Bloque 4. Paradigmas de la Programación

Tarea 3: Corral de gallinas con Monitores

A tener en cuenta:

- La clase Main contiene 28 pollos que serán los hilos.
- Los pollos realizarán 4 actividades: comer, beber, pasear y dormir, las cuales tendrán una limitación de pollos por actividad que regularemos mediante un ArrayList.
- Protegeremos la sección crítica mediante monitores.
- Los pollos nunca finalizan el ciclo (las 4 actividades se ejecutarán en un bucle while).
- Tendremos 5 clases:
  - Main: lanza los 28 hilos
  - Pollo: contiene las 4 opciones/actividades que realizarán los pollos. Primero aseguramos que todos empiezan paseando y tras esto, de forma aleatoria, realizarán otras actividades.
  - Comedero/Bebedero/Cama: cada clase tendrá un monitor y realizará la actividad que durarán un tiempo aleatorio.

Código Clase Main:

```
public class Main {

    public static void main(String[] args) {
        //Variables compartidas, las protegeremos con monitores
        //Un arrayList limitará los pollos por acción
        Comedero comedero = new Comedero(4);
        Bebedero bebedero = new Bebedero(8);
        Cama cama = new Cama(10);
        // Los 28 hilos representan los pollos
        for (int i = 1; i <= 28; i++) {
            Pollo pollo = new Pollo(comedero, bebedero, cama, i);
            pollo.start();
        }
    }

}
```

Código Clase Pollo:

```
import static java.lang.Thread.sleep;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Pollo extends Thread {


    private Comedero comedero;
    private Bebedero bebedero;
    private Cama cama;
    private int id;
    //El pollo tiene un identificador y una actividad aleatoria en cada momento.
    public Pollo(Comedero comedero, Bebedero bebedero, Cama cama, int id) {
        this.comedero = comedero;
```

```
    this.bebedero = bebedero;
    this.cama = cama;
    this.id = id;
}

public void run() {
    //Primero, aseguraremos que la actividad inicial de todos los pollos/hilos será la de pasear
    int accion;
    accion = 1;
    while (true) {
        //En un bucle infinito, se realizarán las actividades aleatorias
        switch (accion) {
            case 1:
                System.out.println("El pollo: " + id + " está paseando");
                int tiempo = (9 + (int) (-5 * Math.random()));
                 {
                    try {
                        sleep(tiempo);
                    } catch (InterruptedException ex) {
                        Logger.getLogger(Pollo.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
                break;
            case 2: {
                try {
                    comedero.comer(id);
                } catch (InterruptedException ex) {
                    Logger.getLogger(Pollo.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            break;
            case 3: {
                try {
                    bebedero.beber(id);
                } catch (InterruptedException ex) {
                    Logger.getLogger(Pollo.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            break;
            case 4: {
                try {
                    cama.dormir(id);
                } catch (InterruptedException ex) {
                    Logger.getLogger(Pollo.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            break;
        }
        //Aquí generaremos un valor aleatorio entre 1 y 4, que decidirá la siguiente acción del pollo
        accion = (4 + (int) (-4 * Math.random()));

    }
}
```

```
}
```

Código Clase Comedero:

```
 import static java.lang.Thread.sleep;
import java.util.ArrayList;
import java.util.concurrent.Semaphore;

public class Comedero {

    private ArrayList<Integer> arl;
    private int maximo = 0, in = 0, out = 0, numElem = 0;

    public Comedero(int max) {
        this.maximo = max;
        this.arl = new ArrayList<>();
    }

    public synchronized void comer(int id) throws InterruptedException {

        while (numElem == maximo) { //Si el ArrayList está lleno... Espera
            wait();
        }

        //Vamos añadiendo los pollos
        arl.add(id);
        System.out.println("El pollo: " + id + " está comiendo");

        numElem++;
        in = (in + 1) % maximo;
        // El pollo realizará la acción durante unos segundos

        int tiempo = (6000 + (int) (-2000 * Math.random()));
        sleep(tiempo);
        id = arl.get(0);
        arl.remove(0);
        numElem = numElem - 1;
        out = (out + 1) % maximo;
        notifyAll(); //El ArrayList ya no está vacío

    }

}
```

Código Clase Bebedero:

```
import static java.lang.Thread.sleep;
import java.util.ArrayList;

public class Bebedero {
```

```
    private ArrayList<Integer> arl; //Array de pollos
    private int maximo = 0, in = 0, out = 0, numElem = 0;

    public Bebedero(int max) {
        this.maximo = max;
        this.arl = new ArrayList<>();
    }

    public synchronized void beber(int id) throws InterruptedException {

        while (numElem == maximo) { //Si el ArrayList está lleno... Espera
            wait();
        }

        //Vamos añadiendo los pollos
        arl.add(id);
        System.out.println("El pollo: " + id + " está bebiendo");
        numElem++;
        in = (in + 1) % maximo;
        // El pollo realizará la acción durante unos segundos
        int tiempo = (3000 + (int) (-1000 * Math.random()));
        sleep(tiempo);
        //Cuando acaba, lo quitamos del array
        id = arl.get(0);
        arl.remove(0);
        numElem = numElem - 1;
        out = (out + 1) % maximo;
        notifyAll();

    }

}
```

Código Clase Cama:

```
import static java.lang.Thread.sleep;
import java.util.ArrayList;

public class Cama {

    private ArrayList<Integer> arl;
    private int maximo = 0, in = 0, out = 0, numElem = 0;

    public Cama(int max) {
        this.maximo = max;
        this.arl = new ArrayList<>();
    }

    public synchronized void dormir(int id) throws InterruptedException {

        while (numElem == maximo) { //Si el ArrayList está lleno... Espera
            wait();
```

```
        }

        //Vamos añadiendo los pollos
        arl.add(id);
        System.out.println("El pollo: " + id + " está durmiendo");
        numElem++;
        in = (in + 1) % maximo;
        // El pollo realizará la acción durante unos segundos
        int tiempo = (19000 + (int) (-15000 * Math.random()));
        sleep(tiempo);
        id = arl.get(0);
        arl.remove(0);
        numElem = numElem - 1;
        out = (out + 1) % maximo;
        notifyAll();

    }

}
```

Resultado:

Los pollos comienzan paseando y luego pasan a otras tareas:

```
run:
El pollo: 1 está paseando
El pollo: 2 está paseando
El pollo: 3 está paseando
El pollo: 4 está paseando
El pollo: 1 está bebiendo
El pollo: 23 está paseando
El pollo: 22 está paseando
El pollo: 21 está paseando
El pollo: 20 está paseando
El pollo: 19 está paseando
El pollo: 18 está paseando
El pollo: 4 está paseando
El pollo: 17 está paseando
El pollo: 16 está paseando
El pollo: 15 está paseando
El pollo: 14 está paseando
El pollo: 13 está paseando
El pollo: 3 está comiendo
El pollo: 12 está paseando
El pollo: 11 está paseando
El pollo: 10 está paseando
El pollo: 9 está paseando
El pollo: 8 está paseando
El pollo: 7 está paseando
El pollo: 6 está paseando
El pollo: 28 está paseando
El pollo: 11 está durmiendo
```

BUILD STOPPED (total time: 21 seconds)