Programación Avanzada

Curso 2022/2023 - Convocatoria Ordinaria

Duque Rey, Tian

03210649X

Índice

Análisis de alto nivel y diseño general del sistema

Parte 1: Programación Concurrente

Exterior Hormiguero

Se realizarán 3 instancias de hormigas (Obreras, Soldados, Crías) en total tendremos 10.000 hormigas. Se debe cumplir que cada 3 hormigas obreras generadas se crean un soldado y una cría. Por tanto, tendremos en conjunto de 6.000 obreras, 2.000 soldados y 2.000 crías. Para cada hormiga generada deberemos introducir un tiempo de espera de 0.8 – 3.5 (s) --> Thread.sleep((long) (800 + Math.random()*2700));

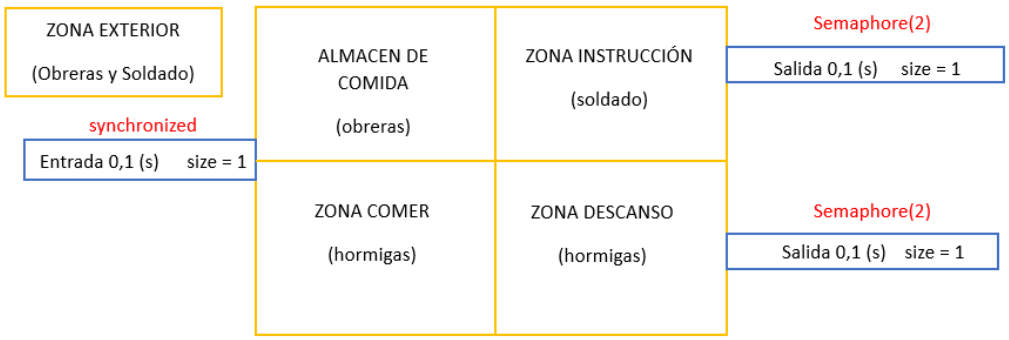
La primera tarea de las obreras será SALIR ("salirHormiguero()") del hormiguero para recolectarComida(). Recolectará 5 elementos de comida y se dirigirá al almacén (Estructura detallada más adelante).

Otro de los métodos de la zonaExterior es defenderInvasor() donde tras pulsar el JBottom "Generar Insecto Invasor", se producirá una InterruptionException que trataremos para que en primer lugar las hormigas soldado salgan al exterior, esperen en un countDownLacth y finalmente acaben con el InsectoInvasor.class . A su vez las hormigas crías durante la InterruptionException se mantendrán en el refugio hasta que InsectoInvasor.class sea repelido.

Métodos	Sleep	Herramientas de sincronización
entrarHormiguero()	0,1 (s)	synchronized
recolectarComida()	4 (s)	synchronized
defenderInvasor()	20 (s)	synchronized, CountDownLatch

Hormiguero

A continuación se representa un esquema general de la estructura interna del hormiguero. Esta clase se compone de una clase principal Hormiguero.class y 6 InnerClases (zonaAlmacen.innerClass, zonaIntruccion.innerClass, zonaComer.innerClass, zonaDescanso.innerClass, zonaRefugio.innerClass y zonaExterior.innerClass). La clase Hormiguero tiene dos métodos salirHormiguero() y viajarZonaComer() que permiten simular el comportamiento de las transiciones entre las diferentes zonas del hormiguero. Se ha incluido la zonaRefugio.innerClass como una parte interna del hormiguero para que las crías puedan refugiarse. Cada zona del hormiguero tiene como etiqueta el tipo de hormigas que pueden estar dentro realizando actividades. Las etiquetan son importantes de cara a las implementaciones y optimización del código, supone un punto fundamental en el comportamiento. Por ejemplo, en el tratamiento de las Interrupciones en la zonaInstruccion solo contendrá excepciones para la pausa y la generación de un insecto, mientras que en la zonaAlmacen solo contendrá excepciones de pausa.



El hormiguero consta de un túnel de entrada y dos de salida que tienen capacidad para un solo hilo y consumen un tiempo de 0,1 (s) --> Thread.sleep(100). En rojo podemos ver las herramientas de sincronización empleadas para cada componente, teniendo en cuenta que en el programa solo se ha hecho instancia de un semaphore(2) de dos permisos para simular el comportamiento de los dos túneles de salida.

Clases	Métodos	Sleep	Herramientas de sincronización
Hormiguero	salirHormiguero()	0.1 (s)	Synchronized
	viajarZonaComer()	1 -2 (s)	Synchronized
ZonaAlmacen	almacenarComida()	2- 4 (s)	Lock Condition
	extraerComida()	1-2 (s)	Lock Condition
ZonaInstruccion	hacerInstruccion()	2-8 (s)	Synchronized
ZonaDescanso	descansar()	Soldado = 2 (s)	Synchronized
		Crias = 4 (s)	Synchronized
		Obreras = 1 (s)	Synchronized
ZonaComer	depositarComida()	1-2 (s)	Synchronized, Lock Condition
	comer()	Soldado y Obreras = 3 (s)	Synchronized, Lock Condition
		Crias = 3-5 (s)	Synchronized, Lock Condition
ZonaRefugio	Refugio()	InsectoInvasor repelido	Synchronized
ZonaExterior	entrarHormiguero()	0.1 (s)	Synchronized
	recolectarComida()	4 (s)	Synchronized
	defenderInvasor()	20 (s)	Synchronized

Tabla de resumen de los métodos de las diferentes clases y principales características

La descripción de los métodos de cada clase las agruparemos en funciona del tipo de hormiga a medida que realizan el flujo de tareas:

- Las hormigas obreras son identificadas como "HOXXXX", donde X es un número único. Las hormigas impares *recolectarComida()* cinco elementos de comida para *almacenarComida()* en el ALMACÉN DE COMIDA. Las hormigas obreras pares *extraerComida()* cinco elementos de comida desde el ALMACÉN DE COMIDA para *depositarComida()* en la ZONA PARA COMER tras viajarZonaComer() donde consumen entre uno y dos segundo. Después de 10 iteraciones completas, todas las hormigas obreras pasan por la ZONA PARA COMER --> comer(), por consiguiente pasan por la ZONA DE DESCANSO --> *descansar()* antes de volver a su actividad habitual.

- Las hormigas soldado se identificarán como "HSXXXX", donde X es un número único. Cada una de ellas, una vez unida a la colonia, repetirá iterativamente: *hacerInstruccion()* en la ZONA DE INSTRUCCIÓN y descansará *descansar()* durante 2 segundos. Cada seis iteraciones, pasarán por la ZONA PARA COMER a reponer fuerzas y *comer(),* consumiendo una unidad de alimento en 3 segundos.

  Si un insecto invasor amenaza la colonia, solo las hormigas soldado generadas antes de pulsar el botón “Generar Insecto invasor” saldrán inmediatamente para *defenderInvasor().* Llevado a la implementación se ha creado un hilo aparte del resto Main Soldado en la clase GenerarSoldado que contiene un hilo encargado de genera hilosSoldado. De este modo, los soldados actúan de forma independiente del resto.

  Finalmente, las hormigas soldado continúan con su comportamiento normal sin alterar el contador de iteraciones. Para ello, trataremos las Interrupciones en las respectivas zonas en la que están realizando la actividad ZONA DE INSTRUCCIÓN o la ZONA PARA COMER. De modo que cuando retorne del manejo de la excepción continúen con la ejecución normal del programa. Las hormigas obreras no se verán afectadas por la presencia de un insecto invasor.

- Las hormigas crías se identifican como "HCXXXX" y acceden a la zona de alimentación para *comer()* durante 3-5 segundos, descansando --> *descansar()* durante 4 segundos en la zona de descanso. Si hay una amenaza de un insecto invasor, todas las hormigas crías deben ir rápidamente a la ZONEA REFUGIO --> *refugio()* hasta que la amenaza desaparezca.

Principales actores

Entre los principales actores encontramos a las hormigas e insecto invasor. En cuanto al diseño del programa se ha hecho una instancia de Hormigas.class que contiene a su vez Obrera.innerClass, Soldado.innerClass y Crias.innerClass. Estas clases internas son hilos que ejecutan los diferentes métodos dentro del hormiguero. La clase Obreras divide sus actividades entre hormigas pares e impares, entre sus características destables respecto a su diseño destaca el manejo de excepciones en la propia clase, y no en los métodos que ejecutan a diferencia de las Soldado y las Crías. Además, la propia clase es la encargada de implementar un monitor para gestionar la cantidad de hormigas interactuando en el almacén. La clase soldado se encarga de ejecutar la tareas principales de la hormiga: *hacerInstruccion()*, *comer()*, *descansar()*. Por último, en cuanto a las crías el hilo se encarga de bloquear las tareas y enviar al refugio los hilos creados durante la invasión de un insecto. Además, la clase se dedica a ejecutar las principales tareas: *comer()* y *descansar().*

La instancia de la clase principal Hormigas contiene los métodos responsables de *detener()* y *reanudar()* la ejecución del programa. Los hilos detenidos se quedarán a la espera en un await en el método *hilosSupendidos()*, conforme se lleven a cabo los métodos anteriores que realizan el cambio de una variable boolean “suspendido” que controla el proceso.

La clase InsectoInvasor.class se encarga de retener en el refugio, y gestionar los hilos de Crías con el método *criasRefugio()* con el uso de un await controlado por los métodos *invasorActivo()* e *invasorRepelido().*

Clase histórico e interfazControl

La clase histórico es la responsable de la escritura y lectura del documento “evolucionColonia.txt” que lleva el registro de la actividad que se produce dentro del hormiguero. Se inicializan dos HashMaps “eventos” y “eventosInterrupt” con las actividades disponibles dentro de la colonia y las interrupciones que pueden ocurrir respectivamente. El registro se lleva a cabo gracias a los métodos *escribirControlInterrupt()* y *escribirTarea()* que emplean FileWritter y BufferWritter para modificar los .txt.

La clase interfazControl se encarga de preparar las cadenas de hormigas mediante la clase StringBuilder que irán más tarde en los JTextField de la interfazHormigas.interfaz. Asi pues, ensambla mediante HashMaps<> las hormigas que se encuentran en cada zona y los imprime en sus respectivos JTextField.

Clase InterfazHormigas

La clase interfazHormigas inicializa todos los componentes de la interfaz gráfica y gestiona las acciones de los JBottom para pausar/reanudar y generar insecto invasor accediendo a los respectivos HashMaps<> de cada hormiga para producir una interrupción de cada hilo necesario.

El propio NetBeans nos permite diseñar la GUI arrastrando y soltando elementos en la biblioteca de componentes Swing. La vista previa en tiempo real permite visualizar los cambios en la GUI de manera inmediata. Además, NetBeans nos ofrece herramientas de enlace de datos para conectar la interfaz gráfica a los componentes de la lógica de la aplicación.

Parte 2: Programación Distribuida

Se implementa un cliente-servidor con sockets para crear un canal de comunicación bidireccional entre un cliente y un servidor. El Cliente se encarga de manera automática de mostrar mediante una interfaz generada en la propia clase la información que nos proporciona el servidor. El servidor situado en el programa de hormigas responde a la solicitud del cliente. En este caso particular, el cliente solo recibe mensajes del servidor y se comunicara enviando un mensaje "INSECTO INVADOR" para generar un insecto invasor en nuestro servidor.

El funcionamiento del programa se inicia cuando el cliente se conecta al servidor a través de un socket. Una vez que se establece la conexión, el cliente espera a recibir mensajes del servidor. El servidor, por su parte, debe estar siempre escuchando solicitudes de los clientes. Cuando recibe una solicitud, procesa la petición y envía la respuesta al cliente correspondiente.

Discusión de las herramientas de sincronización utilizadas

Parte 1: Programación Concurrente

Los métodos de sincronización y protección de secciones criticas facilitan la programación concurrente para evitar problemas y garantizar la consistencia de los datos en entornos multihilo. Estos métodos permiten controlar el acceso al almacén o la cantidad disponible de comida protegiendo al recurso compartido, asegurándose de que solo un hilo pueda ejecutarlas a la vez como observamos en los métodos synchronized del túnel de entrada. La utilización de estructuras como HashMap permite el acceso y la actualización segura de los datos compartidos por diferentes hilos evitando elementos duplicados.

La sincronización y los mecanismos de bloqueo como los semáforos y monitores empleados en la ZONA ALMACEN nos permiten limitar los hilos que se encuentran dentro, y la gestión del estado de los recursos compartidos para evitar la ejecución concurrente de secciones críticas del código y la creación de condiciones de carrera o deadlock.

El CountDownLatch es una herramienta que no permite hacer que un hilo soldado espere a que el resto de hilos completen sus tareas antes de continuar. En términos simples, genera cuenta atrás que se decrementa con la llegada de los hilos, cuando todas las hormigas se encuentran en el exterior proceden a repeler al invasor.

Diagrama de clases

Parte 1: Programación Concurrente

Main

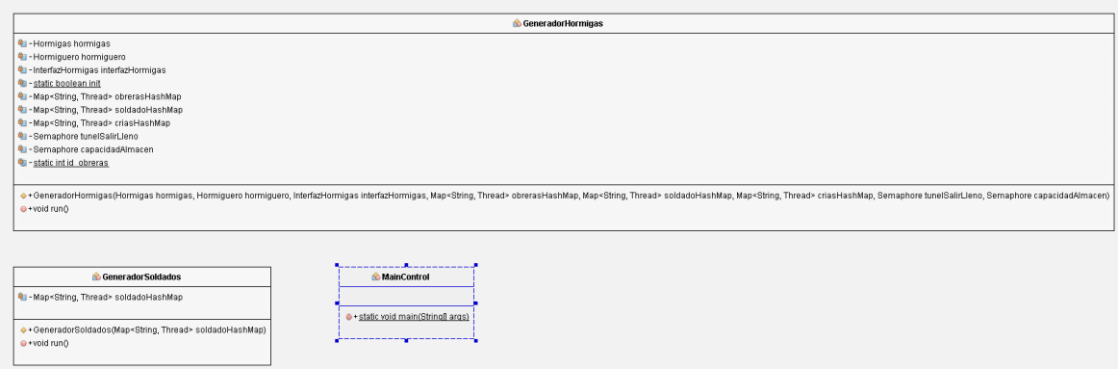


Diagrama Completo UMLClases

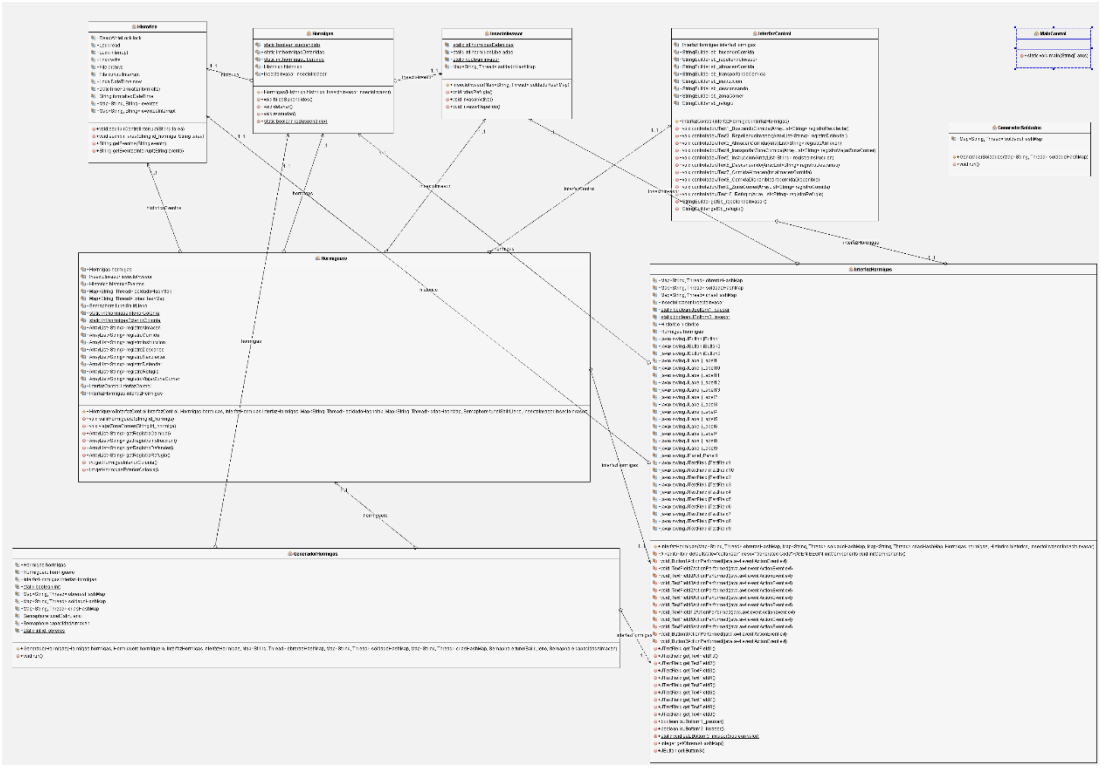
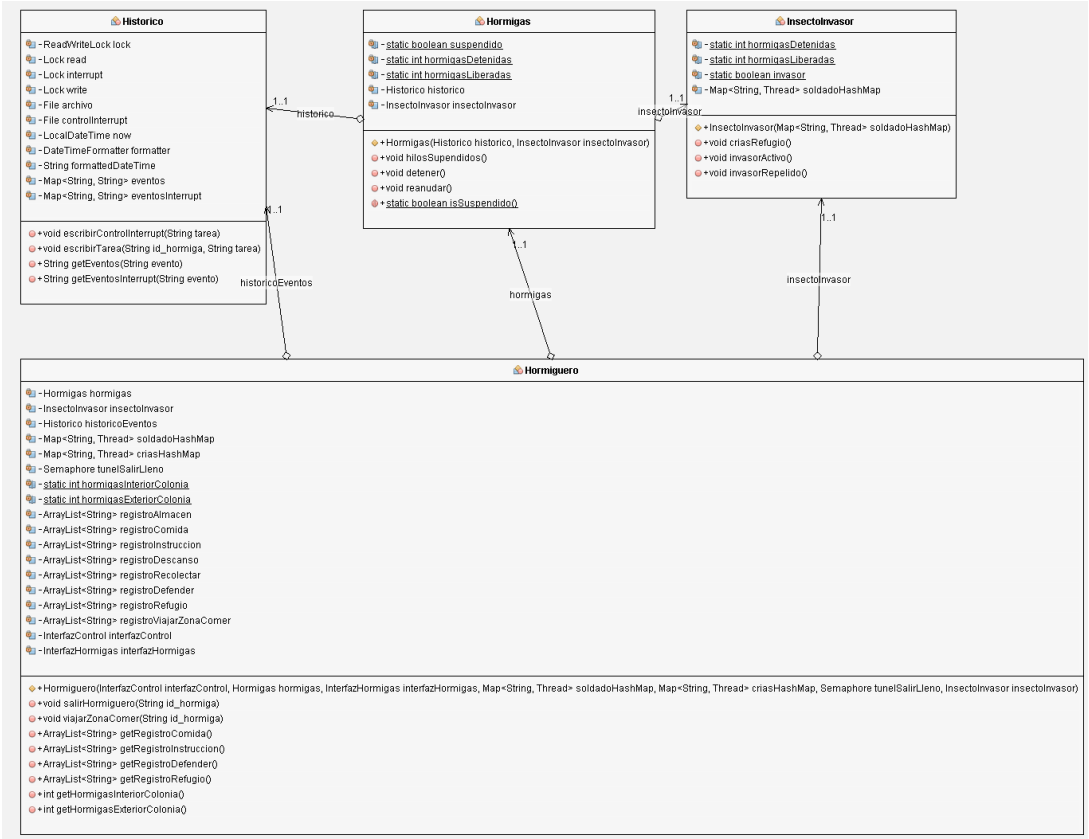
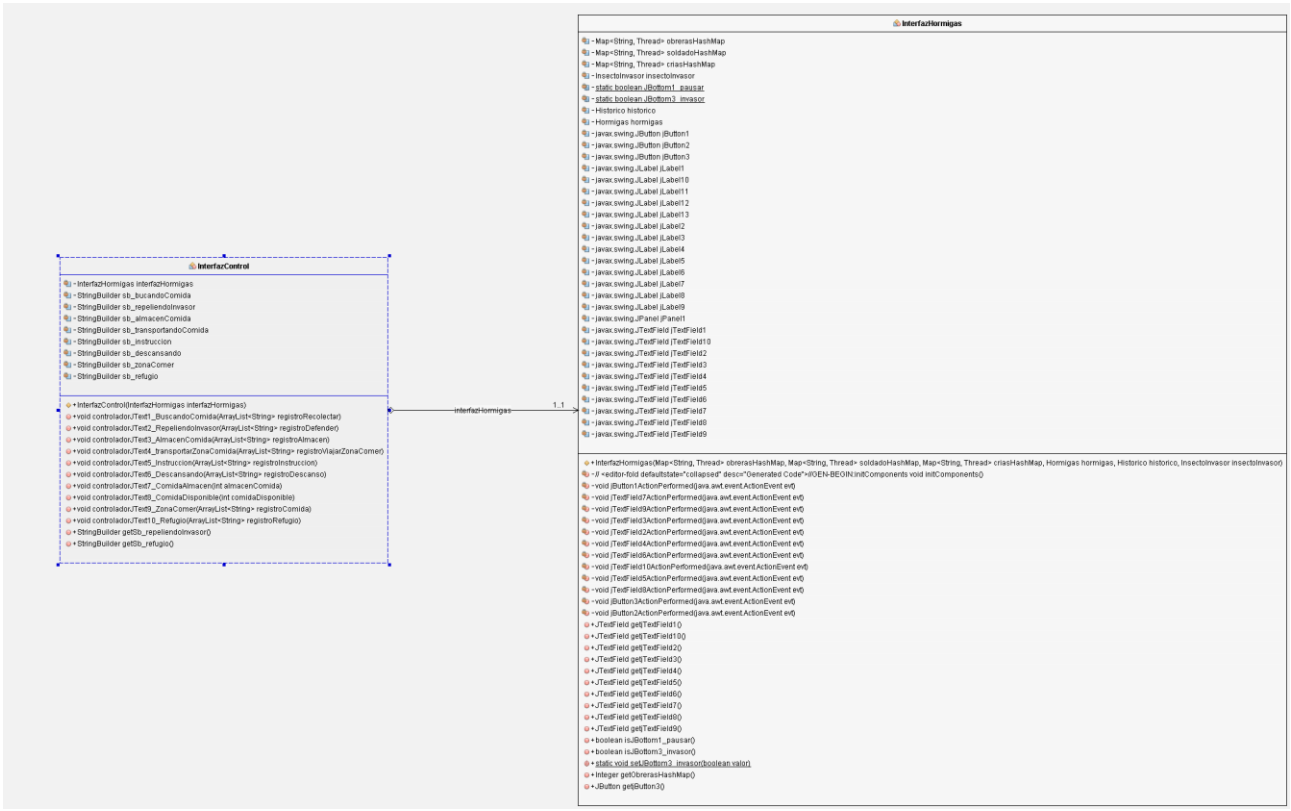


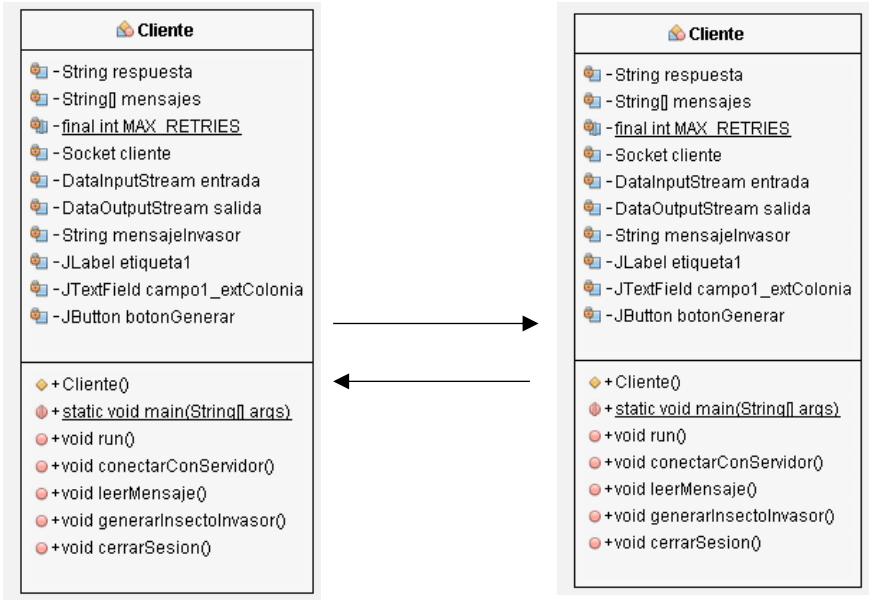
Diagrama Main UMLClass



Interfaz



Parte 2: Programación Distribuida



Anexo Código Fuente

Parte 1: Programación Concurrente

MainControl

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this
template
 */
package Main;

import Interfaz.InterfazControl;
import HormigasPackage.Historico;
import HormigasPackage.Hormigas;
import HormigasPackage.Hormiguero;
import HormigasPackage.InsectoInvasor;
import Interfaz.InterfazHormigas;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.Semaphore;

/**
 *
 * @author tiand
 */
public class MainControl {

    public static void main(String[] args) {

        Historico historico = new Historico();

        Semaphore capacidadAlmacen = new Semaphore(10);
        Semaphore tunelSalirLleno = new Semaphore(2);

        Map<String, Thread> obrerasHashMap = new HashMap<>();
        Map<String, Thread> soldadoHashMap = new HashMap<>();
        Map<String, Thread> criasHashMap = new HashMap<>();

        InsectoInvasor insectoInvasor = new InsectoInvasor(soldadoHashMap);
        Hormigas hormigas = new Hormigas(historico,insectoInvasor);

        InterfazHormigas interfazHormigas = new InterfazHormigas(obrerasHashMap,
soldadoHashMap, criasHashMap, hormigas, historico, insectoInvasor);
        InterfazControl interfazControl = new InterfazControl(interfazHormigas);
        Hormiguero hormiguero = new Hormiguero(interfazControl, hormigas,
interfazHormigas, soldadoHashMap, criasHashMap, tunelSalirLleno, insectoInvasor);

        Servidor servidor = new Servidor(hormiguero, interfazHormigas);
        servidor.start();


    }
}
```

```
/*Si añadimos Thread_MAIN en el hashmap de INTERRUPCIONES
            - Causa problemas la interrupcion de hilo MAIN, por tanto
implementaremos un hilo encargado de inicializar los hilos. Al que podemos manejar
              y realizar operaciones de PARAR Y REANUDAR.
        */

        GeneradorHormigas generadorHormigas = new GeneradorHormigas(hormigas,
hormiguero, interfazHormigas, obrerasHashMap, soldadoHashMap, criasHashMap,
tunelSalirLleno, capacidadAlmacen);
        generadorHormigas.start();

        GeneradorSoldados generadorThreadMainSoldados = new
GeneradorSoldados(soldadoHashMap);
        generadorThreadMainSoldados.start();

        Servidor servidor = new Servidor(hormiguero, interfazHormigas);
        servidor.start();


    }
}
```

GeneradorSoldados

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
 */
package Main;

import java.util.Map;

/**
 *
 * @author tiand
 */
public class GeneradorSoldados extends Thread {

    private final Map<String, Thread> soldadoHashMap;

    public GeneradorSoldados(Map<String, Thread> soldadoHashMap) {

        this.soldadoHashMap = soldadoHashMap;
    }

    @Override
    public void run() {

        soldadoHashMap.put("GENERADOR SOLDADOS", Thread.currentThread());
    }
}
```

GeneradorHormigas

```
package Main;

import HormigasPackage.Hormigas;
import HormigasPackage.Hormiguero;
import Interfaz.InterfazHormigas;
import java.util.Map;
import java.util.concurrent.Semaphore;

/**
 *
 * @author tiand
 */
public class GeneradorHormigas extends Thread{

    private final Hormigas hormigas;
    private final Hormiguero hormiguero;
    private final InterfazHormigas interfazHormigas;
    private static boolean init = true;

    private final Map<String, Thread> obrerasHashMap;
    private final Map<String, Thread> soldadoHashMap;
    private final Map<String, Thread> criasHashMap;

    private final Semaphore tunelSalirLleno;
    private final Semaphore capacidadAlmacen;

    private static int id_obreras = 0, id_soldado = 0, id_crias = 0;

    public GeneradorHormigas(Hormigas hormigas, Hormiguero hormiguero, InterfazHormigas
interfazHormigas, Map<String, Thread> obrerasHashMap, Map<String, Thread> soldadoHashMap,
Map<String, Thread> criasHashMap, Semaphore tunelSalirLleno, Semaphore capacidadAlmacen)
{
        this.hormigas = hormigas;
        this.hormiguero = hormiguero;
        this.interfazHormigas = interfazHormigas;
        this.obrerasHashMap = obrerasHashMap;
        this.soldadoHashMap = soldadoHashMap;
        this.criasHashMap = criasHashMap;
        this.tunelSalirLleno = tunelSalirLleno;
        this.capacidadAlmacen = capacidadAlmacen;
    }

@Override
    public void run() {

        if(init) {
            obrerasHashMap.put("GENERADOR HORMIGAS", Thread.currentThread());
            init = false;
        }
        //criasHashMap.put("GENERADOR HORMIGAS", Thread.currentThread());
```

```
while (!Thread.interrupted()) {
            try {
                for (int i = 0; i < 2000; i++) { //2000
                    Thread.sleep((long) (800 + Math.random()*2700));
                    for (int j = 0; j < 3; j++) {
                        hormigas.new Obreras(String.format("HO%04d", id_obreras),
capacidadAlmacen, tunelSalirLleno, obrerasHashMap, hormiguero).start();
                        id_obreras++;
                    }
                    // CANTIDAD HORMIGAS - OBRERAS
                    //System.out.println("id_obreras " + id_obreras);

                    hormigas.new Soldado(String.format("HS%04d", id_soldado),
tunelSalirLleno, soldadoHashMap, hormiguero).start();
                    id_soldado++;

                    // CANTIDAD HORMIGAS - CRIAS
                    hormigas.new Crias(String.format("HC%04d", id_crias),
criasHashMap, hormiguero, interfazHormigas).start();
                    id_crias++;
                    //System.out.println("id_crias " + id_crias);

                    //TOTAL HORMIGAS EN EJECUCION
                    //System.out.println("TOTAL HORMGIAS = " + (id_obreras +
id_soldado + id_crias));
                }
            } catch (InterruptedException e) {
                //MANJEO DE INTERRUPCIONES
                if(interfazHormigas.isJBottom1_pausar()) {
                    System.out.println("Interrupt - GENERADOR HORMIGAS");
                    hormigas.hilosSupendidos();
                } else if (interfazHormigas.isJBottom3_invasor()) {
                    System.out.println("Interrupt - INVASOR");
                    for(String key: soldadoHashMap.keySet()) {
                            hormiguero.salirHormiguero(key);
                            hormiguero.new ZonaExterior().defenderInvasor(key);
                        }

                    for(String key: criasHashMap.keySet()) {
                        hormiguero.new ZonaRefugio().refugio(key);
                    }
                } else {
                    System.out.println(e);
                }
            }
        }
    }
}
```

InterfazControl

```
package Interfaz;

import java.util.ArrayList;

/**
 *
 * @author tiand
 */
public class InterfazControl {

    private final InterfazHormigas interfazHormigas;
    private final StringBuilder sb_bucandoComida = new StringBuilder();
    private final StringBuilder sb_repeliendoInvasor = new StringBuilder();
    private final StringBuilder sb_almacenComida = new StringBuilder();
    private final StringBuilder sb_transportandoComida = new StringBuilder();
    private final StringBuilder sb_instruccion = new StringBuilder();
    private final StringBuilder sb_descansando = new StringBuilder();
    private final StringBuilder sb_zonaComer = new StringBuilder();
    private final StringBuilder sb_refugio = new StringBuilder();

    //CONTRUCTOR

    public InterfazControl(InterfazHormigas interfazHormigas) {
        this.interfazHormigas = interfazHormigas;
        interfazHormigas.setVisible(true);
    }

    //METODOS
    public synchronized void controladorJText1_BuscandoComida(ArrayList<String>
registroRecolectar) {
        sb_bucandoComida.setLength(0);
        for (int i = 0; i < registroRecolectar.size(); i++) {
            sb_bucandoComida.append(registroRecolectar.get(i));
            sb_bucandoComida.append(", ");
        }
        //System.out.println("Buffer RecolectarComida " + sb_bucandoComida + "
Size: " + registroRecolectar.size());
        interfazHormigas.getjTextField1().setText(sb_bucandoComida.toString());
    }

    public synchronized void controladorJText2_RepeliendoInvasor(ArrayList<String>
registroDefender) {
        //sb_repeliendoInvasor.setLength(0);
        for (int i = 0; i < registroDefender.size(); i++) {
            sb_repeliendoInvasor.append(registroDefender.get(i));
            sb_repeliendoInvasor.append(", ");
        }
        //System.out.println("Buffer RepeliendoInvasor " + sb_repeliendoInvasor + "
Size: " + registroDefender.size());
        interfazHormigas.getjTextField2().setText(sb_repeliendoInvasor.toString());
    }

    public synchronized void controladorJText3_AlmacenComida(ArrayList<String>
registroAlmacen) {
        sb_almacenComida.setLength(0);
        for (int i = 0; i < registroAlmacen.size(); i++) {
            sb_almacenComida.append(registroAlmacen.get(i));
            sb_almacenComida.append(", ");
        }
         //System.out.println("Buffer AlmacenComida " + sb_almacenComida + " Size:
" + registroAlmacen.size());
        interfazHormigas.getjTextField3().setText(sb_almacenComida.toString());
    }
```

```
public synchronized void controladorJText4_transportarZonaComida(ArrayList<String>
registroViajarZonaComer) {
        sb_transportandoComida.setLength(0);
        for (int i = 0; i < registroViajarZonaComer.size(); i++) {
            sb_transportandoComida.append(registroViajarZonaComer.get(i));
            sb_transportandoComida.append(", ");
        }
        //System.out.println("Buffer ZonaComer " + sb_transportandoComida + " Size:
" + registroViajarZonaComer.size());

interfazHormigas.getjTextField4().setText(sb_transportandoComida.toString());
    }

    public synchronized void controladorJText5_Instruccion(ArrayList<String>
registroInstruccion) {
        sb_instruccion.setLength(0);
        for (int i = 0; i < registroInstruccion.size(); i++) {
            sb_instruccion.append(registroInstruccion.get(i));
            sb_instruccion.append(", ");
        }
        //System.out.println("Buffer Intruccion " + sb_instruccion + " Size: " +
registroInstruccion.size());
        interfazHormigas.getjTextField5().setText(sb_instruccion.toString());
    }

    public synchronized void controladorJText6_Descansando(ArrayList<String>
registroDescanso) {
        sb_descansando.setLength(0);
        for (int i = 0; i < registroDescanso.size(); i++) {
            sb_descansando.append(registroDescanso.get(i));
            sb_descansando.append(", ");
        }
        //System.out.println("Buffer Intruccion " + sb_descansando + " Size: " +
registroDescanso.size());
        interfazHormigas.getjTextField6().setText(sb_descansando.toString());
    }

    public synchronized void controladorJText7_ComidaAlmacen(int almacenComida) {
        interfazHormigas.getjTextField7().removeAll();
        interfazHormigas.getjTextField7().setText(Integer.toString(almacenComida));
    }

    public synchronized void controladorJText8_ComidaDisponible(int
comidaDisponible) {
        interfazHormigas.getjTextField8().removeAll();

interfazHormigas.getjTextField8().setText(Integer.toString(comidaDisponible));
    }

public synchronized void controladorJText9_ZonaComer(ArrayList<String>
registroComida) {
        sb_zonaComer.setLength(0);
        for (int i = 0; i < registroComida.size(); i++) {
            sb_zonaComer.append(registroComida.get(i));
            sb_zonaComer.append(", ");
        }
        //System.out.println("Buffer Intruccion " + sb_zonaComer + " Size: " +
registroComida.size());
        interfazHormigas.getjTextField9().setText(sb_zonaComer.toString());
    }
```

```
public synchronized void controladorJText10_Refugio(ArrayList<String>
registroRefugio) {
        //sb_refugio.setLength(0);
        for (int i = 0; i < registroRefugio.size(); i++) {
            sb_refugio.append(registroRefugio.get(i));
            sb_refugio.append(", ");
        }
        //System.out.println("Buffer Intruccion " + sb_refugio + " Size: " +
registroRefugio.size());
        interfazHormigas.getjTextField10().setText(sb_refugio.toString());
    }

    public StringBuilder getSb_repeliendoInvasor() {
        return sb_repeliendoInvasor;
    }

    public StringBuilder getSb_refugio() {
        return sb_refugio;
    }

}
```

InsectoInvasor

```
package HormigasPackage;

import java.util.Map;

/**
 *
 * @author tiand
 */
public class InsectoInvasor {


    private static int hormigasDetenidas = 0;
    private static int hormigasLiberadas = 0;
    private static boolean invasor = false;
    private final Map<String, Thread> soldadoHashMap;

    public InsectoInvasor(Map<String, Thread> soldadoHashMap) {
        this.soldadoHashMap = soldadoHashMap;

    }

    public synchronized void criasRefugio() {
        try {
            hormigasDetenidas++;
            //System.out.println("Cantidad Crias REFUGIO: " + hormigasDetenidas);
            while (invasor) {
                wait();
            }
            //historico.escribirControlInterrupt("CANTIDAD HILOS SUSPENDIDOS: " +
hormigasDetenidas++);
            //historico.escribirControlInterrupt("SOY LIBRE COMO EL VIENTO: " +
Thread.currentThread());
```

```
System.out.println("INVASOR REPELIDO: " + Thread.currentThread());
            hormigasLiberadas++;
        } catch (InterruptedException e) {
            System.out.println(Thread.currentThread() + " ATAQUE INVASOR");
        }
    }

    public synchronized void invasorActivo() {
        invasor = true;
    }

    public synchronized void invasorRepelido() {
        invasor = false;
        System.out.println("CANTIDAD CRIAS EN REFUGIO: " + hormigasDetenidas);
        System.out.println("CANTIDAD CRIAS LIBERADOS DEL REFUGIO: " +
hormigasLiberadas);
        notifyAll();
    }

}
```

Hormiguero (ZonaAlmacen, ZonaComer, ZonaInstruccion, ZonaDescanso)

```
package HormigasPackage;

/**
 *
 * @author tiand
*/

import Interfaz.InterfazHormigas;
import Interfaz.InterfazControl;
import java.util.ArrayList;
import java.util.Map;
import java.util.concurrent.CountDownLatch;
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 Hormiguero {

    private final Hormigas hormigas;
    private final InsectoInvasor insectoInvasor;

    private final Historico historicoEventos = new Historico();

    private final Map<String, Thread> soldadoHashMap;
    private final Map<String, Thread> criasHashMap;
    private final Semaphore tunelSalirLleno;

    private static int hormigasInteriorColonia = 0;
    private static int hormigasExteriorColonia = 0;

     private final ArrayList<String> registroAlmacen = new ArrayList<>();
    private final ArrayList<String> registroComida = new ArrayList<>();
    private final ArrayList<String> registroInstruccion = new ArrayList<>();
```

```
  private final ArrayList<String> registroDescanso = new ArrayList<>();
    private final ArrayList<String> registroRecolectar = new ArrayList<>();
    private final ArrayList<String> registroDefender = new ArrayList<>();
    private final ArrayList<String> registroRefugio = new ArrayList<>();
    private final ArrayList<String> registroViajarZonaComer = new ArrayList<>();
    private final InterfazControl interfazControl;
    private final InterfazHormigas interfazHormigas;

    public Hormiguero(InterfazControl interfazControl, Hormigas hormigas,
InterfazHormigas interfazHormigas, Map<String, Thread> soldadoHashMap, Map<String,
Thread> criasHashMap, Semaphore tunelSalirLleno, InsectoInvasor insectoInvasor) {
        this.interfazControl = interfazControl;
        this.hormigas = hormigas;
        this.interfazHormigas = interfazHormigas;
        this.soldadoHashMap = soldadoHashMap;
        this.criasHashMap = criasHashMap;
        this.tunelSalirLleno = tunelSalirLleno;
        this.insectoInvasor = insectoInvasor;
    }

    public void salirHormiguero(String id_hormiga) {

        try {
            tunelSalirLleno.acquire();
            // System.out.println("La hormiga " + id_hormiga + " SALE del
hormiguero..." + " === Permisos salida " + (tunelSalirLleno.availablePermits()));
            historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("salirHormiguero"));
            Thread.sleep(100);
            hormigasExteriorColonia++;
            hormigasInteriorColonia--;
        } catch (InterruptedException e) {
            //MANJEO DE INTERRUPCIONES
            if(interfazHormigas.isJBottom1_pausar()) {
                    System.out.println("Interrupt PAUSAR - SALIR HOMRIGUERO");

historicoEventos.escribirControlInterrupt(historicoEventos.getEventosInterrupt("sa
lir"));
                    hormigas.hilosSupendidos();
            } else if (interfazHormigas.isJBottom3_invasor()) {
                System.out.println("Interrupt INVASOR - SALIR HOMRIGUERO");

                if(id_hormiga.contains("S")) {
                    salirHormiguero(id_hormiga);
                    new ZonaExterior().defenderInvasor(id_hormiga);
                } else {
                     new ZonaRefugio().refugio(id_hormiga);
                }

            } else {
                System.out.println(e);
            }
        } finally {
             tunelSalirLleno.release();
        }
    }

    public synchronized void viajarZonaComer(String id_hormiga) throws
InterruptedException {
             try {
                 registroViajarZonaComer.add(id_hormiga);
```

```
interfazControl.controladorJText4_transportarZonaComida(registroViajarZonaComer);
                historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("viajarZonaComer"));
                Thread.sleep((long) (1000 + Math.random()* 2000));
            } finally {
                registroViajarZonaComer.remove(id_hormiga);
            }
    }

    // ZONAS DEL HORMIGUERO INTERNO
    public class ZonaAlmacen extends Hormiguero {

        private static int almacenComida;
        private final Lock controlAlmacen = new ReentrantLock();
        private final Condition vacio = controlAlmacen.newCondition();

        public ZonaAlmacen() {
            super(interfazControl, hormigas, interfazHormigas, soldadoHashMap,
criasHashMap, tunelSalirLleno, insectoInvasor);
        }

        public void almacenarComida(String id_hormiga) throws InterruptedException{
            // Control permisos - HORMIGAS ZONA ALMACEN

            //Comienzo SC
            controlAlmacen.lock();
            try {
                registroAlmacen.add(id_hormiga);
                interfazControl.controladorJText3_AlmacenComida(registroAlmacen);
                historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("almacenarComida"));
                historicoEventos.escribirTarea(id_hormiga, "Registro => numHormigas
en el almacen: " + registroAlmacen.size() + " almacenComida = " + almacenComida);
                almacenComida =+ 5;
                Thread.sleep((long) (200 + Math.random()* 200));
                almacenComida += 5;
                interfazControl.controladorJText7_ComidaAlmacen(almacenComida);
                vacio.signalAll();
            } finally {
                registroAlmacen.remove(id_hormiga);
                //System.out.println("Registro almacen size: "+
registroAlmacen.size());
                controlAlmacen.unlock();
            }

        }

        public void extraerComida(String id_hormiga) throws InterruptedException {
            // Control permisos - HORMIGAS ZONA ALMACEN
            // Comienzo SC
            controlAlmacen.lock();
            try {
                registroAlmacen.add(id_hormiga);
                  interfazControl.controladorJText3_AlmacenComida(registroAlmacen);
                while (almacenComida <= 0) {
                    vacio.await();
                }

historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("consumirComida"));
                historicoEventos.escribirTarea(id_hormiga, "Registro => numHormigas
en el almacen: " + registroAlmacen.size() + " almacenComida = " + almacenComida);
                // Tiempo de acceso requerido para acceder al almacen
                Thread.sleep((long) (1000 + Math.random()* 1000));
                almacenComida -= 5;
```

```
  } finally {
                registroAlmacen.remove(id_hormiga);
                controlAlmacen.unlock();
            }
        }
    }
//
=============================================================================================
=========================================

    public class ZonaInstruccion extends Hormiguero {

        public ZonaInstruccion() {
            super(interfazControl, hormigas, interfazHormigas, soldadoHashMap,
criasHashMap, tunelSalirLleno, insectoInvasor);
        }

        public synchronized void hacerInstruccion(String id_hormiga) {
            try {
                registroInstruccion.add(id_hormiga);
                interfazControl.controladorJText5_Instruccion(registroInstruccion);
                historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("hacerInstruccion"));
                Thread.sleep((long) (2000 + Math.random()* 6000));
            } catch (InterruptedException e){
                //MANJEO DE INTERRUPCIONES
                if(interfazHormigas.isJBottom1_pausar()) {
                    System.out.println("Interrupt PAUSAR - ZONA INSTRUCCION");

historicoEventos.escribirControlInterrupt(historicoEventos.getEventosInterrupt("IIn
struccion"));
                    hormigas.hilosSupendidos();
                } else if (interfazHormigas.isJBottom3_invasor()) {
                    System.out.println("Interrupt INVASOR- ZONA INSTRUCCION");
                    //SOLO PUEDEN SER HORMIGAS SOLDADO
                    salirHormiguero(id_hormiga);
                    new ZonaExterior().defenderInvasor(id_hormiga);

                } else {
                    System.out.println(e);
                }
            } finally {
                registroInstruccion.remove(id_hormiga);
            }
        }
    }

//
=============================================================================================


    public class ZonaDescanso extends Hormiguero {

        public ZonaDescanso() {
            super(interfazControl, hormigas, interfazHormigas, soldadoHashMap,
criasHashMap, tunelSalirLleno, insectoInvasor);
        }

        public synchronized void descansar(int taskTimeMiliseconds, String
id_hormiga) {
            try {
                registroDescanso.add(id_hormiga);
                interfazControl.controladorJText6_Descansando(registroDescanso);
                historicoEventos.escribirTarea(id_hormiga, ,
historicoEventos.getEventos("descansar"));
```

```
Thread.sleep((taskTimeMiliseconds));
            } catch (InterruptedException e) {
                //MANJEO DE INTERRUPCIONES;
                if(interfazHormigas.isJBottom1_pausar()) {
                    System.out.println("Interrupt PAUSAR - ZONA DESCANSO");

historicoEventos.escribirControlInterrupt(historicoEventos.getEventosInterrupt("Ide
scanso"));
                    hormigas.hilosSupendidos();
                } else if (interfazHormigas.isJBottom3_invasor()) {
                    System.out.println("Interrupt INVASOR - ZONA DESCANSO");

                    System.out.println(id_hormiga);
                    if(id_hormiga.contains("S")) {
                        salirHormiguero(id_hormiga);
                        new ZonaExterior().defenderInvasor(id_hormiga);
                    } else {
                         new ZonaRefugio().refugio(id_hormiga);
                    }

                } else {
                    System.out.println(e);
                }
            } finally {
                registroDescanso.remove(id_hormiga);
            }
        }
    }

//
==========================================================================================
=======================================
    public class ZonaComer extends Hormiguero{

        private static int comidaDisponible = 0;
        private final Lock control = new ReentrantLock();
        private final Condition vacio = control.newCondition();
        private final Condition lleno = control.newCondition();

        public ZonaComer() {
            super(interfazControl, hormigas, interfazHormigas, soldadoHashMap,
criasHashMap, tunelSalirLleno, insectoInvasor);
        }

        public synchronized void depositarComida(String id_hormiga) throws
InterruptedException {
            control.lock();
            registroComida.add(id_hormiga);
            interfazControl.controladorJText9_ZonaComer(registroComida);

try {
                //Poner MAIXMO aunque no especifique para evitar DEADLOCK, mejora
rendimiento y reparto de recursos
                while(comidaDisponible == 10) {
                    lleno.await();
                }

historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("depositarComida"));
                Thread.sleep((long) (1000 + Math.random()* 1000));
                comidaDisponible += 5;

interfazControl.controladorJText8_ComidaDisponible(comidaDisponible);
                vacio.signal();
```

```
} finally {
                registroComida.remove(id_hormiga);
                control.unlock();
            }
        }

        public synchronized void comer(int taskTimeMiliseconds, String id_hormiga)
{
            control.lock();
            registroComida.add(id_hormiga);
            interfazControl.controladorJText9_ZonaComer(registroComida);
            try {
                while(comidaDisponible <= 0) {
                    historicoEventos.escribirTarea(id_hormiga, " se ha quedado sin
comer");
                    vacio.await();
                }
                if(id_hormiga.contains("C")) {
                    historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("comerCrias"));
                    Thread.sleep((long) (taskTimeMiliseconds + Math.random()*
2000));
                } else {
                    historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("comer"));
                    Thread.sleep((taskTimeMiliseconds));
                }
                lleno.signal();
            } catch (InterruptedException e) {
                //MANEJO DE EXCEPCIONES

                if(interfazHormigas.isJBottom1_pausar()) {
                    System.out.println("Interrupt PAUSAR - ZONA COMER");

historicoEventos.escribirControlInterrupt(historicoEventos.getEventosInterrupt("com
er"));
                    hormigas.hilosSupendidos();

                } else if (interfazHormigas.isJBottom3_invasor()) {
                    System.out.println("Interrupt INVASOR - ZONA COMER");

                    System.out.println(id_hormiga);
                    if(id_hormiga.contains("S")) {

                        System.out.println("Soy un soldado? " + id_hormiga);
                        salirHormiguero(id_hormiga);
                        new ZonaExterior().defenderInvasor(id_hormiga);
                    } else {
                         new ZonaRefugio().refugio(id_hormiga);
                    }

                } else {
                    System.out.println(e);
                }
            } finally {
                comidaDisponible--;
                registroComida.remove(id_hormiga);
                control.unlock();
            }
        }
    }
```

```
public class ZonaRefugio extends Hormiguero {

            public ZonaRefugio() {
                super(interfazControl, hormigas, interfazHormigas, soldadoHashMap,
criasHashMap, tunelSalirLleno, insectoInvasor);
            }

            public synchronized void refugio(String id_hormiga) {
                try {

                    registroRefugio.add(id_hormiga);
                    interfazControl.controladorJText10_Refugio(registroRefugio);
                    historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("refugio"));
                    insectoInvasor.criasRefugio();
                } catch (InterruptedException e) {
                    //MANJEO DE INTERRUPCIONES --> La interrupcion solo puede ser de
PAUSAR
                    System.out.println("Interrupt PAUSAR - REFUGIO");

historicoEventos.escribirControlInterrupt(historicoEventos.getEventosInterrupt("refu
gio"));
                    hormigas.hilosSupendidos();
                } finally {
                    registroRefugio.remove(id_hormiga);
                    interfazControl.getSb_refugio().setLength(0);
                    interfazControl.controladorJText10_Refugio(registroRefugio);
                }
            }
        }

//
==========================================================================================
=====================================
        public class ZonaExterior extends Hormiguero{

            int numHormigasDefensoras;
            private CountDownLatch latch;

            public ZonaExterior() {
                super(interfazControl, hormigas, interfazHormigas, soldadoHashMap,
criasHashMap, tunelSalirLleno, insectoInvasor);
            }

            public synchronized String entrarHormiguero(String id_hormiga) {
                try {
                    historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("entrarHormiguero"));
                    //System.out.println("La hormiga " + id_hormiga + " ENTRANDO al
hormiguero...");
                    Thread.sleep(100);
                    hormigasInteriorColonia++;
                    hormigasExteriorColonia--;
                } catch (InterruptedException e) {
                    //MANJEO DE INTERRUPCIONES
                    if(interfazHormigas.isJBottom1_pausar()) {
                        System.out.println("Interrupt PAUSAR - ENTRAR HORMIGUERO");

historicoEventos.escribirControlInterrupt(historicoEventos.getEventosInterrupt("Ient
rar"));
                        hormigas.hilosSupendidos();
                    } else if (interfazHormigas.isJBottom3_invasor()) {
                        System.out.println("Interrupt INVASOR - ENTRAR HORMIGUERO");
```

```
if(id_hormiga.contains("S")) {
                            salirHormiguero(id_hormiga);
                            new ZonaExterior().defenderInvasor(id_hormiga);
                        } else {
                             new ZonaRefugio().refugio(id_hormiga);
                        }

                    } else {
                        System.out.println(e);
                    }
                }
                return null;
            }

            public synchronized void recolectarComida(String id_hormiga) throws
InterruptedException {
                try {
                    registroRecolectar.add(id_hormiga);

interfazControl.controladorJText1_BuscandoComida(registroRecolectar);
                    historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("recolectarComida"));
                    Thread.sleep(4000);
                } finally {
                    registroRecolectar.remove(id_hormiga);
                }

            }

            public synchronized void defenderInvasor(String id_hormiga) {

                //System.out.println(id_hormiga + " se va a esperar en
CyclicBarrier");

                try {
                    registroDefender.add(id_hormiga);
                    System.out.println(registroDefender.size());

interfazControl.controladorJText2_RepeliendoInvasor(registroDefender);
                    historicoEventos.escribirTarea(id_hormiga,
historicoEventos.getEventos("defenderHormiguero"));
                    System.out.println("Hormiga " + id_hormiga + " defendiendo el
hormiguero");

                    // COMIENZO COUNTDOWN - LACH: Esperar a que todos los soldados
esten LISTOS
                    if(numHormigasDefensoras == 0) {
                        //RETIRAR HILO MAIN -1
                        latch = new CountDownLatch((soldadoHashMap.size() - 1));
                        for (int i = 0; i < (soldadoHashMap.size() - 1); i++) {
                            latch.countDown();
                            numHormigasDefensoras++;
                            System.out.println(id_hormiga + " esperando en COUNTDOWN
- LACH " + numHormigasDefensoras + "/" + (soldadoHashMap.size() - 1));
                            historicoEventos.escribirTarea(id_hormiga, " esperando
en COUNTDOWN - LACH " + numHormigasDefensoras + "/" + (soldadoHashMap.size() - 1));

                        }
                    }

latch.await();
                    // FIN COUNTDOWN - LACH
                    System.out.println("COMIENZA COUNTDOWN - LACH!!!!!");
                    historicoEventos.escribirTarea("COUNTDOWN - LACH", "COMIENZA
COUNTDOWN - LACH!!!!!");
```

```
  Thread.sleep(20000);
                    InterfazHormigas.setJBottom3_invasor(false);
                    System.out.println("INVASOR REPELIDO ");
                } catch (InterruptedException e) {
                    //MANJEO DE INTERRUPCIONES --> La interrupcion solo puede ser de
PAUSAR
                    System.out.println("Interrupt PAUSAR - DEFENDER INVASOR");

historicoEventos.escribirControlInterrupt(historicoEventos.getEventosInterrupt("invas
or"));
                    hormigas.hilosSupendidos();
                } finally {
                    registroDefender.remove(id_hormiga);
                    interfazControl.getSb_repeliendoInvasor().setLength(0);

interfazControl.controladorJText2_RepeliendoInvasor(registroDefender);
                    insectoInvasor.invasorRepelido();
                    entrarHormiguero(id_hormiga);
                }
            }
        }

    public ArrayList<String> getRegistroComida() {
        return registroComida;
    }

    public ArrayList<String> getRegistroInstruccion() {
        return registroInstruccion;
    }

    public ArrayList<String> getRegistroDefender() {
        return registroDefender;
    }

    public ArrayList<String> getRegistroRefugio() {
        return registroRefugio;
    }

    public int getHormigasInteriorColonia() {
        return hormigasInteriorColonia;
    }

    public int getHormigasExteriorColonia() {
        return hormigasExteriorColonia;
    }

}
```

Hormigas (Obreras, Soldado, Crias)

```
package HormigasPackage;

import Interfaz.InterfazHormigas;
import java.util.Map;
import java.util.concurrent.Semaphore;

/**
 *
 * @author tiand
 */
public class Hormigas {

    private static boolean suspendido = false;
    private static int hormigasDetenidas = 0;
    private static int hormigasLiberadas = 0;
    private final Historico historico;
    private final InsectoInvasor insectoInvasor;

    public Hormigas(Historico historico, InsectoInvasor insectoInvasor) {
        this.historico = historico;
        this.insectoInvasor = insectoInvasor;
    }

    public class Obreras extends Thread {

        private final String id_obreras;
        private final String tareaInicial;
        private final Semaphore capacidadAlmacen;
        private final Semaphore tunelSalirLleno;
        private final Map<String, Thread> obrerasHashMap;
        private final Hormiguero hormiguero;


        public Obreras(String id_obreras, Semaphore capacidadAlmacen, Semaphore
tunelSalirLleno, Map<String, Thread> obrerasHashMap, Hormiguero hormiguero) {
                this.id_obreras = id_obreras;
                this.capacidadAlmacen = capacidadAlmacen;
                this.tunelSalirLleno = tunelSalirLleno;
                this.tareaInicial = hormiguero.new
ZonaExterior().entrarHormiguero(id_obreras);
                this.obrerasHashMap = obrerasHashMap;
                this.hormiguero = hormiguero;
        }

        @Override
        public void run() {
            // Clave - Valor "NOTA no ponerlo en el constructor porque sino tiene como
valor MAIN"
            obrerasHashMap.put(id_obreras, Thread.currentThread());

            while(!Thread.interrupted()) {
                for (int i = 0; i < 10; i++) {
                    /*
                    ID Hormigas PARES repetiran iterativamente:
                            1.- intento acceso (1-2s)
                            2.- viajarZonaComer (1-3s)
                            3.- DepositarComida +5 elemntos de comida disponibles
                     */

             if((Integer.parseInt(id_obreras.substring(id_obreras.length() - 4)) % 2)
      == 0) {
                        try {
                        // Obtener permiso en almacen - SEMAFORO
                            capacidadAlmacen.acquire();
```

```
//System.out.println("La hormiga PAR " + id_obreras + " adquirio un permiso" + " ===
Permisos disponibles " + (capacidadAlmacen.availablePermits()));
                            hormiguero.new ZonaAlmacen().extraerComida(id_obreras);
                            //System.out.println("La hormiga " + id_obreras + " ha
completado su tarea y libera el permiso");
                            capacidadAlmacen.release();
                        // Liberar permiso
                            hormiguero.viajarZonaComer(id_obreras);
                            hormiguero.new ZonaComer().depositarComida(id_obreras);
                        } catch (InterruptedException e) {
                            //MANJEO DE INTERRUPCIONES;
                            System.out.println("Interrupt PAUSAR - OBRERA PAR");

//historico.escribirControlInterrupt(historico.getEventosInterrupt("interruptOPAR"));
                            hilosSupendidos();
                        }

                    /*
                    ID Hormigas IMPARES repetiran iterativamente:
                            1.- intento acceso (1-2s)
                            2.- viajarZonaComer (1-3s)
                            3.- DepositarComida +5 elemntos de comida disponibles
                     */
                    } else{

                        try {
                            hormiguero.salirHormiguero(id_obreras);
                            hormiguero.new
ZonaExterior().recolectarComida(id_obreras);
                            hormiguero.new
ZonaExterior().entrarHormiguero(id_obreras);
                        // Obtener permiso en almacen - SEMAFORO
                            capacidadAlmacen.acquire();
                            //System.out.println("La hormiga IMPAR " + id_obreras + "
adquirio un permiso" + " === Permisos disponibles " +
(capacidadAlmacen.availablePermits()));
                            hormiguero.new ZonaAlmacen().almacenarComida(id_obreras);
                            //System.out.println("La hormiga " + id_obreras + " ha
completado su tarea y libera el permiso");
                            capacidadAlmacen.release();
                        //Liberar permiso
                        } catch (InterruptedException e) {
                            //MANJEO DE INTERRUPCIONES;
                            System.out.println("Interrupt PAUSAR - OBRERA IMPAR");

//historico.escribirControlInterrupt(historico.getEventosInterrupt("interruptOIMPAR"))
;
                            hilosSupendidos();
                        }
                }
                // Tareas comunes
                hormiguero.new ZonaComer().comer(3000, id_obreras);
                hormiguero.new ZonaDescanso().descansar(1000, id_obreras);

                }
                hilosSupendidos();
            }
        }
    }
```

```
public class Soldado extends Thread {

        private final String id_soldado;
        private final String tareaSoldado;
        private final Semaphore tunelSalirLleno;
        private final Map<String, Thread> soldadoHashMap;
        private final Hormiguero hormiguero;

        public Soldado(String id_soldado, Semaphore tunelSalirLleno, Map<String,
Thread> soldadoHashMap, Hormiguero hormiguero) {
            this.id_soldado = id_soldado;
            this.tunelSalirLleno = tunelSalirLleno;
            this.tareaSoldado = hormiguero.new
ZonaExterior().entrarHormiguero(id_soldado);
            this.soldadoHashMap = soldadoHashMap;
            this.hormiguero = hormiguero;
        }

        @Override
        public void run() {
            soldadoHashMap.put(id_soldado, Thread.currentThread());
            while(!Thread.interrupted()) {
                for (int i = 0; i < 6; i++) {
                    hormiguero.new ZonaInstruccion().hacerInstruccion(id_soldado);
                    hormiguero.new ZonaDescanso().descansar(2000, id_soldado);
                }
                hormiguero.new ZonaComer().comer(3000, id_soldado);
                }
            }

        }

//
==========================================================================================
====================================

    public class Crias extends Thread {

        private final String id_crias;
        private final String tareaCrias;
        private final Map<String, Thread> criasHashMap;
        private final Hormiguero hormiguero;
        private final InterfazHormigas interfazHormigas;

        public Crias(String id_crias, Map<String, Thread> criasHashMap, Hormiguero
hormiguero, InterfazHormigas interfazHormigas) {
            this.id_crias = id_crias;
            this.tareaCrias = hormiguero.new
ZonaExterior().entrarHormiguero(id_crias);
            this.criasHashMap = criasHashMap;
            this.hormiguero = hormiguero;
            this.interfazHormigas =  interfazHormigas;
        }

        @Override
        public void run() {

            criasHashMap.put(id_crias, Thread.currentThread());
            while (interfazHormigas.isJBottom3_invasor()) {
                  hormiguero.new ZonaRefugio().refugio(id_crias);
            }
```

```
while(!Thread.interrupted()) {
                hormiguero.new ZonaComer().comer(3000, id_crias);
                hormiguero.new ZonaDescanso().descansar(4000, id_crias);
            }

        }
    }


//
================================================================================================
====================================

    public synchronized void hilosSupendidos() {
        try {
            //HORMIGAS DETENIDAS
            //System.out.println("Hormigas detenidas " + hormigasDetenidas++);

            //Control .txt de la ejecucion de las interrupciones
            //historico.escribirControlInterrupt("CANTIDAD HILOS SUSPENDIDOS: " +
hormigasDetenidas++);
            //historico.escribirControlInterrupt("HILO: " + Thread.currentThread() + "
isInterrupted: " + Thread.currentThread().isInterrupted());
            //historico.escribirControlInterrupt("VALOR SUSPENDIDO = " + suspendido);

            System.out.println("VALOR SUSPENDIDO = " + suspendido);
            hormigasDetenidas++;
            while (suspendido) {
                wait();
            }
            //historico.escribirControlInterrupt("CANTIDAD HILOS SUSPENDIDOS: " +
hormigasDetenidas++);
            //historico.escribirControlInterrupt("SOY LIBRE COMO EL VIENTO: " +
Thread.currentThread());
            System.out.println("SOY LIBRE COMO EL VIENTO: " + Thread.currentThread());
            hormigasLiberadas++;
        } catch (InterruptedException e) {
            System.out.println(Thread.currentThread() + " detenido");
        }

    }

    public void detener() {
        suspendido = true;
    }

    public synchronized void reanudar() {
       suspendido = false;
       System.out.println("CANTIDAD HILOS SUSPENDIDOS: " + hormigasDetenidas);
       System.out.println("CANTIDAD HILOS LIBERADOS: " + hormigasLiberadas);
       notifyAll();
    }

    public static boolean isSuspendido() {
        return suspendido;
    }

}
```

Historico

```
package HormigasPackage;

/**
 *
 * @author tiand
 */
import java.io.*;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReadWriteLock;
import java.util.concurrent.locks.ReentrantLock;
import java.util.concurrent.locks.ReentrantReadWriteLock;

public class Historico {

    private final ReadWriteLock lock = new ReentrantReadWriteLock();
    private final Lock read = lock.readLock();
    private final Lock interrupt = new ReentrantLock();
    private final Lock write = lock.writeLock();
    private final File archivo = new File("evolucionColonia.txt");
    private final File controlInterrupt = new File("controlInterrupt.txt");
    private final LocalDateTime now = LocalDateTime.now();
    private final DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy
HH:mm:ss.SSS");
    private final String formattedDateTime = now.format(formatter);

    private final Map<String, String> eventos = new HashMap<>() {{
        put("recolectarComida", " se encuentra recolectando 5 elementos de comida");
        put("almacenarComida", " almacena +5 de comida");
        put("consumirComida", " transportando 5 elementos de comida a la zona para
comer...");
        put("depositarComida", " + 5 elementos de comida disponibles");
        put("entrarHormiguero", " esta entrando al hormiguero");
        put("comer", " consume -1 unidad de comida");
        put("descansar", " descansando profundamente... ");
        put("hacerInstruccion", " esta siendo instruida");
        put("viajarZonaComer", " esta tranportando +5 unidades de comida al comedor");
        put("comerCrias", " una cria tarda entre 3-5 (s) en comer ");
        put("defenderHormiguero", " se encuentra defendiendo el hormiguero
valientemente");
        put("salirHormiguero", " esta saliendo del hormiguero");
        put("refugio", " se encuentra en el REFUGIO");
    }};

    private final Map<String, String> eventosInterrupt = new HashMap<>() {{
        put("interruptM", " Interrupcion - MAIN");
        put("interruptS", " Interrupcion SOLDADO");
        put("interruptOComun", " Interrupcion OBRERAS");
        put("interruptOPAR", " Interrupcion OBRERAS");
        put("interruptOIMPAR", " Interrupcion IMPAR");
        put("interuptC", " Interrupcion CRIAS");
        put("jButton1_pausar", " jButton1_pausar: ");
        put("jButton3_invasor", " jButton3_invasor: ");
        put("Ientrar", " Interrupt PAUSAR - ENTRAR HORMIGUERO ");
        put("Idescanso", " Interrupt PAUSAR - ZONA DESCANSO ");
         put("IInstruccion", " Interrupt PAUSAR - ZONA INSTRUCCION ");
        put("comer", " Interrupt PAUSAR - ZONA COMER");
        put("salir", " Interrupt PAUSAR - SALIR HOMRIGUERO");
        put("invasor", " Interrupt PAUSAR - DEFENDER INVASOR");
        put("refugio", " Interrupt PAUSAR - REFUGIO");

    }};
```

```
public void escribirControlInterrupt(String tarea) {
        interrupt.lock();
        try {
            FileWriter escritor = new FileWriter(controlInterrupt, true);
            BufferedWriter bufferEscritor = new BufferedWriter(escritor);
            bufferEscritor.write(tarea);
            bufferEscritor.newLine();
            bufferEscritor.close();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            interrupt.unlock();
        }
    }

    public void escribirTarea(String id_hormiga, String tarea) throws
InterruptedException {
        read.lock();
        try {
            FileWriter escritor = new FileWriter(archivo, true);
            BufferedWriter bufferEscritor = new BufferedWriter(escritor);
            if(tarea.contains("Registro")) {
                bufferEscritor.write(tarea + " ==> " + formattedDateTime);
            } else {
                bufferEscritor.write("La hormiga " + id_hormiga + tarea + " ==> " +
formattedDateTime);
            }
            bufferEscritor.newLine();
            bufferEscritor.close();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            read.unlock();
        }
    }


    public String getEventos(String evento) {
        return eventos.get(evento);
    }

    public String getEventosInterrupt(String evento) {
        return eventosInterrupt.get(evento);
    }



}
```

InterfazHormigas

```
package Interfaz;

import HormigasPackage.Historico;
import HormigasPackage.Hormigas;
import HormigasPackage.InsectoInvasor;
import java.util.Map;
import javax.swing.*;

/**
 *
 * @author tiand
 */
public class InterfazHormigas extends javax.swing.JFrame {

    private final Map<String, Thread> obrerasHashMap;
    private final Map<String, Thread> soldadoHashMap;
    private final Map<String, Thread> criasHashMap;

    private final InsectoInvasor insectoInvasor;

    private static boolean JBottom1_pausar = false;
    private static boolean JBottom3_invasor = false;
    private final Historico historico;
    private final Hormigas hormigas;

    /**
     * Creates new form InterfazHormigas
     */
    public InterfazHormigas(Map<String, Thread> obrerasHashMap, Map<String, Thread> soldadoHashMap,
Map<String, Thread> criasHashMap, Hormigas hormigas, Historico historico, InsectoInvasor insectoInvasor) {
        initComponents();
        this.obrerasHashMap = obrerasHashMap;
        this.soldadoHashMap = soldadoHashMap;
        this.criasHashMap = criasHashMap;
        this.hormigas = hormigas;
        this.historico = historico;
        this.insectoInvasor = insectoInvasor;
    }

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

```
jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jTextField4 = new javax.swing.JTextField();
        jTextField5 = new javax.swing.JTextField();
        jTextField6 = new javax.swing.JTextField();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jTextField7 = new javax.swing.JTextField();
        jTextField8 = new javax.swing.JTextField();
        jTextField9 = new javax.swing.JTextField();
        jTextField10 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jLabel13 = new javax.swing.JLabel();
        jButton3 = new javax.swing.JButton();
        jTextField1 = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 16)); // NOI18N
        jLabel1.setText("Exterior de la colonia:");

        jLabel2.setText("Hormigas buscando comida:");

        jLabel3.setText("Hormigas repeliendo un insecto invasor:");

        jLabel4.setFont(new java.awt.Font("Segoe UI", 1, 16)); // NOI18N
        jLabel4.setText("Interior de la colonia:");

        jLabel5.setText("Hormigas en el ALMACÉN DE COMIDA:");

        jLabel6.setText("Hormigas llevando comida a la ZONA PARA COMER:");

        jLabel7.setText("Hormigas haciendo INSTRUCCIÓN:");

        jLabel8.setText("Hormigas descansando:");

        jLabel9.setText("ZONA PARA COMER");

        jLabel10.setText("REFUGIO");

        jTextField2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField2ActionPerformed(evt);
            }
        });
```

```
jTextField3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField3ActionPerformed(evt);
            }
        });

        jTextField4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField4ActionPerformed(evt);
            }
        });

        jTextField5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField5ActionPerformed(evt);
            }
        });

        jTextField6.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField6ActionPerformed(evt);
            }
        });

        jLabel11.setText("Unidades de Comida (ALMACÉN)");

        jLabel12.setText("Unidades de Comida (ZONA PARA COMER)");

        jTextField7.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField7ActionPerformed(evt);
            }
        });

        jTextField8.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField8ActionPerformed(evt);
            }
        });

        jTextField9.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField9ActionPerformed(evt);
            }
        });

        jTextField10.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField10ActionPerformed(evt);
            }
        });
```

```
jButton1.setText("Pausar");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Reanudar");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jButton3.setText("Generar Amenaza Insecto Invasor");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

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

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

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(53, 53, 53)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G, false)
                            .addGroup(jPanel1Layout.createSequentialGroup()
                                .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(jTextField2))
                            .addGroup(jPanel1Layout.createSequentialGroup()
                                .addComponent(jLabel2)
                                .addGap(30, 30, 30)
                                .addComponent(jTextField1,
javax.swing.GroupLayout.DEFAULT_SIZE, 789, Short.MAX_VALUE))))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(35, 35, 35)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                            .addComponent(jLabel4)
                            .addComponent(jLabel1)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(55, 55, 55)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
, false)
                            .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                                    .addComponent(jLabel9)
                                    .addComponent(jLabel10)
                                    .addComponent(jButton1))

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

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                                    .addGroup(jPanel1Layout.createSequentialGroup()
                                        .addComponent(jButton2)
                                        .addGap(560, 560, 560)
                                        .addComponent(jLabel13)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                        .addComponent(jButton3)
                                        .addGap(0, 3, Short.MAX_VALUE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                                        .addGap(0, 0, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                                            .addComponent(jTextField9,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE,
851, javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(jTextField10,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE,
851, javax.swing.GroupLayout.PREFERRED_SIZE)))))
                            .addGroup(jPanel1Layout.createSequentialGroup()
                                .addComponent(jLabel8)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jTextField6,
javax.swing.GroupLayout.PREFERRED_SIZE, 473, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(41, 41, 41)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                                    .addComponent(jLabel12)
                                    .addComponent(jLabel11))
                                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                                    .addComponent(jTextField7)
                                    .addComponent(jTextField8)))
                            .addGroup(jPanel1Layout.createSequentialGroup()
                                .addComponent(jLabel7)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(jTextField5))

                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                                .addComponent(jLabel6)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(jTextField4))
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                                .addComponent(jLabel5)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(jTextField3)))))
                .addContainerGap(63, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(32, 32, 32)
                .addComponent(jLabel1)
                .addGap(21, 21, 21)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
                    .addComponent(jLabel2)
                    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
81, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3))
                .addGap(18, 18, 18)
                .addComponent(jLabel4)
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
                    .addComponent(jLabel5)
                    .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
                    .addComponent(jLabel6)
                    .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel7)
                    .addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(32, 32, 32)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
                    .addComponent(jLabel8)
                    .addComponent(jTextField6, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel11)
                    .addComponent(jTextField7, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                    .addComponent(jLabel12)
                    .addComponent(jTextField8,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(15, 15, 15)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                    .addComponent(jTextField9,
javax.swing.GroupLayout.PREFERRED_SIZE, 80, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel9))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                    .addComponent(jLabel10)
                    .addComponent(jTextField10,
javax.swing.GroupLayout.PREFERRED_SIZE, 93, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(28, 28, 28)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                    .addComponent(jButton1)
                    .addComponent(jButton2)
                    .addComponent(jLabel13)
                    .addComponent(jButton3)))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 28, Short.MAX_VALUE))
        );

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

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            synchronized (this) {
                JBottom1_pausar = true;

historico.escribirControlInterrupt(historico.getEventosInterrupt("jButton1_pausar") +
"\n");
                // SUPENDIDO = false
                hormigas.detener();
```

```
//CONTROL INTERRUPCIONES OBRERAS
                for(String key: obrerasHashMap.keySet()) {

                    //Detener AQUI generadorHormigas --> afecta al resto DE HILOS -
soldadoHashMap y criasHashMap
                    if(key.equals("GENERADOR HORMIGAS")) {
                        obrerasHashMap.get(key).interrupt();
                        System.out.println("GENERADOR HORMIGAS OBRERAS --- > Clave:
" + key + " Valor: " + obrerasHashMap.get(key) + " is interrupted " +
obrerasHashMap.get(key).isInterrupted());
                    } else {
                        obrerasHashMap.get(key).interrupt();
                        System.out.println("TAMAÑO HASHMAP: " +
obrerasHashMap.size() + " --- > Clave: " + key + " Valor: " +
obrerasHashMap.get(key) + " is interrupted " +
obrerasHashMap.get(key).isInterrupted());
                        //HILOS INTERUPIDOS - HISTORICO
                        historico.escribirControlInterrupt("Clave: " + key + " : " +
obrerasHashMap.get(key) + " ha sido interrumpido --> " + "isInterrupted: " +
obrerasHashMap.get(key).isInterrupted());
                    }
                }

                //CONTROL INTERRUPCIONES SOLDADOS
                for(String key: soldadoHashMap.keySet()) {
                    //HILOS INTERUPIDOS - HISTORICO
                    if(key.equals("GENERADOR SOLDADOS")) {
                        soldadoHashMap.get(key).interrupt();
                        System.out.println("GENERADOR SOLDADOS --- > Clave: " + key
+ " Valor: " + soldadoHashMap.get(key) + " is interrupted " +
soldadoHashMap.get(key).isInterrupted());
                    } else {
                        soldadoHashMap.get(key).interrupt();
                        //HILOS INTERUPIDOS - HISTORICO
                        System.out.println("TAMAÑO HASHMAP: " +
obrerasHashMap.size() + " --- > Clave: " + key + " Valor: " +
soldadoHashMap.get(key) + " isInterrupted: " +
soldadoHashMap.get(key).isInterrupted());
                        historico.escribirControlInterrupt("Clave: " + key + " : " +
soldadoHashMap.get(key) + " ha sido interrumpido --> " + "isInterrupted: " +
soldadoHashMap.get(key).isInterrupted());
                    }
                }

                //CONTROL INTERRUPCIONES CRIAS
                for(String key: criasHashMap.keySet()) {
//                    if(key.equals("GENERADOR HORMIGAS")) {
//                        criasHashMap.get(key).interrupt();
//                        System.out.println("GENERADOR HORMIGAS CRIAS --- > Clave:
" + key + " Valor: " + criasHashMap.get(key) + " is interrupted " +
criasHashMap.get(key).isInterrupted());
//                    } else {
                        criasHashMap.get(key).interrupt();
                        //HILOS INTERUPIDOS - HISTORICO
                        System.out.println("TAMAÑO HASHMAP: " + criasHashMap.size()
+ " --- > Clave: " + key + " Valor: " + criasHashMap.get(key) + " isInterrupted: " +
criasHashMap.get(key).isInterrupted());
                        historico.escribirControlInterrupt("Clave: " + key + " : " +
criasHashMap.get(key) + " ha sido interrumpido --> " + "isInterrupted: " +
criasHashMap.get(key).isInterrupted());
//                    }
                }
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
```

```
  private void jTextField7ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField9ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField10ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField8ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            synchronized (this) {
                insectoInvasor.invasorActivo();
                JBottom3_invasor = true;

historico.escribirControlInterrupt(historico.getEventosInterrupt("jButton3_invasor")
+ "\n");

                System.out.println("TAMANO EN JBOTTOM3 " + soldadoHashMap.size());

                for(String key: soldadoHashMap.keySet()) {
                    if(key.equals("GENERADOR SOLDADOS")) {
                        System.out.println("HILO main GENERADOR SOLDADOS
continua...");
                    } else {
                        soldadoHashMap.get(key).interrupt();
                        System.out.println("Clave " + key + " Hilo -> " +
Thread.currentThread() + " isInterupted " + soldadoHashMap.get(key).isInterrupted());
                        historico.escribirControlInterrupt(key + " : " +
soldadoHashMap.get(key) + " ha sido interrumpido --> " + "isInterrupted: " +
soldadoHashMap.get(key).isInterrupted());
                    }

                }
```

```
for(String key: criasHashMap.keySet()) {
                    // SOLO SE DEJAN DE GENERAR CRIAS
                    criasHashMap.get(key).interrupt();
                    //System.out.println("Clave " + key + " Hilo -> " +
Thread.currentThread() + " isInterupted " + criasHashMap.get(key).isInterrupted());
                    historico.escribirControlInterrupt(key + " : " +
criasHashMap.get(key) + " ha sido interrumpido --> " + "isInterrupted: " +
criasHashMap.get(key).isInterrupted());

                }
            }
        } catch (Exception e) {
            System.out.println("JBotton3 " + e);
        }
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        JBottom1_pausar = false;
        System.out.println("Reanudar");
        hormigas.reanudar();
    }

    /**
     * @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(InterfazHormigas.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
//        } catch (InstantiationException ex) {
//
java.util.logging.Logger.getLogger(InterfazHormigas.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
//        } catch (IllegalAccessException ex) {
//
java.util.logging.Logger.getLogger(InterfazHormigas.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
//        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
//
java.util.logging.Logger.getLogger(InterfazHormigas.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
//        }
//        //</editor-fold>
//
//        /* Create and display the form */
//        java.awt.EventQueue.invokeLater(new Runnable() {
//            public void run() {
//                new InterfazHormigas(exchangerSoldado).setVisible(true);
//            }
//        });
//    }
```

```
// Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField10;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField5;
    private javax.swing.JTextField jTextField6;
    private javax.swing.JTextField jTextField7;
    private javax.swing.JTextField jTextField8;
    private javax.swing.JTextField jTextField9;
    // End of variables declaration



    public JTextField getjTextField1() {
        return jTextField1;
    }

    public JTextField getjTextField10() {
        return jTextField10;
    }

    public JTextField getjTextField2() {
        return jTextField2;
    }

    public JTextField getjTextField3() {
        return jTextField3;
    }

    public JTextField getjTextField4() {
        return jTextField4;
    }

    public JTextField getjTextField5() {
         return jTextField5;
    }

    public JTextField getjTextField6() {
        return jTextField6;
    }

    public JTextField getjTextField7() {
        return jTextField7;
    }
```

```
public JTextField getjTextField9() {
        return jTextField9;
    }

    public boolean isJBottom1_pausar() {
        return JBottom1_pausar;
    }

    public boolean isJBottom3_invasor() {
        return JBottom3_invasor;
    }

    public static void setJBottom3_invasor(boolean valor) {
        JBottom3_invasor = valor;
    }

    public Integer getObrerasHashMap() {
        return obrerasHashMap.size();
    }

    public JButton getjButton3() {
        return jButton3;
    }

}
```

Parte 2: Programación Distribuida

Cliente

```
package SocketCliente;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import static java.lang.System.exit;
import java.net.InetAddress;
import java.net.Socket;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 *
 * @author tiand
 */

public class Cliente extends Thread {


    //MAIN
    public static void main(String[] args) {
        Cliente cliente = new Cliente();
        VentanaPrincipal ventanaPrincipal = cliente.new VentanaPrincipal();
        cliente.start();
    }
```

```
@Override
    public void run() {

        conectarConServidor();
        System.out.println("Conexión establecida con el servidor");
        while (true) {
            leerMensaje();
        }
    }

    public void conectarConServidor() {

        boolean connected = false;
        int retries = 0;

        while (!connected && retries < MAX_RETRIES) {
            try {
                cliente = new Socket(InetAddress.getLocalHost(),5000);   //Intenta
conectarnos al puerto 5000 del servidor;
                entrada = new DataInputStream(cliente.getInputStream());  //Creamos los
canales de entrada/salida;
                salida = new DataOutputStream(cliente.getOutputStream());
                salida.writeUTF("Tian Duque");
                connected = true; // Si se estableció la conexión, cambia el valor a
true
            } catch (IOException e) {
                retries++; // Si falla la conexión, incrementa el contador de intentos
                if(retries > 10) {
                    exit(1);
                }
                System.out.println("Intento " + retries + " ERROR conexión fallida");
                try {
                    Thread.sleep(1000); // Espera un segundo antes de intentar la
conexión de nuevo
                } catch (InterruptedException ex) {
                    Thread.currentThread().interrupt();
                }
            }
        }
    }

    public void leerMensaje() {

        try {
            respuesta = entrada.readUTF();
            System.out.println(respuesta);
            while (!respuesta.isEmpty()) {

                respuesta = entrada.readUTF();
                if (respuesta.contains("homigasIntruccion")) {
                    System.out.println("Respuesta recibida: " + respuesta + "\n");
                    int indicePalabraClave = respuesta.indexOf("homigasIntruccion");
                    mensajes[0] = respuesta.substring(indicePalabraClave +
"homigasIntruccion".length() + 2); // Obtiene el texto después de la palabra clave
                    campo3_hacerInst.setText(mensajes[0]);


                } else if (respuesta.contains("hormigasInvasion")) {
                    System.out.println("Respuesta recibida: " + respuesta + "\n");
                    int indicePalabraClave = respuesta.indexOf("hormigasInvasion");
                    mensajes[1] = respuesta.substring(indicePalabraClave +
"hormigasInvasion".length() + 2); // Obtiene el texto después de la palabra
clave
                      campo4_invasion.setText(mensajes[1]);
```

```
} else if (respuesta.contains("hormigasRefugio")) {
                    System.out.println("Respuesta recibida: " + respuesta + "\n");
                    int indicePalabraClave = respuesta.indexOf("hormigasRefugio");
                    mensajes[3] = respuesta.substring(indicePalabraClave +
"hormigasRefugio".length() + 2); // Obtiene el texto después de la palabra clave
                    campo6_refugio.setText(mensajes[3]);


                } else if (respuesta.contains("hormigasExteriorColonia")) {
                    System.out.println("Respuesta recibida: " + respuesta + "\n");
                    int indicePalabraClave =
respuesta.indexOf("hormigasExteriorColonia");
                    mensajes[4] = respuesta.substring(indicePalabraClave +
"hormigasExteriorColonia".length() + 2); // Obtiene el texto después de la palabra clave
                    campo1_extColonia.setText(mensajes[4]);


                } else {
                    System.out.println("Respuesta recibida: " + respuesta + "\n");
                    int indicePalabraClave =
respuesta.indexOf("hormigasInteriorColonia");
                    mensajes[5] = respuesta.substring(indicePalabraClave +
"hormigasInteriorColonia".length() + 2); // Obtiene el texto después de la palabra clave
                    campo2_intColonia.setText(mensajes[5]);
                }

            }
        } catch (IOException ex) {

        }
    }

    public void generarInsectoInvasor() {
//        try {
//            salida = new DataOutputStream(cliente.getOutputStream());
//            System.out.println("Se ha generado un " + mensajeInvasor);
//            salida.writeUTF(mensajeInvasor);
//        } catch(IOException ex) {
//
//        }
    }

    public void cerrarSesion() {
        try {
            cliente.close();
            System.out.println("Sesión cerrada\n");
            botonGenerar.setEnabled(false);
            botonCerrar.setEnabled(false);
        } catch (IOException ex) {

        }
    }

public class VentanaPrincipal extends JFrame implements ActionListener {

        public VentanaPrincipal() {

            // Configura la ventana
            super("Ventana Principal");
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setResizable(false);
            setSize(800, 400);
            setTitle("Cliente");
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
   // Crea las etiquetas
            etiqueta1 = new JLabel(" Número de hormigas obreras en el exterior de la
colonia");
            etiqueta2 = new JLabel("Número de hormigas obreras en el interior de la
colonia");
            etiqueta3 = new JLabel("Número de hormigas soldado haciendo instrucción:");
            etiqueta4 = new JLabel("Número de hormigas soldado repeliendo una
invasión");
            etiqueta5 = new JLabel("Número de hormigas crías en la ZONA PARA COMER");
            etiqueta6 = new JLabel(" Número de hormigas crías en el REFUGIO");


            // Crea los campos de texto
            campo1_extColonia = new JTextField(10);
            campo2_intColonia = new JTextField(10);
            campo3_hacerInst = new JTextField(10);
            campo4_invasion = new JTextField(10);
            campo5_zonaComer = new JTextField(10);
            campo6_refugio = new JTextField(10);

            // Crea los botones
            botonGenerar = new JButton("Generar Amenaza Insecto Invasor");
            botonCerrar = new JButton("Cerrar Sesión");

            // Agrega los componentes a la ventana
            JPanel panel = new JPanel(new GridLayout(7, 2, 5, 5));
            panel.setBorder(BorderFactory.createEmptyBorder(50, 50, 50, 50));
            panel.add(etiqueta1);
            panel.add(campo1_extColonia);
            panel.add(etiqueta2);
            panel.add(campo2_intColonia);
            panel.add(etiqueta3);
            panel.add(campo3_hacerInst);
            panel.add(etiqueta4);
            panel.add(campo4_invasion);
            panel.add(etiqueta5);
            panel.add(campo5_zonaComer);
            panel.add(etiqueta6);
            panel.add(campo6_refugio);
            panel.add(botonGenerar);
            panel.add(botonCerrar);

            // Agrega el panel a la ventana
            add(panel);

            // Agrega los listeners a los botones
            botonGenerar.addActionListener(this);
            botonCerrar.addActionListener(this);

            // Muestra la ventana
            setVisible(true);
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            if (e.getSource() == botonGenerar) {
                generarInsectoInvasor();
            } else if (e.getSource() == botonCerrar) {
                 cerrarSesion();
            }
        }
    }

}
```

Servidor

```
package Main;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashMap;
import java.util.Map;
import HormigasPackage.Hormiguero;
import Interfaz.InterfazHormigas;


/**
 *
 * @author tiand
 */
public class Servidor extends Thread {

    private ServerSocket servidor;
    private DataOutputStream salida;
    private DataInputStream entrada;
    private Socket conexion;
    private static int numConexiones = 0;
    private final Map<String, String> mensajes = new HashMap<>();
    private final Hormiguero hormiguero;
    private final InterfazHormigas interfazHormigas;
    private String insectoInvasor;

    public Servidor(Hormiguero hormiguero, InterfazHormigas interfazHormigas) {
        this.hormiguero = hormiguero;
        this.interfazHormigas = interfazHormigas;
    }

    @Override
    public void run() {
        try {
            iniciarServidor();
            while (numConexiones > 0) {
                Thread.sleep(1000);
                enviarMensajes();
                //recibirMensajes();
            }

        } catch (InterruptedException ex) {
            System.out.println(ex);
        }
    }
    public void iniciarServidor() {
        try {
                servidor = new ServerSocket(5000);   //Creamos el socket para
conectarnos al puerto 5000 del servidor
                System.out.println("Servidor Arrancado....");
                conexion = servidor.accept();     // Esperamos una conexión
                numConexiones++;
                System.out.println("Conexión n° " + numConexiones + " desde: " +
conexion.getInetAddress().getHostName()); // Abrimos los canales de E/S
```

```
salida = new DataOutputStream(conexion.getOutputStream());
                String mensaje = entrada.readUTF();
                System.out.println(mensaje + " ha entrado al servidor");
        } catch(IOException ex) {
        }
    }

    public void enviarMensajes() {

        mensajes.put("homigasIntruccion",
String.valueOf(hormiguero.getRegistroInstruccion().size()));
        mensajes.put("hormigasInvasion",
String.valueOf(hormiguero.getRegistroDefender().size()));
        mensajes.put("hormigasZonaComer",
String.valueOf(hormiguero.getRegistroComida().size()));
        mensajes.put("hormigasRefugio",
String.valueOf(hormiguero.getRegistroRefugio().size()));
        mensajes.put("hormigasExteriorColonia",
String.valueOf(Math.abs(hormiguero.getHormigasExteriorColonia())));
        mensajes.put("hormigasInteriorColonia",
String.valueOf(Math.abs(hormiguero.getHormigasInteriorColonia())));


        try{

            for(String key: mensajes.keySet()) {
                //MENSAJES ENVIADOS
                salida.writeUTF(key + " : " + mensajes.get(key));
                System.out.println("MENSAJE: " + key + " = " + mensajes.get(key));
            }
        } catch (IOException e) {

        }
    }


//    public void recibirMensajes() {
//        try {
//            entrada = new DataInputStream(conexion.getInputStream());
//            String mensaje = entrada.readUTF();
//            System.out.println(mensaje);
//        } catch (IOException ex) {
//            Logger.getLogger(Servidor.class.getName()).log(Level.SEVERE, null, ex);
//        }
//
//    }

    public int getNumConexiones() {
        return numConexiones;
    }


}
```