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](#_Toc133748712)

[Herramientas de sincronización utilizadas y diseño general del sistema 4](#_Toc133748713)

[Descripción de las clases principales 7](#_Toc133748714)

[Diagrama de clases 8](#_Toc133748715)

[Código fuente 9](#_Toc133748716)

# 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.

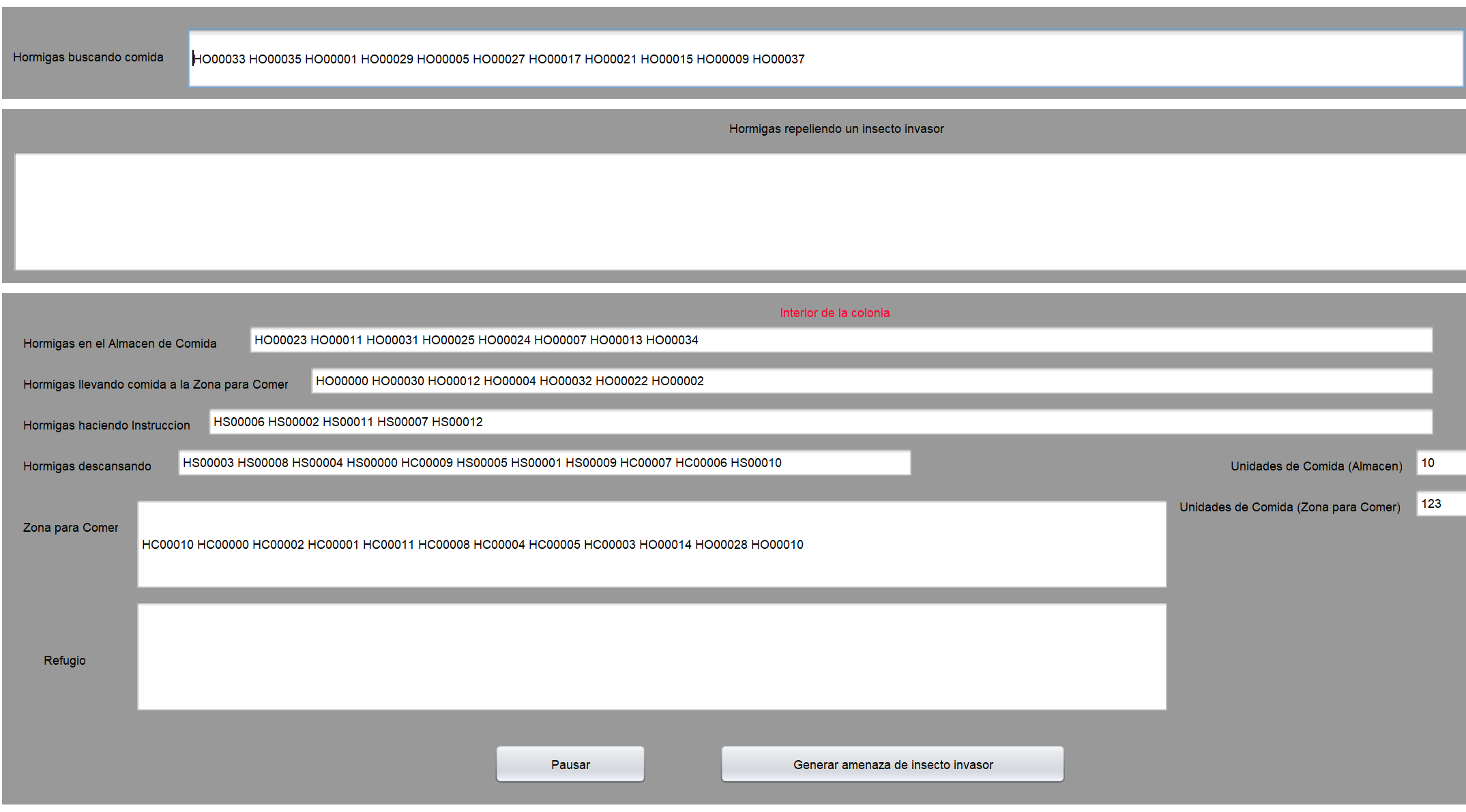
# Herramientas de sincronización utilizadas y diseño general del sistema

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

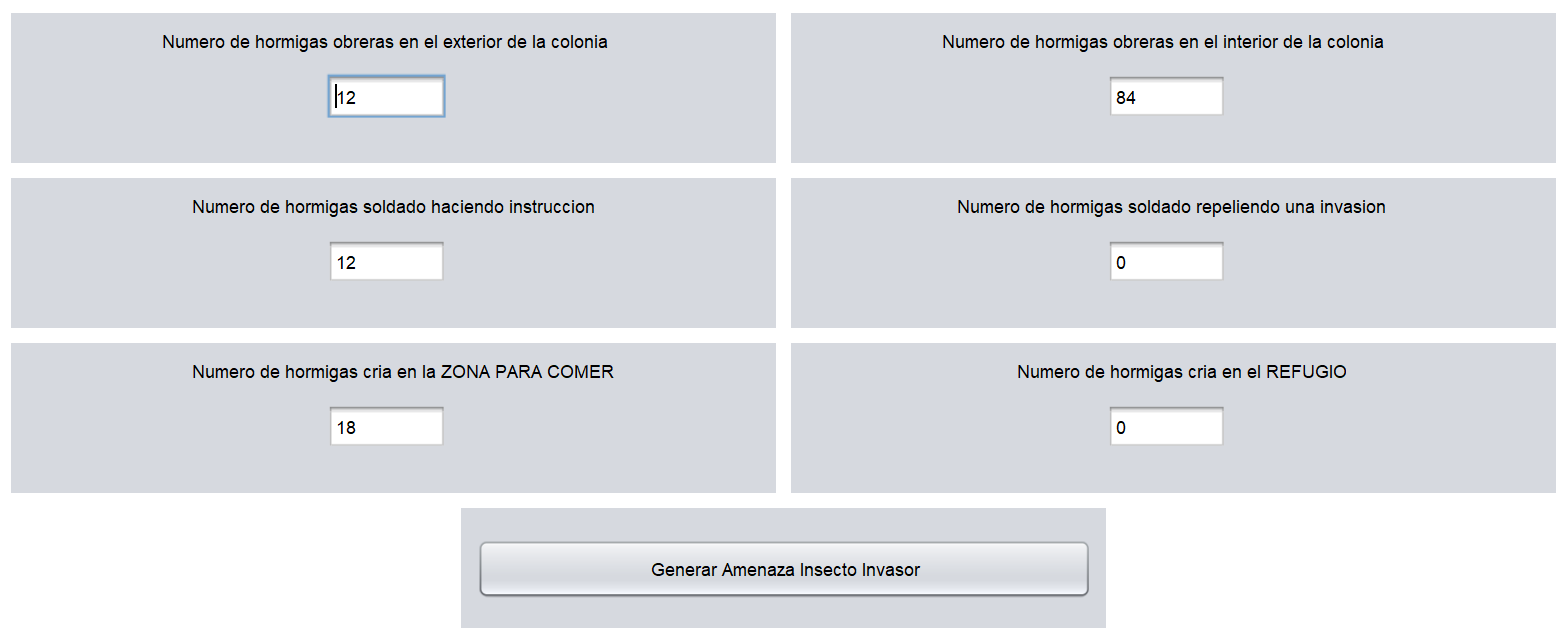
* ***AlmacenComida*** 🡪 Para la clase *AlmacenComida* hemos utilizado las siguientes herramientas:
  + Semáforo *semaforoEntradaSalida:* para asegurarnos mediante los permisos del semáforo que las hormigas no superen el aforo máximo del almacén.
  + Cerrojo *recogeElemento* con Condición *conditionElementoComida:* 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.

# Diagrama de clases



# Código fuente