Imagen en blanco y negro

Descripción generada automáticamente con confianza media

**Grado en Ingeniería de Computadores**

**Curso 2022/2023 – Convocatoria Ordinaria**

03202543T – Sanavia Valdeolivas, Victor

09064004A – Rodríguez Hurtado, Adrián

Índice

[Análisis de alto nivel 3](#_Toc133913723)

[Herramientas de sincronización utilizadas y diseño general del sistema 4](#_Toc133913724)

[Descripción de las clases principales 7](#_Toc133913725)

[Diagrama de clases 19](#_Toc133913726)

[Código fuente 20](#_Toc133913727)

# Análisis de alto nivel

El problema simula el comportamiento de una colonia de hormigas. Dentro del hormiguero, se pueden distinguir tres actores, los cuales tienen características que las diferencian del resto. Los actores serían hormigas obreras, hormigas soldado y hormigas crías.

* Las primeras son las hormigas obreras, que cuentan con un identificador del tipo “HOXXXX”. Dependiendo de si el identificador es par o impar, realizarán distintas actividades. Las hormigas obreras impares repetirán iterativamente un proceso basado en salir al exterior para recoger comida y depositar ésta en el almacén de comida, mientras que las pares entrará al almacén de comida para recoger 5 alimentos y llevarlos a la zona de comer. Además, todas las hormigas obreras, después de 10 iteraciones, accederán a la zona de comer para alimentarse, y de allí accederán a la zona de descanso.
* Las segundas son las hormigas soldado, que cuentan con un identificador del tipo “HSXXXX”. Estas hormigas harán instrucción en la zona de instrucción para posteriormente descansar, además de acudir a la zona de comer cada 6 iteraciones. En cuanto aparezca un insecto invasor, todas las hormigas soldado deberán acudir al exterior para repeler a este insecto.
* Las terceras son las hormigas crías, que cuentan con un identificador del tipo “HCXXXX”. El comportamiento de estas hormigas se basa en acceder a la zona de comer y posteriormente, a la zona de descanso. En caso de invasión de un insecto, estas hormigas acudirán al refugio hasta que la invasión haya finalizado.

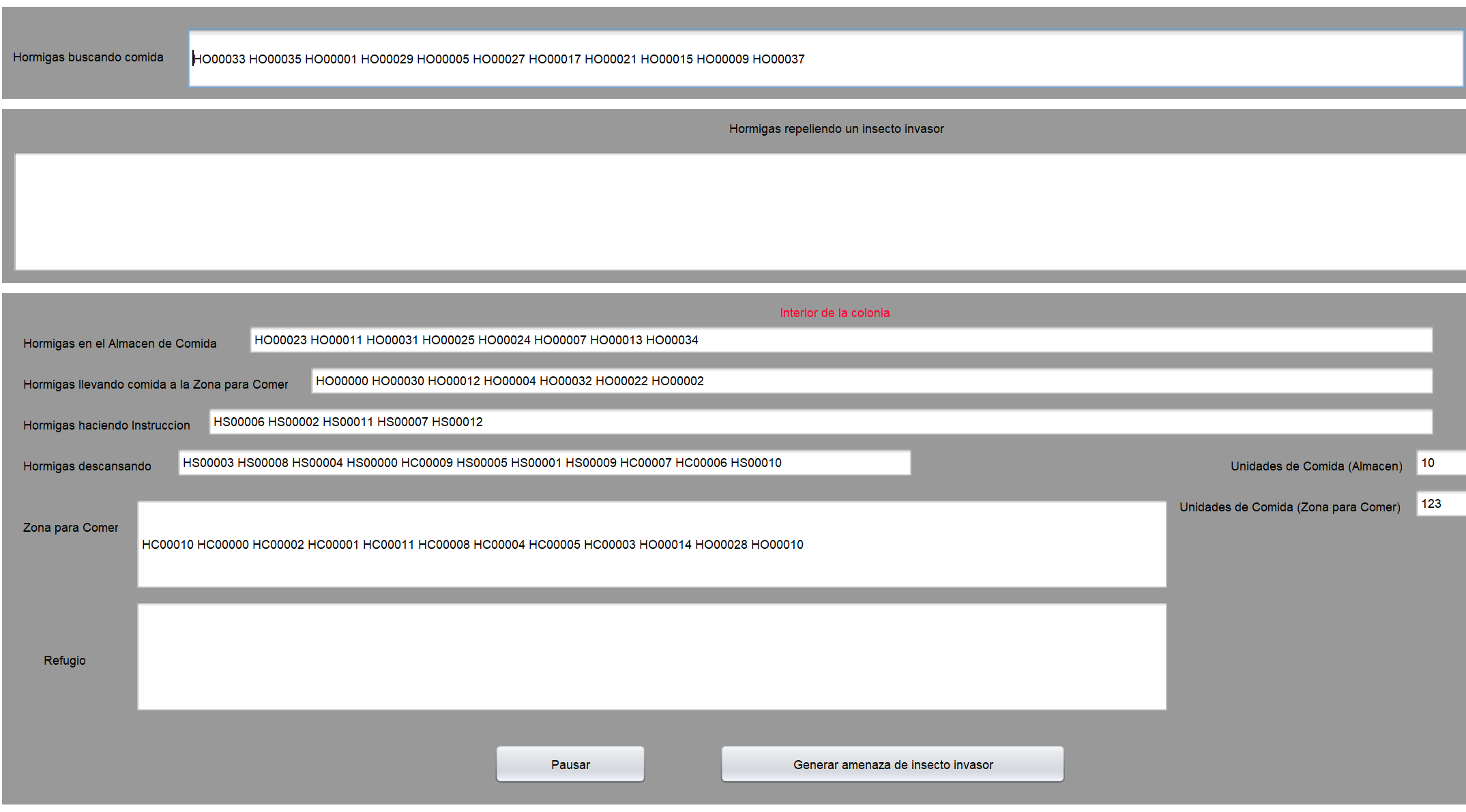
# Herramientas de sincronización utilizadas y diseño general del sistema

En la práctica hemos utilizado diversas herramientas de sincronización, las cuales son Cerrojos, Cerrojos con Condición, Semáforos y Monitores.

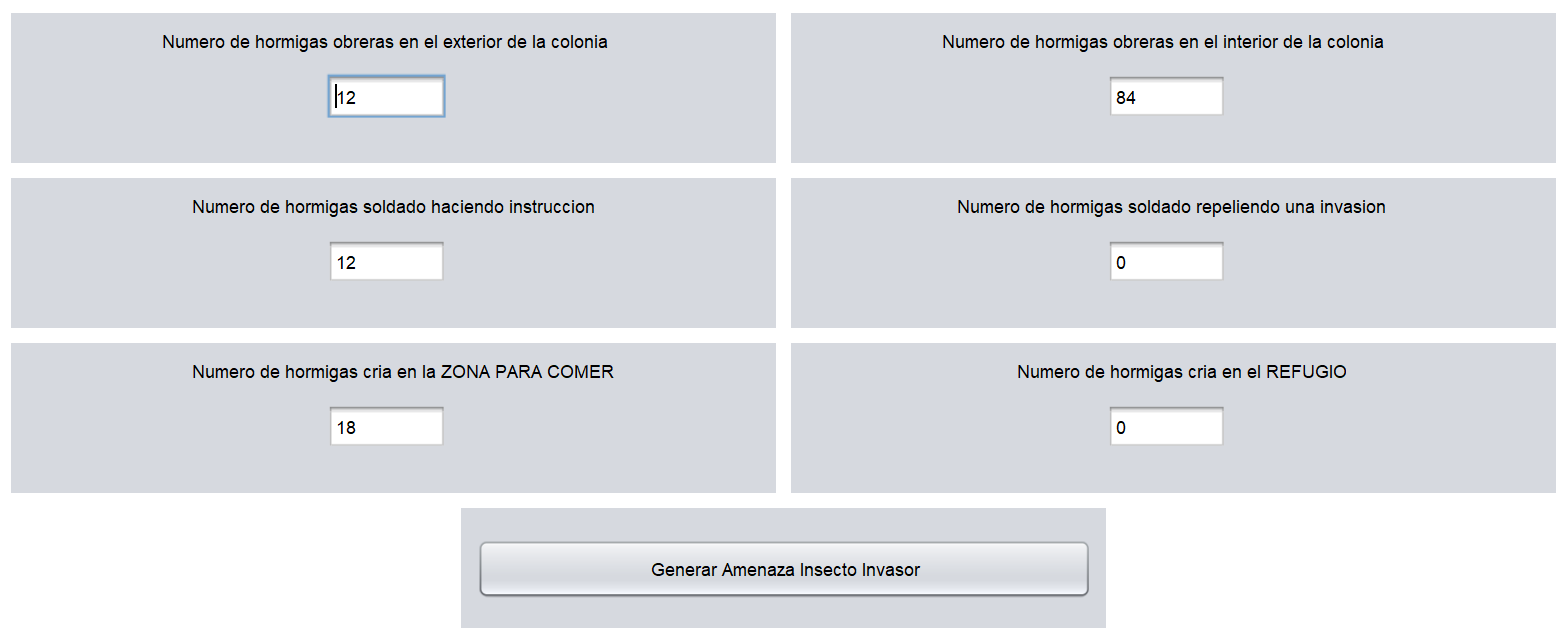
* ***AlmacenComida*** 🡪 Para la clase *AlmacenComida* hemos utilizado las siguientes herramientas:
  + Semáforo *semaforoEntradaSalida:* para asegurarnos mediante los permisos del semáforo que las hormigas no superen el aforo máximo del almacén.
  + Cerrojo *recogeElemento* con Condición *conditionElementoComida:* bloquea a la hormiga obrera para que coja los elementos que tiene que recoger, y en el caso de que no haya, espere.
  + Cerrojo *cerrojoNumElementosComida:* para proteger la lectura y escritura del número de elementos que hay en el almacén.
  + Cerrojo *cerrojoHormigaEsperando*: para proteger la lectura y escritura del número de hormigas que hay esperando.
* ***Colonia*** 🡪 Para la clase *Colonia* hemos utilizado las siguientes herramientas:
  + Cerrojo *entradaColonia*: para asegurar la exclusión mutua de las hormigas que entran a la colonia.
  + Cerrojo *salidaColonia1*: para asegurar la exclusión mutua de las hormigas que salen por la salida uno.
  + Cerrojo *salidaColonia2*:para asegurar la exclusión mutua de las hormigas que salen por la salida dos.
* **Invasion** 🡪 Para la clase *Invasion* hemos utilizado la siguiente herramienta:
  + Cerrojo *cerrojoInvasion* con Condición *conditionInvasion*: las hormigas salen durante 20 segundos a combatir la invasión, pero las hormigas que se crean durante la invasión no salen.
* **ListaThreads** 🡪 Para la clase *ListaThreads* hemos utilizado la siguiente herramienta:
  + Monitores: con el fin de asegurar la exclusión mutua a la hora de imprimir los identificadores de las hormigas en los JTextField.
* **Log** 🡪 Para la clase *Log* hemos utilizado la siguiente herramienta:
  + Monitores: con el fin de asegurar la exclusión mutua a la hora de escribir en el log.
* **Paso** 🡪 Para la clase *Paso* hemos utilizado la siguiente herramienta:
  + Cerrojo *cerrojo* con Condición *parar*: para garantizar la exclusión mutua a la hora de parar o reanudar el programa con la condición de que, si el programa esté parado, al darle de nuevo se reanude y viceversa.
* **Refugio** 🡪 Para la clase *Refugio* hemos utilizado la siguiente herramienta:
  + Cerrojo *cerrojoRefugio* con Condición *finInvasion*: para garantizar la exclusión mutua en las hormigas crías cuando se inicia una invasión, las cuales irán comprobando si pueden salir (es decir, que haya acabado la invasión), y si no pueden salir, esperarán a que acabe ésta.
* **ZonaComer** 🡪 Para la clase *ZonaComer* hemos utilizado las siguientes herramientas:
  + Semáforo *semaforoEntradaSalida*: es un semáforo binario para asegurar la exclusión mutua a la hora de salir y entrar en la zona para comer.
  + Cerrojo *cerrojoElementoComida* con Condición *esperaElementoComida*: asegura exclusión mutua a la hora de coger y depositar comida en la zona para comer, para que no se pueda coger y depositar a la vez.
* **ZonaDescanso** 🡪 Para la clase *ZonaDescanso* hemos utilizado la siguiente herramienta:
  + Cerrojo *entradaSalida*: asegura la exclusión mutua de las hormigas para entrar y salir en la zona para descanso.
* **ZonaInstruccion** 🡪 Para la clase *ZonaInstruccion* hemos utilizado la siguiente herramienta:
  + Cerrojo *entradaSalida*: asegura la exclusión mutua de las hormigas para entrar y salir en la zona para descanso.

Para la parte del diseño del sistema hemos creado dos **interfaces**, una para la parte concurrente y otra para la parte distribuida.

**Interfaz** concurrente 🡪



**Interfaz** distribuida 🡪



# Descripción de las clases principales

El programa está dividido en dos partes, una parte concurrente y otra distribuida.

Parte concurrente: En la parte concurrente hay 15 clases, las cuales son *AlmacenComida*, *Colonia*, *Hormiga*, *HormigaCria*, *HormigaObrera*, *HormigaSoldado*, *Invasion*, *ListaThreads*, *Log*, *Paso*, *ProgPrincipal*, *Refugio*, *ZonaComer*, *ZonaDescanso* y *ZonaInstruccion*.

* **AlmacenComida** 🡪 Para la clase *AlmacenComida* hemos utilizado los siguientes atributos y métodos:
  + Log *log* 🡪 Atributo de tipo log que almacena el log del sistema concurrente.
  + int *numElementosComida* 🡪 Atributo de tipo entero inicializado a 0 que indica el número de elementos de comida que hay en el almacén.
  + int *numHormigasDentro* 🡪 Atributo de tipo entero inicializado a 0 que indica el número de hormigas que hay dentro del almacén.
  + int *numHormigasEsperando* 🡪 Atributo de tipo entero inicializado a 0 que indica el número de hormigas que hay esperando para entrar al almacén.
  + Semaphore *semaforoEntradaSalida*
  + Lock *recogeElemento*
  + Lock *cerrojoNumElementosComida*
  + Lock *cerrojoHormigaEsperando*
  + Condition *conditionElementoComida*
  + ListaThreads *unidadesElementosComida* 🡪 Atributo de tipo ListaThreads que guarda el número de unidades de comida que hay en el almacén.
  + ListaThreads *listaHormigasAlmacenComida* 🡪Atributo de tipo ListaThreads que guarda las hormigas que hay en el almacén.
  + public *AlmacenComida*(Log log, JTextField jTextFieldUnidadesComidaAlmacen, JTextField jTextFieldHormigasAlmacenComida) 🡪 Método constructor de la clase *AlmacenComida*.
  + public void *entra*(HormigaObrera hormiga) 🡪 Método para que una hormiga obrera entre al almacén y, si este está lleno o no hay comida, espere fuera.
  + public void *sale*(HormigaObrera hormiga) 🡪 Método para que la hormiga obrera que se encuentre dentro del almacén salga.
  + public void *depositaElementoComida*(HormigaObrera hormiga) 🡪 Método para que la hormiga deposite los 5 elementos de comida en el almacén y despierte (si hay) a las hormigas esperando para que las transporten.
  + public void *recogeElementoComida*(HormigaObrera hormiga) 🡪 Método para recoger 5 elementos de comida.
  + public void *incrementaNumHormigasDentro*() 🡪 Método que incrementa en uno el atributo *numHormigasDentro*.
  + public void *decrementaNumHormigasDentro*() 🡪 Método que decrementa en uno el atributo *numHormigasDentro*.
  + public int *getNumElementosComida*() 🡪 Método get para el atributo *numElementosComida*.
  + public void *setNumElementosComida*(int numElementosComida) 🡪 Método set para el atributo *numElementosComida*.
  + public int *getNumHormigasEsperando*() 🡪 Método get para el atributo *numHormigasEsperando*.
  + public void *setNumHormigasEsperando*(int numHormigasEsperando) 🡪 Método set para el atributo *numHormigasEsperando*.
  + public ListaThreads *getUnidadesElementosComida*() 🡪 Método get para el atributo *unidadesElementosComida*.
  + public ListaThreads *getListaHormigasAlmacenComida*() 🡪 Método get para el atributo *listaHormigasAlmacenComida*.
  + public Log *getLog*() 🡪 Método get para el atributo *log*.
* **Colonia** 🡪 Para la clase *Colonia* hemos utilizado los siguientes atributos y métodos:
  + int *numHormigasSoldado* 🡪 Atributo de tipo entero inicializado a 0 que indica el número de hormigas soldado que hay en la colonia.
  + Log *log* 🡪 Atributo de tipo log que almacena el log del sistema concurrente.
  + Lock entradaColonia
  + Lock *salidaColonia1*
  + Lock *salidaColonia2*
  + ArrayList<Hormiga> *listaHormigas* = new ArrayList<>() 🡪 Atributo de tipo lista que contiene a todas las hormigas que se encuentran en la colonia.
  + ListaThreads *listaHormigasBuscandoComida* 🡪 Atributo de tipo ListaThreads que contiene a las hormigas que están buscando comida.
  + ListaThreads *listaHormigasLlevandoComida* 🡪 Atributo de tipo ListaThreads que contiene a las hormigas que están llevando comida.
  + AlmacenComida *almacenComida* 🡪 Atributo de tipo AlmacenComida que crea un almacén de comida en la colonia.
  + ZonaComer *zonaComer* 🡪 Atributo de tipo ZonaComer que crea una zona para comer en la colonia.
  + ZonaInstruccion *zonaInstruccion* 🡪 Atributo de tipo ZonaInstruccion que crea una zona para la instrucción en la colonia.
  + ZonaDescanso *zonaDescanso* 🡪 Atributo de tipo ZonaDescanso que crea una zona de descanso en la colonia.
  + Refugio *refugio* 🡪 Atributo de tipo Refugio que crea un refugio en la colonia.
  + Invasion *invasion* 🡪 Atributo de tipo Invasion para permitir generar una invasión en la colonia.
  + Paso *paso* 🡪 Atributo de tipo Paso que va a permitir parar y reanudar el programa.
  + public *Colonia*(Log log, JTextField jTextFieldHormigasBuscandoComida, JTextField jTextFieldHormigasContraInvasor, JTextField jTextFieldHormigasAlmacenComida, JTextField jTextFieldHormiasLlevandoComida, JTextField jTextFieldHormigasHaciendoInstruccion, JTextField jTextFieldUnidadesComidaAlmacen, JTextField jTextFieldUnidadesComidaZonaComer, JTextField jTextFieldHormigasDescansando, JTextField jTextFieldHormigasZonaComer, JTextField jTextFieldHormigasRefugio) 🡪 Método constructor de la clase *Colonia*.
  + public void *entra*(Hormiga hormiga) 🡪 Método para que una hormiga entre a la colonia. Si el túnel de entrada está ocupado, la hormiga esperará hasta que pueda entrar. Ésta será añadida a la lista de hormigas, y en el caso de que haya una invasión activa, las hormigas soldado saldrán a combatir ésta, y las hormigas crías acudirán al refugio.
  + public void *sale*(Hormiga hormiga) 🡪 Método para que una hormiga salga de la colonia. Si el primer túnel de salida está ocupado, la hormiga probará por el segundo túnel.
  + public synchronized void *actualizaEstadoInvasion*(Hormiga hormiga) 🡪 Método para comprobar en que zona está la hormiga (*ZonaComer*, *ZonaInstruccion* o *ZonaDescanso*) para sacarla de ahí.
  + public void *generaInvasion*() 🡪 Método para poder generar una invasión. Este método comprueba que no haya ninguna invasión activa (ya que no puede haber más de una a la vez) y que el refugio esté activo e interrupte a las hormigas soldado (que saldrán a combatir) y a las hormigas crías (que irán al refugio).
  + public Log *getLog*() 🡪 Método get para el atributo *log*.
  + public AlmacenComida *getAlmacenComida*() 🡪 Método get para el atributo *almacenComida*.
  + public ZonaComer *getZonaComer*() 🡪 Método get para el atributo *zonaComer*.
  + public ZonaInstruccion *getZonaInstruccion*() 🡪 Método get para el atributo *zonaInstruccion*.
  + public ZonaDescanso *getZonaDescanso*() 🡪 Método get para el atributo *zonaDescanso*.
  + public Refugio *getRefugio*() 🡪 Método get para el atributo *refugio*.
  + public Invasion *getInvasion*() 🡪 Método get para el atributo *invasion*.
  + public Paso *getPaso*() 🡪 Método get para el atributo *paso*.
  + public ListaThreads *getListaHormigasBuscandoComida*() 🡪 Método get para el atributo *listaHormigasBuscandoComida*.
  + public ListaThreads *getListaHormigasLlevandoComida*() 🡪 Método get para el atributo *listaHormigasLlevandoComida*.
  + public ArrayList<Hormiga> *getListaHormigas*() 🡪 Método get para el atributo *listaHormigas*.
  + public int *getNumHormigasSoldado*() 🡪 Método get para el atributo *numHormigasSoldado*.
  + public void *setNumHormigasSoldado*(int numHormigasSoldado) 🡪 Método set para el atributo *numHormigasSoldado*.
  + public Lock *getEntradaColonia*() 🡪 Método get para el cerrojo *entradaColonia*.
* **Hormiga (Thread)** 🡪 Para la clase *Hormiga* hemos utilizado los siguientes atributos y métodos:
  + String *identificador* 🡪 Atributo de tipo String que indica el identificador de la hormiga.
  + int *numIteraciones* 🡪 Atributo de tipo entero inicializado a 0 que indica el número de iteraciones producidas.
  + String *tipo* 🡪 Atributo de tipo String que indica el tipo de hormiga.
  + Colonia *colonia* 🡪 Atributo de tipo Colonia que crea una colonia.
  + AlmacenComida *almacenComida* 🡪 Atributo de tipo AlmacenComida que crea un almacén de comida en la colonia.
  + ZonaComer *zonaComer* 🡪 Atributo de tipo ZonaComer que crea una zona para comer en la colonia.
  + ZonaInstruccion *zonaInstruccion* 🡪 Atributo de tipo ZonaInstruccion que crea una zona para la instrucción en la colonia.
  + ZonaDescanso *zonaDescanso* 🡪 Atributo de tipo ZonaDescanso que crea una zona de descanso en la colonia.
  + Refugio *refugio* 🡪 Atributo de tipo Refugio que crea un refugio en la colonia.
  + Paso *paso* 🡪 Atributo de tipo Paso que va a permitir parar y reanudar el programa.
  + public *Hormiga*(Colonia colonia, String identificador) 🡪 Método constructor de la clase *Hormiga*, donde se asignará el tipo de hormiga.
  + public String *getIdentificador*() 🡪 Método get para el atributo *identificador*.
  + public int *getNumIteraciones*() 🡪 Método get para el atributo *numIteraciones*.
  + public void *setNumIteraciones*(int numIteraciones) 🡪 Método set para el atributo *numIteraciones*.
  + public String *getTipo*() 🡪 Método get para el atributo *tipo*.
  + public Colonia *getColonia*() 🡪 Método get para el atributo *colonia*.
  + public AlmacenComida *getAlmacenComida*() 🡪 Método get para el atributo *almacenComida*.
  + public ZonaComer *getZonaComer*() 🡪 Método get para el atributo *zonaComer*.
  + public ZonaInstruccion *getZonaInstruccion*() 🡪 Método get para el atributo *zonaInstruccion*.
  + public ZonaDescanso *getZonaDescanso*() 🡪 Método get para el atributo *zonaDescanso*.
  + public Refugio *getRefugio*() 🡪 Método get para el atributo *refugio*.
  + public Paso *getPaso*() 🡪 Método get para el atributo *paso*.
* **HormigaCria** 🡪 Para la clase *HormigaCria* hemos utilizado los siguientes métodos, ya que los atributos son heredados de la clase *Hormiga*:
  + public *HormigaCria*(Colonia colonia, String identificador) 🡪 Método constructor de la clase *HormigaCria*.
  + public void *run*() 🡪 Método run del hilo, donde se llama al método *rutina* y se gestiona el refugio en caso de invasión.
  + private void *rutina*() throws InterruptedException 🡪 Método de la rutina que seguirá la hormiga cría, que consiste en comer y descansar.
* **HormigaObrera** 🡪 Para la clase *HormigaObrera* hemos utilizado el siguiente atributo y métodos, ya que el resto de los atributos son heredados de la clase *Hormiga:*
  + int *numIdentificador* 🡪 Atributo de tipo entero utilizado para dividir las hormigas en pares e impares.
  + public *HormigaObrera*(Colonia colonia, String identificador, int numIdentificador) 🡪 Método constructor de la clase *HormigaObrera*.
  + public void *run*() 🡪 Método run del hilo, donde se llaman a los métodos *rutinaHormigaImpar* o *rutinaHormigaPar* y si el número de iteraciones es superior a 10, se reinicia y la hormiga procede a descansar.
  + private void *rutinaHormigaPar*() throws InterruptedException 🡪Método de la rutina que seguirá la hormiga par, que consiste en recoger comida en el almacén y transportarlo a la zona de comer.
  + private void *rutinaHormigaImpar*() throws InterruptedException 🡪 Método de la rutina que seguirá la hormiga impar, que consiste en salir al exterior a buscar comida para depositar ésta en el almacén.
  + private void *rutinaDescanso*() throws InterruptedException 🡪 Método que usarán las hormigas obreras una vez reiniciadas las iteraciones, donde irán a la zona de comer y tomarán un elemento de comida para posteriormente descansar.
  + public int *getNumIdentificador*() 🡪 Método get para el atributo *numIdentificador*.
* **HormigaSoldado** 🡪 Para la clase *HormigaSoldado* hemos utilizado los siguientes métodos, ya que los atributos son heredados de la clase *Hormiga*:
  + public *HormigaSoldado*(Colonia colonia, String identificador) 🡪 Método constructor de la clase *HormigaSoldado*.
  + public void *run*() 🡪 Método run del hilo, donde se llama al método *rutinaHormigaSoldado*, y una vez que ésta haya hecho 6 iteraciones, se llama al método *rutinaDescanso*. Además, lanza una excepción si se produce una invasión, saliendo las hormigas de la colonia y entrando de vuelta una vez acabada ésta.
  + private void *rutinaHormigaSoldada*() throws InterruptedException 🡪 Método de la rutina que seguirá la hormiga soldado, que consiste en hacer su instrucción y luego descansar.
  + private void *rutinaDescanso*() throws InterruptedException 🡪 Método que usarán las hormigas soldado una vez reiniciadas las iteraciones, donde irán a la zona de comer y tomarán un elemento de comida.
* **Invasion** 🡪 Para la clase *Invasion* hemos utilizado los siguientes atributos y métodos:
  + Log *log* 🡪 Atributo de tipo log que almacena el log del sistema concurrente.
  + int *numHormigasSoldado* 🡪 Atributo de tipo entero inicializado a 0 que indica el número de hormigas soldado que hay en la colonia.
  + boolean *activa* 🡪 Atributo de tipo booleano inicializado como falso que indica si en determinado momento hay una invasión activa.
  + boolean *enCurso* 🡪 Atributo de tipo booleano inicializado como falso que indica si hay una invasión en curso.
  + Lock *cerrojoInvasion*
  + Condition *conditionInvasion*
  + ListaThreads *listaHormigasInvasion* 🡪 Atributo de tipo ListaThreads que contiene a las hormigas que están repeliendo la invasión.
  + public *Invasion*(Log log, JTextField jTextFieldHormigasContraInvasor) 🡪 Método constructor de la clase *Invasion*.
  + public void *realizaInvasion*(Hormiga hormiga) 🡪 Método para que las hormigas combatan al insecto invasor.
  + private void *entra*(Hormiga hormiga) 🡪 Método para que las hormigas entren a combatir al insecto invasor.
  + private boolean *activarInvasionEnCurso*(Hormiga hormiga) 🡪 Método que comprueba si la invasión pudiera estar en curso o no.
  + private void *sale*(Hormiga hormiga) 🡪 Método para que las hormigas salgan tras combatir al insecto invasor.
  + private int *cuentaSoldadasColonia*(ArrayList<Hormiga> lista) 🡪 Método para contar el número de hormigas soldado que hay en la colonia.
  + public Log *getLog*() 🡪 Método get para el atributo *log*.
  + public int *getNumHormigasSoldado*() 🡪 Método get para el atributo *numHormigasSoldado*.
  + public void *setNumHormigasSoldado*(int numHormigasSoldado) 🡪 Método set para el atributo *numHormigasSoldado*.
  + public synchronized boolean *isActiva*() 🡪 Método get para el atributo *activa*.
  + public void *setActiva*(boolean activa) 🡪 Método set para el atributo *activa*.
  + public boolean *isEnCurso*() 🡪 Método get para el atributo *enCurso*.
  + public void *setEnCurso*(boolean enCurso) 🡪 Método set para el atributo *enCurso*.
  + public ListaThreads *getListaHormigasInvasion*() 🡪 Método get para el atributo *listaHormigasInvasion*.
* **ListaThreads** 🡪 Para la clase *ListaThreads* hemos utilizado los siguientes atributos y métodos (esta clase ha sido extraída de las sesiones de laboratorio de la asignatura):
  + ArrayList<Hormiga> *listaHormigas* 🡪 Atributo de tipo lista que almacenará a las hormigas.
  + JTextField *tf* 🡪 Atributo de tipo JTextField el cual mostrará el contenido modificado en *listaHormigas*.
  + public *ListaThreads*(JTextField tf) 🡪 Método constructor de la clase *ListaThreads*.
  + public synchronized void *meterHormiga*(Hormiga hormiga) 🡪 Método para añadir a una hormiga a *listaHormigas*.
  + public synchronized void *sacarHormiga*(Hormiga hormiga) 🡪 Método para eliminar a una hormiga de *listaHormigas*.
  + public synchronized void *insertarNumero*(Integer num) 🡪 Método para poder añadir números a *listaHormigas*.
  + private void *imprimirHormiga*() 🡪 Método para mostrar hormigas en *tf*.
  + public ArrayList<Hormiga> *getListaHormigas*() 🡪 Método get para el atributo *listaHormigas*.
* **Log** 🡪 Para la clase *Log* hemos utilizado los siguientes atributos y métodos:
  + File *file* 🡪 Atributo de tipo File para poder guardar el historial.
  + boolean *debug* 🡪 Atributo de tipo booleano para poder escribir.
  + public *Log*(boolean debug) 🡪 Método constructor de la clase *Log*.
  + public void *crearLog*() 🡪 Método para crear el archivo log. Si éste ya existe, será reemplazado por el nuevo.
  + public synchronized void *escribirEnLog*(String contenido) 🡪 Método empleado para escribir en log. El atributo *debug* deberá ser true para poder escribir.
* **Paso** 🡪 Para la clase *Paso* hemos utilizado los siguientes atributos y métodos métodos (esta clase ha sido extraída de las sesiones de laboratorio de la asignatura):
  + boolean *cerrado* 🡪 Atributo de tipo booleano inicializado como false, empleado para poder detener el programa.
  + Lock *cerrojo*
  + Condition *parar*
  + public *Paso*() 🡪 Método constructor de la clase *Paso*.
  + public void *mirar*() throws InterruptedException 🡪 Método para comprobar si hay que detener o no el programa.
  + public void *abrir*() 🡪 Método para reanudar la ejecución de los hilos.
  + public void *cerrar*() 🡪 Método para detener la ejecución de los hilos.
  + public boolean *isCerrado*() 🡪 Método get para el atributo *cerrado*.
  + public void *setCerrado*(boolean cerrado) 🡪 Método set para el atributo *cerrado*.
* **ProgPrincipal** 🡪 Para la clase *ProgPrincipal* hemos utilizado los siguientes atributos y métodos:
  + int *numTotalHormigas* = 0 🡪 Atributo de tipo entero inicializado a 0 que almacenará el número total de hormigas.
  + int *numHormigasObreras* = 0 🡪 Atributo de tipo entero inicializado a 0 que almacenará el número de hormigas obreras.
  + int *numHormigasSoldado* = 0 🡪 Atributo de tipo entero inicializado a 0 que almacenará el número de hormigas soldado.
  + int *numHormigasCria* = 0 🡪 Atributo de tipo entero inicializado a 0 que almacenará el número de hormigas cría.
  + Log *log* 🡪 Atributo de tipo log que almacena el log del sistema concurrente.
  + Colonia *colonia* 🡪 Atributo de tipo Colonia que crea una colonia.
  + boolean *botonPausarPulsado* 🡪 Atributo de tipo booleano inicializado como false, empleado para poder detener y reanudar el programa.
  + public *ProgPrincipal*() 🡪 Método constructor de la clase *ProgPrincipal*, donde se crea un log y una colonia.
  + public void *crearSistema*() 🡪 Método para crear el sistema concurrente, donde se crean los hilos
  + public void *creaObjetoRemoto*() 🡪 Método para crear el objeto remoto
  + private void *jButtonGenerarAmenazaInsectoInvasorActionPerformed*(java.awt.event.ActionEvent evt) 🡪 Método para generar una invasión
  + private void *jButtonPausarReanudarActionPerformed*(java.awt.event.ActionEvent evt) 🡪 Método para pausar o reanudar el programa.
  + public static void *main*(String args[]) 🡪 Método main del programa, donde se crea un *ProgPrincipal* y se ejecuta.
  + public JTextField getjTextFieldHormiasLlevandoComida() 🡪 Método get para el *JTextField* de las hormigas que llevan comida.
  + public JTextField *getjTextFieldHormigasAlmacenComida*() 🡪 Método get para el *JTextField* de las hormigas que están en el almacén.
  + public JTextField *getjTextFieldHormigasBuscandoComida*() 🡪 Método get para el *JTextField* de las hormigas que están buscando comida.
  + public JTextField *getjTextFieldHormigasContraInvasor*() 🡪 Método get para el *JTextField* de las hormigas que luchan contra el insecto invasor.
  + public JTextField *getjTextFieldHormigasDescansando*() 🡪 Método get para el *JTextField* de las hormigas que están descansando.
  + public JTextField *getjTextFieldHormigasHaciendoInstruccion*() 🡪 Método get para el *JTextField* de las hormigas que hacen instrucción.
  + public JTextField *getjTextFieldHormigasRefugio*() 🡪 Método get para el *JTextField* de las hormigas que están en el refugio.
  + public JTextField *getjTextFieldUnidadesComidaAlmacen*() 🡪 Método get para el *JTextField* de las unidades de comida que hay en el almacén.
  + public JTextField *getjTextFieldUnidadesComidaZonaComer*() 🡪 Método get para el *JTextField* de las unidades de comida que hay en la zona para comer.
  + public JTextField *getjTextFieldHormigasZonaComer*() 🡪 Método get para el *JTextField* de las hormigas que hay en la zona para comer.
  + public JButton *getBotonPausarReanudar*() 🡪 Método get para el *JButton* para pausar o reanudar.
  + public JButton *getBotonGenerarInvasion*() 🡪 Método get para el *JButton* para generar una invasión.
  + public Log *getLog*() 🡪 Método get para el atributo *log*.
  + public Colonia *getColonia*() 🡪 Método get para el atributo *colonia*.
  + public int *getNumTotalHormigas*() 🡪 Método get para el atributo *numTotalHormigas*.
  + public void *setNumTotalHormigas*(int numTotalHormigas) 🡪 Método set para el atributo *numTotalHormigas*.
  + public int *getNumHormigasObreras*() 🡪 Método get para el atributo *numHormigasObreras*.
  + public void *setNumHormigasObreras*(int numHormigasObreras) 🡪 Método set para el atributo *numHormigasObreras*.
  + public int *getNumHormigasSoldado*() 🡪 Método get para el atributo *numHormigasSoldado*.
  + public void *setNumHormigasSoldado*(int numHormigasSoldado) 🡪 Método set para el atributo *numHormigasSoldado*.
  + public int *getNumHormigasCria*() 🡪 Método get para el atributo *numHormigasCria*.
  + public void *setNumHormigasCria*(int numHormigasCria) 🡪 Método set para el atributo *numHormigasCria*.
  + public void *setBotonPausarPulsado*(boolean botonPausarPulsado) 🡪 Método set para el atributo *botonPausarPulsado*.
  + public boolean *isBotonPausarPulsado*() 🡪 Método get para el atributo *botonPausarPulsado*.
* **Refugio** 🡪 Para la clase *Refugio* hemos utilizado los siguientes atributos y métodos:
  + Log *log* 🡪 Atributo de tipo log que almacena el log del sistema concurrente.
  + boolean *activo* 🡪 Atributo de tipo boolean inicializado como false, que indica si hay activa una invasión.
  + int *numHormigasRefugio* 🡪 Atributo de tipo entero inicializado a 0 que indica cuantas hormigas hay dentro del refugio.
  + Lock *cerrojoRefugio*
  + Condition *finInvasion*
  + ListaThreads *listaHormigasRefugio* 🡪 Atributo de tipo ListaThreads para poder almacenar las hormigas que están en el refugio.
  + public *Refugio*(Log log, JTextField jTextFieldHormigasRefugio) 🡪 Método constructor de la clase *Refugio*.
  + public void *protegeRefugio*(Hormiga hormiga) 🡪 Método para que, en caso de que haya activa una invasión, las hormigas cría usen el refugio.
  + private void *entra*(Hormiga hormiga) 🡪 Método para que las hormigas entren en el refugio.
  + private void *sale*(Hormiga hormiga) 🡪 Método para que las hormigas salgan del refugio.
  + public void *despiertaRefugio*() 🡪 Método para despertar a las hormigas que se encuentren en el refugio tras una invasión.
  + public Log *getLog*() 🡪 Método get para el atributo *log*.
  + public ListaThreads *getListaHormigasRefugio*() 🡪 Método get para el atributo *listaHormigasRefugio*.
  + public boolean *isActivo*() 🡪 Método get para el atributo *activo*.
  + public void *setActivo*(boolean activo) 🡪 Método set para el atributo *activo*.
  + public int *getNumHormigasRefugio*() 🡪 Método get para el atributo *numHormigasRefugio*.
  + public void *setNumHormigasRefugio*(int numHormigasRefugio) 🡪 Método set para el atributo *numHormigasRefugio*.
* **ZonaComer** 🡪 Para la clase *ZonaComer* hemos utilizado los siguientes atributos y métodos:
  + Log *log* 🡪 Atributo de tipo log que almacena el log del sistema concurrente.
  + Semaphore *semaforoEntradaSalida*
  + Lock *cerrojoElementoComida*
  + Condition *esperaElementoComida*
  + int *numElementosComida* 🡪 Atributo de tipo entero inicializado a 0 que indica el número de elementos de comida que hay en la zona para comer.
  + int *numHormigasZonaComer* 🡪 Atributo de tipo entero inicializado a 0 que indica el número de hormigas que hay en la zona para comer.
  + ListaThreads *unidadesElementosComida* 🡪 Atributo de tipo ListaThreads para mostrar el número de elementos de comida que hay en la zona para comer.
  + ListaThreads *listaHormigasZonaComer* 🡪 Atributo de tipo ListaThreads para mostrar las hormigas que hay en la zona para comer.
  + public *ZonaComer*(Log log, JTextField jTextFieldUnidadesComidaZonaComer, JTextField jTextFieldHormigasZonaComer) 🡪 Método constructor de la clase *ZonaComer*.
  + public void *entra*(Hormiga hormiga) 🡪 Método para que una hormiga entre a la zona para comer.
  + public void *sale*(Hormiga hormiga) 🡪 Método para que una hormiga salga de la zona para comer.
  + public void *depositaElementoComida*(Hormiga hormiga) 🡪 Método para que una hormiga deposite los 5 elementos de comida en la zona para comer.
  + public void *cogeElementoComida*(Hormiga hormiga) throws InterruptedException 🡪 Método para que una hormiga coja un elemento de comida, y en el caso de que no haya, ésta espere a que hayan repuesto.
  + public Log *getLog*() 🡪 Método get para el atributo *log*.
  + public int *getNumElementosComida*() 🡪 Método get para el atributo *numElementosComida*.
  + public void *setNumElementosComida*(int numElementosComida) 🡪 Método set para el atributo *numElementosComida*.
  + public int *getNumHormigasZonaComer*() 🡪 Método get para el atributo *numHormigasZonaComer*.
  + public void *setNumHormigasZonaComer*(int numHormigasZonaComer) 🡪 Método set para el atributo *numHormigasZonaComer*.
  + public ListaThreads *getUnidadesElementosComida*() 🡪 Método get para el atributo *unidadesElementosComida*.
  + public ListaThreads *getListaHormigasZonaComer*() 🡪 Método get para el atributo *listaHormigasZonaComer*.
* **ZonaDescanso** 🡪 Para la clase *ZonaDescanso* hemos utilizado los siguientes atributos y métodos:
  + Log *log* 🡪 Atributo de tipo log que almacena el log del sistema concurrente.
  + Lock *entradaSalida*
  + ListaThreads *listaHormigasDescansando* 🡪 Atributo de tipo ListaThreads para mostrar las hormigas que hay en la zona para descansar.
  + public *ZonaDescanso*(Log log, JTextField jTextFieldHormigasDescansando) 🡪 Método constructor de la clase *ZonaDescanso*.
  + public void *entra*(Hormiga hormiga) 🡪 Método para que una hormiga entre a la zona para descansar.
  + public void *sale*(Hormiga hormiga) 🡪 Método para que una hormiga salga de la zona para descansar.
  + public void *realizaDescanso*(Hormiga hormiga) throws InterruptedException 🡪 Método para que una hormiga haga el descanso, variando el tiempo de éste en función del tipo de hormiga.
  + public Log *getLog*() 🡪 Método get para el atributo *log*.
  + public ListaThreads *getListaHormigasDescansando*() 🡪 Método get para el atributo *listaHormigasDescansando*.
* **ZonaInstruccion** 🡪 Para la clase *ZonaInstruccion* hemos utilizado los siguientes atributos y métodos:
  + Log *log* 🡪 Atributo de tipo log que almacena el log del sistema concurrente.
  + Lock *entradaSalida*
  + ListaThreads *listaHormigasHaciendoInstruccion* 🡪 Atributo de tipo ListaThreads para mostrar las hormigas que están haciendo instrucción.
  + public *ZonaInstruccion*(Log log, JTextField jTextFieldHormigasHaciendoInstruccion) 🡪 Método constructor de la clase *ZonaInstruccion*.
  + public void *entra*(Hormiga hormiga) 🡪 Método para que una hormiga entre a la zona para hacer instrucción.
  + public void *sale*(Hormiga hormiga) 🡪 Método para que una hormiga salga de la zona para hacer instrucción.
  + public void *realizaInstruccion*(Hormiga hormiga) throws InterruptedException 🡪 Método para que una hormiga realice una instrucción.
  + public Log *getLog*() 🡪 Método get para el atributo *log*.
  + public ListaThreads *getListaHormigasHaciendoInstruccion*() 🡪 Método get para el atributo *listaHormigasHaciendoInstruccion*.

Parte distribuida: En la parte concurrente hay 3 clases, las cuales son *ClienteDeColonia*, *GestorInterfaz* y *InterfazColonia*.

* **ClienteDeColonia** 🡪 Para la clase *ClienteDeColonia* hemos utilizado los siguientes métodos:
  + public *ClienteDeColonia*() 🡪 Método constructor de la clase *ClienteDeColonia*.
  + public void *actualizaJTextFields* 🡪 Método para crear la interfaz remota y los JTextField de la interfaz.
  + private void *jButtonGeneraInvasionActionPerformed*(java.awt.event.ActionEvent evt) 🡪 Método para el JButton de generar invasión.
  + public static void *main*(String args[]) 🡪 Método main del programa, donde se crea un *ClienteDeColonia* y se ejecuta.
  + public JTextField *getjTextFieldNumHomrigasCriaRefugio*() 🡪 Método get para el *JTextField* del número de hormigas cría que hay en el refugio.
  + public JTextField *getjTextFieldNumHomrigasCriaZonaComer*() 🡪 Método get para el JTextField del número de hormigas cría que hay en la zona para comer.
  + public JTextField *getjTextFieldNumHormigasObrerasExteriorColonia*() 🡪 Método get para el JTextField del número de hormigas obreras que hay en el exterior de la colonia.
  + public JTextField *getjTextFieldNumHormigasObrerasInteriorColonia*() 🡪 Método get para el JTextField del número de hormigas obreras que hay en el interior de la colonia.
  + public JTextField *getjTextFieldNumHormigasSoldadoInstruccion*() 🡪 Método get para el JTextField del número de hormigas soldado que hay haciendo instrucción.
  + public JTextField *getjTextFieldNumHormigasSoldadoInvasion*() 🡪 Método get para el JTextField del número de hormigas soldado que hay repeliendo al insecto invasor.
* **GestorInterfaz** 🡪 Para la clase *GestorInterfaz* hemos utilizado el siguiente atributo y los siguientes métodos:
  + Colonia *colonia* 🡪 Atributo de tipo Colonia que crea una colonia.
  + public *GestorInterfaz*(Colonia colonia) throws RemoteException 🡪 Método constructor de la clase *GestorInterfaz*.
  + public String *numHormigasObrerasExteriorColonia*() throws RemoteException 🡪 Método que devuelve el número de hormigas obreras que hay en el exterior de la colonia.
  + public String *numHormigasObrerasInteriorColonia*() throws RemoteException 🡪 Método que devuelve el número de hormigas obreras que hay en el exterior de la colonia.
  + public String *numHormigasSoldadoInstruccion*() throws RemoteException 🡪 Método que devuelve el número de hormigas soldado que hay haciendo instrucción.
  + public String *numHormigasSoldadoInvasion*() throws RemoteException 🡪 Método que devuelve el número de hormigas soldado que hay repeliendo al insecto invasor.
  + public String *numHormigasCriaZonaComer*() throws RemoteException 🡪 Método que devuelve el número de hormigas cría que hay en la zona para comer.
  + public String *numHormigasCriaRefugio*() throws RemoteException 🡪 Método que devuelve el número de hormigas cría que hay en el refugio.
  + public void *generaInvasion*() throws RemoteException 🡪 Método para poder generar una invasión.
  + public Colonia *getColonia*() 🡪 Método get para el atributo *colonia*.
  + private String *cuentaHormigasArraylist*(ArrayList<Hormiga> lista) 🡪 Método para contar el número de hormigas.
  + private String *cuentaHormigasTipoArraylist*(ArrayList<Hormiga> lista, String tipo) 🡪 Método para contar el número de hormigas en función del tipo.
* **InterfazColonia** 🡪 La clase *InterfazColonia* define todos los métodos que puede ejecutar el *RMI*, que son los anteriormente mencionados en la clase *GestorInterfaz*.

# Diagrama de clases



# Código fuente

Clase AlmacenComida:

*package* concurrencia;  
  
*import* javax.swing.\*;  
*import* java.util.concurrent.Semaphore;  
*import* java.util.concurrent.locks.Condition;  
*import* java.util.concurrent.locks.Lock;  
*import* java.util.concurrent.locks.ReentrantLock;  
  
*public class* AlmacenComida {  
 *//Atributos de AlmacenComida  
 private final* Log log;  
 *private int* numElementosComida = 0;  
 *private int* numHormigasDentro = 0;  
 *private int* numHormigasEsperando = 0;  
 *private final* Semaphore semaforoEntradaSalida = *new* Semaphore(10);  
 *private final* Lock recogeElemento = *new* ReentrantLock();  
 *private final* Lock cerrojoNumElementosComida = *new* ReentrantLock();  
 *private final* Lock cerrojoHormigaEsperando = *new* ReentrantLock();  
 *private final* Condition conditionElementoComida = recogeElemento.newCondition();  
 *private final* ListaThreads unidadesElementosComida;  
 *private final* ListaThreads listaHormigasAlmacenComida;  
  
 *//Métodos de la clase AlmacenComida  
  
 //Método constructor  
 public* AlmacenComida(Log log, JTextField jTextFieldUnidadesComidaAlmacen, JTextField jTextFieldHormigasAlmacenComida){  
 *this*.unidadesElementosComida = *new* ListaThreads(jTextFieldUnidadesComidaAlmacen);  
 *this*.listaHormigasAlmacenComida = *new* ListaThreads(jTextFieldHormigasAlmacenComida);  
 *this*.log = log;  
 }  
  
 *//Método para entrar al almacen de comida  
 public void* entra(HormigaObrera hormiga){  
 recogeElemento.lock();  
 *if*(hormiga.getNumIdentificador() % 2 == 0){  
 *if*((getNumHormigasEsperando() > 0) || (getNumElementosComida() <= 0)){  
 *try*{  
 setNumHormigasEsperando(getNumHormigasEsperando() + 1);  
 conditionElementoComida.await();  
 }*catch*(InterruptedException ignored){}  
 }  
 }  
 recogeElemento.unlock();  
 *try*{  
 semaforoEntradaSalida.acquire();  
 incrementaNumHormigasDentro();  
 getListaHormigasAlmacenComida().meterHormiga(hormiga);  
 getLog().escribirEnLog("[ALMACEN COMIDA} --> La hormiga " + hormiga.getIdentificador() + " ha entrado al almacen de comida");  
 }*catch*(InterruptedException ignored){}  
 }  
  
 *//Método para salir del almacen de comida  
 public void* sale(HormigaObrera hormiga){  
 decrementaNumHormigasDentro();  
 getListaHormigasAlmacenComida().sacarHormiga(hormiga);  
 getLog().escribirEnLog("[ALMACEN COMIDA} --> La hormiga " + hormiga.getIdentificador() + " ha salido del almacen de comida");  
 semaforoEntradaSalida.release();  
 }  
  
 *//Método para que una hormiga deposite un elemento de comida  
 public void* depositaElementoComida(HormigaObrera hormiga){  
 recogeElemento.lock();  
 *try*{  
 setNumElementosComida(getNumElementosComida() + 5); *//Primero depositamos 5 elementos de comida* getLog().escribirEnLog("[ALMACEN COMIDA] --> La hormiga " + hormiga.getIdentificador() + " ha depositado 5 elementos de comida en el almacen de comida");  
 *if*(getNumHormigasEsperando() > 0){ *//Mirar si hay hormigas esperando elementos de comida  
 //Verificamos que hay hormigas esperando* conditionElementoComida.signal(); *//Despierta a la hormiga porque ya hay un elemento de comida presente* setNumHormigasEsperando(getNumHormigasEsperando() - 1);  
 }  
 *//Si no hay hormigas esperando, no tiene sentido dar signal, ya que no despertaría a nadie* }  
 *finally*{  
 recogeElemento.unlock();  
 }  
 }  
  
 *//Método para que una hormiga recoja un elemento de comida  
 public void* recogeElementoComida(HormigaObrera hormiga){  
 recogeElemento.lock();  
 *try*{  
 setNumElementosComida(getNumElementosComida() - 5); *//Recogemos un elemento de comida* getLog().escribirEnLog("[ALMACEN COMIDA] --> La hormiga " + hormiga.getIdentificador() + " ha recogido 5 elementos de comida del almacen de comida");  
 }  
 *finally* {  
 recogeElemento.unlock();  
 }  
 }  
  
  
  
 *//Métodos get y set  
  
 public void* incrementaNumHormigasDentro(){  
 numHormigasDentro = numHormigasDentro + 1;  
 }  
  
 *public void* decrementaNumHormigasDentro(){  
 numHormigasDentro = numHormigasDentro - 1;  
 }  
  
 *//Variables para realizar lectura y escritua en el numero de elementos de comida en el almacén  
 public int* getNumElementosComida(){  
 cerrojoNumElementosComida.lock();  
 *try*{  
 *return* numElementosComida;  
 }  
 *finally*{  
 cerrojoNumElementosComida.unlock();  
 }  
 } *//Lectura  
 public void* setNumElementosComida(*int* numElementosComida){  
 cerrojoNumElementosComida.lock();  
 *try*{  
 *this*.numElementosComida = numElementosComida;  
 getUnidadesElementosComida().insertarNumero(getNumElementosComida());  
 }  
 *finally*{  
 cerrojoNumElementosComida.unlock();  
 }  
  
 } *//Escritura  
  
 //Métodos para realizar lectura y escritura sobre el numero de hormigas esperando  
 public int* getNumHormigasEsperando(){  
 cerrojoHormigaEsperando.lock();  
 *try*{  
 *return* numHormigasEsperando;  
 }  
 *finally*{  
 cerrojoHormigaEsperando.unlock();  
 }  
  
 } *//Lectura  
 public void* setNumHormigasEsperando(*int* numHormigasEsperando){  
 cerrojoHormigaEsperando.lock();  
 *try*{  
 *this*.numHormigasEsperando = numHormigasEsperando;  
 }  
 *finally*{  
 cerrojoHormigaEsperando.unlock();  
 }  
 } *//Escritura  
  
 //Métodos get de los ListaThreads  
 public* ListaThreads getUnidadesElementosComida() {  
 *return* unidadesElementosComida;  
 }  
 *public* ListaThreads getListaHormigasAlmacenComida() {  
 *return* listaHormigasAlmacenComida;  
 }  
  
 *//Método get del log  
 public* Log getLog() {  
 *return* log;  
 }  
}

Clase Colonia:

*package* concurrencia;  
  
*import* javax.swing.\*;  
*import* java.util.ArrayList;  
*import* java.util.concurrent.locks.Lock;  
*import* java.util.concurrent.locks.ReentrantLock;  
  
*public class* Colonia { *//Recurso compartido por todos los hilos  
 //Atributos de la clase Colonia  
 private int* numHormigasSoldado = 0;  
 *private final* Log log; *//Log del sistema concurrente  
 private final* Lock entradaColonia = *new* ReentrantLock(); *//Lock del tunel para entrar a la colonia  
 private final* Lock salidaColonia1 = *new* ReentrantLock(); *//Lock del primer tunel de salida de la colonia  
 private final* Lock salidaColonia2 = *new* ReentrantLock(); *//Lock del segundo tunel de salida de la colonia  
 private final* ArrayList<Hormiga> listaHormigas = *new* ArrayList<>(); *//Lista que contiene todas las hormigas que han entrado a la colonia  
 private final* ListaThreads listaHormigasBuscandoComida; *//ListaThreads para manejar el JTextField de hormigas buscando comida  
 private final* ListaThreads listaHormigasLlevandoComida; *//ListaThreads para manejar el JTextField de hormigas llevando comida  
 private final* AlmacenComida almacenComida; *//Almacen de comida de la colonia  
 private final* ZonaComer zonaComer; *//Zona para comer de la colonia  
 private final* ZonaInstruccion zonaInstruccion; *//Zona de instruccion de la colonia  
 private final* ZonaDescanso zonaDescanso; *//Zona de descanso de la colonia  
 private final* Refugio refugio;  
 *private final* Invasion invasion;  
 *private final* Paso paso;  
  
 *//Métodos de la clase colonia  
  
 //Método constructor  
 public* Colonia(Log log, JTextField jTextFieldHormigasBuscandoComida, JTextField jTextFieldHormigasContraInvasor,  
 JTextField jTextFieldHormigasAlmacenComida, JTextField jTextFieldHormiasLlevandoComida,  
 JTextField jTextFieldHormigasHaciendoInstruccion, JTextField jTextFieldUnidadesComidaAlmacen,  
 JTextField jTextFieldUnidadesComidaZonaComer, JTextField jTextFieldHormigasDescansando,  
 JTextField jTextFieldHormigasZonaComer, JTextField jTextFieldHormigasRefugio){  
 *this*.log = log;  
 *this*.listaHormigasBuscandoComida = *new* ListaThreads(jTextFieldHormigasBuscandoComida);  
 *this*.listaHormigasLlevandoComida = *new* ListaThreads(jTextFieldHormiasLlevandoComida);  
 *//Crear aqui todas las zonas, para luego en distribuida pasar un solo objeto que tenga toda la parte concurrente  
 this*.almacenComida = *new* AlmacenComida(log, jTextFieldUnidadesComidaAlmacen, jTextFieldHormigasAlmacenComida);  
 *this*.zonaComer = *new* ZonaComer(log, jTextFieldUnidadesComidaZonaComer, jTextFieldHormigasZonaComer);  
 *this*.zonaInstruccion = *new* ZonaInstruccion(log, jTextFieldHormigasHaciendoInstruccion);  
 *this*.zonaDescanso = *new* ZonaDescanso(log, jTextFieldHormigasDescansando);  
 *this*.refugio = *new* Refugio(log, jTextFieldHormigasRefugio);  
 *this*.invasion = *new* Invasion(log, jTextFieldHormigasContraInvasor);  
 *this*.paso = *new* Paso();  
  
 }  
  
 *//Método para entrar a la colonia  
 public void* entra(Hormiga hormiga){  
 entradaColonia.lock();  
 *try*{  
 Thread.sleep(100);  
 }*catch*(InterruptedException ignored){}  
 *if*(!getListaHormigas().contains(hormiga)){  
 getListaHormigas().add(hormiga); *//A todas las hormigas que entran les añadimos a un arraylist* }  
 getLog().escribirEnLog("[COLONIA] --> La hormiga " + hormiga.getIdentificador() + " ha entrado a la colonia");  
 *if*(hormiga.getTipo().equals("Soldada")){  
 *try*{  
 hormiga.getPaso().mirar();  
 }*catch* (InterruptedException ignored){}  
 setNumHormigasSoldado(getNumHormigasSoldado() + 1);  
 entradaColonia.unlock();  
 getInvasion().realizaInvasion(hormiga); *//Si la invasion no está activa este método serña inutil* }  
 *else if*(hormiga.getTipo().equals("Cria")){  
 entradaColonia.unlock();  
 getRefugio().protegeRefugio(hormiga); *//Si no está activo, esta función será inutil* }  
 *else*{  
 entradaColonia.unlock();  
 }  
 }  
  
 *//Método para que las hormigas obreras salgan de la colonia  
 public void* sale(Hormiga hormiga){  
 *if*(salidaColonia1.tryLock()){  
 *try*{  
 *//Se verifica que el tunel de salida 1 está libre, por tanto puede salir por ese tunel  
 try*{  
 Thread.sleep(100);  
 }*catch*(InterruptedException ignored){}  
 getLog().escribirEnLog("[COLONIA] --> La hormiga " + hormiga.getIdentificador() + " ha salido de la colonia por el tunel de salida 1");  
  
 }*finally*{  
 salidaColonia1.unlock();  
 }  
 }  
 *else*{  
 *//Como el tunel de salida 1 está ocupado en ese momento, la hormiga intentará salir por el tunel de salida 2* salidaColonia2.lock();  
 *try*{  
 Thread.sleep(100);  
 getLog().escribirEnLog("[COLONIA] --> La hormiga " + hormiga.getIdentificador() + " ha salido de la colonia por el tunel de salida 2");  
 }  
 *catch* (InterruptedException ignored) {}  
  
 *finally*{  
 salidaColonia2.unlock();  
 }  
 }  
 }  
  
 *//Método auxiliar a la invasion  
 public synchronized void* actualizaEstadoInvasion(Hormiga hormiga){  
 *try*{  
 *//Tenemos que ver en que zona está la hormiga, que pueden estar en ZonaComer, ZonaInstruccion o ZonaDescanso  
 //ZonaComer  
 if*(getZonaComer().getListaHormigasZonaComer().getListaHormigas().contains(hormiga)){  
 *//Está en ZonaComer, por tanto deberán salir del almacen* getZonaComer().sale(hormiga);  
 }  
 *else if*(getZonaInstruccion().getListaHormigasHaciendoInstruccion().getListaHormigas().contains(hormiga)){  
 *//Está haciendo instruccion, por tanto se va de ZonaInstruccion* getZonaInstruccion().sale(hormiga);  
 }  
 *else if*(getZonaDescanso().getListaHormigasDescansando().getListaHormigas().contains(hormiga)){  
 *//Está haciendo un descanso, por tanto saldrán del descanso* getZonaDescanso().sale(hormiga);  
 }  
 }  
 *catch*(InterruptedException ignored){}  
 }  
  
 *//Método para generar una invasion  
 public void* generaInvasion(){  
 *if* (!getInvasion().isActiva() && !getRefugio().isActivo() && !getPaso().isCerrado()){  
 getInvasion().setActiva(*true*);  
 getRefugio().setActivo(*true*);  
 getLog().escribirEnLog("[INVASION] --> Se ha generado una invasion");  
 *//Una vez activada darmeos interrupt a todas las hormiga soldado y crias que esten presentes en la colonia  
 for*(*int* i = 0; i < getListaHormigas().size() ; i++){  
 Hormiga hormigaActual = getListaHormigas().get(i);  
 String tipoHormiga = hormigaActual.getTipo();  
 *if* (tipoHormiga.equals("Soldada") || tipoHormiga.equals("Cria")){  
 hormigaActual.interrupt();  
 }  
 }  
 }  
 }  
  
 *//Métodos get y set  
 public* Log getLog(){  
 *return this*.log;  
 }  
 *public* AlmacenComida getAlmacenComida(){  
 *return this*.almacenComida;  
 }  
 *public* ZonaComer getZonaComer(){  
 *return this*.zonaComer;  
 }  
 *public* ZonaInstruccion getZonaInstruccion(){  
 *return this*.zonaInstruccion;  
 }  
 *public* ZonaDescanso getZonaDescanso(){  
 *return this*.zonaDescanso;  
 }  
 *public* Refugio getRefugio(){  
 *return this*.refugio;  
 }  
 *public* Invasion getInvasion(){  
 *return this*.invasion;  
 }  
 *public* Paso getPaso() {  
 *return* paso;  
 }  
  
 *public* ListaThreads getListaHormigasBuscandoComida() {  
 *return* listaHormigasBuscandoComida;  
 }  
 *public* ListaThreads getListaHormigasLlevandoComida() {  
 *return* listaHormigasLlevandoComida;  
 }  
 *public* ArrayList<Hormiga> getListaHormigas() {  
 *return* listaHormigas;  
 }  
  
 *public int* getNumHormigasSoldado() {  
 *return* numHormigasSoldado;  
 }  
 *public void* setNumHormigasSoldado(*int* numHormigasSoldado) {  
 *this*.numHormigasSoldado = numHormigasSoldado;  
 }  
  
 *public* Lock getEntradaColonia() {  
 *return* entradaColonia;  
 }  
}

Clase Hormiga:

*package* concurrencia;  
  
*public class* Hormiga *extends* Thread{  
 *//Atributos de la clase Hormiga  
 private final* String identificador;  
 *private int* numIteraciones = 0;  
 *private final* String tipo;  
 *private final* Colonia colonia;  
 *//Zonas de la colonia (escritas para codigo más legible)  
 private final* AlmacenComida almacenComida;  
 *private final* ZonaComer zonaComer;  
 *private final* ZonaInstruccion zonaInstruccion;  
 *private final* ZonaDescanso zonaDescanso;  
 *private final* Refugio refugio;  
 *private final* Paso paso;  
  
 *//Métodos de la clase Hormiga  
  
 //Método constructor  
 public* Hormiga(Colonia colonia, String identificador){  
 *this*.colonia = colonia;  
 *this*.identificador = identificador;  
 *//El tipo dependerá del ID  
 if* (identificador.indexOf("O") == 1){  
 *//La segunda letra es una O de obrera, por tanto esta hormiga será obrera  
 this*.tipo = "Obrera";  
 }  
 *else if* (identificador.indexOf("S") == 1){  
 *//La segunda letra es una S de soldado, por tanto esta hormiga será soldado  
 this*.tipo = "Soldada";  
 }  
 *else*{  
 *//Si no es ni obrera ni soldada, tiene que ser cria  
 this*.tipo = "Cria";  
 }  
 *this*.almacenComida = colonia.getAlmacenComida();  
 *this*.zonaComer = colonia.getZonaComer();  
 *this*.zonaInstruccion = colonia.getZonaInstruccion();  
 *this*.zonaDescanso = colonia.getZonaDescanso();  
 *this*.refugio = colonia.getRefugio();  
 *this*.paso = colonia.getPaso();  
 }  
  
 *//Métodos get y set  
 public* String getIdentificador(){  
 *return this*.identificador;  
 }  
 *public int* getNumIteraciones(){  
 *return this*.numIteraciones;  
 }  
 *public void* setNumIteraciones(*int* numIteraciones){  
 *this*.numIteraciones = numIteraciones;  
 }  
 *public* String getTipo(){  
 *return this*.tipo;  
 }  
 *public* Colonia getColonia(){  
 *return this*.colonia;  
 }  
  
 *public* AlmacenComida getAlmacenComida() {  
 *return* almacenComida;  
 }  
  
 *public* ZonaComer getZonaComer() {  
 *return* zonaComer;  
 }  
  
 *public* ZonaInstruccion getZonaInstruccion() {  
 *return* zonaInstruccion;  
 }  
  
 *public* ZonaDescanso getZonaDescanso() {  
 *return* zonaDescanso;  
 }  
  
 *public* Refugio getRefugio() {  
 *return* refugio;  
 }  
  
 *public* Paso getPaso() {  
 *return* paso;  
 }  
}

Clase HormigaCria:

*package* concurrencia;  
  
*public class* HormigaCria *extends* Hormiga{  
 *//Atributos de la clase HormigaCria  
  
 //Métodos de la clase HormigaCria  
  
 //Método constructor  
 public* HormigaCria(Colonia colonia, String identificador) {  
 *super*(colonia, identificador);  
 }  
  
 *//Método run  
 public void* run(){  
 *try*{  
 getPaso().mirar(); *//Instruccion para que los hilos revisen si deben quedar parados o no, y recordar donde se quedaron parados* getColonia().entra(*this*); *//La hormiga entra a la colonia, sea del tipo que sea  
 while*(*true*){  
 rutina();  
 }  
 }*catch*(InterruptedException ie){  
 *try*{  
 getPaso().mirar();  
 }*catch*(InterruptedException ignored){}  
 *//Código de refugio (ya que son crias)* getColonia().actualizaEstadoInvasion(*this*);  
 *//Se protegen en el refugio  
 try*{  
 getPaso().mirar();  
 }*catch*(InterruptedException ignored){}  
 getColonia().getRefugio().protegeRefugio(*this*);  
 }  
 }  
  
 *//Método que realiza la rutina de una hormiga cria  
 private void* rutina() *throws* InterruptedException{  
 getPaso().mirar(); *//Instruccion para que los hilos revisen si deben quedar parados o no, y recordar donde se quedaron parados  
 //Una vez dentro de la colonia, van a la zona de comer* getColonia().getZonaComer().entra(*this*);  
 getPaso().mirar();  
 *//Una vez dentro de la zona de comer, cogen un alimento* getColonia().getZonaComer().cogeElementoComida(*this*);  
 Thread.sleep((*int*) (((Math.random() + 1) \* 3000) + (5000 - (3000)))); *//Tarda entre 3 y 5 en comer* getPaso().mirar();  
 *//Una vez que haya comido, sale de la zona de comer* getColonia().getZonaComer().sale(*this*);  
 getPaso().mirar();  
 *//Una vez que haya salido de la zona para comer, entran a la zona de descanso* getColonia().getZonaDescanso().entra(*this*);  
 *//Una vez dentro de la zona de descanso, descansan 4 segundos* getColonia().getZonaDescanso().realizaDescanso(*this*);  
 getPaso().mirar();  
 *//Una vez haya realizado el descanso, sale de la zona de descanso* getColonia().getZonaDescanso().sale(*this*);  
 }  
}

Clase HormigaObrera:

*package* concurrencia;  
  
*public class* HormigaObrera *extends* Hormiga{  
 *//Atributos de la clase HormigaObrera  
 private final int* numIdentificador;  
  
  
 *public* HormigaObrera(Colonia colonia, String identificador, *int* numIdentificador) {  
 *super*(colonia, identificador);  
 *this*.numIdentificador = numIdentificador;  
 }  
  
 *public void* run(){  
 *try*{  
 getPaso().mirar(); *//Instruccion para que los hilos revisen si deben quedar parados o no, y recordar donde se quedaron parados* getColonia().entra(*this*); *//La hormiga entra a la colonia, sea del tipo que sea  
 while*(*true*){  
 *//Verificamos que la hormiga es de tipo obrera  
 //Las obreras con ID par tienen distinta rutina que las impares  
 if* ((getNumIdentificador() % 2) != 0){ *//HORMIGA OBRERA IMPAR SALEN FUERA A POR COMIDA* rutinaHormigaImpar();  
 }  
 *else*{ *//HORMIGA OBRERA PAR  
 try*{  
 rutinaHormigaPar();  
 }*catch*(InterruptedException ignored){}  
 }  
 setNumIteraciones(getNumIteraciones() + 1); *//Subimos el numero de interrupciones (local a cada hilo)  
 if* (getNumIteraciones() >= 10){  
 setNumIteraciones(0); *//Reiniciamos las iteraciones* rutinaDescanso();  
 }  
 }  
 }*catch*(InterruptedException ignored){}  
 }  
  
 *private void* rutinaHormigaPar() *throws* InterruptedException{  
 getAlmacenComida().entra(*this*); *//Entra en el almacen de comida* Thread.sleep((*int*) (((Math.random() + 1) \* 1000) + (2000 - (1000 \* 2)))); *//Tarda entre 1 y 2 segundos en recoger un elemento de comida* getPaso().mirar();  
 getAlmacenComida().recogeElementoComida(*this*); *//Recoge un elemento de comida* getPaso().mirar();  
 getAlmacenComida().sale(*this*); *//Sale del almacen de comida* getPaso().mirar();  
 getColonia().getListaHormigasLlevandoComida().meterHormiga(*this*); *//Desde este momento esta llevando comida a la zona para comer* Thread.sleep((*int*) (((Math.random() + 1) \* 1000) + (3000 - (1000 \* 2)))); *//Tarda entre 1 y 3 segundos en viajar a la zona para comer* getPaso().mirar();  
 getColonia().getListaHormigasLlevandoComida().sacarHormiga(*this*);*//Desde que va a entrar a la zona para comer no esta llevando comida* getPaso().mirar();  
 getZonaComer().entra(*this*); *//Entra a la zona de comer* Thread.sleep((*int*) (((Math.random() + 1) \* 1000) + (2000 - (1000 \* 2)))); *//Tarda entre 1 y 2 segundos en depositar un elemento de comida* getPaso().mirar();  
 getZonaComer().depositaElementoComida(*this*); *//Deposita un elemento de comida en la zona para comer* getPaso().mirar();  
 getZonaComer().sale(*this*);  
 getPaso().mirar();  
 }  
  
 *private void* rutinaHormigaImpar() *throws* InterruptedException{  
 getPaso().mirar();  
 getColonia().sale(*this*);  
 getPaso().mirar();*//Sale de la colonia* getColonia().getListaHormigasBuscandoComida().meterHormiga(*this*); *//Desde este momento está buscando comida* Thread.sleep(4000); *//Tarda 4 segundos en recogerlo* getPaso().mirar();  
 getColonia().getListaHormigasBuscandoComida().sacarHormiga(*this*); *//Ya no esta buscando comida* getPaso().mirar();  
 getColonia().entra(*this*); *//Vuelve a entrar a la colonia* getPaso().mirar();  
 getAlmacenComida().entra(*this*); *//Entra al almacen de comida* Thread.sleep((*int*) (((Math.random() + 1) \* 2000) + (4000 - (2000 \* 2)))); *//Tarda entre 2 y 4 segundos en depositar* getPaso().mirar();  
 getAlmacenComida().depositaElementoComida(*this*); *//Deposita el elemento de comida* getPaso().mirar();  
 getAlmacenComida().sale(*this*); *//Sale del almacén de comida* }  
  
 *private void* rutinaDescanso() *throws* InterruptedException{  
 *//Una vez reiniciadas, entramos a la zona para comer* getPaso().mirar();  
 getZonaComer().entra(*this*);  
 getPaso().mirar();  
 *//Una vez dentro cogera un elemento de comida* getZonaComer().cogeElementoComida(*this*);  
 getPaso().mirar();  
 Thread.sleep(3000); *//Tarda 3 segundos en comer* getPaso().mirar();  
 *//Una vez que ha comido saldrá de la zona de comer* getZonaComer().sale(*this*);  
 getPaso().mirar();  
 *//Una vez fuera, entra a la zona de descanso* getZonaDescanso().entra(*this*);  
 getPaso().mirar();  
 *//Una vez dentro de la zona de descanso, realiza el descanso* getZonaDescanso().realizaDescanso(*this*);  
 getPaso().mirar();  
 *//Sale de la zona de descanso* getColonia().getZonaDescanso().sale(*this*);  
 getPaso().mirar();  
 }  
  
 *public int* getNumIdentificador(){  
 *return this*.numIdentificador;  
 }  
}

Clase HormigaSoldado:

*package* concurrencia;  
  
*public class* HormigaSoldado *extends* Hormiga{  
 *//Atributos de la clase HormigaSoldado  
  
 //Métodos de la clase HormigaSoldado  
 public* HormigaSoldado(Colonia colonia, String identificador) {  
 *super*(colonia, identificador);  
 }  
  
 *//Método run  
 public void* run(){  
 *try*{  
 getPaso().mirar(); *//Instruccion para que los hilos revisen si deben quedar parados o no, y recordar donde se quedaron parados* getColonia().entra(*this*); *//La hormiga entra a la colonia, sea del tipo que sea* }*catch*(InterruptedException ignored){}  
 *while*(*true*){  
 *try*{  
 *while*(*true*){  
 rutinaHormigaSoldada();  
 setNumIteraciones(getNumIteraciones() + 1);  
 *if* (getNumIteraciones() >= 6){  
 rutinaDescanso();  
 }  
 }  
 }*catch*(InterruptedException ie){  
 *try*{  
 getPaso().mirar();  
 }*catch*(InterruptedException ignored){}  
 getColonia().actualizaEstadoInvasion(*this*);  
 *try*{  
 getPaso().mirar();  
 }*catch*(InterruptedException ignored){}  
 *//Para realizar la invasión, tienen que salir de la colonia* getColonia().sale(*this*);  
 *try*{  
 getPaso().mirar();  
 }*catch*(InterruptedException ignored){}  
 *//Código de la invasion* getColonia().getInvasion().realizaInvasion(*this*);  
 *//Una vez acabada la invasión, tendrán que entrar de nuevo a la colonia  
 try*{  
 getPaso().mirar();  
 }*catch*(InterruptedException ignored){}  
 getColonia().entra(*this*);  
 }  
 }  
 }  
  
 *private void* rutinaHormigaSoldada() *throws* InterruptedException{  
 *//Verificamos que la hormiga es de tipo soldada  
 //Una vez dentro de la colonia, se irán a la zona de instrucción* getPaso().mirar();  
 getZonaInstruccion().entra(*this*);  
 getPaso().mirar();  
 *//Una vez dentro de la zona de instruccion, hacen entre 2 y 8 segundos de instruccion* getZonaInstruccion().realizaInstruccion(*this*);  
 getPaso().mirar();  
 *//Una vez que haya hecho la instrucción, saldrá de la zona de instrucción* getZonaInstruccion().sale(*this*);  
 getPaso().mirar();  
 *//Una vez fuera de la zona de instrucción, se irá a a zona de descanso* getZonaDescanso().entra(*this*);  
 getPaso().mirar();  
 *//Una vez dentro de la zona de descanso, procede a descansar* getZonaDescanso().realizaDescanso(*this*);  
 getPaso().mirar();  
 *//Una vez que haya realizado el descanso, abandonará la zona de descanso* getZonaDescanso().sale(*this*);  
 getPaso().mirar();  
 *//Como se ha completado una iteración, incrementamos en 1 el numero* }  
  
 *private void* rutinaDescanso() *throws* InterruptedException{  
 *//En primer lugar, reiniciaremos el numero de iteraciones a 0* setNumIteraciones(0);  
 getPaso().mirar();  
 *//Cuando hayan hecho 6 iteraciones, se pasaran por la zona para comer, tardan en comer 3 segundos* getZonaComer().entra(*this*);  
 getPaso().mirar();  
 *//Una vez dentro de la zona de comer, nos ponemos a comer* getZonaComer().cogeElementoComida(*this*);  
 Thread.sleep(3000);  
 getPaso().mirar();  
 *//Cuando haya comido se va de la zona para comer y repite de nuevo* getColonia().getZonaComer().sale(*this*);  
 }  
}

Clase Invasion:

*package* concurrencia;  
  
*import* javax.swing.\*;  
*import* java.util.ArrayList;  
*import* java.util.concurrent.locks.Condition;  
*import* java.util.concurrent.locks.Lock;  
*import* java.util.concurrent.locks.ReentrantLock;  
  
*public class* Invasion {  
 *//Atributos de la clase Invasion  
 private final* Log log;  
 *private int* numHormigasSoldado = 0;  
 *private boolean* activa = *false*;  
 *private boolean* enCurso = *false*;  
 *private final* Lock cerrojoInvasion = *new* ReentrantLock();  
 *private final* Condition conditionInvasion = cerrojoInvasion.newCondition();  
 *private final* ListaThreads listaHormigasInvasion;  
 *//Métodos de la clase Invasion  
  
 //Método constructor  
 public* Invasion(Log log, JTextField jTextFieldHormigasContraInvasor){  
 *this*.log = log;  
 *this*.listaHormigasInvasion = *new* ListaThreads(jTextFieldHormigasContraInvasor);  
 }  
  
 *//Método para unirse a la invasion  
 public void* realizaInvasion(Hormiga hormiga){  
 *try*{  
 cerrojoInvasion.lock();  
 *if*(isActiva() && (!isEnCurso())){  
 hormiga.getPaso().mirar();  
 entra(hormiga);  
 *if*(activarInvasionEnCurso(hormiga)){  
 setEnCurso(*true*); *//Indicamos que la invasion está en curso* getLog().escribirEnLog("[INVASION] --> Las hormigas estan combatiendo al insecto invasor");  
 cerrojoInvasion.unlock();  
 Thread.sleep(20000);  
 hormiga.getPaso().mirar();  
 cerrojoInvasion.lock();  
 getLog().escribirEnLog("[INVASION] --> Las hormigas han vencido al insecto invasor");  
 conditionInvasion.signalAll(); *//Despertamos a las hormigas soldadas* hormiga.getRefugio().despiertaRefugio(); *//Despertamos a las crias pq ya ha terminado la invasion* }  
 *else*{  
 conditionInvasion.await();  
 }  
 hormiga.getPaso().mirar();  
 sale(hormiga);  
 }  
 }*catch* (InterruptedException ignored){}  
 cerrojoInvasion.unlock();  
 }  
  
 *//Método para entrar a la invasion  
 private void* entra(Hormiga hormiga){  
 setNumHormigasSoldado(getNumHormigasSoldado() + 1); *//Incrementamos el numero de hormigas soldado* getListaHormigasInvasion().meterHormiga(hormiga); *//Metemos el identificador de la hormiga al JTextField* getLog().escribirEnLog("[INVASION] --> La hormiga " + hormiga.getIdentificador() + " ha entrado a la invasion");  
 }  
  
 *//Método para comprobar si la invasion deberia estar en curso o no  
 private boolean* activarInvasionEnCurso(Hormiga hormiga){  
 hormiga.getColonia().getEntradaColonia().lock();  
 *boolean* result;  
 *int* numHormigasInvasion = getNumHormigasSoldado();  
 *int* numHormigasSoldadoColonia = cuentaSoldadasColonia(hormiga.getColonia().getListaHormigas());  
 result = numHormigasInvasion == numHormigasSoldadoColonia;  
 hormiga.getColonia().getEntradaColonia().unlock();  
  
 *return* result;  
 }  
  
 *private void* sale(Hormiga hormiga){  
 setNumHormigasSoldado(getNumHormigasSoldado() - 1); *//Decrementamos el numero de hormigas soldado* getListaHormigasInvasion().sacarHormiga(hormiga); *//Quitamos el identificador de la hormiga del JTextField* getLog().escribirEnLog("[INVASION] --> La hormiga " + hormiga.getIdentificador() + " ha salido de la invasion");  
 *if*(getNumHormigasSoldado() == 0){  
 *//Es la última hormiga en salir, por tanto la invasion queda finalizada* setEnCurso(*false*); *//La invasión no estará en curso* setActiva(*false*); *//La invasión dejara de estar activa* }  
 }  
  
 *private int* cuentaSoldadasColonia(ArrayList<Hormiga> lista){  
 *int* resultado = 0;  
 *for* (Hormiga hormigaActual : lista) {  
 String tipo = hormigaActual.getTipo();  
 *if* (tipo.equals("Soldada")) {  
 resultado = resultado + 1;  
 }  
 }  
 *return* resultado;  
 }  
  
 *//Métodos get y set  
  
  
 public* Log getLog() {  
 *return* log;  
 }  
  
 *public int* getNumHormigasSoldado() {  
 *return* numHormigasSoldado;  
 }  
  
 *public void* setNumHormigasSoldado(*int* numHormigasSoldado) {  
 *this*.numHormigasSoldado = numHormigasSoldado;  
 }  
  
 *public synchronized boolean* isActiva() {  
 *return* activa;  
 }  
  
 *public void* setActiva(*boolean* activa) {  
 *this*.activa = activa;  
 }  
  
 *public boolean* isEnCurso() {  
 *return* enCurso;  
 }  
  
 *public void* setEnCurso(*boolean* enCurso) {  
 *this*.enCurso = enCurso;  
 }  
  
 *public* ListaThreads getListaHormigasInvasion() {  
 *return* listaHormigasInvasion;  
 }  
  
}

Clase ListaThreads:

*package* concurrencia;  
*import* java.util.\*;  
*import* javax.swing.JTextField;  
  
*/\* La clase ListaThreads permite gestionar las listas de threads en los monitores,  
con métodos para meter y sacar threads en ella. Cada vez que una lista se modifica,  
se imprime su nuevo contenido en el JTextField que toma como parámetro el constructor. \*/  
public class* ListaThreads{  
 *private final* ArrayList<Hormiga> listaHormigas;  
 *private final* JTextField tf;  
   
 *public* ListaThreads(JTextField tf){  
 listaHormigas = *new* ArrayList<>();  
 *this*.tf = tf;  
 }  
   
 *public synchronized void* meterHormiga(Hormiga hormiga){  
 listaHormigas.add(hormiga);  
 imprimirHormiga();  
 }  
   
 *public synchronized void* sacarHormiga(Hormiga hormiga){  
 listaHormigas.remove(hormiga);  
 imprimirHormiga();  
 }  
  
 *public synchronized void* insertarNumero(Integer num){  
 String texto = String.valueOf(num);  
 tf.setText(texto);  
 }  
  
 *private void* imprimirHormiga() {  
 StringBuilder contenido = *new* StringBuilder();  
 *for* (Hormiga hormigas : listaHormigas) {  
 contenido.append(hormigas.getIdentificador()).append(" ");  
 }  
 tf.setText(contenido.toString());  
 }  
  
 *public* ArrayList<Hormiga> getListaHormigas() {  
 *return* listaHormigas;  
 }  
}

Clase Log:

*package* concurrencia;  
  
*import* java.io.BufferedWriter;  
*import* java.io.File;  
*import* java.io.FileWriter;  
*import* java.io.IOException;  
*import* java.text.DateFormat;  
*import* java.text.SimpleDateFormat;  
*import* java.util.Date;  
  
*public class* Log {  
 *private final* File file;  
 *private final boolean* debug;  
  
 *//Métodos de la clase log  
  
 //Método constructor  
 public* Log(*boolean* debug) {  
 *this*.debug = debug;  
 *//Atributos de la clase log* String ruta = "debug.log";  
 *this*.file = *new* File(ruta);  
 *//Creamos el archivo log  
 if*(debug) {  
 crearLog();  
 }  
 }  
  
 *//Metodo para crear el archivo log  
 public void* crearLog(){  
 *try* {  
 String contenido = "[ARCHIVO LOG COLONIA]\n\n";  
  
 *if*(file.exists()){  
 *//Si existe lo borramos y creamos uno nuevo al inicio de la ejecucion del programa  
 if*(file.delete()){  
 file.createNewFile();  
 }  
 } *else* {  
 file.createNewFile();  
 }  
 *//Mostramos la ruta absoluta del fichero* System.out.println("[LOG] Archivo creado en: " + file.getAbsolutePath());  
  
 FileWriter escribir = *new* FileWriter(file);  
 *try* (BufferedWriter bw = *new* BufferedWriter(escribir)) {  
 bw.write(contenido);  
 }  
 } *catch*(IOException ex) {  
 System.out.println("ERROR: " + ex);  
 }  
 } *// Cierre del método  
  
 //Metodo para escribir en el log en caso de que la variable debug este activada  
 public synchronized void* escribirEnLog(String contenido) {  
 *if*(debug) {  
 Date fecha = *new* Date();  
 DateFormat formatoFecha = *new* SimpleDateFormat("[dd/MM/yyyy HH:mm:ss]");  
 *try* {  
 FileWriter escribir = *new* FileWriter(file, *true*);  
 *try* (BufferedWriter bw = *new* BufferedWriter(escribir)) {  
 bw.write( formatoFecha.format(fecha) + " --> " + contenido + "\n" );  
 }  
 } *catch*(IOException ex) {  
 System.out.println("ERROR: " + ex);  
 }  
 }  
 }  
}

Clase Paso:

*package* concurrencia;  
  
*import* java.util.concurrent.locks.Condition;  
*import* java.util.concurrent.locks.Lock;  
*import* java.util.concurrent.locks.ReentrantLock;  
  
*// Si el booleano vale false (abierto) el proceso puede continuar, pero si es true(cerrado) el proceso se detiene  
public class* Paso{  
 *//Atributos de la clase Paso  
 private boolean* cerrado = *false*;  
 *private final* Lock cerrojo = *new* ReentrantLock();  
 *private final* Condition parar = cerrojo.newCondition();  
  
 *//Métodos de la clase Paso  
  
 //Método constructor  
 public* Paso(){  
  
 }  
 *//Método que mira si nos tenemos que detener o no  
 public void* mirar() *throws* InterruptedException{  
 *try* {  
 cerrojo.lock();  
 *while*(isCerrado()) {  
 parar.await();  
 }  
 }  
 *finally* {  
 cerrojo.unlock();  
 }  
 }  
 *//Método para reanudar la ejecucion de hilos  
 public void* abrir() {  
 *try*{  
 cerrojo.lock();  
 setCerrado(*false*);  
 parar.signalAll();  
 }  
 *finally* {  
 cerrojo.unlock();  
 }  
 }  
 *//Métodos para detener la ejecucion de hilos  
 public void* cerrar() {  
 *try*{  
 cerrojo.lock();  
 setCerrado(*true*);  
 }  
 *finally* {  
 cerrojo.unlock();  
 }  
 }  
  
 *public boolean* isCerrado() {  
 *return* cerrado;  
 }  
  
 *public void* setCerrado(*boolean* cerrado) {  
 *this*.cerrado = cerrado;  
 }  
}

Clase ProgPrincipal:

*/\*  
 \* To change this license header, choose License Headers in Project Properties.  
 \* To change this template file, choose Tools | Templates  
 \* and open the template in the editor.  
 \*/  
package* concurrencia;  
  
*import* distribuida.GestorInterfaz;  
  
*import* javax.swing.\*;  
*import* java.rmi.Naming;  
*import* java.rmi.registry.LocateRegistry;  
*import* java.rmi.registry.Registry;  
*import* java.util.Formatter;  
  
*/\*\*  
 \*  
 \* @author vsnv  
 \*/  
public class* ProgPrincipal *extends* javax.swing.JFrame {  
 *//Atributos de la clase ProgPrincipal  
 private int* numTotalHormigas = 0, numHormigasObreras = 0, numHormigasSoldado = 0, numHormigasCria = 0;  
 *private* Log log;  
 *private* Colonia colonia;  
 *private boolean* botonPausarPulsado = *false*;  
  
 */\*\*  
 \* Creates new form ProgPrincipal  
 \*/  
 public* ProgPrincipal() {  
 initComponents();  
 *//Creamos el log  
 this*.log = *new* Log(*true*);  
 *//Creamos la colonia  
 this*.colonia = *new* Colonia(getLog(), getjTextFieldHormigasBuscandoComida(), getjTextFieldHormigasContraInvasor(),  
 getjTextFieldHormigasAlmacenComida(), getjTextFieldHormiasLlevandoComida(), getjTextFieldHormigasHaciendoInstruccion(),  
 getjTextFieldUnidadesComidaAlmacen(), getjTextFieldUnidadesComidaZonaComer(), getjTextFieldHormigasDescansando(),  
 getjTextFieldHormigasZonaComer(), getjTextFieldHormigasRefugio());  
  
 *//Inicializamos el objeto remoto* creaObjetoRemoto();  
 }  
  
 *//Método que crea el sistema concurrente  
 public void* crearSistema(){  
 *//Creamos los hilos* Formatter fmt;  
 *while*(getNumTotalHormigas() < 10000){  
 *for* (*int* i = 0; i < 3; i++){  
 fmt = *new* Formatter();  
 fmt.format("%05d", getNumHormigasObreras());  
 String identificadorObrera = "HO" + fmt;  
 HormigaObrera hormigaObrera = *new* HormigaObrera(getColonia(), identificadorObrera, getNumHormigasObreras());  
 hormigaObrera.setName(identificadorObrera);  
 hormigaObrera.start();  
 identificadorObrera = *null*;  
 setNumHormigasObreras(getNumHormigasObreras() + 1);  
 }  
 *//Por cada 3 obreras, se hace una soldada y una cria  
 //Creamos una hormiga soldado* fmt = *new* Formatter();  
 fmt.format("%05d", getNumHormigasSoldado());  
 String identificadorSoldado = "HS" + fmt;  
 HormigaSoldado hormigaSoldado = *new* HormigaSoldado(getColonia(), identificadorSoldado);  
 hormigaSoldado.setName(identificadorSoldado);  
 hormigaSoldado.start();  
 identificadorSoldado = *null*;  
 setNumHormigasSoldado(getNumHormigasSoldado() + 1);  
 *//getColonia().setNumHormigasSoldado(getNumHormigasSoldado());  
  
 //Creamos una hormiga cria* fmt = *new* Formatter();  
 fmt.format("%05d", getNumHormigasCria());  
 String identificadorCria = "HC" + fmt;  
 HormigaCria hormigaCria = *new* HormigaCria(getColonia(), identificadorCria);  
 hormigaCria.setName(identificadorCria);  
 hormigaCria.start();  
 identificadorCria = *null*;  
 setNumHormigasCria(getNumHormigasCria() + 1);  
 *//Esperamos entre 0.8 y 3.5 segundos para hacer la siguiente ronda  
 try*{  
 Thread.sleep((*int*) (((Math.random() + 1) \* 800) + (3500 - (800 \* 2))));  
 }*catch*(InterruptedException ignored){}  
 setNumTotalHormigas(getNumTotalHormigas() + 5);  
 }  
 }  
  
 *//Método para crear el objeto remoto  
 public void* creaObjetoRemoto(){  
 *try*{  
 GestorInterfaz objetoRemoto = *new* GestorInterfaz(getColonia());  
 Registry registry = LocateRegistry.createRegistry(1099);  
 Naming.rebind("//localhost/ObjetoColonia", objetoRemoto);  
 System.out.println("El objeto remoto ha sido creado");  
 getLog().escribirEnLog("[RMI] --> El objeto remoto ha sido creado");  
 }*catch*(Exception e){  
 System.out.println("Error --> " + e);  
 e.printStackTrace();  
 }  
 }  
  
 */\*\*  
 \* 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">//GEN-BEGIN:initComponents  
 private void* initComponents() {  
  
 jPanelPrincipal = *new* javax.swing.JPanel();  
 JPanelHormigasBuscandoComida = *new* javax.swing.JPanel();  
 jLabelHormigasBuscandoComida = *new* javax.swing.JLabel();  
 jTextFieldHormigasBuscandoComida = *new* javax.swing.JTextField();  
 jPanel1 = *new* javax.swing.JPanel();  
 jLabelHormigasContraInvasor = *new* javax.swing.JLabel();  
 jTextFieldHormigasContraInvasor = *new* javax.swing.JTextField();  
 jPanel2 = *new* javax.swing.JPanel();  
 jLabelInteriorColonia = *new* javax.swing.JLabel();  
 jLabelHormigasAlmacenComida = *new* javax.swing.JLabel();  
 jTextFieldHormigasAlmacenComida = *new* javax.swing.JTextField();  
 jLabelHormigasLlevandoComida = *new* javax.swing.JLabel();  
 jTextFieldHormiasLlevandoComida = *new* javax.swing.JTextField();  
 jLabelHormigasHaciendoInstruccion = *new* javax.swing.JLabel();  
 jTextFieldHormigasHaciendoInstruccion = *new* javax.swing.JTextField();  
 jLabel2 = *new* javax.swing.JLabel();  
 jTextFieldHormigasDescansando = *new* javax.swing.JTextField();  
 jLabelUnidadesComidaAlmacen = *new* javax.swing.JLabel();  
 jTextFieldUnidadesComidaAlmacen = *new* javax.swing.JTextField();  
 jLabelUnidadesComidaZonaComer = *new* javax.swing.JLabel();  
 jTextFieldUnidadesComidaZonaComer = *new* javax.swing.JTextField();  
 jLabelZonaComer = *new* javax.swing.JLabel();  
 jTextFieldHormigasZonaComer = *new* javax.swing.JTextField();  
 jLabelRefugio = *new* javax.swing.JLabel();  
 jTextFieldHormigasRefugio = *new* javax.swing.JTextField();  
 jButtonPausarReanudar = *new* javax.swing.JButton();  
 jButtonGenerarAmenazaInsectoInvasor = *new* javax.swing.JButton();  
  
 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);  
 getContentPane().setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 jPanelPrincipal.setBackground(*new* java.awt.Color(255, 255, 255));  
 jPanelPrincipal.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 JPanelHormigasBuscandoComida.setBackground(*new* java.awt.Color(153, 153, 153));  
 JPanelHormigasBuscandoComida.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 jLabelHormigasBuscandoComida.setBackground(*new* java.awt.Color(255, 255, 255));  
 jLabelHormigasBuscandoComida.setText("Hormigas buscando comida");  
 JPanelHormigasBuscandoComida.add(jLabelHormigasBuscandoComida, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(10, 40, -1, -1));  
 JPanelHormigasBuscandoComida.add(jTextFieldHormigasBuscandoComida, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(180, 20, 1250, 60));  
  
 jPanelPrincipal.add(JPanelHormigasBuscandoComida, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, 1440, 90));  
  
 jPanel1.setBackground(*new* java.awt.Color(153, 153, 153));  
 jPanel1.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 jLabelHormigasContraInvasor.setBackground(*new* java.awt.Color(255, 255, 255));  
 jLabelHormigasContraInvasor.setText("Hormigas repeliendo un insecto invasor ");  
 jPanel1.add(jLabelHormigasContraInvasor, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(710, 10, -1, -1));  
 jPanel1.add(jTextFieldHormigasContraInvasor, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(10, 40, 1430, 120));  
  
 jPanelPrincipal.add(jPanel1, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(10, 110, 1450, 170));  
  
 jPanel2.setBackground(*new* java.awt.Color(153, 153, 153));  
 jPanel2.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 jLabelInteriorColonia.setBackground(*new* java.awt.Color(0, 0, 0));  
 jLabelInteriorColonia.setForeground(*new* java.awt.Color(255, 0, 51));  
 jLabelInteriorColonia.setText("Interior de la colonia");  
 jPanel2.add(jLabelInteriorColonia, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(760, 10, -1, -1));  
  
 jLabelHormigasAlmacenComida.setBackground(*new* java.awt.Color(255, 255, 255));  
 jLabelHormigasAlmacenComida.setText("Hormigas en el Almacen de Comida");  
 jPanel2.add(jLabelHormigasAlmacenComida, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(20, 40, -1, -1));  
 jPanel2.add(jTextFieldHormigasAlmacenComida, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(240, 30, 1160, 30));  
  
 jLabelHormigasLlevandoComida.setBackground(*new* java.awt.Color(0, 0, 0));  
 jLabelHormigasLlevandoComida.setText("Hormigas llevando comida a la Zona para Comer");  
 jPanel2.add(jLabelHormigasLlevandoComida, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(20, 80, -1, -1));  
 jPanel2.add(jTextFieldHormiasLlevandoComida, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(300, 70, 1100, 30));  
  
 jLabelHormigasHaciendoInstruccion.setBackground(*new* java.awt.Color(255, 255, 255));  
 jLabelHormigasHaciendoInstruccion.setText("Hormigas haciendo Instruccion");  
 jPanel2.add(jLabelHormigasHaciendoInstruccion, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(20, 120, -1, -1));  
 jPanel2.add(jTextFieldHormigasHaciendoInstruccion, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(200, 110, 1200, 30));  
  
 jLabel2.setBackground(*new* java.awt.Color(255, 255, 255));  
 jLabel2.setText("Hormigas descansando");  
 jPanel2.add(jLabel2, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(20, 160, -1, -1));  
 jPanel2.add(jTextFieldHormigasDescansando, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(170, 150, 720, 30));  
  
 jLabelUnidadesComidaAlmacen.setBackground(*new* java.awt.Color(255, 255, 255));  
 jLabelUnidadesComidaAlmacen.setText("Unidades de Comida (Almacen)");  
 jPanel2.add(jLabelUnidadesComidaAlmacen, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(1200, 160, -1, -1));  
 jPanel2.add(jTextFieldUnidadesComidaAlmacen, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(1380, 150, 60, 30));  
  
 jLabelUnidadesComidaZonaComer.setBackground(*new* java.awt.Color(255, 255, 255));  
 jLabelUnidadesComidaZonaComer.setText("Unidades de Comida (Zona para Comer)");  
 jPanel2.add(jLabelUnidadesComidaZonaComer, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(1150, 200, -1, -1));  
 jPanel2.add(jTextFieldUnidadesComidaZonaComer, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(1380, 190, 60, 30));  
  
 jLabelZonaComer.setBackground(*new* java.awt.Color(255, 255, 255));  
 jLabelZonaComer.setText("Zona para Comer");  
 jPanel2.add(jLabelZonaComer, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(20, 220, -1, -1));  
 jPanel2.add(jTextFieldHormigasZonaComer, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(130, 200, 1010, 90));  
  
 jLabelRefugio.setBackground(*new* java.awt.Color(255, 255, 255));  
 jLabelRefugio.setText("Refugio");  
 jPanel2.add(jLabelRefugio, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(40, 350, -1, -1));  
 jPanel2.add(jTextFieldHormigasRefugio, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(130, 300, 1010, 110));  
  
 jButtonPausarReanudar.setText("Pausar");  
 jButtonPausarReanudar.setCursor(*new* java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));  
 jButtonPausarReanudar.addActionListener(*new* java.awt.event.ActionListener() {  
 *public void* actionPerformed(java.awt.event.ActionEvent evt) {  
 jButtonPausarReanudarActionPerformed(evt);  
 }  
 });  
 jPanel2.add(jButtonPausarReanudar, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(480, 440, 150, 40));  
  
 jButtonGenerarAmenazaInsectoInvasor.setText("Generar amenaza de insecto invasor");  
 jButtonGenerarAmenazaInsectoInvasor.setCursor(*new* java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));  
 jButtonGenerarAmenazaInsectoInvasor.addActionListener(*new* java.awt.event.ActionListener() {  
 *public void* actionPerformed(java.awt.event.ActionEvent evt) {  
 jButtonGenerarAmenazaInsectoInvasorActionPerformed(evt);  
 }  
 });  
 jPanel2.add(jButtonGenerarAmenazaInsectoInvasor, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(700, 440, 340, 40));  
  
 jPanelPrincipal.add(jPanel2, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(10, 290, 1450, 500));  
  
 getContentPane().add(jPanelPrincipal, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1470, 800));  
  
 pack();  
 }*// </editor-fold>//GEN-END:initComponents  
  
 private void* jButtonGenerarAmenazaInsectoInvasorActionPerformed(java.awt.event.ActionEvent evt) {*//GEN-FIRST:event\_jButtonGenerarAmenazaInsectoInvasorActionPerformed  
 // TODO add your handling code here:* getColonia().generaInvasion();  
 }*//GEN-LAST:event\_jButtonGenerarAmenazaInsectoInvasorActionPerformed  
  
 private void* jButtonPausarReanudarActionPerformed(java.awt.event.ActionEvent evt) {*//GEN-FIRST:event\_jButtonPausarReanudarActionPerformed  
 // TODO add your handling code here:  
 if*(!isBotonPausarPulsado()){  
 *//Si no se ha pulsado el boton* setBotonPausarPulsado(*true*); *//Actualizamos el booleano a que se ha pulsado* getBotonPausarReanudar().setText("Reanudar"); *//Ponemos el texto a Reanudar, ya que a próxima vez que se pulse el botón será para reanudar* getColonia().getPaso().cerrar(); *//Cerramos el paso para que los hilos se detengan en él* }  
 *else*{  
 *//Si ya se habia pulsado el boton previamente* setBotonPausarPulsado(*false*); *//Cambiamos el estado para que la proxima vez que pulsemos sea para parar* getBotonPausarReanudar().setText("Pausar"); *//Cambiamos el texto del boton a pausar, ya que la proxima vez que se pulse el boton será para pausar* getColonia().getPaso().abrir(); *//Abrimos el paso para que los hilos puedan seguir su ejecucion* }  
  
 }*//GEN-LAST:event\_jButtonPausarReanudarActionPerformed  
  
 /\*\*  
 \* @param args the command line arguments  
 \*/  
 public static void* main(String args[]) {  
 */\* Set the Nimbus look and feel \*/  
 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html   
 \*/  
 try* {  
 *for* (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  
 *if* ("Nimbus".equals(info.getName())) {  
 javax.swing.UIManager.setLookAndFeel(info.getClassName());  
 *break*;  
 }  
 }  
 } *catch* (ClassNotFoundException ex) {  
 java.util.logging.Logger.getLogger(ProgPrincipal.*class*.getName()).log(java.util.logging.Level.SEVERE, *null*, ex);  
 } *catch* (InstantiationException ex) {  
 java.util.logging.Logger.getLogger(ProgPrincipal.*class*.getName()).log(java.util.logging.Level.SEVERE, *null*, ex);  
 } *catch* (IllegalAccessException ex) {  
 java.util.logging.Logger.getLogger(ProgPrincipal.*class*.getName()).log(java.util.logging.Level.SEVERE, *null*, ex);  
 } *catch* (javax.swing.UnsupportedLookAndFeelException ex) {  
 java.util.logging.Logger.getLogger(ProgPrincipal.*class*.getName()).log(java.util.logging.Level.SEVERE, *null*, ex);  
 }  
 *//</editor-fold>* ProgPrincipal main = *new* ProgPrincipal();  
 */\* Create and display the form \*/* java.awt.EventQueue.invokeLater(*new* Runnable() {  
 *public void* run() {  
 main.setVisible(*true*);  
 }  
 });  
 *//Creamos el sistema concurrente* main.crearSistema();  
 }  
  
 *// Variables declaration - do not modify//GEN-BEGIN:variables  
 private* javax.swing.JPanel JPanelHormigasBuscandoComida;  
 *private* javax.swing.JButton jButtonGenerarAmenazaInsectoInvasor;  
 *private* javax.swing.JButton jButtonPausarReanudar;  
 *private* javax.swing.JLabel jLabel2;  
 *private* javax.swing.JLabel jLabelHormigasAlmacenComida;  
 *private* javax.swing.JLabel jLabelHormigasBuscandoComida;  
 *private* javax.swing.JLabel jLabelHormigasContraInvasor;  
 *private* javax.swing.JLabel jLabelHormigasHaciendoInstruccion;  
 *private* javax.swing.JLabel jLabelHormigasLlevandoComida;  
 *private* javax.swing.JLabel jLabelInteriorColonia;  
 *private* javax.swing.JLabel jLabelRefugio;  
 *private* javax.swing.JLabel jLabelUnidadesComidaAlmacen;  
 *private* javax.swing.JLabel jLabelUnidadesComidaZonaComer;  
 *private* javax.swing.JLabel jLabelZonaComer;  
 *private* javax.swing.JPanel jPanel1;  
 *private* javax.swing.JPanel jPanel2;  
 *private* javax.swing.JPanel jPanelPrincipal;  
 *private* javax.swing.JTextField jTextFieldHormiasLlevandoComida;  
 *private* javax.swing.JTextField jTextFieldHormigasAlmacenComida;  
 *private* javax.swing.JTextField jTextFieldHormigasBuscandoComida;  
 *private* javax.swing.JTextField jTextFieldHormigasContraInvasor;  
 *private* javax.swing.JTextField jTextFieldHormigasDescansando;  
 *private* javax.swing.JTextField jTextFieldHormigasHaciendoInstruccion;  
 *private* javax.swing.JTextField jTextFieldHormigasRefugio;  
 *private* javax.swing.JTextField jTextFieldHormigasZonaComer;  
 *private* javax.swing.JTextField jTextFieldUnidadesComidaAlmacen;  
 *private* javax.swing.JTextField jTextFieldUnidadesComidaZonaComer;  
 *// End of variables declaration//GEN-END:variables  
  
 //Métodos get y set  
  
 public* JTextField getjTextFieldHormiasLlevandoComida() {  
 *return this*.jTextFieldHormiasLlevandoComida;  
 }  
  
 *public* JTextField getjTextFieldHormigasAlmacenComida() {  
 *return this*.jTextFieldHormigasAlmacenComida;  
 }  
  
 *public* JTextField getjTextFieldHormigasBuscandoComida() {  
 *return this*.jTextFieldHormigasBuscandoComida;  
 }  
  
 *public* JTextField getjTextFieldHormigasContraInvasor() {  
 *return this*.jTextFieldHormigasContraInvasor;  
 }  
  
 *public* JTextField getjTextFieldHormigasDescansando() {  
 *return this*.jTextFieldHormigasDescansando;  
 }  
  
 *public* JTextField getjTextFieldHormigasHaciendoInstruccion() {  
 *return this*.jTextFieldHormigasHaciendoInstruccion;  
 }  
  
 *public* JTextField getjTextFieldHormigasRefugio() {  
 *return this*.jTextFieldHormigasRefugio;  
 }  
  
 *public* JTextField getjTextFieldUnidadesComidaAlmacen() {  
 *return this*.jTextFieldUnidadesComidaAlmacen;  
 }  
  
 *public* JTextField getjTextFieldUnidadesComidaZonaComer() {  
 *return this*.jTextFieldUnidadesComidaZonaComer;  
 }  
  
 *public* JTextField getjTextFieldHormigasZonaComer() {  
 *return this*.jTextFieldHormigasZonaComer;  
 }  
  
 *public* JButton getBotonPausarReanudar() {  
 *return* jButtonPausarReanudar;  
 }  
  
 *public* JButton getBotonGenerarInvasion(){  
 *return* jButtonGenerarAmenazaInsectoInvasor;  
 }  
  
 *public* Log getLog(){  
 *return this*.log;  
 }  
  
 *public* Colonia getColonia(){  
 *return this*.colonia;  
 }  
  
 *public int* getNumTotalHormigas(){  
 *return this*.numTotalHormigas;  
 }  
  
 *public void* setNumTotalHormigas(*int* numTotalHormigas){  
 *this*.numTotalHormigas = numTotalHormigas;  
 }  
  
 *public int* getNumHormigasObreras(){  
 *return this*.numHormigasObreras;  
 }  
  
 *public void* setNumHormigasObreras(*int* numHormigasObreras){  
 *this*.numHormigasObreras = numHormigasObreras;  
 }  
  
 *public int* getNumHormigasSoldado(){  
 *return this*.numHormigasSoldado;  
 }  
  
 *public void* setNumHormigasSoldado(*int* numHormigasSoldado){  
 *this*.numHormigasSoldado = numHormigasSoldado;  
 }  
  
 *public int* getNumHormigasCria(){  
 *return this*.numHormigasCria;  
 }  
  
 *public void* setNumHormigasCria(*int* numHormigasCria){  
 *this*.numHormigasCria = numHormigasCria;  
 }  
  
 *public void* setBotonPausarPulsado(*boolean* botonPausarPulsado) {  
 *this*.botonPausarPulsado = botonPausarPulsado;  
 }  
  
 *public boolean* isBotonPausarPulsado() {  
 *return* botonPausarPulsado;  
 }  
  
  
}

Clase Refugio:

*package* concurrencia;  
  
*import* javax.swing.\*;  
*import* java.util.concurrent.locks.Condition;  
*import* java.util.concurrent.locks.Lock;  
*import* java.util.concurrent.locks.ReentrantLock;  
  
*public class* Refugio {  
 *//Atributos de la clase Refugio  
 private final* Log log; *//Log del sistema concurrente  
 private boolean* activo = *false*;  
 *private int* numHormigasRefugio = 0;  
 *private final* Lock cerrojoRefugio = *new* ReentrantLock();  
 *private final* Condition finInvasion = cerrojoRefugio.newCondition(); *//Condition para que las hormigas esperen al fin de la invasion  
 private final* ListaThreads listaHormigasRefugio; *//ListaThreads para manejar el JTextField del refugio de la interfaz  
  
 //Métodos de la clase Refugio  
  
 //Método constructor  
 public* Refugio(Log log, JTextField jTextFieldHormigasRefugio){  
 *this*.log = log;  
 *this*.listaHormigasRefugio = *new* ListaThreads(jTextFieldHormigasRefugio);  
 }  
  
 *//Método para que las crias usen el refugio  
 public void* protegeRefugio(Hormiga hormiga){  
 *try*{  
 cerrojoRefugio.lock();  
 *if*(isActivo()){  
 hormiga.getPaso().mirar();  
 entra(hormiga);  
 hormiga.getPaso().mirar();  
 *try*{  
 finInvasion.await();  
 hormiga.getPaso().mirar();  
 }*catch*(InterruptedException ignored){}  
 sale(hormiga);  
 }  
 }  
 *catch*(InterruptedException ignored){}  
 *finally*{  
 cerrojoRefugio.unlock();  
 }  
 }  
  
 *//Método para que las hormigas entren al refugio  
 private void* entra(Hormiga hormiga){  
 *//Añadimos a la hormiga a la lista del refugio* getListaHormigasRefugio().meterHormiga(hormiga);  
 setNumHormigasRefugio(getNumHormigasRefugio() + 1);  
 *//Una vez que esta metida, escribimos el evento en el log* getLog().escribirEnLog("[REFUGIO] --> La hormiga " + hormiga.getIdentificador() + " ha entrado al refugio");  
 }  
  
 *//Método para salir del refugio  
 private void* sale(Hormiga hormiga){  
 *//Para salir, quitamos al hormiga del JTextField correspondiente* getListaHormigasRefugio().sacarHormiga(hormiga);  
 setNumHormigasRefugio(getNumHormigasRefugio() - 1);  
 *//Una vez fuera del JTextField, escribimos el evento en el log* getLog().escribirEnLog("[REFUGIO] --> La hormiga " + hormiga.getIdentificador() + " ha salido del refugio");  
 *if*(getNumHormigasRefugio() == 0){  
 setActivo(*false*); *//Como el refugio está vacio, no estará activo* }  
 }  
  
 *//Método para despertar a las hormigas cria despues de la invasion  
 public void* despiertaRefugio(){  
 *try*{  
 cerrojoRefugio.lock();  
 finInvasion.signalAll(); *//Despertamos a las crias que están dentro del refugio* }  
 *finally*{  
 cerrojoRefugio.unlock();  
 }  
 }  
  
 *//Métodos get y set  
  
 public* Log getLog() {  
 *return* log;  
 }  
  
 *public* ListaThreads getListaHormigasRefugio() {  
 *return* listaHormigasRefugio;  
 }  
  
 *public boolean* isActivo() {  
 *try*{  
 cerrojoRefugio.lock();  
 *return this*.activo;  
 }*finally*{  
 cerrojoRefugio.unlock();  
 }  
 }  
  
 *public void* setActivo(*boolean* activo){  
 *this*.activo = activo;  
 }  
  
 *public int* getNumHormigasRefugio() {  
 *return* numHormigasRefugio;  
 }  
  
 *public void* setNumHormigasRefugio(*int* numHormigasRefugio) {  
 *this*.numHormigasRefugio = numHormigasRefugio;  
 }  
}

Clase ZonaComer:

*package* concurrencia;  
  
*import* javax.swing.\*;  
*import* java.util.concurrent.Semaphore;  
*import* java.util.concurrent.locks.Condition;  
*import* java.util.concurrent.locks.Lock;  
*import* java.util.concurrent.locks.ReentrantLock;  
  
*public class* ZonaComer {  
 *//Atrubutos de la clase ZonaComer  
 private final* Log log;  
 *private final* Semaphore semaforoEntradaSalida = *new* Semaphore(1, *true*); *//Semáforo para entrar y salir  
 private final* Lock cerrojoElementoComida = *new* ReentrantLock();  
 *private final* Condition esperaElementoComida = cerrojoElementoComida.newCondition();  
 *private int* numElementosComida = 0, numHormigasZonaComer = 0;  
 *private final* ListaThreads unidadesElementosComida;  
 *private final* ListaThreads listaHormigasZonaComer;  
  
 *//Métodos de la clase ZonaComer  
  
 //Método constructor  
 public* ZonaComer(Log log, JTextField jTextFieldUnidadesComidaZonaComer, JTextField jTextFieldHormigasZonaComer){  
 *this*.log = log;  
 *this*.unidadesElementosComida = *new* ListaThreads(jTextFieldUnidadesComidaZonaComer);  
 *this*.listaHormigasZonaComer = *new* ListaThreads(jTextFieldHormigasZonaComer);  
 }  
  
 *//Método para entrar a la zona para comer  
 public void* entra(Hormiga hormiga){  
 *try*{  
 semaforoEntradaSalida.acquire();  
 setNumHormigasZonaComer(getNumHormigasZonaComer() + 1); *//Incrementamos el numero de hormigas en la ZonaComer* getListaHormigasZonaComer().meterHormiga(hormiga);  
 getLog().escribirEnLog("[ZONA COMER] --> La hormiga " + hormiga.getIdentificador() + " ha entrado a la zona para comer");  
 semaforoEntradaSalida.release();  
 }  
 *catch*(InterruptedException ignored){}  
 }  
  
 *//Método para salir de la zona para comer  
 public void* sale(Hormiga hormiga){  
 *try*{  
 semaforoEntradaSalida.acquire();  
 getListaHormigasZonaComer().sacarHormiga(hormiga); *//Nos quitamos de la lista* getLog().escribirEnLog("[ZONA COMER] --> La hormiga " + hormiga.getIdentificador() + " ha salido de la zona para comer"); *//Escribimos el evento en el log* semaforoEntradaSalida.release();  
 }  
 *catch*(InterruptedException ignored){}  
 }  
  
 *//Método para depositar un elemento de comida  
 public void* depositaElementoComida(Hormiga hormiga){  
 *try*{  
 cerrojoElementoComida.lock();  
 setNumElementosComida(getNumElementosComida() + 5); *//Depositamos los 5 elementos de comida* getUnidadesElementosComida().insertarNumero(getNumElementosComida());  
 getLog().escribirEnLog("[ZONA COMER] --> La hormiga " + hormiga.getIdentificador() + " ha depositado un elemento de comida en la zona para comer");  
 *for*(*int* i = 0; i < 5; i++){  
 esperaElementoComida.signal(); *//Como hemos depositado 5 elementos de comida, despertaremos a 5 hormigas para que puedan coger su comida* }  
 }  
 *finally*{  
 cerrojoElementoComida.unlock();  
 }  
 }  
  
 *//Método para coger un elemento de comida  
 public void* cogeElementoComida(Hormiga hormiga) *throws* InterruptedException{  
 *try*{  
 cerrojoElementoComida.lock();  
 *if*(getNumElementosComida() <= 0){  
 esperaElementoComida.await(); *//Si no hay elementos de comida, tendrá que esperar a que haya uno* }  
 setNumElementosComida(getNumElementosComida() - 1); *//Cogemos el elemento de comida* getUnidadesElementosComida().insertarNumero(getNumElementosComida()); *//Imprimios el numero de elementos de comida* getLog().escribirEnLog("[ZONA COMER] --> La hormiga " + hormiga.getIdentificador() + " ha cogido un elemento de comida de la zona para comer");  
 }*finally*{  
 cerrojoElementoComida.unlock();  
 }  
 }  
  
 *//Métodos get y set  
  
 public* Log getLog() {  
 *return* log;  
 }  
  
 *public int* getNumElementosComida() {  
 *return* numElementosComida;  
 }  
  
 *public void* setNumElementosComida(*int* numElementosComida) {  
 *this*.numElementosComida = numElementosComida;  
 }  
  
 *public int* getNumHormigasZonaComer() {  
 *return* numHormigasZonaComer;  
 }  
  
 *public void* setNumHormigasZonaComer(*int* numHormigasZonaComer) {  
 *this*.numHormigasZonaComer = numHormigasZonaComer;  
 }  
  
 *public* ListaThreads getUnidadesElementosComida() {  
 *return* unidadesElementosComida;  
 }  
  
 *public* ListaThreads getListaHormigasZonaComer() {  
 *return* listaHormigasZonaComer;  
 }  
}

Clase ZonaDescanso:

*package* concurrencia;  
  
*import* javax.swing.\*;  
*import* java.util.concurrent.locks.Lock;  
*import* java.util.concurrent.locks.ReentrantLock;  
  
*public class* ZonaDescanso {  
 *//Atributos de la clase ZonaDescanso  
 private final* Log log;  
 *private final* Lock entradaSalida = *new* ReentrantLock();  
 *private final* ListaThreads listaHormigasDescansando;  
  
 *//Métodos de la clase ZonaDescanso  
  
 //Método constructor  
 public* ZonaDescanso(Log log, JTextField jTextFieldHormigasDescansando){  
 *this*.log = log;  
 *this*.listaHormigasDescansando = *new* ListaThreads(jTextFieldHormigasDescansando);  
 }  
  
 *//Método para entrar a la zona de descanso  
 public void* entra(Hormiga hormiga){  
 entradaSalida.lock();  
 *try*{  
 *//Para entrar, metemos el identificador de la hormiga en el JTextField correspondiente* getListaHormigasDescansando().meterHormiga(hormiga);  
 *//Una vez dentro, escribimos el evento en el log* getLog().escribirEnLog("[Zona Descanso] --> La hormiga " + hormiga.getIdentificador() + " ha entrado a la zona de descanso");  
  
 }*finally*{  
 entradaSalida.unlock();  
 }  
 }  
  
 *//Método para salir de la zona de descanso  
 public void* sale(Hormiga hormiga) *throws* InterruptedException{  
 entradaSalida.lock();  
 *try*{  
 *//Para salir, tenemos que quitar el identificador de la hormiga del JTextField correspondiente* getListaHormigasDescansando().sacarHormiga(hormiga);  
 *//Una vez fuera, escribimos el evento en el log* getLog().escribirEnLog("[ZONA DESCANSO] --> La hormiga " + hormiga.getIdentificador() + " ha salido de la zona de descanso");  
  
 }*finally*{  
 entradaSalida.unlock();  
 }  
 }  
  
 *//Método para que una hormiga haga su descanso  
 public void* realizaDescanso(Hormiga hormiga) *throws* InterruptedException{  
 *//Escribimos el evento en el log* getLog().escribirEnLog("[ZONA DESCANSO] --> La hormiga " + hormiga.getIdentificador() + " esta descansando");  
 *if* (hormiga.getTipo().equals("Soldada")){  
 *//Una hormiga soldada descansa un tiempo aleatorio entre 2 y 8 segundos* Thread.sleep((*int*) (((Math.random() + 1) \* 2000) + (8000 - (2000 \* 2))));  
 }  
 *else if* (hormiga.getTipo().equals("Obrera")){  
 *//Una hormiga obrera realiza un descanso de 1 segundo* Thread.sleep(1000);  
 }  
 *else*{  
 *//Una hormiga cria realiza un descanso de 4 segundos* Thread.sleep(4000);  
 }  
 }  
  
 *//Métodos get y set  
  
 public* Log getLog() {  
 *return* log;  
 }  
  
 *public* ListaThreads getListaHormigasDescansando() {  
 *return* listaHormigasDescansando;  
 }  
}

Clase ZonaInstruccion:

*package* concurrencia;  
  
*import* javax.swing.\*;  
*import* java.util.concurrent.locks.Lock;  
*import* java.util.concurrent.locks.ReentrantLock;  
  
*public class* ZonaInstruccion {  
 *//Atributos de la clase ZonaInstruccion  
 private final* Log log;  
 *private final* Lock entradaSalida = *new* ReentrantLock();  
 *private final* ListaThreads listaHormigasHaciendoInstruccion;  
  
 *//Métodos de la clase ZonaInstruccion  
  
 //Método constructor  
 public* ZonaInstruccion(Log log, JTextField jTextFieldHormigasHaciendoInstruccion){  
 *this*.log = log;  
 *this*.listaHormigasHaciendoInstruccion = *new* ListaThreads(jTextFieldHormigasHaciendoInstruccion);  
 }  
  
 *//Método para entrar a la zona de instruccion  
 public void* entra(Hormiga hormiga){  
 entradaSalida.lock();  
 *try*{  
 *//Cuando entra una hormiga, mete su ID al JTextField correspondiente mediante el ListaThreads* getListaHormigasHaciendoInstruccion().meterHormiga(hormiga);  
 *//Una vez metida, escribe el evento en el log* getLog().escribirEnLog("[Zona Instruccion] --> La hormiga " + hormiga.getIdentificador() + " ha entrado a la zona de instruccion");  
  
 }*finally*{  
 entradaSalida.unlock();  
 }  
 }  
 *//Método para salir de la zona de instruccion  
 public void* sale(Hormiga hormiga) *throws* InterruptedException{  
 entradaSalida.lock();  
 *try*{  
 *//Para salir, en primer lugar tenemos que quitar su ID del JTextField* getListaHormigasHaciendoInstruccion().sacarHormiga(hormiga);  
 *//Una vez fuera, escribe el mensaje en el log* getLog().escribirEnLog("[Zona Instruccion] --> La hormiga " + hormiga.getIdentificador() + " ha salido de la zona de instruccion");  
  
 }*finally*{  
 entradaSalida.unlock();  
 }  
 }  
  
 *//Método para hacer instruccion  
 public void* realizaInstruccion(Hormiga hormiga) *throws* InterruptedException{  
 *//Hacen instrucción entre 2 y 8 segundos* getLog().escribirEnLog("[Zona Instruccion] --> La hormiga " + hormiga.getIdentificador() + " esta haciendo instruccion");  
 Thread.sleep((*int*) (((Math.random() + 1) \* 2000) + (8000 - (2000 \* 2))));  
 getLog().escribirEnLog("[Zona Instruccion] --> La hormiga " + hormiga.getIdentificador() + " ha terminado instruccion");  
 }  
 *//Métodos get y set  
  
 public* Log getLog() {  
 *return* log;  
 }  
  
 *public* ListaThreads getListaHormigasHaciendoInstruccion() {  
 *return* listaHormigasHaciendoInstruccion;  
 }  
}

Clase ClienteDeColonia:

*/\*  
 \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  
 \* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template  
 \*/  
package* distribuida;  
  
*import* javax.swing.\*;  
*import* java.rmi.Naming;  
  
*/\*\*  
 \*  
 \* @author victorsanavia  
 \*/  
public class* ClienteDeColonia *extends* javax.swing.JFrame {  
  
 */\*\*  
 \* Creates new form ClienteDeColonia  
 \*/  
 public* ClienteDeColonia() {  
 initComponents();  
 }  
  
 *public void* actualizaJTextFields(){  
 *try*{  
 InterfazColonia objetoRemoto = (InterfazColonia) Naming.lookup("//localhost/ObjetoColonia");  
 getjTextFieldNumHormigasObrerasExteriorColonia().setText(objetoRemoto.numHormigasObrerasExteriorColonia());  
 getjTextFieldNumHormigasObrerasInteriorColonia().setText(objetoRemoto.numHormigasObrerasInteriorColonia());  
 getjTextFieldNumHormigasSoldadoInstruccion().setText(objetoRemoto.numHormigasSoldadoInstruccion());  
 getjTextFieldNumHormigasSoldadoInvasion().setText(objetoRemoto.numHormigasSoldadoInvasion());  
 getjTextFieldNumHomrigasCriaZonaComer().setText(objetoRemoto.numHormigasCriaZonaComer());  
 getjTextFieldNumHomrigasCriaRefugio().setText(objetoRemoto.numHormigasCriaRefugio());  
 }*catch* (Exception ignored){}  
 }  
  
 */\*\*  
 \* 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">//GEN-BEGIN:initComponents  
 private void* initComponents() {  
  
 jPanel1 = *new* javax.swing.JPanel();  
 panelNumHomrigasObrerasExteriorColonia = *new* javax.swing.JPanel();  
 labelNumHormigasObrerasExteriorColonia = *new* javax.swing.JLabel();  
 jTextFieldNumHormigasObrerasExteriorColonia = *new* javax.swing.JTextField();  
 panelNumHomrigasObrerasInteriorColonia = *new* javax.swing.JPanel();  
 labelNumHormigasObrerasInteriorColonia = *new* javax.swing.JLabel();  
 jTextFieldNumHormigasObrerasInteriorColonia = *new* javax.swing.JTextField();  
 panelNumHomrigasSoldadoInstruccion = *new* javax.swing.JPanel();  
 labelNumHormigasSoldadoInstruccion = *new* javax.swing.JLabel();  
 jTextFieldNumHormigasSoldadoInstruccion = *new* javax.swing.JTextField();  
 panelNumHomrigasSoldadoInvasion = *new* javax.swing.JPanel();  
 labelNumHormigasSoldadoInvasion = *new* javax.swing.JLabel();  
 jTextFieldNumHormigasSoldadoInvasion = *new* javax.swing.JTextField();  
 panelNumHomrigasCriaZonaComer = *new* javax.swing.JPanel();  
 labelNumHomrigasCriaZonaComer = *new* javax.swing.JLabel();  
 jTextFieldNumHomrigasCriaZonaComer = *new* javax.swing.JTextField();  
 panelNumHomrigasCriaRefugio = *new* javax.swing.JPanel();  
 labelNumHomrigasCriaRefugio = *new* javax.swing.JLabel();  
 jTextFieldNumHomrigasCriaRefugio = *new* javax.swing.JTextField();  
 jPanel2 = *new* javax.swing.JPanel();  
 jButtonGeneraInvasion = *new* javax.swing.JButton();  
  
 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);  
 getContentPane().setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 jPanel1.setBackground(*new* java.awt.Color(255, 255, 255));  
 jPanel1.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 panelNumHomrigasObrerasExteriorColonia.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 labelNumHormigasObrerasExteriorColonia.setText("Numero de hormigas obreras en el exterior de la colonia");  
 panelNumHomrigasObrerasExteriorColonia.add(labelNumHormigasObrerasExteriorColonia, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(100, 10, 310, -1));  
  
 jTextFieldNumHormigasObrerasExteriorColonia.setText("0");  
 panelNumHomrigasObrerasExteriorColonia.add(jTextFieldNumHormigasObrerasExteriorColonia, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(210, 40, 80, 30));  
  
 jPanel1.add(panelNumHomrigasObrerasExteriorColonia, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, 510, 100));  
  
 panelNumHomrigasObrerasInteriorColonia.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 labelNumHormigasObrerasInteriorColonia.setText("Numero de hormigas obreras en el interior de la colonia");  
 panelNumHomrigasObrerasInteriorColonia.add(labelNumHormigasObrerasInteriorColonia, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(100, 10, 310, -1));  
  
 jTextFieldNumHormigasObrerasInteriorColonia.setText("0");  
 panelNumHomrigasObrerasInteriorColonia.add(jTextFieldNumHormigasObrerasInteriorColonia, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(210, 40, 80, 30));  
  
 jPanel1.add(panelNumHomrigasObrerasInteriorColonia, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(530, 10, 510, 100));  
  
 panelNumHomrigasSoldadoInstruccion.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 labelNumHormigasSoldadoInstruccion.setText("Numero de hormigas soldado haciendo instruccion");  
 panelNumHomrigasSoldadoInstruccion.add(labelNumHormigasSoldadoInstruccion, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(120, 10, 290, -1));  
  
 jTextFieldNumHormigasSoldadoInstruccion.setText("0");  
 panelNumHomrigasSoldadoInstruccion.add(jTextFieldNumHormigasSoldadoInstruccion, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(210, 40, 80, 30));  
  
 jPanel1.add(panelNumHomrigasSoldadoInstruccion, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(10, 120, 510, 100));  
  
 panelNumHomrigasSoldadoInvasion.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 labelNumHormigasSoldadoInvasion.setText("Numero de hormigas soldado repeliendo una invasion");  
 panelNumHomrigasSoldadoInvasion.add(labelNumHormigasSoldadoInvasion, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(110, 10, 290, -1));  
  
 jTextFieldNumHormigasSoldadoInvasion.setText("0");  
 panelNumHomrigasSoldadoInvasion.add(jTextFieldNumHormigasSoldadoInvasion, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(210, 40, 80, 30));  
  
 jPanel1.add(panelNumHomrigasSoldadoInvasion, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(530, 120, 510, 100));  
  
 panelNumHomrigasCriaZonaComer.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 labelNumHomrigasCriaZonaComer.setText("Numero de hormigas cria en la ZONA PARA COMER");  
 panelNumHomrigasCriaZonaComer.add(labelNumHomrigasCriaZonaComer, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(120, 10, 290, -1));  
  
 jTextFieldNumHomrigasCriaZonaComer.setText("0");  
 panelNumHomrigasCriaZonaComer.add(jTextFieldNumHomrigasCriaZonaComer, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(210, 40, 80, 30));  
  
 jPanel1.add(panelNumHomrigasCriaZonaComer, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(10, 230, 510, 100));  
  
 panelNumHomrigasCriaRefugio.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 labelNumHomrigasCriaRefugio.setText("Numero de hormigas cria en el REFUGIO");  
 panelNumHomrigasCriaRefugio.add(labelNumHomrigasCriaRefugio, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(150, 10, 220, -1));  
  
 jTextFieldNumHomrigasCriaRefugio.setText("0");  
 panelNumHomrigasCriaRefugio.add(jTextFieldNumHomrigasCriaRefugio, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(210, 40, 80, 30));  
  
 jPanel1.add(panelNumHomrigasCriaRefugio, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(530, 230, 510, 100));  
  
 jPanel2.setLayout(*new* org.netbeans.lib.awtextra.AbsoluteLayout());  
  
 jButtonGeneraInvasion.setText("Generar Amenaza Insecto Invasor");  
 jButtonGeneraInvasion.setCursor(*new* java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));  
 jButtonGeneraInvasion.addActionListener(*new* java.awt.event.ActionListener() {  
 *public void* actionPerformed(java.awt.event.ActionEvent evt) {  
 jButtonGeneraInvasionActionPerformed(evt);  
 }  
 });  
 jPanel2.add(jButtonGeneraInvasion, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(10, 20, 410, 40));  
  
 jPanel1.add(jPanel2, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(310, 340, 430, 80));  
  
 getContentPane().add(jPanel1, *new* org.netbeans.lib.awtextra.AbsoluteConstraints(6, 5, 1050, 440));  
  
 pack();  
 }*// </editor-fold>//GEN-END:initComponents  
  
 private void* jButtonGeneraInvasionActionPerformed(java.awt.event.ActionEvent evt) {*//GEN-FIRST:event\_jButtonGeneraInvasionActionPerformed  
 // TODO add your handling code here:  
 try*{  
 InterfazColonia objetoRemoto = (InterfazColonia) Naming.lookup("//localhost/ObjetoColonia");  
 objetoRemoto.generaInvasion();  
 }*catch* (Exception ignored){}  
 }*//GEN-LAST:event\_jButtonGeneraInvasionActionPerformed  
  
 /\*\*  
 \* @param args the command line arguments  
 \*/  
 public static void* main(String args[]) {  
 */\* Set the Nimbus look and feel \*/  
 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html   
 \*/  
 try* {  
 *for* (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  
 *if* ("Nimbus".equals(info.getName())) {  
 javax.swing.UIManager.setLookAndFeel(info.getClassName());  
 *break*;  
 }  
 }  
 } *catch* (ClassNotFoundException ex) {  
 java.util.logging.Logger.getLogger(ClienteDeColonia.*class*.getName()).log(java.util.logging.Level.SEVERE, *null*, ex);  
 } *catch* (InstantiationException ex) {  
 java.util.logging.Logger.getLogger(ClienteDeColonia.*class*.getName()).log(java.util.logging.Level.SEVERE, *null*, ex);  
 } *catch* (IllegalAccessException ex) {  
 java.util.logging.Logger.getLogger(ClienteDeColonia.*class*.getName()).log(java.util.logging.Level.SEVERE, *null*, ex);  
 } *catch* (javax.swing.UnsupportedLookAndFeelException ex) {  
 java.util.logging.Logger.getLogger(ClienteDeColonia.*class*.getName()).log(java.util.logging.Level.SEVERE, *null*, ex);  
 }  
 *//</editor-fold>* ClienteDeColonia mainDistribuida = *new* ClienteDeColonia();  
  
 */\* Create and display the form \*/* java.awt.EventQueue.invokeLater(*new* Runnable() {  
 *public void* run() {  
 mainDistribuida.setVisible(*true*);  
 }  
 });  
 *while*(*true*){  
 mainDistribuida.actualizaJTextFields();  
 *try*{  
 Thread.sleep(1500);  
 }*catch*(InterruptedException ignored){}  
 }  
 }  
  
 *// Variables declaration - do not modify//GEN-BEGIN:variables  
 private* javax.swing.JButton jButtonGeneraInvasion;  
 *private* javax.swing.JPanel jPanel1;  
 *private* javax.swing.JPanel jPanel2;  
 *private* javax.swing.JTextField jTextFieldNumHomrigasCriaRefugio;  
 *private* javax.swing.JTextField jTextFieldNumHomrigasCriaZonaComer;  
 *private* javax.swing.JTextField jTextFieldNumHormigasObrerasExteriorColonia;  
 *private* javax.swing.JTextField jTextFieldNumHormigasObrerasInteriorColonia;  
 *private* javax.swing.JTextField jTextFieldNumHormigasSoldadoInstruccion;  
 *private* javax.swing.JTextField jTextFieldNumHormigasSoldadoInvasion;  
 *private* javax.swing.JLabel labelNumHomrigasCriaRefugio;  
 *private* javax.swing.JLabel labelNumHomrigasCriaZonaComer;  
 *private* javax.swing.JLabel labelNumHormigasObrerasExteriorColonia;  
 *private* javax.swing.JLabel labelNumHormigasObrerasInteriorColonia;  
 *private* javax.swing.JLabel labelNumHormigasSoldadoInstruccion;  
 *private* javax.swing.JLabel labelNumHormigasSoldadoInvasion;  
 *private* javax.swing.JPanel panelNumHomrigasCriaRefugio;  
 *private* javax.swing.JPanel panelNumHomrigasCriaZonaComer;  
 *private* javax.swing.JPanel panelNumHomrigasObrerasExteriorColonia;  
 *private* javax.swing.JPanel panelNumHomrigasObrerasInteriorColonia;  
 *private* javax.swing.JPanel panelNumHomrigasSoldadoInstruccion;  
 *private* javax.swing.JPanel panelNumHomrigasSoldadoInvasion;  
 *// End of variables declaration//GEN-END:variables  
  
 //Métodos get de los JTextField  
  
 public* JTextField getjTextFieldNumHomrigasCriaRefugio() {  
 *return* jTextFieldNumHomrigasCriaRefugio;  
 }  
  
 *public* JTextField getjTextFieldNumHomrigasCriaZonaComer() {  
 *return* jTextFieldNumHomrigasCriaZonaComer;  
 }  
  
 *public* JTextField getjTextFieldNumHormigasObrerasExteriorColonia() {  
 *return* jTextFieldNumHormigasObrerasExteriorColonia;  
 }  
  
 *public* JTextField getjTextFieldNumHormigasObrerasInteriorColonia() {  
 *return* jTextFieldNumHormigasObrerasInteriorColonia;  
 }  
  
 *public* JTextField getjTextFieldNumHormigasSoldadoInstruccion() {  
 *return* jTextFieldNumHormigasSoldadoInstruccion;  
 }  
  
 *public* JTextField getjTextFieldNumHormigasSoldadoInvasion() {  
 *return* jTextFieldNumHormigasSoldadoInvasion;  
 }  
}

Clase GestorInterfaz:

*package* distribuida;  
  
*import* concurrencia.\*;  
  
*import* java.rmi.RemoteException;  
*import* java.rmi.server.UnicastRemoteObject;  
*import* java.util.ArrayList;  
  
*public class* GestorInterfaz *extends* UnicastRemoteObject *implements* InterfazColonia {  
 *//Métodos de la clase colonia  
 private final* Colonia colonia;  
  
 *//Método constructor  
 public* GestorInterfaz(Colonia colonia) *throws* RemoteException {  
 *this*.colonia = colonia;  
 }  
  
 *//Método que devuelve el numero de hormigas obreras en el exterior de la colonia  
 public* String numHormigasObrerasExteriorColonia() *throws* RemoteException {  
 ArrayList<Hormiga> lista = getColonia().getListaHormigasBuscandoComida().getListaHormigas();  
 *return* cuentaHormigasArraylist(lista);  
 }  
  
 *//Método que devuelve el numero de hormigas obreras dentro de la colonia  
 public* String numHormigasObrerasInteriorColonia() *throws* RemoteException {  
 ArrayList<Hormiga> lista = getColonia().getListaHormigas();  
 *return* cuentaHormigasTipoArraylist(lista, "Obrera");  
 }  
  
 *//Método que devuelve el numero de hormigas soldado haciendo instruccion  
 public* String numHormigasSoldadoInstruccion() *throws* RemoteException {  
 ZonaInstruccion zonaInstruccion = getColonia().getZonaInstruccion();  
 ArrayList<Hormiga> lista = zonaInstruccion.getListaHormigasHaciendoInstruccion().getListaHormigas();  
 *return* cuentaHormigasArraylist(lista);  
 }  
  
 *//Método que devuelve el numero de hormigas luchando en la invasion  
 public* String numHormigasSoldadoInvasion() *throws* RemoteException {  
 Invasion invasion = getColonia().getInvasion();  
 ArrayList<Hormiga> lista = invasion.getListaHormigasInvasion().getListaHormigas();  
 *return* cuentaHormigasArraylist(lista);  
 }  
  
 *//Método que devuelve el número de hormigas cría en la zona para comer  
 public* String numHormigasCriaZonaComer() *throws* RemoteException {  
 ZonaComer zonaComer = getColonia().getZonaComer();  
 ArrayList<Hormiga> lista = zonaComer.getListaHormigasZonaComer().getListaHormigas();  
 *return* cuentaHormigasTipoArraylist(lista, "Cria");  
 }  
  
 *//Método que devuelve el número de hormigas cría en el refugio  
 public* String numHormigasCriaRefugio() *throws* RemoteException {  
 Refugio refugio = getColonia().getRefugio();  
 ArrayList<Hormiga> lista = refugio.getListaHormigasRefugio().getListaHormigas();  
 *return* cuentaHormigasArraylist(lista);  
 }  
  
 *//Método que genera una invasión  
 public void* generaInvasion() *throws* RemoteException {  
 getColonia().generaInvasion();  
 }  
  
 *//Método get de la colonia  
 public* Colonia getColonia(){  
 *return this*.colonia;  
 }  
  
 *private* String cuentaHormigasArraylist(ArrayList<Hormiga> lista){  
 String resultado;  
 *int* num;  
 num = lista.size();  
 resultado = String.valueOf(num);  
 *return* resultado;  
 }  
  
 *private* String cuentaHormigasTipoArraylist(ArrayList<Hormiga> lista, String tipo){  
 *int* num = 0;  
 *for* (Hormiga hormigaActual : lista) {  
 *if* (hormigaActual.getTipo().equals(tipo)) {  
 num = num + 1;  
 }  
 }  
 *return* String.valueOf(num);  
 }  
}

Clase InterfazColonia:

*package* distribuida;  
  
*import* java.rmi.Remote;  
*import* java.rmi.RemoteException;  
  
*public interface* InterfazColonia *extends* Remote {  
 String numHormigasObrerasExteriorColonia() *throws* RemoteException;  
 String numHormigasObrerasInteriorColonia() *throws* RemoteException;  
 String numHormigasSoldadoInstruccion() *throws* RemoteException;  
 String numHormigasSoldadoInvasion() *throws* RemoteException;  
 String numHormigasCriaZonaComer() *throws* RemoteException;  
 String numHormigasCriaRefugio() *throws* RemoteException;  
 *void* generaInvasion() *throws* RemoteException;  
   
}