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.

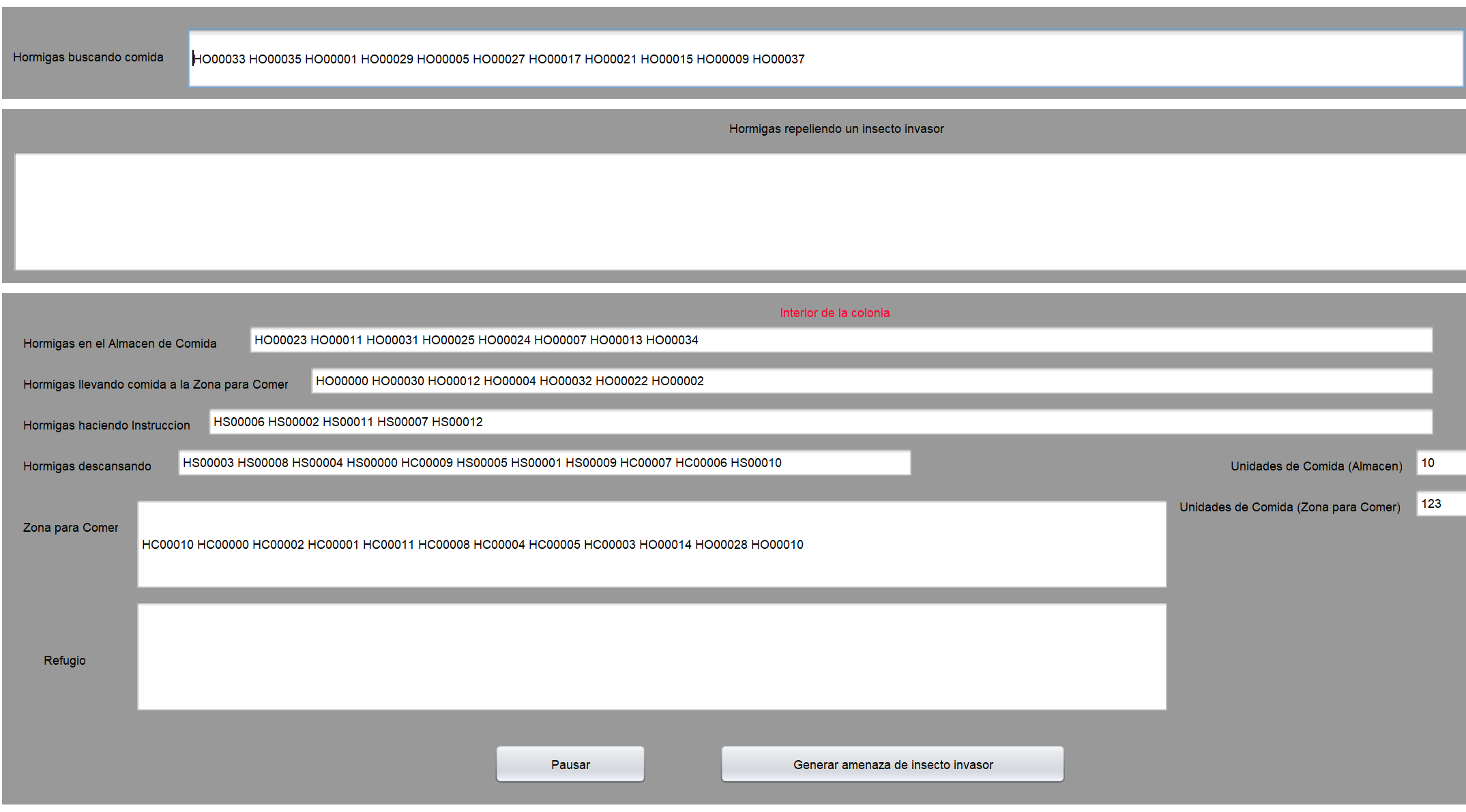
# Herramientas de sincronización utilizadas y diseño general del sistema

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

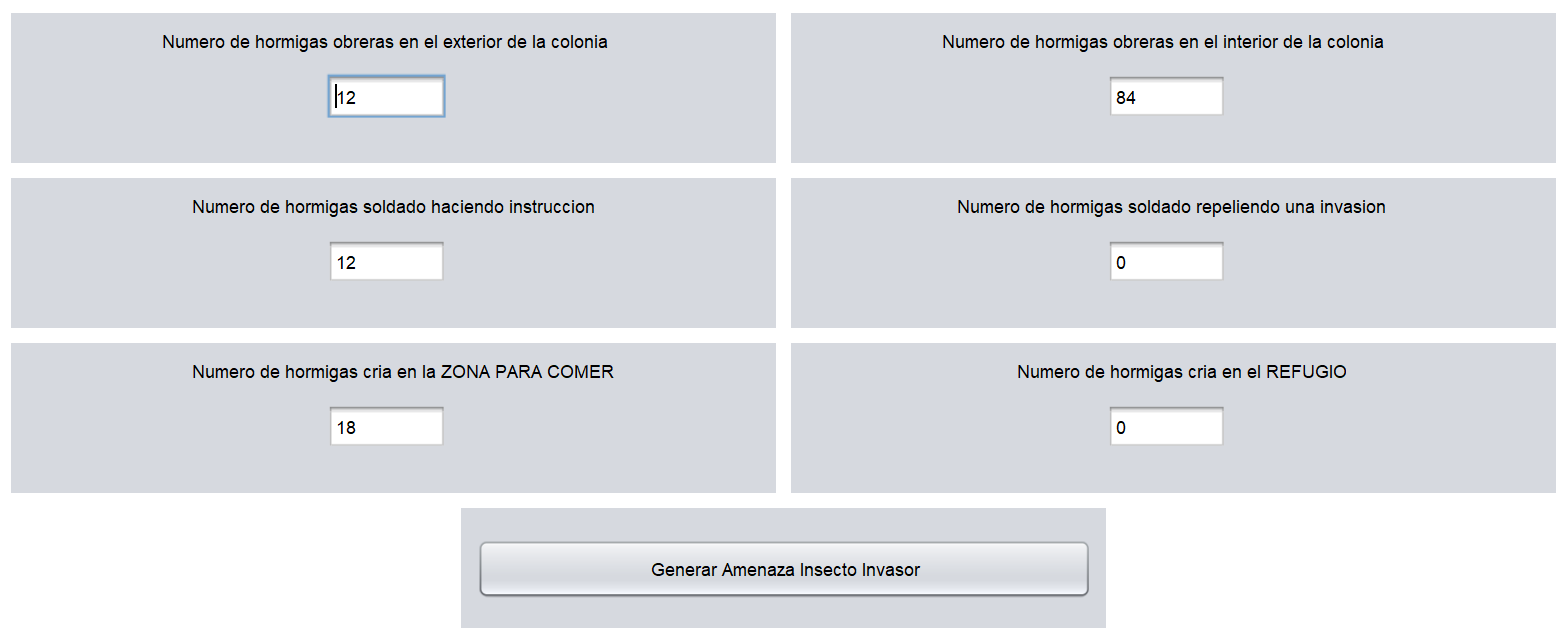
* ***AlmacenComida*** 🡪 Para la clase *AlmacenComida* hemos utilizado las siguientes herramientas:
  + Semáforo *semaforoEntradaSalida:* para asegurarnos mediante los permisos del semáforo que las hormigas no superen el aforo máximo del almacén.
  + Cerrojo *recogeElemento* con Condición *conditionElementoComida:*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

# Diagrama de clases



# Código fuente