Pràctiques de laboratori
Problema de les formigues
(1 sessió)

Concurrència i
Sistemes Distribuïts

Introducció

L'objectiu d'aquesta pràctica és analitzar i completar un programa concurrent en el qual s'apliquen diferents condicions de sincronització entre fils, fent ús d'algunes de les eines Java proporcionades a la biblioteca java.util.concurrent.

- Ús de ReentrantLock .
- Generació i ús de les Condicions associades a un determinat ReentrantLock
- Aplicar solucions per al problema dels interbloquejos

Aquesta pràctica es desenvolupa en una única sessió. Tinga en compte que necessitarà destinar un poc de temps del seu treball personal per a concloure la pràctica. Al llarg de la pràctica ha de completar una sèrie d'exercicis. Es recomana resoldre'ls i anotar els seus resultats per a facilitar l'estudi posterior.

El problema de les formigues

Es disposa d'un territori en el qual viu un conjunt de formigues. El territori es modela com una matriu rectangular NxN, sent N un paràmetre del programa. Cada formiga s'implementa com un fil Java, amb una posició inicial aleatòria dins del territori.
Cada formiga es mou lliurement pel territori: cada moviment es realitza a una cel·la contigua a la qual es troba (a dalt, a baix, esquerra o dreta). Es pot indicar la quantitat inicial de formigues, així com el nombre de moviments que realitza cadascuna abans d'acabar. Quan una formiga acaba, desapareix del territori (alliberant la cel·la que ocupava).
Les formigues es modelen mitjançant la classe Ant, que estén a Thread . El codi bàsic de les formigues és:

```
class Ant extends Thread {
    . . .
```

```
    public void run() {
        ...
        t.hi(a); // t = terrain. put ant a in a random cell
        while (movs > 0) {
            movs--;      // decrement remaining moves
            t.move(a); // move ant a to random next cell
            delay();    // applies a random delay
        }
        t.bye(a );       // ant a disappears.
        ...
    }
}
```

Com a restricció als moviments de les formigues, en cada cel·la del territori pot haver-hi com a màxim una única formiga:

- Inicialment el programa situa a cada formiga en una cel·la lliure
- Si una formiga desitja moure's a una cel·la ocupada, ha d'esperar que quede lliure

Resolem el problema utilitzant el concepte de monitor. Quan una formiga vol desplaçar-se a una cel·la i aquesta es troba ocupada, haurà de suspendre's en una variable condition fins que siga reactivada per una altra formiga. El interface Terrain modela la interfície d'aquest monitor:

```
interface Terrain {
    void hi  (int a);
    void bye (int a );.
    void move(int a) throws InterruptedException;
}
```

Els mètodes hi, bye, move s'invoquen des del codi de cada formiga. La simulació acaba quan es compleix alguna de les següents condicions:

1. Totes les formigues han acabat la seua execució i abandonat el territori.
2. S'ha arribat a una situació d'interbloqueix, de manera que les formigues que queden vives mai podran acabar.

Per a modelar el territori plantegem diferents classes que implementen Terrain:

- Terrain0.- Monitor bàsic (lock i condition implícits).
- Terrain1.- Monitor general (java.util.concurrent) amb una única variable condition per a tot el territori.
- Terrain2.- Monitor general amb una variable condition per cel·la del territori: una formiga se suspèn en la variable condició associada a la cel·la ocupada a la qual vol desplaçar-se.
- Terrain3.- Monitor general amb una variable condition per cel·la del territori. Inclou un mecanisme per a resoldre el problema dels interbloquejos.

La classe Terrain0 ja està implementada, i ha d'utilitzar-se com a punt de partida per als restants tipus de territori. El següent apartat presenta el codi proporcionat: la resta dels apartats del butlletí corresponen a les diferents activitats que es plantegen, una per tipus de territori.

Codi proporcionat

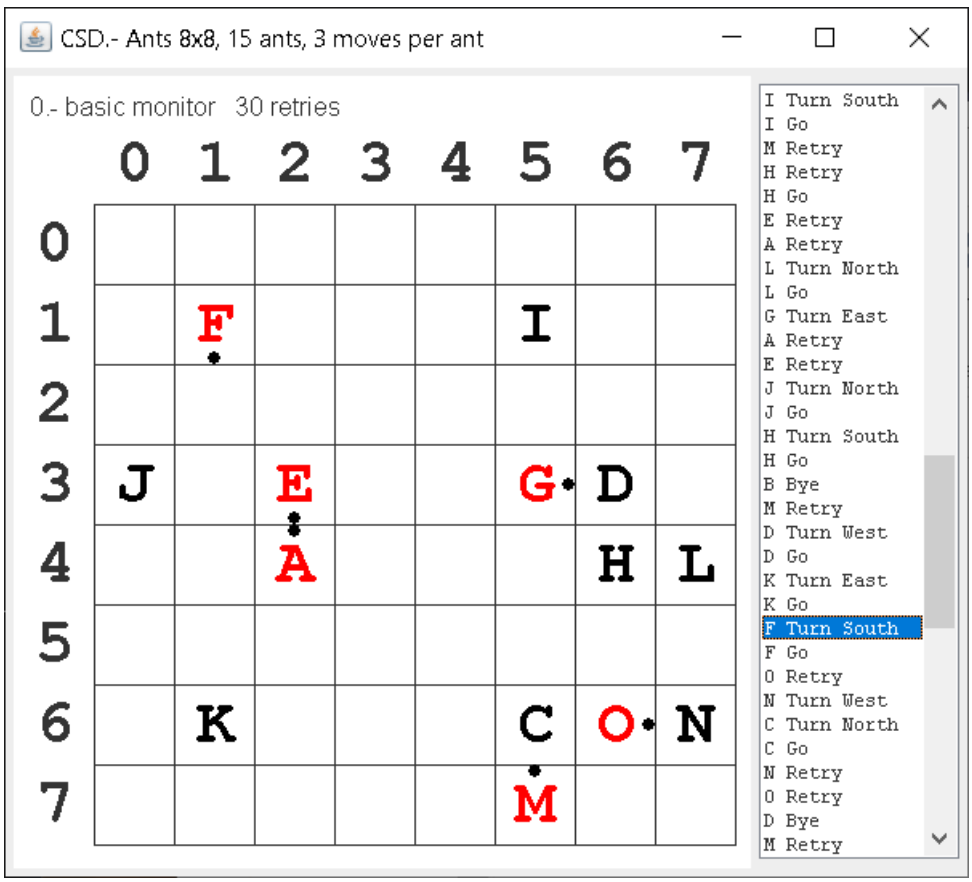
Pot descarregar el codi necessari per a la pràctica des del Poliformat de l'assignatura (fitxer Ants.jar).
La classe Ants conté el mètode principal main. Des d'un terminal, escriu java Ants (i possiblement arguments opcionals). Els arguments possibles són:

- Tipus de Territori (0,1,2,3, per defecte 0).
- Talla del territori (sencer entre 6 i 10, per defecte 8).
- Nombre de formigues (sencer entre 10 i 26, per defecte 15).
- Nombre de moviments per cada formiga (sencer entre 2 i 5, per defecte 3).

Exemples:

- java Ants (Terrain0, 8x8 cel·les, 15 formigues, 3 moviments).
- java Ants `1 10 26 2` (Terrain1, 10x10, 26 formigues, 2 moviments).

En executar l'aplicació, es mostra en pantalla una finestra amb el territori i una llista de esdeveniments. En seleccionar qualsevol esdeveniment de la llista es mostra el corresponent estat en el
Territori. La següent figura mostra l'aspecte de l'aplicació després de seleccionar un esdeveniment (`F Turn South`):



Les claus per a interpretar la figura són:

- La quadrícula de l'esquerra representa l'estat del territori (inicialment buit), i la llista de la dreta la seqüència d'esdeveniments. El contingut del territori representa la situació en el punt seleccionat en la llista d'esdeveniments (inicialment cap).
- En seleccionar un esdeveniment de la llista es mostra l'estat del territori. La selecció d'esdeveniments és lliure (avance, arrere, salt a l'esdeveniment desitjat, ...).
- Una cel·la del territori pot estar en blanc (buida) o contenir una formiga, representada amb una lletra.
- Si la formiga vol desplaçar-se a una altra cel·la apareix en color roig i amb un punt en la direcció en la qual desitja desplaçar-se. Tota formiga que no s'està desplaçant es dibuixa en negre.
- Per a la llista d'esdeveniments s'utilitza la següent notació:

Esdeveni-ment	**Significat**
`X hi (f,c)`	La formiga X arriba al territori, i se situa en la posició (fila,columna).
`X turn dir`	La formiga X vol desplaçar-se a la cel·la veïna en la direcció dir (North, West, South, East).
`X go`	La formiga X completa el moviment (es desplaça a la casella veïna corresponent).
`X retry`	La formiga X estava esperant (destinació ocupada) i reintenta el moviment.
`X chgDir`	La formiga X estava esperant (destinació ocupada), i ha decidit canviar d'adreça.
`X bye`	La formiga X abandona el territori (la casella queda buida).

Sobre la figura que mostra el resultat de l'execució podem interpretar el següent:

- Formigues que no intenten cap moviment: I, J, D, H, L, K, C, N.
- Formigues que intenten un moviment cap a una destinació lliure: F
  - Quan la destinació està lliure, el següent esdeveniment correspon al desplaçament pròpiament dit (en la figura esdeveniment F go).
  - Únicament pot haver-hi com a màxim una formiga en aqueixa situació
- Formigues que intenten un moviment cap a una destinació ocupada: E, G, A, O, M.
  - E i A estan en situació d'interbloqueix: mai podran completar els seus respectius moviments.
  - La resta de formigues en espera (G,O,M) podran completar el seu moviment quan queden lliures les respectives caselles destine.
- En el títol de la finestra s'indiquen els paràmetres de la simulació
- En la part superior apareix com a missatge el tipus de monitor i la quantitat de reintents (esdeveniments retry) realitzats per part de totes les formigues durant la simulació completa.

Anàlisi del codi

El codi proporcionat implementa diferents classes. Cap d'elles ha de modificar-se

- *Post,* Op, Hi, Bye, Turn, Retry, Go, ChgDir, Viewer *són* "classes opaques", necessàries per al correcte funcionament de l'aplicació, però sense interès per a l'alumne.
- Ant *és* la classe que estén a Thread i implementa el concepte de formiga. Ha de comprendre's el funcionament del seu mètode run (ja presentat).
- Interface Terrain (ja comentat).

- Terrain0 és el punt de partida per a dissenyar la resta de territoris . Ha d'analitzar-se fins a comprendre el seu funcionament. Les operacions sobre v (atribut de tipus Viewer) són necessàries per al funcionament del programa: en desenvolupar Terrain1, Terrain2, Terrain3 han de mantenir-se.

```
class Terrain0 implements Terrain {
    Viewer v;
    public Terrain0 (int t, int ants, int movs) {
        v=new Viewer(t, ants,*movs,"0.- basic monitor");
        for (int i==0; i<ants; i++)
            new Ant(i, this, movs) .start(); // llança les formigues
    }
    public synchronized void hi (int a) {v.hi(a); }
    public synchronized void bye (int a) {v.bye(a); }
    public synchronized void move (int a)
            throws InterruptedException {
        v.turn(a); Post dest=v.dest(a);
        while (v.occupied(dest)) {
            wait();
            v.retry(a );
        }
        v.go(a );
        notifyAll();
    }
}
```

- No es proporciona el codi de les classes Terrain1, Terrain2, Terrain3 (han de desenvolupar-se en aquesta pràctica).
- Ants és **la classe principal**. Arreplega els arguments des de línia d'ordres i llança les simulacions utilitzant els diferents tipus de territori.

Activitat 0 (Sincronització bàsica en Java)

Llance diverses execucions del codi proporcionat usant Terrain0 i els valors per defecte (java Ants), i complete la següent taula, on s'indique per a cada execució:

- el número total de reintents (retries) realitzades per les formigues
- i la quantitat de formigues que s'han quedat al final bloquejades.

Prova	Total Retries	Núm. formigues interbloquejades
1		
2		
3		
4		
5		
6		

Repetisca les execucions amb l'ordre java Ants *0 8 10 3*

Prova	Total Retries	Núm. formigues interbloqueadas
1		

2		
3		
4		
5		
6		

Activitat 1 (Sincronització amb ReentrantLocks en Java)

Utilitzant les eines ReentrantLock i Condition proporcionades en java.*util.concurrent, desenvolupe Terrain1.

Emplene les taules següents
java Ants *1*

Prova	Total Retries	Núm. formigues interbloquejades
1		
2		
3		
4		
5		
6		

java Ants *1 8 10 3*

Prova	Total Retries	Núm. formigues interbloquejades
1		
2		
3		
4		
5		
6		

S'aprecien diferències significatives entre Terrain0 i Terrain1?

- es produeixen menys reactivacions "innecessàries" de fils en Terrain1 que en Terrain0?

Activitat 2 (Ús de diverses variables Condition)

Desenvolupe Terrain2, utilitzant un array de condicions (tantes condicions com cel·les)
NOTA.- La sentència v.getPos*(a)* retorna la posició actual de la formiga a.
NOTA.- A partir d'una posició (ex. *Post* dest=v.dest*(a)*;) podem accedir a les seues components *x* e i (dest.x i dest.i)

Emplene les taules següents
java Ants *2*

Prova	Total Retries	Núm. formigues interbloquejades
1		
2		
3		
4		
5		
6		

java Ants *2 8 10 3*

Prova	Total Retries	Núm. formigues interbloquejades
1		
2		
3		
4		
5		
6		

S'aprecien diferències significatives entre Terrain0, Terrain1 i Terrain2?

Activitat 3 (Gestió de interbloquejos)

Desenvolupe Terrain*3* . Terrain3 utilitza una condició per cel·la (com Terrain*2*), però a més resol el problema del interbloqueix. Quan una formiga completa un temps màxim d'espera sense poder completar el moviment (ej 300ms), realitza un canvi de direcció v.chgDir*(a);* (la qual cosa obliga a recalcular la destinació);
NOTA.- await*(*long timeout*,* TimeUnit unit*)* bloqueja fins al signal corresponent (en el cas del qual retorna true) o bé fins al venciment del període timeout indicat (en aquest cas retorna false)

Indica com classificar el mecanisme utilitzat per a resoldre el problema dels interbloquejos (prevenció, evitació, detecció+recuperació), i si escau la condició de Coffman que es trenca.