**GRADO EN INGENIERÍA DE COMPUTADORES**

**CURSO 2024-2025 CONVOCATORIA ORDINARIA**

**03223193L – MARTÍNEZ MARTÍN, ALEJANDRO**

**09147371H – CABRERA ROZALÉN, RAÚL**

ÍNDICE

Contenido

[1. ANÁLISIS DE ALTO NIVEL 4](#_Toc197294194)

[1.1. DESCRIPCIÓN GENERAL DEL PROBLEMA 4](#_Toc197294195)

[1.2. ACTORES PRINCIPALES 4](#_Toc197294196)

[1.2.1. Humanos 4](#_Toc197294197)

[1.2.2. Zombis 4](#_Toc197294198)

[1.2.3. Zonas del refugio. 5](#_Toc197294199)

[1.2.4. Zonas de riesgo 5](#_Toc197294200)

[1.2.5. Túneles 5](#_Toc197294201)

[2. DISEÑO GENERAL DEL SISTEMA Y DISCUSIÓN DE LAS HERRAMIENTAS DE SINCRONIZACIÓN UTILIZADAS 6](#_Toc197294202)

[2.1. DISEÑO GENERAL DEL SISTEMA 6](#_Toc197294203)

[2.2. DISCUSIÓN DE LAS HERRAMIENTAS DE SINCRONIZACIÓN UTILIZADAS 10](#_Toc197294204)

[3. DESCRIPCIÓN DE LAS CLASES PRINCIPALES 13](#_Toc197294205)

[Parte1 13](#_Toc197294206)

[Parte2 17](#_Toc197294207)

[4. DIAGRAMA DE CLASES 18](#_Toc197294208)

[5. ANEXO - CÓDIGO FUENTE 20](#_Toc197294209)

[5.1. PARTE1 20](#_Toc197294210)

[5.1.1. Arranque.java 20](#_Toc197294211)

[5.1.2. Comedor.java 22](#_Toc197294212)

[5.1.3. Humano.java 25](#_Toc197294213)

[5.1.4. Logger.java 27](#_Toc197294214)

[5.1.5. Pausa.java 28](#_Toc197294215)

[5.1.6. Ranking.java 28](#_Toc197294216)

[5.1.7. Servidor.java 30](#_Toc197294217)

[5.1.8. Tunel.java 33](#_Toc197294218)

[5.1.9. Zombie 37](#_Toc197294219)

[5.1.10. ZonaComun.java 40](#_Toc197294220)

[5.1.11. ZonaDescanso.java 41](#_Toc197294221)

[5.1.12. ZonaRiesgo.java 43](#_Toc197294222)

[5.2. PARTE2 44](#_Toc197294223)

[5.2.1. Cliente.java 44](#_Toc197294224)

[5.3. Visuals.ApocalipsisZombi 49](#_Toc197294225)

[5.3.1. VentanaCli.java 49](#_Toc197294226)

[5.3.2. VentanaServ.java 63](#_Toc197294227)

# ANÁLISIS DE ALTO NIVEL

## DESCRIPCIÓN GENERAL DEL PROBLEMA

El enunciado nos pide generar un sistema concurrente que simule un escenario en el que la humanidad ha sufrido un apocalipsis zombi mediante hilos en Java. En este mundo apocalíptico los humanos viven en un refugio seguro donde los zombis nunca pueden entrar, pero necesitan comer, por lo que van al exterior a través de cuatro túneles que conectan el refugio con cuatro zonas de riesgo distintas. En estas zonas de riesgo, los humanos tratan de recolectar comida para posteriormente llevarla al refugio, pero pueden ser atacados por zombis, lo que conllevará a que regresen al refugio sin comida o a que mueran y se conviertan en zombis. Dentro del refugio hay una zona común donde los humanos se preparan para ir al exterior, un comedor donde comen y depositan la comida al volver del exterior, y una zona de descanso donde descansan al volver del exterior o para recuperarse del ataque de un zombi.

Se debe gestionar el sistema de manera concurrente en diferentes zonas, como en el acceso a los túneles y la manera en la que los zombis interactúan con los humanos. Para ello se utilizará programación concurrente en Java.

* Túneles: Los túneles solo permiten el paso de un humano a la vez entre el exterior y el refugio. Además, los humanos que quieran volver al refugio desde el exterior tendrán prioridad frente a los que quieran salir del mismo.
* Zombis: Los zombis solo pueden atacar a un humano a la vez, y un humano solo puede ser atacado por un zombi al mismo tiempo.

Finalmente, se debe permitir un acceso remoto desde un cliente al servidor, que contiene toda la lógica del programa. Este cliente recibirá información en tiempo real sobre los humanos y zombis en las diferentes zonas, así como el contador de muertes de los zombis. Además, se deben generar interfaces gráficas que muestren esta información de manera interactiva.

## ACTORES PRINCIPALES

### Humanos

Son los personajes que deben sobrevivir en el refugio y a los ataques de los zombis cuando salen en busca de comida para llevar al refugio. Los humanos interactúan entre sí en los túneles e interactúan con los zombis en las zonas de riesgo.

### Zombis

Son los personajes que tratan de acabar con la humanidad. Se mueven por las zonas de riesgo aleatoriamente en busca de humanos para atacarlos. Inicialmente solo hay uno, pero a medida que van muriendo humanos van creándose más zombis. Un humano muere cuando es atacado por un zombi y el ataque resulta exitoso, lo cual ocurre con una probabilidad de 1/3.

### Zonas del refugio.

Aunque las zonas no son actores como tal, son esenciales para las interacciones entre los humanos, al igual que las zonas de riesgo y los túneles.

El refugio se divide en 3 zonas:

* Zona común: zona en la que los humanos se preparan y seleccionan el túnel por el que accederán a la zona de riesgo.
* Zona de descanso: zona en la que los humanos descansan según vuelven del exterior y donde, si han sido atacados pero han sobrevivido, descansarán por segunda vez después de comer para curarse del ataque del zombi.
* Comedor: zona en la que los humanos depositan la comida al volver del exterior si no han sido atacados y donde comen tras descansar antes de volver a la zona común y prepararse para volver a salir.

### Zonas de riesgo

Las zonas de riesgo están fuera del refugio y es donde interactúan los humanos y los zombis. Los humanos acceden a ella en busca de comida, pero pueden ser atacados si hay zombis en esa zona.

Si son atacados y no mueren, regresan al refugio marcados, lo que significa que tiene que descansar dos veces, y sin comida.

En el caso en el que un zombi ataque de manera exitosa a un humano, este morirá y se convertirá en zombi.

### Túneles

Son los medios mediante los que los humanos acceden al refugio y salen de este. Solo un humano puede pasar por cada túnel a la vez, y se necesita un grupo de tres humanos listos para salir por ese túnel para que puedan salir. Para volver esto no es necesario, puede volver únicamente un humano. Un humano que quiere volver al refugio tiene prioridad sobre los humanos que quieren salir del refugio.

# DISEÑO GENERAL DEL SISTEMA Y DISCUSIÓN DE LAS HERRAMIENTAS DE SINCRONIZACIÓN UTILIZADAS

## DISEÑO GENERAL DEL SISTEMA

El diseño del sistema está basado en programación concurrente, en la que se utilizan hilos (threads) para modelar las entidades del sistema y sus interacciones.

Componentes:

1. Humanos (Clase Humano.java):

Los humanos son hilos que realizan diferentes tareas en las distintas zonas que existen. Cada humano sigue un ciclo de vida definido:

1. Preparación: los humanos inician en la zona común del refugio y se preparan para salir.
2. Salida del refugio: después de prepararse, el humano elige uno de los cuatro túneles para dirigirse a una zona de riesgo. En ese túnel espera a que sean tres para comenzar a acceder al túnel y salir del refugio.
3. Regreso al refugio: una vez hayan terminado su estancia en la zona de riesgo, ya sea porque han recolectado la comida sin problema o porque han sido atacados sin éxito por un zombi, vuelven al túnel marcados o sin marcar y con o sin comida, y regresan al refugio con prioridad frente a los humanos que quieren salir.
4. Recuperación en el refugio: una vez hayan regresado, lo primero que hacen es depositar la comida las dos unidades de comida si han conseguido volver con ella, después se dirigen a la zona de descanso y descansan, al terminar se dirigen al comedor y comen una pieza de comida, y a continuación, si han sido atacados y están marcados, descansan por segunda vez para recuperarse del ataque. Lo próximo que hacen es volver a la zona común y repetir el ciclo.

Cada uno de estos comportamientos son gestionados por hilos independientes, lo que permite que varios humanos interactúen simultáneamente con el sistema.

1. Zombis (Clase Zombie.java)

Los zombis también son hilos que van alternando entre las zonas de riesgo de manera aleatoria en busca de humanos para atacarlos.

Inicialmente hay un solo zombi que buscará humanos para matarlos y convertirlos también en zombis.

Comportamiento de los zombis:

1. Deambulación: Los zombis se mueven aleatoriamente entre las zonas de riesgo. Cuando acceden a una y no hay humanos, esperan entre dos y tres segundos y cambian de zona.
2. Ataque: encuentran a uno o varios humanos, eligen a uno y lo atacan. El ataque tiene una probabilidad de éxito de una entre tres. Si el ataque resulta exitoso, el humano se convierte en zombi, y si no, el humano vuelve al refugio.
3. Cambio de zona: Ya sea porque no han encontrado humanos en la zona de riesgo en la que está o porque ha atacado a un humano, el zombi espera entre dos y tres segundos y cambian aleatoriamente de zona.
4. Túneles (Clase Tunel.java)

Son conexiones entre el refugio y las zonas de riesgo. Los túneles se pueden dividir principalmente en 3 zonas:

* + Zona de espera: los humanos antes de acceder al túnel desde el refugio tienen que esperar para formar un grupo de tres humanos.
  + Zona de paso: Una vez que se hayan formado el grupo, de uno en uno y fijándose siempre en que no haya humanos queriendo volver, los humanos del grupo previamente formado entran al túnel y acceden a la zona de riesgo correspondiente.
  + Zona de espera para volver: Cuando los humanos terminan su estancia en la zona de riesgo, estos vuelven por el mismo túnel por el que salieron con prioridad frente a los humanos que quieren salir.

Por lo tanto, las características del túnel son:

1. Acceso exclusivo al túnel, donde solo un humano puede pasar por el túnel a la vez.
2. Prioridad para regresar de los humanos que se encuentran fuera del refugio con respecto a los que están dentro.

1. Zonas del refugio (Clases Comedor.java, ZonaDescanso.java y ZonaComun.java)

Dentro del refugio hay diferentes zonas donde los humanos realizan diferentes actividades. Estas zonas se gestionan de tal forma para que los humanos no interfieran entre sí.

1. Comedor: es una zona del refugio donde los humanos depositan la comida que traen del exterior, siempre que no sean atacados, y donde comen. Cuando un humano regresa con comida, deposita dos piezas en el comedor. Por otro lado, cuando un humano come, lo hace siempre en orden de llegada, y si no hay comida, los humanos esperarán igualmente en el orden en el que han llegado hasta que algún humano deposite comida en el comedor para poder comer. Este proceso de alimentación debe gestionarse de manera sincronizada, ya que varios humanos pueden intentar acceder al comedor al mismo tiempo, y el número de piezas de comida es limitado.

Funcionamiento del comedor:

1. Acceso ordenado a la comida.
2. Sincronización en el acceso al comedor para garantizar que los humanos no interfieran entre sí al acceder a la comida.
3. Zona de descanso: es una zona del refugio donde los humanos van a descansar antes de comer al volver del exterior, y para curarse antes de comenzar de nuevo su ciclo si han sido atacados por un zombi en las zonas de riesgo.
4. Zona común: En esta zona del refugio los humanos se preparan para ir al exterior. Aleatoriamente eligen un túnel por el que van a salir y esperan a que tres humanos, incluido él, quieran salir por ese túnel hacia la zona de riesgo correspondiente. Es una zona de preparación donde no se realiza nada más a parte de la selección del túnel y la formación de grupos para salir por el túnel.
5. Zonas de riesgo (Clase ZonaRiesgo.java)

Son áreas externas al refugio a las que se accede mediante los túneles donde los humanos intentan permanecer entre tres y cinco segundos sin ser atacados por un zombi para recolectar dos piezas de comida. En el caso de que concluya el tiempo y no hayan sido atacados, vuelve al refugio y depositan la comida recolectada en el comedor. Si son atacados por un zombi mientras intentan recolectar comida hay dos opciones:

1. El ataque del zombi falla: Si esto ocurre vuelven inmediatamente al refugio marcados y soltando la comida. Después de comer deberán volver a descansar para dejar de estar marcados.
2. El ataque del zombi resulta exitoso: El humano muere y se crea un zombi con el mismo identificador del humano, cambiando la H de humano por la Z de zombi.
3. Arranque (Clase Arranque.java)

La clase arranque es la encargada de iniciar la simulación. Dentro de esta clase se crea el zombi inicial y se crean los humanos, así como las zonas que componen el escenario apocalíptico. La clase crea constantemente humanos de manera ordenada cada 0,5-2 segundos, asignándoles un identificador a cada uno de manera ordenada. En esta clase también se interactúa con la clase pausa, que se encarga de pasar la actividad de los hilos (zombis y humanos).

Por último, la información que genera la simulación se muestra de manera gráfica en una ventana (VentanaServ).

1. Pausa (Clase Pausa.java)

Esta clase se encarga de pausar y reanudar la ejecución del sistema, y además tiene un método para que los hilos comprueben si se ha pausado o no la simulación para que puedan detenerse.

1. Ranking (Clase Ranking.java)

Esta clase se encarga de conocer todos los contadores de muertes de los zombis que haya en el sistema y, a partir de ellos, crear un ranking de los tres zombis con más muertes causadas a los humanos.

1. Servidor (Clase Servidor.java)

El servidor se encarga de crear un servidor donde se inicie una simulación del apocalipsis. Este servidor intercambiará información con el cliente, donde envía datos de los humanos y zombis que hay en cada zona del sistema, así como los datos de las muertes causadas por los tres mejores zombis. También se encarga de recibir las ordenes de los clientes para parar y reanudar la simulación.

1. Cliente (Clase Cliente.java)

El cliente recibe información sobre la cantidad de humanos y zombis que hay en cada zona. También recibe información sobre el ranking de zombis en cuanto a las muertes que han causado. Con esta información se crea una ventana donde se muestran gráficamente los resultados obtenidos del servidor (VetanaCli).

1. Ventanas (Clases VentanaServ.java y VentanaCli.java)

Son ventanas gráficas que muestran información sobre los humanos y zombis que hay en cada zona, así como las muertes causadas por los tres zombis más letales en el caso de la ventana del cliente (VentanaCli).

1. Logger (Clase Logger.java)

Esta clase se encarga de registrar los eventos que van sucediendo en el sistema, como los accesos de los humanos a diferentes zonas, los depósitos de comida, las acciones que realizan los humanos y las interacciones que tienen con los zombis. Todos estos sucesos se escriben en un documento de texto llamado Apocalipsis.txt. Esta clase es crucial para conocer el comportamiento del sistema.

## DISCUSIÓN DE LAS HERRAMIENTAS DE SINCRONIZACIÓN UTILIZADAS

**Clase Tunel.java**   
Herramientas empleadas:

* Semaphore(3, true) esperarAntes: Controla el acceso a la zona de espera para formar grupos de tres humanos. El parámetro true asegura que el semáforo sea FIFO.
* CyclicBarrier(3) esperar: Asegura que los humanos solo accedan al túnel cuando el grupo de tres esté formado.
* Semaphore pasar: Gestiona el acceso al túnel de uno en uno, asegurando que solo un humano pase a la vez.
* LinkedBlockingQueue<Humano> listaPasar, listaPasado y listaRegresar: Utilizada para gestionar las listas de espera y de acceso al túnel de los humanos, garantizando un acceso ordenado y limitado.
* ReentrantLock con condition: Verifica que no haya humanos intentando regresar antes de permitir que otros salgan, gestionando el acceso con prioridad.

**Clase Comedor.java**   
Herramientas empleadas:

* AtomicInteger comidaDisponible: permite un acceso atómico a la comida disponible, asegurando operaciones seguras y sin interferencias.
* Semaphore (1,true) comer: Controla que solo un humano pueda comerá ala vez, mientras los demás esperan en orden.
* ReentrantLock con condition comidaEsperar y noComida: gestiona la espera de los humanos cuando no hay comida disponible.
* LinkedBlockingQueue<Humano> listaComedor: organiza la lista de los humanos en el comedor, gestionando su acceso y salida de forma eficiente y ordenada.

**Clase Humano.java   
Herramientas empleadas:**

* **CyclicBarrier(2) esperarAtaque: Se utiliza para coordinar que el humano y el zombi se sincronicen durante el ataque. La barrera se activa solamente cuando el humano y el zombi están listos,. Esto asegura que el ataque no ocurra hasta que ambos estén preparados. Espera a que los dos estén listos para continuar, ya que es un CyclicBarrier de 2.**
* **AtomicBoolean esperandoAtaque: Se usa para indicar de manera atómica si el humano está esperando un ataque de un zombi. Si este está en true significa que puede ser atacado por un zombi, mientras que si está en false significa que ya está siendo atacado por un zombi, para que no ataquen dos zombis al mismo humano.**

**Clase Logger.java**Herramientas empleadas:

* Semaphore(1, true) sem: controla el acceso al archivo de log, para asegurar que solo un hilo escriba a la vez en el y no haya interferencias. Al ser true significa que es un semáforo FIFO y que se accede y se espera en orden de llegada.

**Clase Pausa.java**   
Herramientas empleadas:

* ReentrantLock lockDetener y condition condición: se usan para gestionar la espera y la reanudación de los hilos, permitiendo que se bloqueen y desbloqueen de manera sincronizada según indique la simulación.

**Clase ZonaComun.java**   
Herramientas empleadas:

* LinkedBlockingQueue<Humano> listaHumanos: se utiliza para obtener y modificar la lista de los humanos que están en la zona común del refugio de manera eficiente y ordenada.

**Clase ZonaDescanso.java**   
Herramientas empleadas:

* LinkedBlockingQueue<Humano> listaDescansando: se utiliza para obtener y modificar la lista de los humanos que están descansando en la zona de descanso del refugio de manera eficiente y ordenada.

**Clase ZonaRiesgo.java**   
Herramientas empleadas:

* LinkedBlockingQueue<Humano> listaZombies: se utiliza para obtener y modificar la lista de los zombis que están en la zona de riesgo indicada de manera eficiente y ordenada.

# DESCRIPCIÓN DE LAS CLASES PRINCIPALES

Para no repetir el 2.1, se describirán los métodos principales de las clases

## Parte1

**Clase Sevidor.java**iniciarServ()

Se crea un socket para el servidor y se llama al arranque para que empiece la ejecución del programa. Una vez el cliente acepta la conexión se llama al método conexionCliente(Socket cli)

conexionCliente(Socket cli)

Cuando este método se llama se crean los I/O stream para mandar y recibir los Objetos. Se manda toda la información de la ejecución y se recibe del cliente un Boolean por si se quisiera parar la ejecución.

**Clase Arranque.java**crearSimulacionSegundoPlano()

Lanza un hilo llamando a iniciarSimulacion() para que la clase Arranque no se quede bloqueada instanciando los miles de hilos.

iniciarSimulacion()

Aquí es donde se instancias las principales clases del proyecto, tanto los arrays de los túneles como los Humanos y Zombies

pausarEjecucion()

Este método es llamado para hacer uso de la clase Pausa y que pare la ejecución.

reanudarEjecucion()

Realiza una función similar a pausarEjecucion(), pero esta vez la clase Pausa reanuda la ejecución.

Los métodos getters() de las instancias que hemos creado de las clases principales.

**Clase Humano.java**run()

El bucle while en donde el humano realiza sus tareas y comprueba si tiene que parar llamando a la clase Pausa.

defensa()

Hace uso de un CyclicBarrier para esperarse a que el Zombie termine de atacarle.

comprobarPausaHumano()

Se utiliza en Clases ajenas para que el humano compruebe si tiene que pausarse.

**Clase Zombie.java**run()

Aquí es donde se encuentra el while en el que reside la lógica del Zombie. Comprueba si hay que pausarse, elige aleatoriamente una zona de Riesgo y escoge un humano para atacarlo.

atacar()

Si el humano ha tenido la mala suerte de morir, se crea un nuevo zombie con su mismo ID y se llama a matarHumano(). En el caso contrario se marca al humano y se le deja ir.

matarHumano()

Se le cambia la condición al humano de No Muerto a Muerto mediante una llamada a la función morir() del humano, esta función pone a true el booleano de muerto.

**Clase Tunel.java**irExterior(Humano hu)

Aquí se encuentra la lógica del funcionamiento de los túneles, donde se esperan los humanos y van pasando de uno en uno de forma ordenada.

venirDelExterior(Humano hu)

Este método se encarga de que los humanos vuelvan por el túnel y tengan prioridad sobre los que van a pasar a las zonas de riesgo.

**Clase Comedor.java**depositarComida(Humano hu)

Se comprueba si el humano ha sido marcado o no y en el caso de que no esté marcado depositará 2 unidades de comida.

comer(Humano hu)

El humano tomará una pieza de comida y si no queda más comida esperan de manera ordenada a que otro superviviente la traiga.

**Clase ZonaRiesgo.java**devolverHumanoAleatorio(ZonaRiesgo zonaActual)

Este método se encarga de que los zombies cojan aleatoriamente a un Humano de manera que solamente un Zombie pueda coger a un humano en concreto

**Clase Pausa.java**pararEjecucion()

Se cambia el valor de booleano “parar” a true y los hilos se van esperando en la cola de la Condition que tiene esta clase cuando comprueban el booleano.

continuarEjecucion()

Se cambia el valor del booleano a false y se realiza un signalAll para despertar a todos los hilos que están a la espera.

comprobarPausa()

Los hilos comprueban si hay que detener la ejecución del programa y si es así realizan un await() para esperar.

**Clase Ranking.java**hacerRanking()

Se crea una lista que incluye a todos los zombies de todas las zonas de riesgo y se ordena de manera que los tres primeros sean los que más muertes llevan.

**Clase Logger.java**escribir(String msg)

Haciendo uso del semáforo, los hilos escriben el archivo apocalipsis.txt todo lo que va sucediendo en el programa.

**Clase ZonaComun.java**entrarZonaComun(Humano hu)

Se añaden a la lista de la zona común y actualizan la ventana del servidor.

prepararse(Humano hu)

Aquí se preparan ejecutando un sleep de entre 1 o 2 segundos.

vidaFueraRefugio(Humano hu)

Aquí es donde el humano elige aleatoriamente el túnel por el que va a pasar y se elimina de la lista de la zona común.

**Clase ZonaDescanso.java**

descansarVuelta(Humano hu)

El humano descansa tras su vuelta descansará entre 2 y 4 segundos registrándolo en el log.

descansarMarcado(Humano hu)

Si el humano ha sido marcado ejecutará este método en el que volverá a la zona de descanso para dormir entre 3 y 5 segundos.

**Clase VentanaServ.java**

Métodos de actualización

A esta clase se le pasa como parámetro la clase Arranque, de la que se sacan todo el resto de clases con métodos get() para posteriormente obtener las listas pertinentes de donde sacar la información con otro get().

## Parte2

**Clase Cliente.java**

conectarServ()

En este método reside la lógica con la que se comunica el cliente con el servidor. Se crea el socket para conectarse en el puerto correspondiente y los I/O streams para recibir y mandar objetos. Se reciben los datos que muestra la pantalla del cliente y se manda un Boolean para indicar si se pausa o no la ejecución.

Métodos get() de los atributos recibidos del servidor.

# Diagrama, Dibujo de ingeniería El contenido generado por IA puede ser incorrecto.DIAGRAMA DE CLASES

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

# ANEXO - CÓDIGO FUENTE

## PARTE1

### Arranque.java

package Parte1;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import static java.lang.Thread.sleep;

import Visuals.ApocalipsisZombi.\*;

public class Arranque {

private int num = 0;

private ZonaDescanso zonaDescanso;

private VentanaServ vent = new VentanaServ(this);

private ZonaComun zonaComun;

private Tunel[] arrayTunel = new Tunel[4];

private ZonaRiesgo[] arrayZonaRiesgo = new ZonaRiesgo[4];

private Logger logger = new Logger();

private Pausa pausa = new Pausa(logger);

private String[] idHumanos = new String[6];

private String[] idZombie = new String[6];

private ExecutorService executor = Executors.newSingleThreadExecutor();

private Comedor comedor = new Comedor(logger, vent);

private Ranking ranking = new Ranking(arrayZonaRiesgo);

public Arranque(){

vent.setVisible(true);

}

private void iniciarSimulacion() {

for (int i = 0; i < 4; i++) {

arrayZonaRiesgo[i] = new ZonaRiesgo(i, logger, vent);

}

for (int i = 0; i < 4; i++) {

arrayTunel[i] = new Tunel(i, arrayZonaRiesgo[i], logger,vent);

}

zonaComun = new ZonaComun(arrayTunel, logger, vent);

zonaDescanso = new ZonaDescanso(logger, vent);

//creacion zombie

idZombie[0] = "Z";

for (int i = 1; i <= 5; i++) {

idZombie[i] = "0";

}

new Zombie(idZombie, arrayZonaRiesgo, pausa, logger).start();

//creacion humanos

int contadorHumano = 1;

for (int t = 0; t < 10; t++) {

for (int k = 0; k < 10; k++) {

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

String num = String.format("%05d", contadorHumano);

idHumanos[0] = "H";

idHumanos[1] = String.valueOf(num.charAt(0));

idHumanos[2] = String.valueOf(num.charAt(1));

idHumanos[3] = String.valueOf(num.charAt(2));

idHumanos[4] = String.valueOf(num.charAt(3));

idHumanos[5] = String.valueOf(num.charAt(4));

System.out.println(idHumanos[0] + idHumanos[1] + idHumanos[2] + idHumanos[3] + idHumanos[4]+idHumanos[5]);

new Humano(idHumanos.clone(), comedor, arrayTunel, zonaComun, zonaDescanso, pausa).start();

contadorHumano++;

try {

sleep(500 + (int) (1500 \* Math.random()));

} catch (InterruptedException ie) {

System.out.println("Error creando humano");

}

}

}

}

}

}

public void crearSimulacionSegundoPlano() {

executor.submit(new Runnable() {

@Override

public void run() {

// Llamar a la creación de zombies y humanos en segundo plano

iniciarSimulacion();

}

});

}

public void pausarEjecucion(){

pausa.pararEjecucion();

}

public void reanudarEjecucion(){

pausa.continuarEjecucion();

}

public ZonaDescanso getZonaDescanso() {

return zonaDescanso;

}

public ZonaComun getZonaComun() {

return zonaComun;

}

public Tunel[] getArrayTunel() {

return arrayTunel;

}

public ZonaRiesgo[] getArrayZonaRiesgo() {

return arrayZonaRiesgo;

}

public Comedor getComedor() {

return comedor;

}

public Ranking getRanking() {

return ranking;

}

}

### Comedor.java

package Parte1;

import java.util.concurrent.Semaphore;

import java.util.concurrent.atomic.AtomicInteger;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.ReentrantLock;

import static java.lang.Thread.sleep;

import java.util.concurrent.LinkedBlockingQueue;

import javax.swing.SwingUtilities;

import Visuals.ApocalipsisZombi.\*;

public class Comedor {

private AtomicInteger comidaDisponible = new AtomicInteger(0);

private Semaphore comer = new Semaphore(1, true);

private ReentrantLock comidaEsperar = new ReentrantLock();

private Condition noComida = comidaEsperar.newCondition();

private VentanaServ ventana;

private LinkedBlockingQueue<Humano> listaComedor = new LinkedBlockingQueue<>();

private Logger logger;

public Comedor(Logger log, VentanaServ vent) {

ventana = vent;

logger = log;

}

public void depositarComida(Humano hu) {

if (!hu.getMarcado() && !hu.isMuerto()) {

hu.comprobarPausaHumano();

comidaDisponible.addAndGet(2); // Si todo ha ido bien depositan comida

logger.escribir("El humano " + hu.getIdHumanoStr() + " ha traido 2 piezas de comida. Comida restante: " + comidaDisponible.toString());

hu.comprobarPausaHumano();

comidaEsperar.lock(); //Toman el lock para realizar el signall

try {

noComida.signalAll(); // Al hacer el signal despiertan a los que estuvieran esperando por comida

}

finally {

comidaEsperar.unlock();

}

}

}

public void comer(Humano hu) {

if (!hu.isMuerto()) {

try {

hu.comprobarPausaHumano(); // Comprueba la pausa

listaComedor.add(hu); // Se aÃ±aden a la lista del comedor

SwingUtilities.invokeLater(new Runnable() { // Se actualiza la ventana

public void run() {

ventana.actualizarHumanosComedor();

}

});

comer.acquire(); // Toma el semÃ¡foro para la espera ordenada

comidaEsperar.lock(); //Se toma el lock para comprobar la comida

hu.comprobarPausaHumano(); // Comprueba la pausa

try {

while (comidaDisponible.get() == 0) { // Comprueban si no hay comida para quedarse esperando

logger.escribir("El humano " + hu.getIdHumanoStr() + " esta esperando para comer pero no hay comida.");

noComida.await(); // Se queda esperando a que traigan comida

}

}finally {

comidaEsperar.unlock();

}

hu.comprobarPausaHumano();

comer.release(); // Dejan el semÃ¡foro para que pase el siguiente

comidaDisponible.decrementAndGet(); // Se decrementa en una unidad ya que comen

hu.comprobarPausaHumano();

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarComida();

}

});

logger.escribir("El humano " + hu.getIdHumanoStr() + " esta comiendo. Comida restante: " + comidaDisponible.toString());

sleep(3000 + (int) (2000 \* Math.random())); // Tiempo que tardan en comer

logger.escribir("El humano " + hu.getIdHumanoStr() + " ha terminado de comer.");

listaComedor.remove(hu); // Se eliminan de la lista

hu.comprobarPausaHumano();

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarHumanosComedor();

}

});

} catch (InterruptedException e) {

return; //ha muerto

}

} else {

return;

}

}

public LinkedBlockingQueue<Humano> getListaHumanosComedor() {

return listaComedor;

}

public AtomicInteger getComida() {

return comidaDisponible;

}

}

### Humano.java

package Parte1;

import java.util.concurrent.BrokenBarrierException;

import java.util.concurrent.CyclicBarrier;

import java.util.concurrent.atomic.AtomicBoolean;

public class Humano extends Thread {

private String[] id = new String[5];

private boolean marcado = false;

private Comedor comedor;

private Tunel[] arrayTunel;

private ZonaDescanso zonaDescanso;

private ZonaComun zonaComun;

private boolean muerto = false;

private CyclicBarrier esperarAtaque = new CyclicBarrier(2);

private AtomicBoolean esperandoAtaque = new AtomicBoolean(false);

private Pausa pausa;

public Humano(String[] identificador, Comedor come, Tunel[] at, ZonaComun zc, ZonaDescanso zd, Pausa pa) {

id = identificador;

comedor = come;

arrayTunel = at;

zonaComun = zc;

zonaDescanso = zd;

pausa = pa;

}

public boolean getEsperandoAtaque() {

return esperandoAtaque.get();

}

public void setEsperandoAtaque(boolean esperandoAtaque) {

this.esperandoAtaque.set(esperandoAtaque);

}

public String[] getIdHumano() {

return id;

}

public String getIdHumanoStr() {

String str = "";

for (int i = 0; i < 6; i++) {

str += id[i];

}

return str;

}

public Tunel[] getArrayTunel() {

return arrayTunel;

}

public void setIdHumano(String[] id) {

this.id = id;

}

public boolean getMarcado() {

return marcado;

}

public void setMarcado(boolean bool) {

marcado = bool;

}

public boolean isMuerto(){

return muerto;

}

public void morir(){

muerto = true;

}

public void defensa(){

try {

esperarAtaque.await();

} catch (InterruptedException | BrokenBarrierException e) {

throw new RuntimeException(e);

}

}

public void comprobarPausaHumano(){

pausa.comprobarPausa();

}

public void run() {

while (!muerto) {

pausa.comprobarPausa();

zonaComun.entrarZonaComun(this);

pausa.comprobarPausa();

zonaComun.prepararse(this);

pausa.comprobarPausa();

zonaComun.vidaFueraRefugio(this);

pausa.comprobarPausa();

comedor.depositarComida(this);

pausa.comprobarPausa();

zonaDescanso.descansarVuelta(this);

pausa.comprobarPausa();

comedor.comer(this);

pausa.comprobarPausa();

if (marcado) {

zonaDescanso.descansarMarcado(this);

pausa.comprobarPausa();

}

}

System.out.println("Terminó el humano" + getIdHumanoStr());

}

}

### Logger.java

package Parte1;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.concurrent.Semaphore;

/\*\*

\*

\* @author Alex y Raul

\*/

public class Logger {

private String FICHERO = "apocalipsis.txt";

private Semaphore sem = new Semaphore(1, true);

public void escribir(String msg){

try{

sem.acquire();

try{

BufferedWriter bw = new BufferedWriter(new FileWriter(FICHERO, true));

String tiempo = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(new Date());

bw.write(tiempo + " - " + msg);

bw.newLine();

bw.close();

}catch(IOException e){

System.out.println("Error escribiendo en el log.");

}

}catch(InterruptedException ie){

System.out.println("Interrumpido mientras escribia en el log.");

}finally{

sem.release();

}

}

}

### Pausa.java

package Parte1;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.ReentrantLock;

public class Pausa {

private ReentrantLock lockDetener = new ReentrantLock();

private Condition condicion = lockDetener.newCondition();

private boolean parar = false;

private Logger logger;

public Pausa(Logger log){

logger = log;

}

public void pararEjecucion() {

logger.escribir("SE PARA LA EJECUCIÓN DEL PROGRAMA");

parar = true;

}

public void continuarEjecucion(){

logger.escribir("SE REANUDA LA EJECUCIÓN DEL PROGRAMA");

parar = false;

lockDetener.lock();

condicion.signalAll();

lockDetener.unlock();

}

public void comprobarPausa(){

try {

lockDetener.lock();

if (parar) {

condicion.await();

}

}

catch (InterruptedException e) {

throw new RuntimeException(e);

}

finally {

lockDetener.unlock();

}

}

}

### Ranking.java

package Parte1;

import java.util.ArrayList;

import java.util.Comparator;

public class Ranking {

private ZonaRiesgo[] arrayZonaRiesgo;

public Ranking(ZonaRiesgo[] zr){

arrayZonaRiesgo = zr;

}

Comparator<Zombie> porMuertes = (z1, z2) -> z2.getContadorMuertes() - z1.getContadorMuertes();

public ArrayList<String> hacerRanking(){

ArrayList<Zombie> listaCompleta = new ArrayList<>(arrayZonaRiesgo[0].getListaZombies());

listaCompleta.addAll(arrayZonaRiesgo[1].getListaZombies());

listaCompleta.addAll(arrayZonaRiesgo[2].getListaZombies());

listaCompleta.addAll(arrayZonaRiesgo[3].getListaZombies());

listaCompleta.sort(porMuertes);

ArrayList<String> listaMejores = new ArrayList<>(3);

if(listaCompleta.size()==0){

listaMejores.add("No hay muertos");

listaMejores.add("");

listaMejores.add("");

return listaMejores;

}

else if (listaCompleta.size()==1){

listaMejores.add(0, listaCompleta.get(0).getIdZombie() + " - " + listaCompleta.get(0).getContadorMuertes() + " muertes");

listaMejores.add("");

listaMejores.add("");

return listaMejores;

}else if(listaCompleta.size()==2){

listaMejores.add(0, listaCompleta.get(0).getIdZombie() + " - " + listaCompleta.get(0).getContadorMuertes() + " muertes");

listaMejores.add(1, listaCompleta.get(1).getIdZombie() + " - " + listaCompleta.get(1).getContadorMuertes() + " muertes");

listaMejores.add("");

return listaMejores;

}else{

listaMejores.add(0, listaCompleta.get(0).getIdZombie() + " - " + listaCompleta.get(0).getContadorMuertes() + " muertes");

listaMejores.add(1, listaCompleta.get(1).getIdZombie() + " - " + listaCompleta.get(1).getContadorMuertes() + " muertes");

listaMejores.add(2, listaCompleta.get(2).getIdZombie() + " - " + listaCompleta.get(2).getContadorMuertes() + " muertes");

return listaMejores;

}

}

}

### Servidor.java

package Parte1;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

public class Servidor {

private ServerSocket servidor;

private Arranque arranque = new Arranque();

private Boolean parar = false;

public Servidor() {

}

public Arranque getArranque() {

return arranque;

}

public void iniciarServ() {

try {

servidor = new ServerSocket(5002);

System.out.println("Servidor Arrancando....");

arranque.crearSimulacionSegundoPlano();

while (true) {

Socket cliente = servidor.accept();

conexionCliente(cliente);

}

} catch (IOException e) {

System.out.println("Error al iniciar server");

}

}

public void conexionCliente(Socket cli) {

try {

ObjectOutputStream oos = new ObjectOutputStream(cli.getOutputStream());

ObjectInputStream ois = new ObjectInputStream(cli.getInputStream());

while (true) {

parar = (Boolean) ois.readObject();

if (parar) {

arranque.pausarEjecucion();

} else {

arranque.reanudarEjecucion();

}

//listas de la Zona Segura

int listaComedor = arranque.getComedor().getListaHumanosComedor().size();

oos.writeObject(listaComedor);

int listaZonaComun = arranque.getZonaComun().getListaHumanosZonaComun().size();

oos.writeObject(listaZonaComun);

int listaZonaDescanso = arranque.getZonaDescanso().getListaHumanosDescansando().size();

oos.writeObject(listaZonaDescanso);

//Listas de zonas de Riesgo

int listaZonaRiesgo1 = arranque.getArrayZonaRiesgo()[0].getListaHumanos().size();

oos.writeObject(listaZonaRiesgo1);

int listaZonaRiesgo2 = arranque.getArrayZonaRiesgo()[1].getListaHumanos().size();

oos.writeObject(listaZonaRiesgo2);

int listaZonaRiesgo3 = arranque.getArrayZonaRiesgo()[2].getListaHumanos().size();

oos.writeObject(listaZonaRiesgo3);

int listaZonaRiesgo4 = arranque.getArrayZonaRiesgo()[3].getListaHumanos().size();

oos.writeObject(listaZonaRiesgo4);

//Lista de humanos para pasar en los Túneles

int listaPasarTunel1 = arranque.getArrayTunel()[0].getListaPasar().size();

oos.writeObject(listaPasarTunel1);

int listaPasarTunel2 = arranque.getArrayTunel()[1].getListaPasar().size();

oos.writeObject(listaPasarTunel2);

int listaPasarTunel3 = arranque.getArrayTunel()[2].getListaPasar().size();

oos.writeObject(listaPasarTunel3);

int listaPasarTunel4 = arranque.getArrayTunel()[3].getListaPasar().size();

oos.writeObject(listaPasarTunel4);

//Listas de humanos pasando en cada momento por los túneles

int listaPasandoTunel1 = arranque.getArrayTunel()[0].getListaPasando().size();

oos.writeObject(listaPasandoTunel1);

int listaPasandoTunel2 = arranque.getArrayTunel()[1].getListaPasando().size();

oos.writeObject(listaPasandoTunel2);

int listaPasandoTunel3 = arranque.getArrayTunel()[2].getListaPasando().size();

oos.writeObject(listaPasandoTunel3);

int listaPasandoTunel4 = arranque.getArrayTunel()[3].getListaPasando().size();

oos.writeObject(listaPasandoTunel4);

//Listas de humanos regresando en cada tunel

int listaRegresandoTunel1 = arranque.getArrayTunel()[0].getListaRegresar().size();

oos.writeObject(listaRegresandoTunel1);

int listaRegresandoTunel2 = arranque.getArrayTunel()[1].getListaRegresar().size();

oos.writeObject(listaRegresandoTunel2);

int listaRegresandoTunel3 = arranque.getArrayTunel()[2].getListaRegresar().size();

oos.writeObject(listaRegresandoTunel3);

int listaRegresandoTunel4 = arranque.getArrayTunel()[3].getListaRegresar().size();

oos.writeObject(listaRegresandoTunel4);

//Listas de Zombies en cada zona de Riesgo

int listaZombieZonaRiesgo1 = arranque.getArrayZonaRiesgo()[0].getListaZombies().size();

oos.writeObject(listaZombieZonaRiesgo1);

int listaZombieZonaRiesgo2 = arranque.getArrayZonaRiesgo()[1].getListaZombies().size();

oos.writeObject(listaZombieZonaRiesgo2);

int listaZombieZonaRiesgo3 = arranque.getArrayZonaRiesgo()[2].getListaZombies().size();

oos.writeObject(listaZombieZonaRiesgo3);

int listaZombieZonaRiesgo4 = arranque.getArrayZonaRiesgo()[3].getListaZombies().size();

oos.writeObject(listaZombieZonaRiesgo4);

ArrayList<String> listaMejoresZombies = arranque.getRanking().hacerRanking();

oos.writeObject(listaMejoresZombies);

oos.flush();

oos.reset();

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error en la conexion con el cliente");

throw new RuntimeException(e);

}

}

public static void main(String[] args) {

Servidor servidor = new Servidor();

servidor.iniciarServ();

}

}

### Tunel.java

package Parte1;

import Visuals.ApocalipsisZombi.VentanaServ;

import java.util.concurrent.BrokenBarrierException;

import java.util.concurrent.CyclicBarrier;

import java.util.concurrent.LinkedBlockingQueue;

import java.util.concurrent.Semaphore;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.ReentrantLock;

import static java.lang.Thread.sleep;

import javax.swing.SwingUtilities;

public class Tunel {

private int id;

private Semaphore esperarAntes = new Semaphore(3, true);

private CyclicBarrier esperar = new CyclicBarrier(3);

private Semaphore pasar = new Semaphore(1, true);

private LinkedBlockingQueue<Humano> listaPasar = new LinkedBlockingQueue<>();

private LinkedBlockingQueue<Humano> listaPasando = new LinkedBlockingQueue<>();

private LinkedBlockingQueue<Humano> listaRegresar = new LinkedBlockingQueue<>();

private ZonaRiesgo zonaRiesgo;

private ReentrantLock candado = new ReentrantLock();

private Condition condicion = candado.newCondition();

private Logger logger;

private VentanaServ ventana;

public Tunel(int pid, ZonaRiesgo zr, Logger log, VentanaServ vServ) {

id = pid;

zonaRiesgo = zr;

logger = log;

ventana = vServ;

}

public LinkedBlockingQueue<Humano> getListaPasar() {

return listaPasar;

}

public LinkedBlockingQueue<Humano> getListaPasando() {

return listaPasando;

}

public LinkedBlockingQueue<Humano> getListaRegresar() {

return listaRegresar;

}

public int getId() {

return id;

}

public void irExterior(Humano hu) {

try {

hu.comprobarPausaHumano(); //Comprueba la pausa de la ejecución

esperarAntes.acquire(); //Mira si hay hueco para pasar a la zona de espera

listaPasar.add(hu); // Se añade en la zona de espera

SwingUtilities.invokeLater(new Runnable() { //Se actualiza la ventana del servidor

public void run() {

ventana.actualizarListaPasarTunel(getId());

}

});

// En función de la gente esperando se guarda en el log un mensaje u otro

if (esperar.getNumberWaiting() + 1 == 3) {

logger.escribir("Humano " + hu.getIdHumanoStr() + " ha llegado al tunel " + id + " y ya son 3. Listos para salir!");

} else {

logger.escribir("Humano " + hu.getIdHumanoStr() + " esta esperando para salir por el tunel " + id + ". Humanos esperando: " + (1 + esperar.getNumberWaiting()));

}

hu.comprobarPausaHumano(); // Comprueba la pausa

esperar.await(); // Cyclic Barrier para esperar a que sean 3

hu.comprobarPausaHumano(); // Comprueba la pausa

candado.lock();

try {

hu.comprobarPausaHumano(); // Comprueba la pausa

// Mientras que la lista de gente que quiere regresar al refugio no esté vacía, entran al while para esperar

while (!listaRegresar.isEmpty()) {

logger.escribir("Humano " + hu.getIdHumanoStr() + " esta esperando a que entre otro humano al refugio por el tunel " + id);

condicion.await();

}

} finally {

candado.unlock();

}

hu.comprobarPausaHumano(); // Comprueba la pausa

pasar.acquire(); // Toma el semáforo para que pasen de 1 en 1

logger.escribir("Humano " + hu.getIdHumanoStr() + " está pasando por el tunel " + id);

esperarAntes.release(); // Se libera un token del semáforo de la zona de espera del tunel

listaPasar.remove(hu); // Se elimina de la lista que se muestra en la interfaz

SwingUtilities.invokeLater(new Runnable() { // Se actualiza la interfaz

public void run() {

ventana.actualizarListaPasarTunel(getId());

}

});

listaPasando.add(hu); //Se añade en la lista de personas pasando

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarListaPasandoTunel(getId());

}

});

sleep(1000); // 1000 milisegundos que es el tiempo que tarda en pasar

listaPasando.remove(hu); //Se elimina de la lista de personas pasando

SwingUtilities.invokeLater(new Runnable() { // Se actualiza la ventana

public void run() {

ventana.actualizarListaPasandoTunel(getId());

}

});

logger.escribir("Humano " + hu.getIdHumanoStr() + " ha terminado de pasar por el tunel " + id);

pasar.release(); // Se libera el semáforo para que pase el siguiente

hu.comprobarPausaHumano(); //Comprueba la pausa

hu.comprobarPausaHumano();

logger.escribir("Humano " + hu.getIdHumanoStr() + " ha entrado a la zona de riesgo " + id);

zonaRiesgo.entrarHumano(hu);

SwingUtilities.invokeLater(new Runnable() { // Se actualiza la ventana

public void run() {

ventana.actualizarHumanosZP(getId());

}

});

sleep((long) (Math.random() \* 3000 + 2000));

hu.comprobarPausaHumano();

} catch (InterruptedException | BrokenBarrierException e) {

if (hu.getEsperandoAtaque()) {

System.out.println(hu.getIdHumanoStr() + " Está siendo atacado y puede morir.");

hu.defensa();

}

}

}

public void venirDelExterior(Humano hu) {

try {

hu.comprobarPausaHumano(); //Se comprueba si necesita parar la ejecución

zonaRiesgo.salirHumano(hu); //Sale de la Zona de Riesgo y se elimina de la lista

logger.escribir("Humano " + hu.getIdHumanoStr() + " ha terminado en la zona de riesgo " + id + " y vuelve al refugio");

//Si sale marcado se registra en el log

if (hu.getMarcado()) {

logger.escribir("Vuelve el humano " + hu.getIdHumanoStr() + " marcado y sin comida.");

} else {

logger.escribir("Vuelve el humano " + hu.getIdHumanoStr() + " con comida.");

}

listaRegresar.add(hu); // Se añade en la lista de personas que quieren regresar

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarListaRegresarTunel(getId());

}

});

logger.escribir("Humano " + hu.getIdHumanoStr() + " esta esperando a entrar por el tunel " + id);

pasar.acquire(); //Toma el semáforo para pasar por la zona estrecha del túnel

listaRegresar.remove(hu); //Sale de lista de personas esperando para regresar

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarListaRegresarTunel(getId());

}

});

listaPasando.add(hu); //Se añade a la lista de gente pasando por la zona estrecha

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarListaPasandoTunel(getId());

}

});

logger.escribir("Humano " + hu.getIdHumanoStr() + " volviendo por el tunel " + id);

sleep(1000); // Tiempo que tarda en pasar

listaPasando.remove(hu); // Se elimina de la lista de personas pasando una vez ha pasado el tiempo

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarListaPasandoTunel(getId());

}

});

logger.escribir("Humano " + hu.getIdHumanoStr() + " pasando por el tunel " + id);

pasar.release(); // Libera el semáforo para que pase el siguiente

// Despierta a los hilos que estuvieran esperando para pasar a las zonas de riesgo

candado.lock();

try {

condicion.signalAll();

} finally {

candado.unlock();

}

hu.comprobarPausaHumano(); // Comprueba la pausa de la ejecución

} catch (InterruptedException e) {

System.out.println("muerto");

}

}

}

### Zombie

package Parte1;

public class Zombie extends Thread {

private int contadorMuertes = 0;

private String[] id = new String[6];

private ZonaRiesgo[] arrayZonaRiesgo;

private Pausa pausa;

private Logger logger;

public Zombie(String[] identificador, ZonaRiesgo[] zr, Pausa pa, Logger log) {

id = identificador;

arrayZonaRiesgo = zr;

pausa = pa;

logger = log;

}

public String getIdZombie() {

String str = "";

for (int i = 0; i < 6; i++) {

str += id[i];

}

return str;

}

public int getContadorMuertes() {

return contadorMuertes;

}

public void run() {

try {

while (true) {

pausa.comprobarPausa(); // Comprueba la pausa

int rand1 = (int) (4 \* Math.random()); // Elige aleatoriamente la zona de riesgo

logger.escribir("El zombie " + getIdZombie() + " entra a la zona de riesgo " + rand1);

ZonaRiesgo zonaActual = arrayZonaRiesgo[rand1];

pausa.comprobarPausa();

zonaActual.entrarZombie(this); // Se añade a la lista de la zona de riesgo

pausa.comprobarPausa();

// Coge a un humano aleatoriamente

Humano huAtacado = zonaActual.devolverHumanoAleatorio(zonaActual);

pausa.comprobarPausa();

if(!(huAtacado == null)){

pausa.comprobarPausa();

huAtacado.setEsperandoAtaque(true);

huAtacado.interrupt();

pausa.comprobarPausa();

atacar(huAtacado, zonaActual);

pausa.comprobarPausa();

}

sleep(2000 + (int) (1000 \* Math.random())); // Se espera en la zona de riesgo

pausa.comprobarPausa();

logger.escribir("El zombie " + getIdZombie() + " sale a la zona de riesgo " + rand1);

zonaActual.salirZombie(this);

pausa.comprobarPausa();

}

} catch (InterruptedException ie) {

System.out.println("Error");

}

}

public void atacar(Humano hu, ZonaRiesgo zonaActual) {

try {

pausa.comprobarPausa();

sleep(500 + (int) (1000 \* Math.random()));

pausa.comprobarPausa();

int prob = (int) (3 \* Math.random());

if (prob == 0) {

pausa.comprobarPausa();

String[] idH = hu.getIdHumano();

pausa.comprobarPausa();

matarHumano(hu, zonaActual);

pausa.comprobarPausa();

String[] idZ = new String[]{"Z", idH[1], idH[2], idH[3], idH[4], idH[5]};

new Zombie(idZ, arrayZonaRiesgo, pausa, logger).start();

pausa.comprobarPausa();

contadorMuertes++;

hu.defensa();

}

else {

pausa.comprobarPausa();

logger.escribir("Humano " + hu.getIdHumanoStr() + " ha sido marcado por el zombie " + this.getIdZombie());

hu.setMarcado(true);

//El zombie realiza un await en la cyclic barrier para que el humano y el zombie sigan con la ejecución

hu.defensa();

//El humano ya no está esperando para el ataque, lo marcamos a false

pausa.comprobarPausa();

hu.setEsperandoAtaque(false);

}

} catch (InterruptedException ie) {

System.out.println("Error");

}

}

public void matarHumano(Humano hu, ZonaRiesgo zonaActual) {

pausa.comprobarPausa();

logger.escribir("Humano " + hu.getIdHumanoStr() + " ha muerto a manos del zombie " + this.getIdZombie() + " en la zona " + zonaActual.getId() + "y ahora tambien es un zombie.");

hu.morir();

}

}

### ZonaComun.java

package Parte1;

import static java.lang.Thread.sleep;

import java.util.concurrent.LinkedBlockingQueue;

import javax.swing.SwingUtilities;

import Visuals.ApocalipsisZombi.\*;

public class ZonaComun {

private Tunel[] arrayTunel;

private LinkedBlockingQueue<Humano> listaHumanos = new LinkedBlockingQueue<>();

private VentanaServ ventana;

private Logger logger;

public ZonaComun(Tunel[] at, Logger log, VentanaServ vServ) {

arrayTunel = at;

ventana = vServ;

logger = log;

}

public void entrarZonaComun(Humano hu) {

listaHumanos.add(hu);

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarHumanosZonaComun();

}

});

logger.escribir("Humano " + hu.getIdHumanoStr() + " ha entrado a la Zona Común.");

}

public void prepararse(Humano hu) {

try {

logger.escribir("Humano " + hu.getIdHumanoStr() + " se esta preparando.");

sleep((long) (Math.random() \* 1000 + 1000));

} catch (InterruptedException e) {

throw new RuntimeException(e);

}

}

public void vidaFueraRefugio(Humano hu) {

int eleccion = (int) (Math.random() \* 4);

logger.escribir("Humano " + hu.getIdHumanoStr() + " ha decidido salir de la Zona Común por el tunel " + eleccion);

listaHumanos.remove(hu);

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarHumanosZonaComun();

}

});

arrayTunel[eleccion].irExterior(hu);

if (!hu.isMuerto()) {

arrayTunel[eleccion].venirDelExterior(hu);

}

}

public LinkedBlockingQueue<Humano> getListaHumanosZonaComun() {

return listaHumanos;

}

}

### ZonaDescanso.java

package Parte1;

import java.util.concurrent.LinkedBlockingQueue;

import static java.lang.Thread.sleep;

import javax.swing.SwingUtilities;

import Visuals.ApocalipsisZombi.\*;

public class ZonaDescanso {

private LinkedBlockingQueue<Humano> listaDescansando = new LinkedBlockingQueue<>();

private VentanaServ ventana;

private Logger logger;

public ZonaDescanso(Logger log, VentanaServ vent) {

ventana = vent;

logger = log;

}

public LinkedBlockingQueue<Humano> getListaHumanosDescansando() {

return listaDescansando;

}

public void descansarVuelta(Humano hu) {

if (!hu.isMuerto()) {

try {

listaDescansando.add(hu); //Se añade a la lista de descansar

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarHumanosDescansando();

}

});

logger.escribir("Humano " + hu.getIdHumanoStr() + " está descansando tras volver del exterior.");

sleep(2000 + (int) (2000 \* Math.random())); // Tiempo en el que está descansando

listaDescansando.remove(hu); // Sale de la lista de descansar

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarHumanosDescansando();

}

});

logger.escribir("Humano " + hu.getIdHumanoStr() + " ha terminado de descansar.");

} catch (InterruptedException ie) {

System.out.println("Error descansar al volver humano " + hu.getIdHumanoStr());

}

}

}

public void descansarMarcado(Humano hu) {

try {

listaDescansando.add(hu); // Se añade a la lista de descansar, misma lista para los sanos y marcados

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarHumanosDescansando();

}

});

logger.escribir("Humano marcado " + hu.getIdHumanoStr() + " está descansando tras ser atacado para curarse.");

hu.setMarcado(false); // Se le quita la condición de marcado

sleep(3000 + (int) (2000 \* Math.random()));

listaDescansando.remove(hu); // Sale de la lista de marcados

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarHumanosDescansando();

}

});

logger.escribir("Humano " + hu.getIdHumanoStr() + " se ha curado y ya no esta marcado.");

} catch (InterruptedException ie) {

System.out.println("Error descansar humano marcado " + hu.getIdHumanoStr());

}

}

}

### ZonaRiesgo.java

package Parte1;

import Visuals.ApocalipsisZombi.VentanaServ;

import java.util.ArrayList;

import java.util.concurrent.LinkedBlockingQueue;

import javax.swing.SwingUtilities;

public class ZonaRiesgo {

private LinkedBlockingQueue<Zombie> listaZombies = new LinkedBlockingQueue<>();

private ArrayList<Humano> listaHumanos = new ArrayList<>();

private int id;

private VentanaServ ventana;

private Logger logger;

public ZonaRiesgo(int pid, Logger log, VentanaServ vent) {

id = pid;

ventana = vent;

logger = log;

}

public int getId() {

return id;

}

public ArrayList<Humano> getListaHumanos() {

return listaHumanos;

}

public LinkedBlockingQueue<Zombie> getListaZombies() {

return listaZombies;

}

public void entrarHumano(Humano hu) {

listaHumanos.add(hu);

}

public void salirHumano(Humano hu) {

listaHumanos.remove(hu);

}

public void entrarZombie(Zombie zo) {

listaZombies.add(zo);

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarZombiesZP(id);

}

});

}

public void salirZombie(Zombie zo) {

listaZombies.remove(zo);

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarZombiesZP(id);

}

});

}

public synchronized Humano devolverHumanoAleatorio(ZonaRiesgo zonaActual) {

if (zonaActual.getListaHumanos().isEmpty()) {

return null;

} else {

int numHumanos = zonaActual.getListaHumanos().size();

int rand2 = (int) (numHumanos \* Math.random());

Humano huAtacado = (Humano) zonaActual.getListaHumanos().get(rand2);

zonaActual.getListaHumanos().remove(huAtacado);

SwingUtilities.invokeLater(new Runnable() {

public void run() {

ventana.actualizarHumanosZP(id);

}

});

return huAtacado;

}

}

}

## PARTE2

### Cliente.java

package Parte2;

import java.io.\*;

import java.net.InetAddress;

import java.net.Socket;

import java.util.ArrayList;

import Visuals.ApocalipsisZombi.\*;

public class Cliente {

private VentanaCli ventana = new VentanaCli(this);

private boolean seguir = true;

private Boolean parar = false;

private Integer listaComedorRecibida;

private Integer listaZonaComunRecibida;

private Integer listaZonaDescansoRecibida;

private Integer listaZonaRiesgoRecibida1;

private Integer listaZonaRiesgoRecibida2;

private Integer listaZonaRiesgoRecibida3;

private Integer listaZonaRiesgoRecibida4;

private Integer listaPasarTunelRecibida1;

private Integer listaPasarTunelRecibida2;

private Integer listaPasarTunelRecibida3;

private Integer listaPasarTunelRecibida4;

private Integer listaPasandoTunelRecibida1;

private Integer listaPasandoTunelRecibida2;

private Integer listaPasandoTunelRecibida3;

private Integer listaPasandoTunelRecibida4;

private Integer listaRegresandoTunelRecibida1;

private Integer listaRegresandoTunelRecibida2;

private Integer listaRegresandoTunelRecibida3;

private Integer listaRegresandoTunelRecibida4;

private Integer listaZombieZonaRiesgo1;

private Integer listaZombieZonaRiesgo2;

private Integer listaZombieZonaRiesgo3;

private Integer listaZombieZonaRiesgo4;

private ArrayList<String> listaMejoresZombies;

public Cliente() {

ventana.setVisible(true);

}

public void conectarServ() {

Socket cliente;

try {

cliente = new Socket(InetAddress.getLocalHost(), 5002); // Conectar al servidor

ObjectInputStream ois = new ObjectInputStream(cliente.getInputStream());

ObjectOutputStream oos = new ObjectOutputStream(cliente.getOutputStream());

while (true) {

System.out.println(parar);

oos.writeObject(parar);

System.out.println(parar);

oos.flush();

oos.reset();

//Listas de la zona segura

listaComedorRecibida = (Integer) ois.readObject();

System.out.println("Tamaño de la lista de comedor: " + listaComedorRecibida);

listaZonaComunRecibida = (Integer) ois.readObject();

listaZonaDescansoRecibida = (Integer) ois.readObject();

//Listas de humanos en zonas de riesgo

listaZonaRiesgoRecibida1 = (Integer) ois.readObject();

listaZonaRiesgoRecibida2 = (Integer) ois.readObject();

listaZonaRiesgoRecibida3 = (Integer) ois.readObject();

listaZonaRiesgoRecibida4 = (Integer) ois.readObject();

//Listas de humanos para pasar por los túneles

listaPasarTunelRecibida1 = (Integer) ois.readObject();

listaPasarTunelRecibida2 = (Integer) ois.readObject();

listaPasarTunelRecibida3 = (Integer) ois.readObject();

listaPasarTunelRecibida4 = (Integer) ois.readObject();

//Listas de humanos pasando por los tuneles

listaPasandoTunelRecibida1 = (Integer) ois.readObject();

listaPasandoTunelRecibida2 = (Integer) ois.readObject();

listaPasandoTunelRecibida3 = (Integer) ois.readObject();

listaPasandoTunelRecibida4 = (Integer) ois.readObject();

//Listas de humanos regresando por los tuneles

listaRegresandoTunelRecibida1 = (Integer) ois.readObject();

listaRegresandoTunelRecibida2 = (Integer) ois.readObject();

listaRegresandoTunelRecibida3 = (Integer) ois.readObject();

listaRegresandoTunelRecibida4 = (Integer) ois.readObject();

//Listas de Zombies en la zona de Riesgo

listaZombieZonaRiesgo1 = (Integer) ois.readObject();

listaZombieZonaRiesgo2 = (Integer) ois.readObject();

listaZombieZonaRiesgo3 = (Integer) ois.readObject();

listaZombieZonaRiesgo4 = (Integer) ois.readObject();

//Mejores zombies

listaMejoresZombies = (ArrayList<String>) ois.readObject();

System.out.println(listaMejoresZombies);

//cambiarEstadoParar();

//sleep(10000);

ventana.actualizarVentana();

}

} catch (NullPointerException | IOException | ClassNotFoundException e) {

System.out.println("Error");

throw new RuntimeException(e);

}

}

public boolean isPausado() {

return !seguir;

}

public void cambiarEstadoParar() {

parar = !parar;

}

public void pausarServidor() {

parar = true;

}

public void reanudarServidor() {

parar = true;

}

public static void main(String args[]) {

Cliente cliente = new Cliente();

new Thread(() -> cliente.conectarServ()).start();

}

public Integer getListaZombieZonaRiesgo4() {

return listaZombieZonaRiesgo4;

}

public Integer getListaZombieZonaRiesgo3() {

return listaZombieZonaRiesgo3;

}

public Integer getListaZombieZonaRiesgo1() {

return listaZombieZonaRiesgo1;

}

public Integer getListaZombieZonaRiesgo2() {

return listaZombieZonaRiesgo2;

}

public Integer getListaRegresandoTunelRecibida4() {

return listaRegresandoTunelRecibida4;

}

public Integer getListaRegresandoTunelRecibida3() {

return listaRegresandoTunelRecibida3;

}

public Integer getListaRegresandoTunelRecibida2() {

return listaRegresandoTunelRecibida2;

}

public Integer getListaRegresandoTunelRecibida1() {

return listaRegresandoTunelRecibida1;

}

public Integer getListaPasandoTunelRecibida4() {

return listaPasandoTunelRecibida4;

}

public Integer getListaPasandoTunelRecibida3() {

return listaPasandoTunelRecibida3;

}

public Integer getListaPasandoTunelRecibida2() {

return listaPasandoTunelRecibida2;

}

public Integer getListaPasandoTunelRecibida1() {

return listaPasandoTunelRecibida1;

}

public Integer getListaPasarTunelRecibida4() {

return listaPasarTunelRecibida4;

}

public Integer getListaPasarTunelRecibida3() {

return listaPasarTunelRecibida3;

}

public Integer getListaPasarTunelRecibida2() {

return listaPasarTunelRecibida2;

}

public Integer getListaPasarTunelRecibida1() {

return listaPasarTunelRecibida1;

}

public Integer getListaZonaRiesgoRecibida4() {

return listaZonaRiesgoRecibida4;

}

public Integer getListaZonaRiesgoRecibida3() {

return listaZonaRiesgoRecibida3;

}

public Integer getListaZonaRiesgoRecibida2() {

return listaZonaRiesgoRecibida2;

}

public Integer getListaZonaRiesgoRecibida1() {

return listaZonaRiesgoRecibida1;

}

public Integer getListaZonaDescansoRecibida() {

return listaZonaDescansoRecibida;

}

public Integer getListaZonaComunRecibida() {

return listaZonaComunRecibida;

}

public Integer getListaComedorRecibida() {

return listaComedorRecibida;

}

public ArrayList<String> getListaMejoresZombies() {

return listaMejoresZombies;

}

}

## Visuals.ApocalipsisZombi

### VentanaCli.java

package Visuals.ApocalipsisZombi;

import javax.swing.DefaultListModel;

import Parte2.Cliente;

/\*\*

\*

\* @author Alex

\*/

public class VentanaCli extends javax.swing.JFrame {

private Cliente cliente;

/\*\*

\* Creates new form VentanaCli

\*/

public VentanaCli(Cliente cli) {

initComponents();

cliente = cli;

}

public void actualizarVentana(){

borrarValores();

actualizarValores();

}

public void borrarValores(){

jTextFieldHumanosRefugio.removeAll();

jTextFieldTunel1.removeAll();

jTextFieldTunel2.removeAll();

jTextFieldTunel3.removeAll();

jTextFieldTunel4.removeAll();

jTextFieldHumanosZR1.removeAll();

jTextFieldHumanosZR2.removeAll();

jTextFieldHumanosZR3.removeAll();

jTextFieldHumanosZR4.removeAll();

jTextFieldZombisZR1.removeAll();

jTextFieldZombisZR2.removeAll();

jTextFieldZombisZR3.removeAll();

jTextFieldZombisZR4.removeAll();

jList1.removeAll();

}

public void actualizarValores(){

//refugio

jTextFieldHumanosRefugio.setText(""+(cliente.getListaZonaComunRecibida()+cliente.getListaComedorRecibida()+cliente.getListaZonaDescansoRecibida()));

jTextFieldHumanosZComun.setText(""+cliente.getListaZonaComunRecibida());

jTextFieldHumanosZDescanso.setText(""+cliente.getListaZonaDescansoRecibida());

jTextFieldHumanosComedor.setText(""+cliente.getListaComedorRecibida());

//tuneles

jTextFieldTunel1.setText(""+(cliente.getListaPasarTunelRecibida1()+cliente.getListaPasandoTunelRecibida1()+cliente.getListaRegresandoTunelRecibida1()));

jTextFieldTunel2.setText(""+(cliente.getListaPasarTunelRecibida2()+cliente.getListaPasandoTunelRecibida2()+cliente.getListaRegresandoTunelRecibida2()));

jTextFieldTunel3.setText(""+(cliente.getListaPasarTunelRecibida3()+cliente.getListaPasandoTunelRecibida3()+cliente.getListaRegresandoTunelRecibida3()));

jTextFieldTunel4.setText(""+(cliente.getListaPasarTunelRecibida4()+cliente.getListaPasandoTunelRecibida4()+cliente.getListaRegresandoTunelRecibida4()));

//humanos zona riesgo

jTextFieldHumanosZR1.setText(""+cliente.getListaZonaRiesgoRecibida1());

jTextFieldHumanosZR2.setText(""+cliente.getListaZonaRiesgoRecibida2());

jTextFieldHumanosZR3.setText(""+cliente.getListaZonaRiesgoRecibida3());

jTextFieldHumanosZR4.setText(""+cliente.getListaZonaRiesgoRecibida4());

//zombis zona riesgo

jTextFieldZombisZR1.setText(""+cliente.getListaZombieZonaRiesgo1());

jTextFieldZombisZR2.setText(""+cliente.getListaZombieZonaRiesgo2());

jTextFieldZombisZR3.setText(""+cliente.getListaZombieZonaRiesgo3());

jTextFieldZombisZR4.setText(""+cliente.getListaZombieZonaRiesgo4());

//lista zombis mas mortales

DefaultListModel<String> model = new DefaultListModel<>();

jList1.removeAll();

for (int i = 0; i<cliente.getListaMejoresZombies().size(); i++) {

model.addElement(cliente.getListaMejoresZombies().get(i));

}

jList1.setModel(model);

}

/\*\*

\* 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();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jLabel5 = new javax.swing.JLabel();

jScrollPane1 = new javax.swing.JScrollPane();

jList1 = new javax.swing.JList<>();

jToggleButtonParar = new javax.swing.JToggleButton();

jTextFieldTunel1 = new javax.swing.JTextField();

jTextFieldHumanosRefugio = new javax.swing.JTextField();

jTextFieldHumanosZR1 = new javax.swing.JTextField();

jTextFieldHumanosZR2 = new javax.swing.JTextField();

jTextFieldHumanosZR3 = new javax.swing.JTextField();

jTextFieldHumanosZR4 = new javax.swing.JTextField();

jTextFieldZombisZR1 = new javax.swing.JTextField();

jTextFieldZombisZR2 = new javax.swing.JTextField();

jTextFieldZombisZR3 = new javax.swing.JTextField();

jTextFieldZombisZR4 = new javax.swing.JTextField();

jTextFieldTunel2 = new javax.swing.JTextField();

jTextFieldTunel3 = new javax.swing.JTextField();

jTextFieldTunel4 = new javax.swing.JTextField();

jLabel6 = new javax.swing.JLabel();

jLabel7 = new javax.swing.JLabel();

jLabel8 = new javax.swing.JLabel();

jLabel9 = new javax.swing.JLabel();

jTextFieldHumanosZComun = new javax.swing.JTextField();

jTextFieldHumanosZDescanso = new javax.swing.JTextField();

jTextFieldHumanosComedor = new javax.swing.JTextField();

jLabel10 = new javax.swing.JLabel();

jLabel11 = new javax.swing.JLabel();

jLabel12 = new javax.swing.JLabel();

jLabel13 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jLabel1.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jLabel1.setText("NÚMERO DE HUMANOS EN EL REFUGIO");

jLabel2.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jLabel2.setText("NÚMERO DE HUMANOS EN LAS ZONAS DE RIESGO");

jLabel3.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jLabel3.setText("NÚMERO DE ZOMBIS EN LAS ZONAS DE RIESGO");

jLabel4.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jLabel4.setText("NÚMERO DE HUMANOS EN LOS TÚNELES");

jLabel5.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jLabel5.setText("ZOMBIS MÁS LETALES");

jList1.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jScrollPane1.setViewportView(jList1);

jToggleButtonParar.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N

jToggleButtonParar.setText("DETENER EJECUCIÓN");

jToggleButtonParar.addActionListener(new java.awt.event.ActionListener() {

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

jToggleButtonPararActionPerformed(evt);

}

});

jTextFieldTunel1.setEditable(false);

jTextFieldTunel1.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldTunel1ActionPerformed(evt);

}

});

jTextFieldHumanosRefugio.setEditable(false);

jTextFieldHumanosRefugio.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldHumanosRefugioActionPerformed(evt);

}

});

jTextFieldHumanosZR1.setEditable(false);

jTextFieldHumanosZR1.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldHumanosZR1ActionPerformed(evt);

}

});

jTextFieldHumanosZR2.setEditable(false);

jTextFieldHumanosZR2.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldHumanosZR2ActionPerformed(evt);

}

});

jTextFieldHumanosZR3.setEditable(false);

jTextFieldHumanosZR3.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldHumanosZR3ActionPerformed(evt);

}

});

jTextFieldHumanosZR4.setEditable(false);

jTextFieldHumanosZR4.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldHumanosZR4ActionPerformed(evt);

}

});

jTextFieldZombisZR1.setEditable(false);

jTextFieldZombisZR1.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldZombisZR1ActionPerformed(evt);

}

});

jTextFieldZombisZR2.setEditable(false);

jTextFieldZombisZR2.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldZombisZR2ActionPerformed(evt);

}

});

jTextFieldZombisZR3.setEditable(false);

jTextFieldZombisZR3.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldZombisZR3ActionPerformed(evt);

}

});

jTextFieldZombisZR4.setEditable(false);

jTextFieldZombisZR4.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldZombisZR4ActionPerformed(evt);

}

});

jTextFieldTunel2.setEditable(false);

jTextFieldTunel2.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldTunel2ActionPerformed(evt);

}

});

jTextFieldTunel3.setEditable(false);

jTextFieldTunel3.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldTunel3ActionPerformed(evt);

}

});

jTextFieldTunel4.setEditable(false);

jTextFieldTunel4.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldTunel4ActionPerformed(evt);

}

});

jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel6.setText("1");

jLabel7.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel7.setText("2");

jLabel8.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel8.setText("3");

jLabel9.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel9.setText("4");

jTextFieldHumanosZComun.setEditable(false);

jTextFieldHumanosZComun.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldHumanosZComunActionPerformed(evt);

}

});

jTextFieldHumanosZDescanso.setEditable(false);

jTextFieldHumanosZDescanso.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldHumanosZDescansoActionPerformed(evt);

}

});

jTextFieldHumanosComedor.setEditable(false);

jTextFieldHumanosComedor.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldHumanosComedorActionPerformed(evt);

}

});

jLabel10.setText("Z Común");

jLabel11.setText("Z Descanso");

jLabel12.setText("Comedor");

jLabel13.setText("Total");

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addGroup(layout.createSequentialGroup()

.addContainerGap()

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

.addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED\_SIZE, 192, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addGroup(layout.createSequentialGroup()

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 192, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(101, 101, 101)

.addComponent(jToggleButtonParar, javax.swing.GroupLayout.PREFERRED\_SIZE, 359, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

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

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

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, 428, Short.MAX\_VALUE)

.addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel4, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addComponent(jLabel1, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.PREFERRED\_SIZE, 428, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

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

.addGroup(layout.createSequentialGroup()

.addComponent(jTextFieldZombisZR1, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jTextFieldZombisZR2, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jTextFieldZombisZR3, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jTextFieldZombisZR4, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addComponent(jTextFieldHumanosZR1, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jTextFieldHumanosZR2, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jTextFieldHumanosZR3, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jTextFieldHumanosZR4, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

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

.addComponent(jLabel6, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jTextFieldTunel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

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

.addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldTunel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

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

.addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldTunel3, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

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

.addComponent(jTextFieldTunel4, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel9, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGroup(layout.createSequentialGroup()

.addComponent(jTextFieldHumanosZComun, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jTextFieldHumanosZDescanso, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jTextFieldHumanosComedor, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jTextFieldHumanosRefugio, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE))))

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel10, javax.swing.GroupLayout.PREFERRED\_SIZE, 50, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jLabel11)

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

.addComponent(jLabel12)

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

.addComponent(jLabel13)

.addGap(7, 7, 7))))

.addGap(26, 70, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

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

.addGroup(layout.createSequentialGroup()

.addGap(20, 20, 20)

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

.addComponent(jLabel10)

.addComponent(jLabel11)

.addComponent(jLabel12)

.addComponent(jLabel13))

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

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

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldHumanosRefugio, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldHumanosZComun, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldHumanosZDescanso, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldHumanosComedor, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

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

.addComponent(jLabel6)

.addComponent(jLabel7)

.addComponent(jLabel8)

.addComponent(jLabel9))

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

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

.addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldTunel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldTunel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldTunel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldTunel4, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

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

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldHumanosZR1, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldHumanosZR2, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldHumanosZR3, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldHumanosZR4, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

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

.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldZombisZR1, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldZombisZR2, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldZombisZR3, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextFieldZombisZR4, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(17, 17, 17)

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

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 98, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(jToggleButtonParar, javax.swing.GroupLayout.PREFERRED\_SIZE, 51, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap(42, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

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

if (cliente.isPausado()){

jToggleButtonParar.setText("DETENER EJECUCIÓN");

}else{

jToggleButtonParar.setText("REANUDAR EJECUCIÓN");

}

cliente.cambiarEstadoParar();

}

private void jTextFieldTunel1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

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

// TODO add your handling code here:

}

private void jTextFieldHumanosZR1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jTextFieldHumanosZR2ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jTextFieldHumanosZR3ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jTextFieldHumanosZR4ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jTextFieldZombisZR1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jTextFieldZombisZR2ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jTextFieldZombisZR3ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jTextFieldZombisZR4ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jTextFieldTunel2ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jTextFieldTunel3ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jTextFieldTunel4ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

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

// TODO add your handling code here:

}

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

// TODO add your handling code here:

}

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

// TODO add your handling code here:

}

/\*\*

\* @param args the command line arguments

\*/

// Variables declaration - do not modify

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 jLabel2;

private javax.swing.JLabel jLabel3;

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.JList<String> jList1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTextField jTextFieldHumanosComedor;

private javax.swing.JTextField jTextFieldHumanosRefugio;

private javax.swing.JTextField jTextFieldHumanosZComun;

private javax.swing.JTextField jTextFieldHumanosZDescanso;

private javax.swing.JTextField jTextFieldHumanosZR1;

private javax.swing.JTextField jTextFieldHumanosZR2;

private javax.swing.JTextField jTextFieldHumanosZR3;

private javax.swing.JTextField jTextFieldHumanosZR4;

private javax.swing.JTextField jTextFieldTunel1;

private javax.swing.JTextField jTextFieldTunel2;

private javax.swing.JTextField jTextFieldTunel3;

private javax.swing.JTextField jTextFieldTunel4;

private javax.swing.JTextField jTextFieldZombisZR1;

private javax.swing.JTextField jTextFieldZombisZR2;

private javax.swing.JTextField jTextFieldZombisZR3;

private javax.swing.JTextField jTextFieldZombisZR4;

private javax.swing.JToggleButton jToggleButtonParar;

// End of variables declaration

}

### VentanaServ.java

package Visuals.ApocalipsisZombi;

import Parte1.Humano;

import Parte1.Zombie;

import Parte1.Arranque;

import javax.swing.\*;

/\*\*

\*

\* @author Alex y Raul

\*/

public class VentanaServ extends javax.swing.JFrame {

private Arranque arr;

public VentanaServ(Arranque pArr) {

setTitle("Apocalipsis Zombi");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

initComponents();

arr = pArr;

}

public void actualizarHumanosDescansando() {

jTextAreaHumanosDescanso.setText(""); // Limpiar el JTextArea

for (Humano hu : arr.getZonaDescanso().getListaHumanosDescansando()) {

jTextAreaHumanosDescanso.append(hu.getIdHumanoStr()+" ");

}

jTextAreaHumanosDescanso.setCaretPosition(jTextAreaHumanosDescanso.getDocument().getLength());

}

public void actualizarListaPasarTunel(int id) {

if (id == 0) {

jListPasar0.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[0].getListaPasar()) {

model.addElement(hu.getIdHumanoStr());

}

jListPasar0.setModel(model);

} else if (id == 1) {

jListPasar1.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[1].getListaPasar()) {

model.addElement(hu.getIdHumanoStr());

}

jListPasar1.setModel(model);

} else if (id == 2) {

jListPasar2.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[2].getListaPasar()) {

model.addElement(hu.getIdHumanoStr());

}

jListPasar2.setModel(model);

} else if (id == 3) {

jListPasar3.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[3].getListaPasar()) {

model.addElement(hu.getIdHumanoStr());

}

jListPasar3.setModel(model);

}

}

public void actualizarListaPasandoTunel(int id) {

if (id == 0) {

jListPasando0.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[0].getListaPasando()) {

model.addElement(hu.getIdHumanoStr());

}

jListPasando0.setModel(model);

} else if (id == 1) {

jListPasando1.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[1].getListaPasando()) {

model.addElement(hu.getIdHumanoStr());

}

jListPasando1.setModel(model);

} else if (id == 2) {

jListPasando2.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[2].getListaPasando()) {

model.addElement(hu.getIdHumanoStr());

}

jListPasando2.setModel(model);

} else if (id == 3) {

jListPasando3.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[3].getListaPasando()) {

model.addElement(hu.getIdHumanoStr());

}

jListPasando3.setModel(model);

}

}

public void actualizarListaRegresarTunel(int id) {

if (id == 0) {

jListRegresando0.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[0].getListaRegresar()) {

model.addElement(hu.getIdHumanoStr());

}

jListRegresando0.setModel(model);

} else if (id == 1) {

jListRegresando1.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[1].getListaRegresar()) {

model.addElement(hu.getIdHumanoStr());

}

jListRegresando1.setModel(model);

} else if (id == 2) {

jListRegresando2.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[2].getListaRegresar()) {

model.addElement(hu.getIdHumanoStr());

}

jListRegresando2.setModel(model);

} else if (id == 3) {

jListRegresando3.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayTunel()[3].getListaRegresar()) {

model.addElement(hu.getIdHumanoStr());

}

jListRegresando3.setModel(model);

}

}

public void actualizarHumanosZP(int id) {

if (id == 0) {

jListHumanosZona0.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayZonaRiesgo()[0].getListaHumanos()) {

model.addElement(hu.getIdHumanoStr());

}

jListHumanosZona0.setModel(model);

} else if (id == 1) {

jListHumanosZona1.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayZonaRiesgo()[0].getListaHumanos()) {

model.addElement(hu.getIdHumanoStr());

}

jListHumanosZona0.setModel(model);

} else if (id == 2) {

jListHumanosZona2.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayZonaRiesgo()[2].getListaHumanos()) {

model.addElement(hu.getIdHumanoStr());

}

jListHumanosZona2.setModel(model);

} else if (id == 3) {

jListHumanosZona3.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Humano hu : arr.getArrayZonaRiesgo()[3].getListaHumanos()) {

model.addElement(hu.getIdHumanoStr());

}

jListHumanosZona3.setModel(model);

}

}

public void actualizarZombiesZP(int id) {

if (id == 0) {

jListZombis0.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Zombie zo : arr.getArrayZonaRiesgo()[0].getListaZombies()) {

model.addElement(zo.getIdZombie());

}

jListZombis0.setModel(model);

} else if (id == 1) {

jListZombis1.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Zombie zo : arr.getArrayZonaRiesgo()[1].getListaZombies()) {

model.addElement(zo.getIdZombie());

}

jListZombis1.setModel(model);

} else if (id == 2) {

jListZombis2.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Zombie zo : arr.getArrayZonaRiesgo()[2].getListaZombies()) {

model.addElement(zo.getIdZombie());

}

jListZombis2.setModel(model);

} else if (id == 3) {

jListZombis3.removeAll();

DefaultListModel<String> model = new DefaultListModel<>();

for (Zombie zo : arr.getArrayZonaRiesgo()[3].getListaZombies()) {

model.addElement(zo.getIdZombie());

}

jListZombis3.setModel(model);

}

}

public void actualizarHumanosComedor() {

jTextAreaHumanosComedor.setText(""); // Limpiar el JTextArea

for (Humano hu : arr.getComedor().getListaHumanosComedor()) {

jTextAreaHumanosComedor.append(" " + hu.getIdHumanoStr());

}

jTextAreaHumanosComedor.setCaretPosition(jTextAreaHumanosComedor.getDocument().getLength());

}

public void actualizarComida() {

jTextFieldNumComida.setText(arr.getComedor().getComida().toString());

}

public void actualizarHumanosZonaComun() {

jTextAreaHumanosZonaComun.setText(""); // Limpiar el JTextArea

for (Humano hu : arr.getZonaComun().getListaHumanosZonaComun()) {

jTextAreaHumanosZonaComun.append(" " + hu.getIdHumanoStr());

}

jTextAreaHumanosZonaComun.setCaretPosition(jTextAreaHumanosZonaComun.getDocument().getLength());

}

/\*\*

\* 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() {

jPanelRefugio = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

jPanelDescanso = new javax.swing.JPanel();

jLabel2 = new javax.swing.JLabel();

jScrollPane1 = new javax.swing.JScrollPane();

jTextAreaHumanosDescanso = new javax.swing.JTextArea();

jPanel1 = new javax.swing.JPanel();

jLabel3 = new javax.swing.JLabel();

jScrollPane2 = new javax.swing.JScrollPane();

jTextAreaHumanosComedor = new javax.swing.JTextArea();

jLabel4 = new javax.swing.JLabel();

jTextFieldNumComida = new javax.swing.JTextField();

jPanel2 = new javax.swing.JPanel();

jLabel5 = new javax.swing.JLabel();

jScrollPane3 = new javax.swing.JScrollPane();

jTextAreaHumanosZonaComun = new javax.swing.JTextArea();

jPanelTuneles = new javax.swing.JPanel();

jLabel6 = new javax.swing.JLabel();

jPanelTunel0 = new javax.swing.JPanel();

jLabel7 = new javax.swing.JLabel();

jScrollPaneListaRegresando0 = new javax.swing.JScrollPane();

jListRegresando0 = new javax.swing.JList<>();

jScrollPane5 = new javax.swing.JScrollPane();

jListPasando0 = new javax.swing.JList<>();

jScrollPane6 = new javax.swing.JScrollPane();

jListPasar0 = new javax.swing.JList<>();

jPanelTunel1 = new javax.swing.JPanel();

jLabel8 = new javax.swing.JLabel();

jScrollPaneListaRegresando6 = new javax.swing.JScrollPane();

jListRegresando1 = new javax.swing.JList<>();

jScrollPane21 = new javax.swing.JScrollPane();

jListPasando1 = new javax.swing.JList<>();

jScrollPane22 = new javax.swing.JScrollPane();

jListPasar1 = new javax.swing.JList<>();

jPanelTunel2 = new javax.swing.JPanel();

jLabel9 = new javax.swing.JLabel();

jScrollPane23 = new javax.swing.JScrollPane();

jListPasar2 = new javax.swing.JList<>();

jScrollPane24 = new javax.swing.JScrollPane();

jListPasando2 = new javax.swing.JList<>();

jScrollPaneListaRegresando7 = new javax.swing.JScrollPane();

jListRegresando2 = new javax.swing.JList<>();

jPanelTunel3 = new javax.swing.JPanel();

jLabel10 = new javax.swing.JLabel();

jScrollPane25 = new javax.swing.JScrollPane();

jListPasar3 = new javax.swing.JList<>();

jScrollPane26 = new javax.swing.JScrollPane();

jListPasando3 = new javax.swing.JList<>();

jScrollPaneListaRegresando8 = new javax.swing.JScrollPane();

jListRegresando3 = new javax.swing.JList<>();

jPaneZonaRiesgo = new javax.swing.JPanel();

jPanelTunel4 = new javax.swing.JPanel();

jLabel11 = new javax.swing.JLabel();

jScrollPaneListaRegresando1 = new javax.swing.JScrollPane();

jListHumanosZona0 = new javax.swing.JList<>();

jScrollPane8 = new javax.swing.JScrollPane();

jListZombis0 = new javax.swing.JList<>();

jPanelTunel5 = new javax.swing.JPanel();

jLabel12 = new javax.swing.JLabel();

jScrollPaneListaRegresando2 = new javax.swing.JScrollPane();

jListHumanosZona1 = new javax.swing.JList<>();

jScrollPane9 = new javax.swing.JScrollPane();

jListZombis1 = new javax.swing.JList<>();

jPanelTunel6 = new javax.swing.JPanel();

jLabel13 = new javax.swing.JLabel();

jScrollPaneListaRegresando3 = new javax.swing.JScrollPane();

jListHumanosZona3 = new javax.swing.JList<>();

jScrollPane10 = new javax.swing.JScrollPane();

jListZombis3 = new javax.swing.JList<>();

jPanelTunel7 = new javax.swing.JPanel();

jLabel14 = new javax.swing.JLabel();

jScrollPaneListaRegresando4 = new javax.swing.JScrollPane();

jListHumanosZona2 = new javax.swing.JList<>();

jScrollPane11 = new javax.swing.JScrollPane();

jListZombis2 = new javax.swing.JList<>();

jLabel15 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jPanelRefugio.setBackground(new java.awt.Color(102, 255, 255));

jPanelRefugio.setPreferredSize(new java.awt.Dimension(200, 400));

jLabel1.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N

jLabel1.setForeground(new java.awt.Color(0, 0, 0));

jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel1.setText("REFUGIO");

jPanelDescanso.setBackground(new java.awt.Color(102, 102, 102));

jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel2.setText("Descansando");

jLabel2.setVerticalAlignment(javax.swing.SwingConstants.TOP);

jTextAreaHumanosDescanso.setColumns(20);

jTextAreaHumanosDescanso.setRows(5);

jScrollPane1.setViewportView(jTextAreaHumanosDescanso);

javax.swing.GroupLayout jPanelDescansoLayout = new javax.swing.GroupLayout(jPanelDescanso);

jPanelDescanso.setLayout(jPanelDescansoLayout);

jPanelDescansoLayout.setHorizontalGroup(

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

.addGroup(jPanelDescansoLayout.createSequentialGroup()

.addContainerGap()

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

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE))

.addContainerGap())

);

jPanelDescansoLayout.setVerticalGroup(

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

.addGroup(jPanelDescansoLayout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 65, javax.swing.GroupLayout.PREFERRED\_SIZE))

);

jPanel1.setBackground(new java.awt.Color(102, 102, 102));

jPanel1.setForeground(new java.awt.Color(153, 255, 255));

jLabel3.setBackground(new java.awt.Color(0, 0, 0));

jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel3.setText("Comedor");

jTextAreaHumanosComedor.setColumns(20);

jTextAreaHumanosComedor.setRows(5);

jScrollPane2.setViewportView(jTextAreaHumanosComedor);

jLabel4.setText("Comida");

jTextFieldNumComida.addActionListener(new java.awt.event.ActionListener() {

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

jTextFieldNumComidaActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 115, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(0, 0, Short.MAX\_VALUE)

.addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED\_SIZE, 55, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jTextFieldNumComida, javax.swing.GroupLayout.PREFERRED\_SIZE, 71, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 1, Short.MAX\_VALUE)))))

.addContainerGap())

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 57, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jLabel4)

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

.addComponent(jTextFieldNumComida, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addContainerGap())

);

jPanel2.setBackground(new java.awt.Color(102, 102, 102));

jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel5.setText("Zona común");

jTextAreaHumanosZonaComun.setColumns(20);

jTextAreaHumanosZonaComun.setRows(5);

jScrollPane3.setViewportView(jTextAreaHumanosZonaComun);

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE)

.addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addContainerGap())

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jLabel5)

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

.addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED\_SIZE, 68, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

javax.swing.GroupLayout jPanelRefugioLayout = new javax.swing.GroupLayout(jPanelRefugio);

jPanelRefugio.setLayout(jPanelRefugioLayout);

jPanelRefugioLayout.setHorizontalGroup(

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

.addGroup(jPanelRefugioLayout.createSequentialGroup()

.addContainerGap()

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

.addGroup(jPanelRefugioLayout.createSequentialGroup()

.addGroup(jPanelRefugioLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanelDescanso, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanel1, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGap(0, 29, Short.MAX\_VALUE))

.addComponent(jLabel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addContainerGap())

);

jPanelRefugioLayout.setVerticalGroup(

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

.addGroup(jPanelRefugioLayout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 28, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jPanelDescanso, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, 0))

);

jPanelTuneles.setBackground(new java.awt.Color(255, 153, 0));

jPanelTuneles.setPreferredSize(new java.awt.Dimension(200, 400));

jLabel6.setBackground(new java.awt.Color(0, 0, 0));

jLabel6.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N

jLabel6.setForeground(new java.awt.Color(0, 0, 0));

jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel6.setText("TUNELES");

jLabel7.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel7.setText("TUNEL 0");

jScrollPaneListaRegresando0.setViewportView(jListRegresando0);

jListPasando0.setVisibleRowCount(3);

jScrollPane5.setViewportView(jListPasando0);

jListPasar0.setModel(new javax.swing.AbstractListModel<String>() {

String[] strings = { " " };

public int getSize() { return strings.length; }

public String getElementAt(int i) { return strings[i]; }

});

jScrollPane6.setViewportView(jListPasar0);

javax.swing.GroupLayout jPanelTunel0Layout = new javax.swing.GroupLayout(jPanelTunel0);

jPanelTunel0.setLayout(jPanelTunel0Layout);

jPanelTunel0Layout.setHorizontalGroup(

jPanelTunel0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel0Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanelTunel0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel7, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(jPanelTunel0Layout.createSequentialGroup()

.addGap(12, 12, 12)

.addComponent(jScrollPane6, javax.swing.GroupLayout.PREFERRED\_SIZE, 78, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jScrollPane5, javax.swing.GroupLayout.PREFERRED\_SIZE, 87, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jScrollPaneListaRegresando0, javax.swing.GroupLayout.PREFERRED\_SIZE, 81, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 9, Short.MAX\_VALUE)))

.addContainerGap())

);

jPanelTunel0Layout.setVerticalGroup(

jPanelTunel0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel0Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel7)

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

.addGroup(jPanelTunel0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(jScrollPane6, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE)

.addComponent(jScrollPane5, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, 74, Short.MAX\_VALUE)

.addComponent(jScrollPaneListaRegresando0, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

jLabel8.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel8.setText("TUNEL 1");

jScrollPaneListaRegresando6.setViewportView(jListRegresando1);

jListPasando1.setVisibleRowCount(3);

jScrollPane21.setViewportView(jListPasando1);

jListPasar1.setModel(new javax.swing.AbstractListModel<String>() {

String[] strings = { " " };

public int getSize() { return strings.length; }

public String getElementAt(int i) { return strings[i]; }

});

jScrollPane22.setViewportView(jListPasar1);

javax.swing.GroupLayout jPanelTunel1Layout = new javax.swing.GroupLayout(jPanelTunel1);

jPanelTunel1.setLayout(jPanelTunel1Layout);

jPanelTunel1Layout.setHorizontalGroup(

jPanelTunel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel1Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanelTunel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel8, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanelTunel1Layout.createSequentialGroup()

.addGap(9, 9, 9)

.addComponent(jScrollPane22, javax.swing.GroupLayout.PREFERRED\_SIZE, 78, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jScrollPane21, javax.swing.GroupLayout.PREFERRED\_SIZE, 87, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jScrollPaneListaRegresando6, javax.swing.GroupLayout.PREFERRED\_SIZE, 81, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE)))

.addContainerGap())

);

jPanelTunel1Layout.setVerticalGroup(

jPanelTunel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel1Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel8)

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

.addGroup(jPanelTunel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jScrollPaneListaRegresando6, javax.swing.GroupLayout.DEFAULT\_SIZE, 79, Short.MAX\_VALUE)

.addComponent(jScrollPane21, javax.swing.GroupLayout.DEFAULT\_SIZE, 79, Short.MAX\_VALUE)

.addComponent(jScrollPane22, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE))

.addContainerGap())

);

jLabel9.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel9.setText("TUNEL 2");

jListPasar2.setModel(new javax.swing.AbstractListModel<String>() {

String[] strings = { " " };

public int getSize() { return strings.length; }

public String getElementAt(int i) { return strings[i]; }

});

jScrollPane23.setViewportView(jListPasar2);

jListPasando2.setVisibleRowCount(3);

jScrollPane24.setViewportView(jListPasando2);

jScrollPaneListaRegresando7.setViewportView(jListRegresando2);

javax.swing.GroupLayout jPanelTunel2Layout = new javax.swing.GroupLayout(jPanelTunel2);

jPanelTunel2.setLayout(jPanelTunel2Layout);

jPanelTunel2Layout.setHorizontalGroup(

jPanelTunel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel2Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanelTunel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel9, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(jPanelTunel2Layout.createSequentialGroup()

.addGap(6, 6, 6)

.addComponent(jScrollPane23, javax.swing.GroupLayout.PREFERRED\_SIZE, 84, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jScrollPane24, javax.swing.GroupLayout.PREFERRED\_SIZE, 87, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jScrollPaneListaRegresando7, javax.swing.GroupLayout.PREFERRED\_SIZE, 81, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE)))

.addContainerGap())

);

jPanelTunel2Layout.setVerticalGroup(

jPanelTunel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel2Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel9)

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

.addGroup(jPanelTunel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jScrollPane23, javax.swing.GroupLayout.PREFERRED\_SIZE, 80, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jScrollPane24)

.addComponent(jScrollPaneListaRegresando7, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE))

.addContainerGap(15, Short.MAX\_VALUE))

);

jLabel10.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel10.setText("TUNEL 3");

jListPasar3.setModel(new javax.swing.AbstractListModel<String>() {

String[] strings = { " " };

public int getSize() { return strings.length; }

public String getElementAt(int i) { return strings[i]; }

});

jScrollPane25.setViewportView(jListPasar3);

jListPasando3.setVisibleRowCount(3);

jScrollPane26.setViewportView(jListPasando3);

jScrollPaneListaRegresando8.setViewportView(jListRegresando3);

javax.swing.GroupLayout jPanelTunel3Layout = new javax.swing.GroupLayout(jPanelTunel3);

jPanelTunel3.setLayout(jPanelTunel3Layout);

jPanelTunel3Layout.setHorizontalGroup(

jPanelTunel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel3Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanelTunel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel10, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(jPanelTunel3Layout.createSequentialGroup()

.addComponent(jScrollPane25, javax.swing.GroupLayout.PREFERRED\_SIZE, 84, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jScrollPane26, javax.swing.GroupLayout.PREFERRED\_SIZE, 87, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jScrollPaneListaRegresando8, javax.swing.GroupLayout.PREFERRED\_SIZE, 81, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE)))

.addContainerGap())

);

jPanelTunel3Layout.setVerticalGroup(

jPanelTunel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel3Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel10)

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

.addGroup(jPanelTunel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel3Layout.createSequentialGroup()

.addComponent(jScrollPane25, javax.swing.GroupLayout.PREFERRED\_SIZE, 80, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE))

.addComponent(jScrollPane26)

.addComponent(jScrollPaneListaRegresando8, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE))

.addContainerGap())

);

javax.swing.GroupLayout jPanelTunelesLayout = new javax.swing.GroupLayout(jPanelTuneles);

jPanelTuneles.setLayout(jPanelTunelesLayout);

jPanelTunelesLayout.setHorizontalGroup(

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

.addGroup(jPanelTunelesLayout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanelTunelesLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(jLabel6, javax.swing.GroupLayout.DEFAULT\_SIZE, 303, Short.MAX\_VALUE)

.addComponent(jPanelTunel3, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanelTunel0, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanelTunel1, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanelTunel2, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addContainerGap(28, Short.MAX\_VALUE))

);

jPanelTunelesLayout.setVerticalGroup(

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

.addGroup(jPanelTunelesLayout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED\_SIZE, 27, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jPanelTunel0, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jPanelTunel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jPanelTunel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jPanelTunel3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(20, Short.MAX\_VALUE))

);

jPaneZonaRiesgo.setBackground(new java.awt.Color(51, 255, 51));

jPaneZonaRiesgo.setPreferredSize(new java.awt.Dimension(200, 400));

jLabel11.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel11.setText("ZONA 0");

jScrollPaneListaRegresando1.setViewportView(jListHumanosZona0);

jListZombis0.setModel(new javax.swing.AbstractListModel<String>() {

String[] strings = { " " };

public int getSize() { return strings.length; }

public String getElementAt(int i) { return strings[i]; }

});

jScrollPane8.setViewportView(jListZombis0);

javax.swing.GroupLayout jPanelTunel4Layout = new javax.swing.GroupLayout(jPanelTunel4);

jPanelTunel4.setLayout(jPanelTunel4Layout);

jPanelTunel4Layout.setHorizontalGroup(

jPanelTunel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel4Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel11, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addContainerGap())

.addGroup(jPanelTunel4Layout.createSequentialGroup()

.addGap(33, 33, 33)

.addComponent(jScrollPaneListaRegresando1, javax.swing.GroupLayout.PREFERRED\_SIZE, 81, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 73, Short.MAX\_VALUE)

.addComponent(jScrollPane8, javax.swing.GroupLayout.PREFERRED\_SIZE, 78, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(29, 29, 29))

);

jPanelTunel4Layout.setVerticalGroup(

jPanelTunel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel4Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel11)

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

.addGroup(jPanelTunel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(jScrollPane8, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE)

.addComponent(jScrollPaneListaRegresando1, javax.swing.GroupLayout.DEFAULT\_SIZE, 74, Short.MAX\_VALUE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

jLabel12.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel12.setText("ZONA 1");

jScrollPaneListaRegresando2.setViewportView(jListHumanosZona1);

jListZombis1.setModel(new javax.swing.AbstractListModel<String>() {

String[] strings = { " " };

public int getSize() { return strings.length; }

public String getElementAt(int i) { return strings[i]; }

});

jScrollPane9.setViewportView(jListZombis1);

javax.swing.GroupLayout jPanelTunel5Layout = new javax.swing.GroupLayout(jPanelTunel5);

jPanelTunel5.setLayout(jPanelTunel5Layout);

jPanelTunel5Layout.setHorizontalGroup(

jPanelTunel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel5Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel12, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addContainerGap())

.addGroup(jPanelTunel5Layout.createSequentialGroup()

.addGap(33, 33, 33)

.addComponent(jScrollPaneListaRegresando2, javax.swing.GroupLayout.PREFERRED\_SIZE, 81, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jScrollPane9, javax.swing.GroupLayout.PREFERRED\_SIZE, 78, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(29, 29, 29))

);

jPanelTunel5Layout.setVerticalGroup(

jPanelTunel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel5Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel12)

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

.addGroup(jPanelTunel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(jScrollPane9, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE)

.addComponent(jScrollPaneListaRegresando2, javax.swing.GroupLayout.DEFAULT\_SIZE, 74, Short.MAX\_VALUE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

jLabel13.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel13.setText("ZONA 3");

jScrollPaneListaRegresando3.setViewportView(jListHumanosZona3);

jListZombis3.setModel(new javax.swing.AbstractListModel<String>() {

String[] strings = { " " };

public int getSize() { return strings.length; }

public String getElementAt(int i) { return strings[i]; }

});

jScrollPane10.setViewportView(jListZombis3);

javax.swing.GroupLayout jPanelTunel6Layout = new javax.swing.GroupLayout(jPanelTunel6);

jPanelTunel6.setLayout(jPanelTunel6Layout);

jPanelTunel6Layout.setHorizontalGroup(

jPanelTunel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel6Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel13, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addContainerGap())

.addGroup(jPanelTunel6Layout.createSequentialGroup()

.addGap(33, 33, 33)

.addComponent(jScrollPaneListaRegresando3, javax.swing.GroupLayout.PREFERRED\_SIZE, 81, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jScrollPane10, javax.swing.GroupLayout.PREFERRED\_SIZE, 78, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(29, 29, 29))

);

jPanelTunel6Layout.setVerticalGroup(

jPanelTunel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel6Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel13)

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

.addGroup(jPanelTunel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(jScrollPane10, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE)

.addComponent(jScrollPaneListaRegresando3, javax.swing.GroupLayout.DEFAULT\_SIZE, 74, Short.MAX\_VALUE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

jLabel14.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel14.setText("ZONA 2");

jScrollPaneListaRegresando4.setViewportView(jListHumanosZona2);

jListZombis2.setModel(new javax.swing.AbstractListModel<String>() {

String[] strings = { " " };

public int getSize() { return strings.length; }

public String getElementAt(int i) { return strings[i]; }

});

jScrollPane11.setViewportView(jListZombis2);

javax.swing.GroupLayout jPanelTunel7Layout = new javax.swing.GroupLayout(jPanelTunel7);

jPanelTunel7.setLayout(jPanelTunel7Layout);

jPanelTunel7Layout.setHorizontalGroup(

jPanelTunel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel7Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel14, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addContainerGap())

.addGroup(jPanelTunel7Layout.createSequentialGroup()

.addGap(33, 33, 33)

.addComponent(jScrollPaneListaRegresando4, javax.swing.GroupLayout.PREFERRED\_SIZE, 81, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jScrollPane11, javax.swing.GroupLayout.PREFERRED\_SIZE, 78, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(29, 29, 29))

);

jPanelTunel7Layout.setVerticalGroup(

jPanelTunel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanelTunel7Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel14)

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

.addGroup(jPanelTunel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(jScrollPane11, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE)

.addComponent(jScrollPaneListaRegresando4, javax.swing.GroupLayout.DEFAULT\_SIZE, 74, Short.MAX\_VALUE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

jLabel15.setBackground(new java.awt.Color(0, 0, 0));

jLabel15.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N

jLabel15.setForeground(new java.awt.Color(0, 0, 0));

jLabel15.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel15.setText("ZONAS DE RIESGO");

javax.swing.GroupLayout jPaneZonaRiesgoLayout = new javax.swing.GroupLayout(jPaneZonaRiesgo);

jPaneZonaRiesgo.setLayout(jPaneZonaRiesgoLayout);

jPaneZonaRiesgoLayout.setHorizontalGroup(

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

.addGroup(jPaneZonaRiesgoLayout.createSequentialGroup()

.addContainerGap()

.addGroup(jPaneZonaRiesgoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(jLabel15, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanelTunel4, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanelTunel5, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanelTunel6, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanelTunel7, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addContainerGap(14, Short.MAX\_VALUE))

);

jPaneZonaRiesgoLayout.setVerticalGroup(

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

.addGroup(jPaneZonaRiesgoLayout.createSequentialGroup()

.addGap(14, 14, 14)

.addComponent(jLabel15, javax.swing.GroupLayout.PREFERRED\_SIZE, 27, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jPanelTunel4, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jPanelTunel5, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(24, 24, 24)

.addComponent(jPanelTunel7, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanelTunel6, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(26, 26, 26))

);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addGroup(layout.createSequentialGroup()

.addComponent(jPanelRefugio, javax.swing.GroupLayout.PREFERRED\_SIZE, 252, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(12, 12, 12)

.addComponent(jPanelTuneles, javax.swing.GroupLayout.PREFERRED\_SIZE, 337, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jPaneZonaRiesgo, javax.swing.GroupLayout.PREFERRED\_SIZE, 314, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap())

);

layout.setVerticalGroup(

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

.addComponent(jPanelRefugio, javax.swing.GroupLayout.DEFAULT\_SIZE, 583, Short.MAX\_VALUE)

.addComponent(jPaneZonaRiesgo, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, 583, Short.MAX\_VALUE)

.addComponent(jPanelTuneles, javax.swing.GroupLayout.DEFAULT\_SIZE, 583, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

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

// TODO add your handling code here:

}

/\*\*

\* @param args the command line arguments

\*/

// Variables declaration - do not modify

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 jLabel2;

private javax.swing.JLabel jLabel3;

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.JList<String> jListHumanosZona0;

private javax.swing.JList<String> jListHumanosZona1;

private javax.swing.JList<String> jListHumanosZona2;

private javax.swing.JList<String> jListHumanosZona3;

private javax.swing.JList<String> jListPasando0;

private javax.swing.JList<String> jListPasando1;

private javax.swing.JList<String> jListPasando2;

private javax.swing.JList<String> jListPasando3;

private javax.swing.JList<String> jListPasar0;

private javax.swing.JList<String> jListPasar1;

private javax.swing.JList<String> jListPasar2;

private javax.swing.JList<String> jListPasar3;

private javax.swing.JList<String> jListRegresando0;

private javax.swing.JList<String> jListRegresando1;

private javax.swing.JList<String> jListRegresando2;

private javax.swing.JList<String> jListRegresando3;

private javax.swing.JList<String> jListZombis0;

private javax.swing.JList<String> jListZombis1;

private javax.swing.JList<String> jListZombis2;

private javax.swing.JList<String> jListZombis3;

private javax.swing.JPanel jPaneZonaRiesgo;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanelDescanso;

private javax.swing.JPanel jPanelRefugio;

private javax.swing.JPanel jPanelTunel0;

private javax.swing.JPanel jPanelTunel1;

private javax.swing.JPanel jPanelTunel2;

private javax.swing.JPanel jPanelTunel3;

private javax.swing.JPanel jPanelTunel4;

private javax.swing.JPanel jPanelTunel5;

private javax.swing.JPanel jPanelTunel6;

private javax.swing.JPanel jPanelTunel7;

private javax.swing.JPanel jPanelTuneles;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane10;

private javax.swing.JScrollPane jScrollPane11;

private javax.swing.JScrollPane jScrollPane2;

private javax.swing.JScrollPane jScrollPane21;

private javax.swing.JScrollPane jScrollPane22;

private javax.swing.JScrollPane jScrollPane23;

private javax.swing.JScrollPane jScrollPane24;

private javax.swing.JScrollPane jScrollPane25;

private javax.swing.JScrollPane jScrollPane26;

private javax.swing.JScrollPane jScrollPane3;

private javax.swing.JScrollPane jScrollPane5;

private javax.swing.JScrollPane jScrollPane6;

private javax.swing.JScrollPane jScrollPane8;

private javax.swing.JScrollPane jScrollPane9;

private javax.swing.JScrollPane jScrollPaneListaRegresando0;

private javax.swing.JScrollPane jScrollPaneListaRegresando1;

private javax.swing.JScrollPane jScrollPaneListaRegresando2;

private javax.swing.JScrollPane jScrollPaneListaRegresando3;

private javax.swing.JScrollPane jScrollPaneListaRegresando4;

private javax.swing.JScrollPane jScrollPaneListaRegresando6;

private javax.swing.JScrollPane jScrollPaneListaRegresando7;

private javax.swing.JScrollPane jScrollPaneListaRegresando8;

private javax.swing.JTextArea jTextAreaHumanosComedor;

private javax.swing.JTextArea jTextAreaHumanosDescanso;

private javax.swing.JTextArea jTextAreaHumanosZonaComun;

private javax.swing.JTextField jTextFieldNumComida;

// End of variables declaration

}