcion.lock();
        try{
            colaEspera.offer(paciente);
            colaEsperaLista.meter(paciente);
            String text=("El paciente "+ paciente.getId_paciente() +" entra en la cola.");
            escribirEnLog(text, LocalDateTime.now());
            synchronized (colaEspera) {  // notifica que la cola está vacía a los hilos que estaban
esperando para hacer poll;
                colaEspera.notifyAll();
            }
        }
        finally{
            cerrojoRecepcion.unlock();
        }


    }

    public Paciente sacarDeCola() {
        if (colaEspera.isEmpty()){ // si la cola está vacía, esperar a que haya algun elemento.
```

```
        synchronized (colaEspera){
            try {
                String text=("La cola esta vacía, esperando que lleguen pacientes...");
                escribirEnLog(text, LocalDateTime.now());
                colaEspera.wait();
            } catch (InterruptedException ex) {}
        }
    }
    cerrojoRecepcion.lock();
    try{
      Paciente paciente= (Paciente) colaEspera.poll();
      colaEsperaLista.sacar(paciente);
      String text=("El paciente "+ paciente.getId_paciente()+" sale de la cola.");
      escribirEnLog(text, LocalDateTime.now());
      return paciente;
    }
    finally{
        cerrojoRecepcion.unlock();
    }

}

 public void entrarSalaDescanso(String id_sanitario) {
    cerrojoSalaDescanso.lock();
    try{
        for (int i=0;i<10;i++){
            if(sanitarioSalaDescanso[i]==null){
                sanitarioSalaDescanso[i]=id_sanitario;
                break;
            }
        }
        String contenido="";
        for (int i=0;i<10;i++){
            if (sanitarioSalaDescanso[i]!=null){
                contenido=contenido+" " + sanitarioSalaDescanso[i];
            }
        }
        if(aux1==""){
            contenido = contenido+" A1";
        }
        if(aux2==""){
            contenido=contenido+" A2";
        }
        salaDescansoText=contenido;
    }
    finally{
        cerrojoSalaDescanso.unlock();
```

```
    }
}

public void salirSalaDescanso(String id_sanitario) {
    cerrojoSalaDescanso.lock();
    try{
        for (int i=0;i<10;i++){
            if(sanitarioSalaDescanso[i]==id_sanitario){
                sanitarioSalaDescanso[i]=null;
                break;
            }
        }
        String contenido="";
        for (int i=0; i<10; i++){
            if (sanitarioSalaDescanso[i]!=null){
                contenido=contenido+" " + sanitarioSalaDescanso[i];
            }
        }
        if(aux1==""){
            contenido = contenido+" A1";
        }
        if(aux2==""){
            contenido=contenido+" A2";
        }
        salaDescansoText=contenido;
    }
    finally{
        cerrojoSalaDescanso.unlock();
    }
}

public int asignarPuestoVacunacionSanitario(Sanitario id_sanitario) {
    cerrojoSalaVacunacion.lock();
    int puesto=-1;
    try{
        for (int i=0;i<10;i++){
            if(sanitarioPuestoVacunacion[i]==null){
                sanitarioPuestoVacunacion[i]=id_sanitario;
                puestosVacunacion[i]=id_sanitario.getId_sanitario();
                puesto=i;
                break;
            }
        }
    }
    finally{
        cerrojoSalaVacunacion.unlock();
    }
```

```
    return puesto;
}

public Paciente[] getPacientePuestoVacunacion() {
    return pacientePuestoVacunacion;
}

public synchronized Paciente vacunar(int puesto) {
    while(!hayPaciente(puesto)){
        try {
            wait();
        } catch (InterruptedException ex) {}
    }
    Paciente vacunado = pacientePuestoVacunacion[puesto];
    return vacunado;
}
private synchronized boolean hayPaciente(int puesto){
    if(pacientePuestoVacunacion[puesto]==null){
        return false;
    }
    return true;
}


public void dejarPuestoVacunacionSanitario(int puestoAsignado) {
    cerrojoSalaVacunacion.lock();
    try{
        sanitarioPuestoVacunacion[puestoAsignado]=null;
        puestosVacunacion[puestoAsignado]="";
    }
    finally{
        cerrojoSalaVacunacion.unlock();
    }
}

public int hayAlertas() {
    int puestoAlerta=-1;
    cerrojoSalaObservacion.lock();
    try{
        for (int i=0; i<20;i++){
            if(alertasObservacion[i]==true){
                puestoAlerta=i;
                break;
            }
        }
    }
    finally{
```

```
            cerrojoSalaObservacion.unlock();
        }
        return puestoAlerta;
    }

    public String atenderAlerta(int puestoAlerta, String id_sanitario) {
        cerrojoSalaObservacion.lock();
        try{
            sanitarioPuestoObservacion[puestoAlerta]=id_sanitario;
            alertasObservacion[puestoAlerta]=false;

puestosObservacion[puestoAlerta]=puestosObservacion[puestoAlerta]+"->"+id_sanitario;
        }
        finally{
            cerrojoSalaObservacion.unlock();
        }
        return pacientePuestoObservacion[puestoAlerta];
    }

    public synchronized int asignarPuestoVacunacionPaciente(Paciente paciente) {
        try {
            salaVacunacion.acquire();
        } catch (InterruptedException ex) {}
        cerrojoSalaVacunacion.lock();
        int puestoAsignado=-1;
        try{
            for (int i=0; i<10; i++){

                if ((sanitarioPuestoVacunacion[i]!=null) && (pacientePuestoVacunacion[i]==null)
&& (sanitarioPuestoVacunacion[i].getContador()<15)){
                    puestoAsignado=i;
                    pacientePuestoVacunacion[i]=paciente;
                    puestosVacunacion[i]=puestosVacunacion[i]+"->"+ paciente.getId_paciente();
                    notifyAll();
                    break;
                }

            }
        }
        finally{
            cerrojoSalaVacunacion.unlock();
        }
        return puestoAsignado;
    }

    private void dejarPuestoSalaVacunacionPaciente(int puesto){
        cerrojoSalaVacunacion.lock();
```

```
        try{
            pacientePuestoVacunacion[puesto]=null;
            puestosVacunacion[puesto]=sanitarioPuestoVacunacion[puesto].getId_sanitario();
            salaVacunacion.release();
        }
        finally{
            cerrojoSalaVacunacion.unlock();
        }
    }

    public void asignarPuestoSalaObservacion(String paciente, int puestoVacunacion) {
        try {
            salaObservacion.acquire(); //si consigue un puesto en la sala de observacion, deja la
de vacunacion
        } catch (InterruptedException ex) {}
        dejarPuestoSalaVacunacionPaciente(puestoVacunacion);
        cerrojoSalaObservacion.lock();
        try{
            for (int i=0;i<20;i++){
                if(pacientePuestoObservacion[i]==null){
                    pacientePuestoObservacion[i]=paciente;
                    puestosObservacion[i]=paciente;
                    break;
                }
            }
        }
        finally{
            cerrojoSalaObservacion.unlock();
        }


    }

    public String getSanitarioPuestoVacunacion(int puesto){
        cerrojoSalaVacunacion.lock();
        try{
            return sanitarioPuestoVacunacion[puesto].getId_sanitario();
        }
        finally{
            cerrojoSalaVacunacion.unlock();
        }

    }

    public void generarAlerta(String id_paciente) {
        cerrojoSalaObservacion.lock();
        try{
```

```
            for (int i=0;i<20;i++){
                if(id_paciente==pacientePuestoObservacion[i]){
                    alertasObservacion[i]=true;
                    String text= ("Alerta activada puesto: "+i);
                    escribirEnLog(text, LocalDateTime.now());
                    break;
                }
            }
        }
        finally{
            cerrojoSalaObservacion.unlock();
        }
    }

    public void anadirVacuna() {
        Object vacunaNueva= new Object();
        colaVacunas.offer(vacunaNueva);
        vacunasDisponibles=colaVacunas.size()+"";
        synchronized (colaVacunas) {  // notifica que la cola está vacía a los hilos que estaban
esperando para hacer poll;
            colaVacunas.notifyAll();
        }
    }

    public void cogerVacuna() {
        if (colaVacunas.isEmpty()){ // si la cola está vacía, esperar a que haya algun elemento.
            synchronized (colaVacunas){
                try {
                    String text= ("La cola esta vacía, esperando que lleguen vacunas...");
                    escribirEnLog(text, LocalDateTime.now());
                    colaVacunas.wait();
                } catch (InterruptedException ex) {}
            }
        }
        colaVacunas.poll();
        vacunasDisponibles=colaVacunas.size()+"";
    }

    public void dejarPuestoSalaObservacionPaciente(String id_paciente) {
        cerrojoSalaObservacion.lock();
        try{
            for (int i=0;i<20;i++){
                if(pacientePuestoObservacion[i]==id_paciente){
                    pacientePuestoObservacion[i]=null;
                    puestosObservacion[i]="";              }
            }
            salaObservacion.release();
```

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

}

public void dejarPuestoSalaObservacionSanitario(int puestoAlerta) {
    cerrojoSalaObservacion.lock();
    try{
        sanitarioPuestoObservacion[puestoAlerta]=null;
        puestosObservacion[puestoAlerta]="";
    }
    finally{
        cerrojoSalaObservacion.unlock();
    }
}

public void salirAuxSalaDescanso(){
    cerrojoSalaDescanso.lock();
    try{
        String contenido="";
        for (int i=0; i<10; i++){
            if (sanitarioSalaDescanso[i]!=null){
                contenido=contenido+" " + sanitarioSalaDescanso[i];
            }
        }
        if(aux2==""){
            contenido = contenido + " " + "A2";
        }
        if(aux1==""){
            contenido = contenido + " " + "A1";
        }
        salaDescansoText=contenido;
    }
    finally{
        cerrojoSalaDescanso.unlock();
    }
}

public boolean[] getAlertasObservacion() {
    return alertasObservacion;
}

public String getColaEsperaText() {
    return colaEsperaLista.getCuadroColaEspera();
}
```

```
public String getSalaDescansoText() {
    return salaDescansoText;
}

public String[] getPuestosVacunacion() {
    return puestosVacunacion;
}

public String[] getPuestosObservacion() {
    return puestosObservacion;
}

public String getVacunasDisponibles() {
    return vacunasDisponibles;
}

public String getAux1() {
    return aux1;
}

public String getAux2() {
    return aux2;
}

public String getPacienteRegistro(){
    return pacienteRegistro;
}

public void setPacienteRegistro(String pacienteRegistro) {
    this.pacienteRegistro = pacienteRegistro;
}

public void setAux1(String aux1) {
    this.aux1 = aux1;
}

public void setAux2(String aux2) {
    this.aux2 = aux2;
}

public void setSalaDescansoText(String salaDescansoText) {
    this.salaDescansoText = salaDescansoText;
}
```

```
    public void escribirEnLog(String texto, LocalDateTime fecha){
        try {
            escribirLog.acquire();
        } catch (InterruptedException ex) {}
        try {
            texto = "\n"+texto+" [ "+fecha+" ]";
            File log = new File("EvolucionHospital.txt");
            // Si el archivo no existe, se crea
            if (!log.exists()) {
                log.createNewFile();
            }
            // flag true, indica adjuntar información al archivo.
            fw = new FileWriter(log.getAbsoluteFile(), true);
            bw = new BufferedWriter(fw);
            bw.write(texto);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                //Cierra instancias de FileWriter y BufferedWriter
                if (bw != null){
                    bw.close();
                }
                if (fw != null){
                    fw.close();
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
        escribirLog.release();

    }

    public synchronized void limpiarPuesto(int puestoCerrado) {
        if(pacientePuestoVacunacion[puestoCerrado]==null){
            Paciente pacienteFicticio=new Paciente("", this);
            pacientePuestoVacunacion[puestoCerrado]=pacienteFicticio;
        }
        sanitarioPuestoVacunacion[puestoCerrado].setCerrarPuesto(true);
    }


}
```

ListaThreads

```
package com.mycompany.plhospital;

import java.util.ArrayList;

public class ListaThreads {
   ArrayList<Paciente> listaPacientes;
   String cuadroColaEspera;
   public ListaThreads(String cuadroColaEspera)
   {
      listaPacientes=new ArrayList<Paciente>();
      this.cuadroColaEspera=cuadroColaEspera;
   }

   public synchronized void meter(Paciente t)
   {
      listaPacientes.add(t);
      imprimir();
   }
   public synchronized void sacar(Paciente t)
   {
      listaPacientes.remove(t);
      imprimir();
   }
   public void imprimir()
   {
      String contenido="";
      for(int i=0; i<listaPacientes.size(); i++)
      {
        contenido=contenido+listaPacientes.get(i).getId_paciente()+" ";
      }
      cuadroColaEspera=contenido;
   }

   public String getCuadroColaEspera() {
      return cuadroColaEspera;
   }

}
```

Paciente

```
package com.mycompany.plhospital;

import java.time.LocalDateTime;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
public class Paciente extends Thread {
    String id_paciente;
    Hospital hospital;
    boolean vacunado;
    boolean registroCorrecto;


    public Paciente(String id_paciente, Hospital hospital) {
        this.id_paciente = id_paciente;
        this.hospital = hospital;
        this.vacunado=false;
        this.registroCorrecto=true;

    }

    public void run(){
        int aleatorio;
        hospital.hacerCola(this);
        while (!vacunado && registroCorrecto){
            try {
                sleep(10);
            } catch (InterruptedException ex) {}
        }
        if(vacunado){
            try {
                sleep(10000);
            } catch (InterruptedException ex) {}
            aleatorio= (int) (Math.random()*100+1);
            if(aleatorio>0 && aleatorio<=5){
                String text =("El paciente "+this.id_paciente+" está sufriendo una reaccion a la
vacuna.");
                hospital.escribirEnLog(text, LocalDateTime.now());
                hospital.generarAlerta(this.id_paciente); // aleatoriamente tendrá problemas con la
vacuna o no
            }
            else{
                String text=("El paciente "+this.id_paciente+" deja la sala de observacion. Se va a
casa.");
                hospital.escribirEnLog(text, LocalDateTime.now());
                hospital.dejarPuestoSalaObservacionPaciente(this.id_paciente);
            }

        }
    }

    public String getId_paciente() {
        return id_paciente;
```

```
    }

    public void setVacunado(boolean vacunado) {
        this.vacunado = vacunado;
    }

    public void setRegistroCorrecto(boolean registroCorrecto) {
        this.registroCorrecto = registroCorrecto;
    }




}
```

Sanitario

```
package com.mycompany.plhospital;

import java.time.LocalDateTime;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Sanitario extends Thread {
    private String id_sanitario;
    private Hospital hospital;
    private int contador=0;
    private boolean cerrarPuesto=false;

    public Sanitario(String id_sanitario, Hospital hospital) {
        this.id_sanitario = id_sanitario;
        this.hospital = hospital;
    }

    public void run(){
        //Cuando llegan al hospital van entre 1 y 3 segundos a cambiarse en la sala de
descanso
        hospital.entrarSalaDescanso(this.id_sanitario);
        int tiempoAleatorio;
        try{
            tiempoAleatorio= (int) (Math.random()*3+1)*1000;
            sleep(tiempoAleatorio);
        } catch (InterruptedException ex) {}
        hospital.salirSalaDescanso(this.id_sanitario);

        while(true){
```

```
        int puestoAlerta=hospital.hayAlertas();
        if(puestoAlerta!=-1){
            int tiempoObservacion = (int)(Math.random()*4+2)*1000;
            String paciente= hospital.atenderAlerta(puestoAlerta, this.id_sanitario);
            String text =("El sanitario "+ this.id_sanitario+" acude a atender una reaccion del
paciente "+paciente);
            hospital.escribirEnLog(text, LocalDateTime.now());
            try {
                sleep(tiempoObservacion);
            } catch (InterruptedException ex) {}
            text=("El paciente "+ paciente+ " ya se ha recuperado, se va a casa.");
            hospital.escribirEnLog(text, LocalDateTime.now());
            hospital.dejarPuestoSalaObservacionPaciente(paciente);
            hospital.dejarPuestoSalaObservacionSanitario(puestoAlerta);

        }
        int puestoAsignado= hospital.asignarPuestoVacunacionSanitario(this);
        String text= ("Al sanitario "+this.id_sanitario+" se le asigna el puesto:
"+puestoAsignado);
        hospital.escribirEnLog(text, LocalDateTime.now());
        while(contador<15 && !cerrarPuesto){
            hospital.cogerVacuna();
            Paciente pacienteVacunar= hospital.vacunar(puestoAsignado);
            if(cerrarPuesto){
                break;
            }
            tiempoAleatorio=(int)(Math.random()*3+3)*1000;
            try {
                sleep(tiempoAleatorio);
            } catch (InterruptedException ex) {}
            text= ("Paciente "+pacienteVacunar.getId_paciente()+" vacunado, se va a la sala
de observacion.");
            hospital.escribirEnLog(text, LocalDateTime.now());
            hospital.asignarPuestoSalaObservacion(pacienteVacunar.getId_paciente(),
puestoAsignado);
            pacienteVacunar.setVacunado(true);
            contador++;
        }
        hospital.dejarPuestoVacunacionSanitario(puestoAsignado);
        tiempoAleatorio= (int) (Math.random()*4+5)*1000;
        if(cerrarPuesto){
            text= ("El sanitario "+this.id_sanitario+" va a descansar "+tiempoAleatorio+"
segundos porque cierran su puesto para limpiarlo");
            hospital.escribirEnLog(text, LocalDateTime.now());
            cerrarPuesto=false;
        }
        else{
```

```
            text= ("El sanitario "+this.id_sanitario+" ha vacunado a 15 pacientes, va a
descansar "+tiempoAleatorio+" segundos");
            hospital.escribirEnLog(text, LocalDateTime.now());

        }

        hospital.entrarSalaDescanso(id_sanitario);
        try {
            sleep(tiempoAleatorio);
        }catch (InterruptedException ex) {}
        hospital.salirSalaDescanso(id_sanitario);
        contador=0;
      }
   }

   public int getContador() {
      return contador;
   }

   public String getId_sanitario() {
      return id_sanitario;
   }



   public void setCerrarPuesto(boolean cerrarPuesto) {
      this.cerrarPuesto = cerrarPuesto;
   }



}
```

Servidor

```
package com.mycompany.plhospital;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Servidor {


   public static void main(String[] args) {
```

```
        String[] puestosVacunacion= new String[10];
        String[] puestosObservacion = new String[20];
        ServerSocket servidor;
        Socket conexion;
        DataOutputStream salida;
        DataInputStream entrada;

        Hospital hospital = new Hospital();
        HiloCargarDatos inicializador = new HiloCargarDatos(hospital);
        inicializador.start();
        try{
           servidor = new ServerSocket(5000); // Creamos un ServerSocket en el puerto 5000
           System.out.println("Servidor Arrancado....");
           while (true)
           {
              conexion = servidor.accept();     // Esperamos una conexión
              salida  = new DataOutputStream(conexion.getOutputStream());
              entrada = new DataInputStream(conexion.getInputStream());
              salida.writeUTF(hospital.getAux1());
              salida.writeUTF(hospital.getAux2());
              salida.writeUTF(hospital.getVacunasDisponibles());
              salida.writeUTF(hospital.getColaEsperaText());
              puestosVacunacion=hospital.getPuestosVacunacion();
              for (int i=0;i<10;i++){
                 salida.writeUTF(puestosVacunacion[i]);
              }
              puestosObservacion=hospital.getPuestosObservacion();
              for (int i=0;i<20;i++){
                 salida.writeUTF(puestosObservacion[i]);
              }
              salida.writeUTF(hospital.getSalaDescansoText());
              salida.writeUTF(hospital.getPacienteRegistro());
              int puestoCerrado =entrada.readInt();
              if(puestoCerrado!=-1){

                 hospital.limpiarPuesto(puestoCerrado);
              }
              conexion.close();// Y cerramos la conexión
           }
        } catch (IOException e) {}

     }

}
```

VentanaHospital

```
package com.mycompany.plhospital;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import static java.lang.Thread.sleep;
import java.net.InetAddress;
import java.net.Socket;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.Timer;

public class VentanaHospital extends javax.swing.JFrame {

    /**
     * Creates new form VentanaHospital
     */
    JTextField puestosVacunacion[]=new JTextField[10];
    JTextField puestosObservacion[]=new JTextField[20];

    public VentanaHospital() {
        initComponents();
        HiloCliente cliente = new HiloCliente(this);
        cliente.start();


        puestosVacunacion[0]=puestoVacunacion_0;
        puestosVacunacion[1]=puestoVacunacion_1;
        puestosVacunacion[2]=puestoVacunacion_2;
        puestosVacunacion[3]=puestoVacunacion_3;
        puestosVacunacion[4]=puestoVacunacion_4;
        puestosVacunacion[5]=puestoVacunacion_5;
        puestosVacunacion[6]=puestoVacunacion_6;
        puestosVacunacion[7]=puestoVacunacion_7;
        puestosVacunacion[8]=puestoVacunacion_8;
        puestosVacunacion[9]=puestoVacunacion_9;

        puestosObservacion[0]=puestoObservacion_0;
        puestosObservacion[1]=puestoObservacion_1;
        puestosObservacion[2]=puestoObservacion_2;
        puestosObservacion[3]=puestoObservacion_3;
        puestosObservacion[4]=puestoObservacion_4;
        puestosObservacion[5]=puestoObservacion_5;
        puestosObservacion[6]=puestoObservacion_6;
        puestosObservacion[7]=puestoObservacion_7;
```

```
        puestosObservacion[8]=puestoObservacion_8;
        puestosObservacion[9]=puestoObservacion_9;
        puestosObservacion[10]=puestoObservacion_10;
        puestosObservacion[11]=puestoObservacion_11;
        puestosObservacion[12]=puestoObservacion_12;
        puestosObservacion[13]=puestoObservacion_13;
        puestosObservacion[14]=puestoObservacion_14;
        puestosObservacion[15]=puestoObservacion_15;
        puestosObservacion[16]=puestoObservacion_16;
        puestosObservacion[17]=puestoObservacion_17;
        puestosObservacion[18]=puestoObservacion_18;
        puestosObservacion[19]=puestoObservacion_19;



    }

    public void rellenarVentana(String aux1, String aux2, String vacunasDisponibles, String
colaEspera,
            String[] puestosVacunacion, String[] puestosObservacion, String salaDescanso,
String pacienteRegistro){
        this.aux1.setText(aux1);
        this.aux2.setText(aux2);
        this.vacunasDisponibles.setText(vacunasDisponibles);
        this.colaEspera.setText(colaEspera);
        this.salaDescanso.setText(salaDescanso);
        this.pacienteRegistro.setText(pacienteRegistro);
        for (int i=0; i<10;i++){
            this.puestosVacunacion[i].setText(puestosVacunacion[i]);
            this.puestosObservacion[i].setText(puestosObservacion[i]);
        }
        for (int i=10; i<20;i++){
            this.puestosObservacion[i].setText(puestosObservacion[i]);
        }
    }

    public void enviarPuestoCerrado(int puesto){
        Socket cliente;
        DataOutputStream salida;
        DataInputStream entrada;
        String[] puestosVacunacion = new String[10];
        String[] puestosObservacion = new String[20];

        try {
            cliente = new Socket(InetAddress.getLocalHost(),5000);
            salida = new DataOutputStream(cliente.getOutputStream());
```

```
            entrada = new DataInputStream(cliente.getInputStream());  //Creamos los canales de
entrada/salida
            String aux1 = entrada.readUTF();
            String aux2 = entrada.readUTF();
            String vacunasDisponibles = entrada.readUTF();
            String colaEspera= entrada.readUTF();
            for (int i=0;i<10;i++){
               puestosVacunacion[i]=entrada.readUTF();
            }
            for (int i=0;i<20;i++){
               puestosObservacion[i]=entrada.readUTF();
            }
            String salaDescanso=entrada.readUTF();
            String pacienteRegistro=entrada.readUTF();
         salida.writeInt(puesto);
         cliente.close();
         } catch (IOException ex) {}
      }

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

         jLabel1 = new javax.swing.JLabel();
         jLabel3 = new javax.swing.JLabel();
         jLabel4 = new javax.swing.JLabel();
         jLabel5 = new javax.swing.JLabel();
         colaEspera = new javax.swing.JTextField();
         aux1 = new javax.swing.JTextField();
         jLabel2 = 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();
         jLabel16 = new javax.swing.JLabel();
         puestoVacunacion_8 = new javax.swing.JTextField();
```

```
puestoVacunacion_0 = new javax.swing.JTextField();
puestoVacunacion_5 = new javax.swing.JTextField();
puestoVacunacion_7 = new javax.swing.JTextField();
puestoVacunacion_9 = new javax.swing.JTextField();
puestoVacunacion_6 = new javax.swing.JTextField();
puestoVacunacion_4 = new javax.swing.JTextField();
puestoVacunacion_2 = new javax.swing.JTextField();
puestoVacunacion_3 = new javax.swing.JTextField();
puestoVacunacion_1 = new javax.swing.JTextField();
jLabel17 = new javax.swing.JLabel();
salaDescanso = new javax.swing.JTextField();
jLabel18 = new javax.swing.JLabel();
jLabel19 = new javax.swing.JLabel();
jLabel20 = new javax.swing.JLabel();
puestoObservacion_3 = new javax.swing.JTextField();
puestoObservacion_6 = new javax.swing.JTextField();
puestoObservacion_0 = new javax.swing.JTextField();
puestoObservacion_1 = new javax.swing.JTextField();
puestoObservacion_4 = new javax.swing.JTextField();
puestoObservacion_2 = new javax.swing.JTextField();
puestoObservacion_5 = new javax.swing.JTextField();
puestoObservacion_7 = new javax.swing.JTextField();
puestoObservacion_8 = new javax.swing.JTextField();
puestoObservacion_11 = new javax.swing.JTextField();
puestoObservacion_9 = new javax.swing.JTextField();
puestoObservacion_10 = new javax.swing.JTextField();
jLabel21 = new javax.swing.JLabel();
jLabel22 = new javax.swing.JLabel();
puestoObservacion_14 = new javax.swing.JTextField();
puestoObservacion_13 = new javax.swing.JTextField();
puestoObservacion_12 = new javax.swing.JTextField();
jLabel23 = new javax.swing.JLabel();
puestoObservacion_15 = new javax.swing.JTextField();
puestoObservacion_16 = new javax.swing.JTextField();
puestoObservacion_17 = new javax.swing.JTextField();
aux2 = new javax.swing.JTextField();
jLabel25 = new javax.swing.JLabel();
jLabel24 = new javax.swing.JLabel();
vacunasDisponibles = new javax.swing.JTextField();
jLabel26 = new javax.swing.JLabel();
puestoObservacion_19 = new javax.swing.JTextField();
puestoObservacion_18 = new javax.swing.JTextField();
jTextField38 = new javax.swing.JTextField();
jTextField39 = new javax.swing.JTextField();
jTextField40 = new javax.swing.JTextField();
jTextField41 = new javax.swing.JTextField();
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();
        jTextField43 = new javax.swing.JTextField();
        ButtPuesto9 = new javax.swing.JButton();
        ButtPuesto0 = new javax.swing.JButton();
        ButtPuesto1 = new javax.swing.JButton();
        ButtPuesto2 = new javax.swing.JButton();
        ButtPuesto3 = new javax.swing.JButton();
        ButtPuesto4 = new javax.swing.JButton();
        ButtPuesto6 = new javax.swing.JButton();
        ButtPuesto8 = new javax.swing.JButton();
        ButtPuesto5 = new javax.swing.JButton();
        ButtPuesto7 = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        pacienteRegistro = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("HOSPITAL");
        setAlwaysOnTop(true);
        setBackground(new java.awt.Color(204, 204, 255));
        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 13)); // NOI18N
        jLabel1.setForeground(new java.awt.Color(102, 102, 255));
        jLabel1.setText("SALA DE ESPERA");
        getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(40,
10, -1, -1));

        jLabel3.setFont(new java.awt.Font("Tahoma", 1, 13)); // NOI18N
        jLabel3.setForeground(new java.awt.Color(102, 102, 255));
        jLabel3.setText("SALA DE DESCANSO PERSONAL SANITARIO");
        getContentPane().add(jLabel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(44,
472, -1, -1));

        jLabel4.setFont(new java.awt.Font("Tahoma", 1, 13)); // NOI18N
        jLabel4.setForeground(new java.awt.Color(102, 102, 255));
        jLabel4.setText("SALA DE OBSERVACIÓN");
```

```
        getContentPane().add(jLabel4, new org.netbeans.lib.awtextra.AbsoluteConstraints(454,
472, -1, -1));

        jLabel5.setFont(new java.awt.Font("Tahoma", 1, 13)); // NOI18N
        jLabel5.setForeground(new java.awt.Color(102, 102, 255));
        jLabel5.setText("SALA DE VACUNACIÓN");
        getContentPane().add(jLabel5, new org.netbeans.lib.awtextra.AbsoluteConstraints(40,
241, -1, -1));

        colaEspera.setEditable(false);
        colaEspera.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
        getContentPane().add(colaEspera, new
org.netbeans.lib.awtextra.AbsoluteConstraints(40, 62, 812, 30));

        aux1.setEditable(false);
        getContentPane().add(aux1, new org.netbeans.lib.awtextra.AbsoluteConstraints(110,
120, 90, -1));

        jLabel2.setText("Auxiliar");
        getContentPane().add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(42,
126, -1, -1));

        jLabel6.setText("Paciente");
        getContentPane().add(jLabel6, new org.netbeans.lib.awtextra.AbsoluteConstraints(40,
152, -1, -1));

        jLabel7.setText("Puesto 0");
        getContentPane().add(jLabel7, new org.netbeans.lib.awtextra.AbsoluteConstraints(180,
274, -1, -1));

        jLabel8.setText("Puesto 2");
        getContentPane().add(jLabel8, new org.netbeans.lib.awtextra.AbsoluteConstraints(368,
274, -1, -1));

        jLabel9.setText("Puesto 1");
        getContentPane().add(jLabel9, new org.netbeans.lib.awtextra.AbsoluteConstraints(180,
330, -1, -1));

        jLabel10.setText("Puesto 3");
        getContentPane().add(jLabel10, new
org.netbeans.lib.awtextra.AbsoluteConstraints(368, 333, -1, -1));

        jLabel11.setText("Puesto 4");
        getContentPane().add(jLabel11, new
org.netbeans.lib.awtextra.AbsoluteConstraints(525, 231, -1, -1));

        jLabel12.setText("Puesto 5");
```

```
        getContentPane().add(jLabel12, new
org.netbeans.lib.awtextra.AbsoluteConstraints(713, 231, -1, -1));

        jLabel13.setText("Puesto 6");
        getContentPane().add(jLabel13, new
org.netbeans.lib.awtextra.AbsoluteConstraints(525, 301, -1, -1));

        jLabel14.setText("Puesto 7");
        getContentPane().add(jLabel14, new
org.netbeans.lib.awtextra.AbsoluteConstraints(713, 301, -1, -1));

        jLabel15.setText("Puesto 8");
        getContentPane().add(jLabel15, new
org.netbeans.lib.awtextra.AbsoluteConstraints(525, 377, -1, -1));

        jLabel16.setText("Puesto 9");
        getContentPane().add(jLabel16, new
org.netbeans.lib.awtextra.AbsoluteConstraints(713, 377, -1, -1));

        puestoVacunacion_8.setEditable(false);
        getContentPane().add(puestoVacunacion_8, new
org.netbeans.lib.awtextra.AbsoluteConstraints(525, 404, 130, -1));

        puestoVacunacion_0.setEditable(false);
        getContentPane().add(puestoVacunacion_0, new
org.netbeans.lib.awtextra.AbsoluteConstraints(180, 300, 140, -1));

        puestoVacunacion_5.setEditable(false);
        getContentPane().add(puestoVacunacion_5, new
org.netbeans.lib.awtextra.AbsoluteConstraints(713, 258, 139, -1));

        puestoVacunacion_7.setEditable(false);
        getContentPane().add(puestoVacunacion_7, new
org.netbeans.lib.awtextra.AbsoluteConstraints(713, 328, 139, -1));

        puestoVacunacion_9.setEditable(false);
        getContentPane().add(puestoVacunacion_9, new
org.netbeans.lib.awtextra.AbsoluteConstraints(713, 404, 139, -1));

        puestoVacunacion_6.setEditable(false);
        getContentPane().add(puestoVacunacion_6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(525, 328, 130, -1));

        puestoVacunacion_4.setEditable(false);
        getContentPane().add(puestoVacunacion_4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(525, 258, 130, -1));
```

```
        puestoVacunacion_2.setEditable(false);
        getContentPane().add(puestoVacunacion_2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(368, 296, 130, -1));

        puestoVacunacion_3.setEditable(false);
        getContentPane().add(puestoVacunacion_3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(368, 360, 139, -1));

        puestoVacunacion_1.setEditable(false);
        getContentPane().add(puestoVacunacion_1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(180, 359, 139, -1));

        jLabel17.setText("Cola de espera:");
        getContentPane().add(jLabel17, new org.netbeans.lib.awtextra.AbsoluteConstraints(40,
40, -1, -1));

        salaDescanso.setEditable(false);
        getContentPane().add(salaDescanso, new
org.netbeans.lib.awtextra.AbsoluteConstraints(44, 506, 306, 261));

        jLabel18.setText("Puesto 0");
        getContentPane().add(jLabel18, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 501, -1, -1));

        jLabel19.setText("Puesto 3");
        getContentPane().add(jLabel19, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 554, -1, -1));

        jLabel20.setText("Puesto 6");
        getContentPane().add(jLabel20, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 606, -1, -1));

        puestoObservacion_3.setEditable(false);
        getContentPane().add(puestoObservacion_3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 574, 137, -1));

        puestoObservacion_6.setEditable(false);
        getContentPane().add(puestoObservacion_6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 628, 137, -1));

        puestoObservacion_0.setEditable(false);
        getContentPane().add(puestoObservacion_0, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 524, 137, -1));

        puestoObservacion_1.setEditable(false);
        getContentPane().add(puestoObservacion_1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(600, 524, 137, -1));
```

```
        puestoObservacion_4.setEditable(false);
        getContentPane().add(puestoObservacion_4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(600, 576, 137, -1));

        puestoObservacion_2.setEditable(false);
        getContentPane().add(puestoObservacion_2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(753, 524, 137, -1));

        puestoObservacion_5.setEditable(false);
        getContentPane().add(puestoObservacion_5, new
org.netbeans.lib.awtextra.AbsoluteConstraints(753, 576, 137, -1));

        puestoObservacion_7.setEditable(false);
        getContentPane().add(puestoObservacion_7, new
org.netbeans.lib.awtextra.AbsoluteConstraints(603, 628, 137, -1));

        puestoObservacion_8.setEditable(false);
        getContentPane().add(puestoObservacion_8, new
org.netbeans.lib.awtextra.AbsoluteConstraints(752, 628, 137, -1));

        puestoObservacion_11.setEditable(false);
        getContentPane().add(puestoObservacion_11, new
org.netbeans.lib.awtextra.AbsoluteConstraints(753, 678, 137, -1));

        puestoObservacion_9.setEditable(false);
        getContentPane().add(puestoObservacion_9, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 678, 137, -1));

        puestoObservacion_10.setEditable(false);
        getContentPane().add(puestoObservacion_10, new
org.netbeans.lib.awtextra.AbsoluteConstraints(600, 679, 137, -1));

        jLabel21.setText("Puesto 9");
        getContentPane().add(jLabel21, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 658, -1, -1));

        jLabel22.setText("Puesto 12");
        getContentPane().add(jLabel22, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 709, -1, -1));

        puestoObservacion_14.setEditable(false);
        getContentPane().add(puestoObservacion_14, new
org.netbeans.lib.awtextra.AbsoluteConstraints(752, 732, 137, -1));

        puestoObservacion_13.setEditable(false);
```

```
        getContentPane().add(puestoObservacion_13, new
org.netbeans.lib.awtextra.AbsoluteConstraints(603, 732, 137, -1));

        puestoObservacion_12.setEditable(false);
        getContentPane().add(puestoObservacion_12, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 732, 137, -1));

        jLabel23.setText("Puesto 18");
        getContentPane().add(jLabel23, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 812, -1, -1));

        puestoObservacion_15.setEditable(false);
        getContentPane().add(puestoObservacion_15, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 782, 137, -1));

        puestoObservacion_16.setEditable(false);
        getContentPane().add(puestoObservacion_16, new
org.netbeans.lib.awtextra.AbsoluteConstraints(603, 782, 137, -1));

        puestoObservacion_17.setEditable(false);
        getContentPane().add(puestoObservacion_17, new
org.netbeans.lib.awtextra.AbsoluteConstraints(752, 782, 137, -1));

        aux2.setEditable(false);
        getContentPane().add(aux2, new org.netbeans.lib.awtextra.AbsoluteConstraints(40,
283, 84, -1));

        jLabel25.setText("<html><body>Vacunas <br>disponibles: </body></html>\"");
        jLabel25.setVerticalAlignment(javax.swing.SwingConstants.TOP);
        jLabel25.setAutoscrolls(true);
        getContentPane().add(jLabel25, new org.netbeans.lib.awtextra.AbsoluteConstraints(40,
312, 84, -1));

        jLabel24.setText("Auxiliar");
        getContentPane().add(jLabel24, new org.netbeans.lib.awtextra.AbsoluteConstraints(40,
263, -1, -1));

        vacunasDisponibles.setEditable(false);
        getContentPane().add(vacunasDisponibles, new
org.netbeans.lib.awtextra.AbsoluteConstraints(40, 350, 84, -1));

        jLabel26.setText("Puesto 15");
        getContentPane().add(jLabel26, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 762, -1, -1));

        puestoObservacion_19.setEditable(false);
```

```
        getContentPane().add(puestoObservacion_19, new
org.netbeans.lib.awtextra.AbsoluteConstraints(603, 832, 137, -1));

        puestoObservacion_18.setEditable(false);
        getContentPane().add(puestoObservacion_18, new
org.netbeans.lib.awtextra.AbsoluteConstraints(450, 832, 137, -1));

        jTextField38.setBackground(new java.awt.Color(204, 204, 255));
        jTextField38.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1, 1));
        jTextField38.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
        getContentPane().add(jTextField38, new
org.netbeans.lib.awtextra.AbsoluteConstraints(419, 193, 470, 20));

        jTextField39.setBackground(new java.awt.Color(204, 204, 255));
        jTextField39.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1, 1));
        getContentPane().add(jTextField39, new
org.netbeans.lib.awtextra.AbsoluteConstraints(41, 443, 580, 20));

        jTextField40.setBackground(new java.awt.Color(204, 204, 255));
        jTextField40.setForeground(new java.awt.Color(255, 255, 204));
        jTextField40.setToolTipText("");
        jTextField40.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1, 1));
        getContentPane().add(jTextField40, new
org.netbeans.lib.awtextra.AbsoluteConstraints(670, 443, 223, 20));

        jTextField41.setBackground(new java.awt.Color(204, 204, 255));
        jTextField41.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1, 1));
        getContentPane().add(jTextField41, new
org.netbeans.lib.awtextra.AbsoluteConstraints(390, 453, 20, 410));

        jLabel27.setText("Puesto 2");
        getContentPane().add(jLabel27, new
org.netbeans.lib.awtextra.AbsoluteConstraints(753, 501, -1, -1));

        jLabel28.setText("Puesto 1");
        getContentPane().add(jLabel28, new
org.netbeans.lib.awtextra.AbsoluteConstraints(600, 501, -1, -1));

        jLabel29.setText("Puesto 5");
        getContentPane().add(jLabel29, new
org.netbeans.lib.awtextra.AbsoluteConstraints(753, 559, -1, -1));

        jLabel30.setText("Puesto 4");
        getContentPane().add(jLabel30, new
org.netbeans.lib.awtextra.AbsoluteConstraints(600, 559, -1, -1));

        jLabel31.setText("Puesto 8");
```

```
        getContentPane().add(jLabel31, new
org.netbeans.lib.awtextra.AbsoluteConstraints(753, 611, -1, -1));

        jLabel32.setText("Puesto 10");
        getContentPane().add(jLabel32, new
org.netbeans.lib.awtextra.AbsoluteConstraints(600, 658, -1, -1));

        jLabel33.setText("Puesto 7");
        getContentPane().add(jLabel33, new
org.netbeans.lib.awtextra.AbsoluteConstraints(603, 611, -1, -1));

        jLabel34.setText("Puesto 11");
        getContentPane().add(jLabel34, new
org.netbeans.lib.awtextra.AbsoluteConstraints(753, 658, -1, -1));

        jLabel35.setText("Puesto 14");
        getContentPane().add(jLabel35, new
org.netbeans.lib.awtextra.AbsoluteConstraints(753, 709, -1, -1));

        jLabel36.setText("Puesto 13");
        getContentPane().add(jLabel36, new
org.netbeans.lib.awtextra.AbsoluteConstraints(600, 708, -1, -1));

        jLabel37.setText("Puesto 16");
        getContentPane().add(jLabel37, new
org.netbeans.lib.awtextra.AbsoluteConstraints(600, 767, -1, -1));

        jLabel38.setText("Puesto 17");
        getContentPane().add(jLabel38, new
org.netbeans.lib.awtextra.AbsoluteConstraints(753, 767, -1, -1));

        jLabel39.setText("Puesto 19");
        getContentPane().add(jLabel39, new
org.netbeans.lib.awtextra.AbsoluteConstraints(603, 812, -1, -1));

        jTextField43.setBackground(new java.awt.Color(204, 204, 255));
        jTextField43.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1, 1));
        jTextField43.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
        getContentPane().add(jTextField43, new
org.netbeans.lib.awtextra.AbsoluteConstraints(40, 193, 320, 20));

        ButtPuesto9.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
        ButtPuesto9.setText("CERRAR");
        ButtPuesto9.setAlignmentY(0.0F);
        ButtPuesto9.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        ButtPuesto9.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            ButtPuesto9ActionPerformed(evt);
        }
    });
    getContentPane().add(ButtPuesto9, new
org.netbeans.lib.awtextra.AbsoluteConstraints(780, 370, 70, -1));

    ButtPuesto0.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
    ButtPuesto0.setText("CERRAR");
    ButtPuesto0.setAlignmentY(0.0F);
    ButtPuesto0.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
    ButtPuesto0.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            ButtPuesto0ActionPerformed(evt);
        }
    });
    getContentPane().add(ButtPuesto0, new
org.netbeans.lib.awtextra.AbsoluteConstraints(240, 270, 70, -1));

    ButtPuesto1.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
    ButtPuesto1.setText("CERRAR");
    ButtPuesto1.setAlignmentY(0.0F);
    ButtPuesto1.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
    ButtPuesto1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            ButtPuesto1ActionPerformed(evt);
        }
    });
    getContentPane().add(ButtPuesto1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(250, 330, 70, -1));

    ButtPuesto2.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
    ButtPuesto2.setText("CERRAR");
    ButtPuesto2.setAlignmentY(0.0F);
    ButtPuesto2.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
    ButtPuesto2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            ButtPuesto2ActionPerformed(evt);
        }
    });
    getContentPane().add(ButtPuesto2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(430, 270, 70, -1));

    ButtPuesto3.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
    ButtPuesto3.setText("CERRAR");
    ButtPuesto3.setAlignmentY(0.0F);
    ButtPuesto3.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
    ButtPuesto3.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ButtPuesto3ActionPerformed(evt);
            }
        });
        getContentPane().add(ButtPuesto3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(430, 330, 70, -1));

        ButtPuesto4.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
        ButtPuesto4.setText("CERRAR");
        ButtPuesto4.setAlignmentY(0.0F);
        ButtPuesto4.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        ButtPuesto4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ButtPuesto4ActionPerformed(evt);
            }
        });
        getContentPane().add(ButtPuesto4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(590, 230, 70, -1));

        ButtPuesto6.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
        ButtPuesto6.setText("CERRAR");
        ButtPuesto6.setAlignmentY(0.0F);
        ButtPuesto6.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        ButtPuesto6.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ButtPuesto6ActionPerformed(evt);
            }
        });
        getContentPane().add(ButtPuesto6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(590, 300, 70, -1));

        ButtPuesto8.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
        ButtPuesto8.setText("CERRAR");
        ButtPuesto8.setAlignmentY(0.0F);
        ButtPuesto8.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        ButtPuesto8.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ButtPuesto8ActionPerformed(evt);
            }
        });
        getContentPane().add(ButtPuesto8, new
org.netbeans.lib.awtextra.AbsoluteConstraints(590, 370, 70, -1));

        ButtPuesto5.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
        ButtPuesto5.setText("CERRAR");
        ButtPuesto5.setAlignmentY(0.0F);
        ButtPuesto5.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
```

```
        ButtPuesto5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ButtPuesto5ActionPerformed(evt);
            }
        });
        getContentPane().add(ButtPuesto5, new
org.netbeans.lib.awtextra.AbsoluteConstraints(780, 230, 70, -1));

        ButtPuesto7.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
        ButtPuesto7.setText("CERRAR");
        ButtPuesto7.setAlignmentY(0.0F);
        ButtPuesto7.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        ButtPuesto7.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ButtPuesto7ActionPerformed(evt);
            }
        });
        getContentPane().add(ButtPuesto7, new
org.netbeans.lib.awtextra.AbsoluteConstraints(780, 300, 70, -1));

        jPanel1.setBackground(new java.awt.Color(252, 224, 224));
        jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        pacienteRegistro.setEditable(false);
        jPanel1.add(pacienteRegistro, new org.netbeans.lib.awtextra.AbsoluteConstraints(115,
150, 90, -1));

        getContentPane().add(jPanel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(0,
0, 960, 890));

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

    private void ButtPuesto0ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        enviarPuestoCerrado(0);
    }

    private void ButtPuesto1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        enviarPuestoCerrado(1);
    }

    private void ButtPuesto2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        enviarPuestoCerrado(2);
    }
```

```
    private void ButtPuesto3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        enviarPuestoCerrado(3);
    }

    private void ButtPuesto4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        enviarPuestoCerrado(4);
    }

    private void ButtPuesto6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        enviarPuestoCerrado(6);
    }

    private void ButtPuesto8ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        enviarPuestoCerrado(8);
    }

    private void ButtPuesto5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        enviarPuestoCerrado(5);
    }

    private void ButtPuesto7ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        enviarPuestoCerrado(7);
    }

    private void ButtPuesto9ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        enviarPuestoCerrado(9);
    }

    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
```

```
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(VentanaHospital.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(VentanaHospital.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(VentanaHospital.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(VentanaHospital.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    }
    //</editor-fold>

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

        }
    });

}

// Variables declaration - do not modify
private javax.swing.JButton ButtPuesto0;
private javax.swing.JButton ButtPuesto1;
private javax.swing.JButton ButtPuesto2;
private javax.swing.JButton ButtPuesto3;
private javax.swing.JButton ButtPuesto4;
private javax.swing.JButton ButtPuesto5;
private javax.swing.JButton ButtPuesto6;
private javax.swing.JButton ButtPuesto7;
private javax.swing.JButton ButtPuesto8;
private javax.swing.JButton ButtPuesto9;
private javax.swing.JTextField aux1;
private javax.swing.JTextField aux2;
```

```
private javax.swing.JTextField colaEspera;
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 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 jPanel1;
private javax.swing.JTextField jTextField38;
private javax.swing.JTextField jTextField39;
private javax.swing.JTextField jTextField40;
private javax.swing.JTextField jTextField41;
private javax.swing.JTextField jTextField43;
private javax.swing.JTextField pacienteRegistro;
```

```
    private javax.swing.JTextField puestoObservacion_0;
    private javax.swing.JTextField puestoObservacion_1;
    private javax.swing.JTextField puestoObservacion_10;
    private javax.swing.JTextField puestoObservacion_11;
    private javax.swing.JTextField puestoObservacion_12;
    private javax.swing.JTextField puestoObservacion_13;
    private javax.swing.JTextField puestoObservacion_14;
    private javax.swing.JTextField puestoObservacion_15;
    private javax.swing.JTextField puestoObservacion_16;
    private javax.swing.JTextField puestoObservacion_17;
    private javax.swing.JTextField puestoObservacion_18;
    private javax.swing.JTextField puestoObservacion_19;
    private javax.swing.JTextField puestoObservacion_2;
    private javax.swing.JTextField puestoObservacion_3;
    private javax.swing.JTextField puestoObservacion_4;
    private javax.swing.JTextField puestoObservacion_5;
    private javax.swing.JTextField puestoObservacion_6;
    private javax.swing.JTextField puestoObservacion_7;
    private javax.swing.JTextField puestoObservacion_8;
    private javax.swing.JTextField puestoObservacion_9;
    private javax.swing.JTextField puestoVacunacion_0;
    private javax.swing.JTextField puestoVacunacion_1;
    private javax.swing.JTextField puestoVacunacion_2;
    private javax.swing.JTextField puestoVacunacion_3;
    private javax.swing.JTextField puestoVacunacion_4;
    private javax.swing.JTextField puestoVacunacion_5;
    private javax.swing.JTextField puestoVacunacion_6;
    private javax.swing.JTextField puestoVacunacion_7;
    private javax.swing.JTextField puestoVacunacion_8;
    private javax.swing.JTextField puestoVacunacion_9;
    private javax.swing.JTextField salaDescanso;
    private javax.swing.JTextField vacunasDisponibles;
    // End of variables declaration


}
```