Atelierul lui Mos Craciun

I. Vine Craciunul!

Craciunul se apropie! Mos Craciun si angajatii sai (elfii si renii) se pregatesc pentru un nou Craciun care aduce bucurie si cadouri tuturor copiilor din jurul lumii.

Dar Mos Craciun are nevoie de ajutor!

El este proprietarul unui atelier care contine mai multe fabrici (in fiecare an el reconstruieste fabricile). Angajatii sunt elfii care construiesc jucarii, renii care asteapta sa impacheteze jucariile sub forma de cadouri. Mos Craciun are nevoie disperata ca atelierul sa functioneze, tinand cont ca ultimul manager batran al atelierului a plecat. Planul atelierului contine reguli care ne asigura ca toate cadourile sunt create la timp astfel incat nici un copil sa nu ramana fara darul de Craciun.

Mos Craciun doreste sa-l ajuti la implementarea planului fabricii utilizand concurenta in Java (el stie ca esti expert in acest domeniu).

(Mos Craciun fericit ca tu il vei ajuta !)

In continuare prezentam cateva informatii utile despre atelierul lui Mos Craciun, cat si sarcinile pe care le aveti de realizat.

II. Instructiunile primite de la Mos Craciun

Mos Craciun ne-a transmis urmatoarele informatii despre atelierul sau:

1. El detine **mai multe** fabrici de jucarii:

(Fabrica de jucarii)

- Numarul de fabrici este aleator (intre 2-5). O fabrica creaza jucarii (nu se stie numarul jucariilor necesare pana in momentul in care planul fabricii este creat si executat)
- Toate fabricile au o dispunere matriciala de forma $N \times N$ (unde N este un numar aleator intre 100-500)
- Fiecare fabrica contine o multime de elfi.
- La fiecare interval de cateva secunde fabrica isi intreaba toti elfii sa-i transmita pozitiile lor curente
- Nu pot exista mai mult de $\frac{N}{2}$ elfi in fabrica

2. **Elfii** sunt foarte importanti. Ei construiesc toate jucariile, astfel:

(Elf fericit muncind din greu)

- Elfii sunt creati aleator in fiecare fabrica intr-o pozitie aleatoare, la un moment aleator de timp (intre 500-1000 millisecunde)
- Imediat dupa creare, fiecare elf trebuie sa informeze fabrica despre existenta lui
- Doi elfi nu se pot afla in aceeasi pozitie
- Fiecare elf lucreaza independent de ceilalti elfi

- Elfii creaza cadouri astfel:
    - Mutandu-se intr-una din directiile: stanga, dreapta, sus, jos)
    - Cand ajung la un zid al fabricii se muta in oricare din alta directive decat inspre zidul fabricii
    - La fiecare mutare ei creaza un cadou
    - Daca un elf este inconjurat de alti elfi si nu se poate misca el opreste lucrul pentru o durata aleatoare de timp (ingtre 10-50 millisecunde)
    - Cand un cadou este creat elful care l-a creat trebuie sa informeze fabrica despre ce cadou s-a creat, iar fabrica v-a interoga toti elfii sa comunice locatia lor pentru a fi transmisa lui Mos Craciun
    - Dupa ce un elf a creat un cadou, el trebuie sa se odihneasca pentru 30 millisecunde
        *) Elfii sunt rapizi
    - Elfii lucreaza… continuu (cel putin pana la 25 Decembrie). Astfel ca ei nu trebuie sa se opreasca.

3. Mos Craciun are si o multime de **reni**:

(Ren in asteptarea cadourilor pentru a fi transmise lui Mos Craciun)

- Renii asteapta sa primeasca cadouri de la fabrici. Renii sunt disponibili in atelier
- Ei pot intotdeauna citi de la fabrici numarul cadourilor (cat si lista acestora). Dar accesul la fabrica nu le va fi permis in timp ce elfii instiinteaza fabricile despre noile cadouri sau in timp ce fabrica interogheaza elfii despre pozitiile lor
- Un numar maxim de 10 reni pot citi la un moment dat de la o fabrica
- Intre doua citiri consecutive de la o fabrica trece un interval de timp aleator
- Renii vor transmite toate cadourile printr-o conducta catre Mos Craciun
- Mai multi reni pot transmite simultan cadouri catre Mos Craciun, in timp ce Mos Craciun va prelua cate un cadou la un moment dat
- Renii se cunosc de la inceput si sunt in numar de cel putin 8, intrucat Mos Craciun are nevoie de rezerve.

4. Atelierul lui Mos Craciun va realiza urmatoarele:

(Semn catre casa ascunsa de la Polul Nord a lui Mos Craciun)

- Atelierul contine toate fabricile
- Se creaza fabricile
- Se creaza elfii aleator in fabrici si fiecare elf se inscrie la fabrica in care a fost creat

III. Lucruri necesare

Fluxul de control concurent in acest exemplu este destul de subtil. Miscarea elfilor (ce include si raportarea la fabrica) este initiata independent de fiecare elf, fiecare ruland pe cate un fir separat.

Astfel ca elfi diferiti se pot misca si raporta la fabrica in mod concurent. Raportarea concurenta la fabrica trebuie realizata in mod sincronizat.

Apelul de raportare declanseaza metoda individuala de raportare pentru toti elfii inregistrati la fabrica. Acest aspect trebuie tratat cu grija deoarece ambele metode de raportare si miscare sunt sincronizate, astfel ca nu vor putea fi apelate concurent pentru un acelasi obiect.

IV. Indicatii

Mos Craciun stie ca il puteti ajuta. El stie ca aveti la dispozitie urmatoarele instrumente utile de programare concurenta:
- Semafoare, monitoare si zavoare
- Fire de executie. Fiecare elf poate fi reprezentat de un fir separate de executie in cadrul planului atelierului
- Ele trebuie sa comunice: elfii creaza cadouri si renii le primesc pentru a le transmite lui Mos Craciun care le va livra copiilor
- Mos Craciun vrea ca renii sa-I transmita cadourile prin TCP/IP, pentru a nu se pierde nici un cadou.

V. Sarcini suplimentare

1. Retragerea unui elf

Puteti adauga un nou fir la program care retrage cate un elfi la interval aleatoare de timp. Elfii vor fi retrasi intr-o ordine aleatoare. Retragerea unui elf inseamna ca metoda sa *run* trebuie sa se termine. Acest lucru se va realiza prin terminarea normal a buclei, si nu prin folosirea apelului invechit *stop*. In plus, elful trebuie eliminat din fabrica in mod corect astfel ca spatiul de memorie alocat obiectului elf sa poata fi refolosit.

Pentru aceasta puteti folosi un semafor pe care elfii incearca sa-l achizitioneze pentru “a primi permisiunea sa se retraga”. Semaforul este initial zero (adica “blocat”) si poate fi eliberat de un numar de ori intr-un fir pornit in main. Este foarte util sa se incerce achizitionarea semaforului cu metoda *tryAcquire*.

Implementati si testati programul fiind siguri ca intelegeti cum se realizeaza retragerea elfilor intr-o ordine nepredictibila. Optional, puteti extinde programul mai mult astfel incat elfii retrasi se fie recreate la un moment aleator ulterior.

2. “Odihnirea” unui elf

Modificati programul astfel incat atunci cand un elf se afla in zona diagonalei principale ($|x - y| \leq \delta$, unde x si y sunt coordonatele elfului si δ este un parametru cu valoare fixata, de exemplu $\delta = 1$), el se va opri din miscare. Optional se permite ca un elf sa sara peste zona diagonalei principale intr-o singura miscare, evitand odihnirea. Daca toti efii s-au oprit in zona diagonalei principale, ei vor fi treziti astfel incat sa-si continue miscarea pana ce vor ajunge sa se odihneasca din nou in zona diagonalei principale. Secventa miscare/odihnire se va repeta la infinit.

Acest comportament se poate realiza printr-o sincronizare de tip “bariera” ce se poate realiza cu $M + 1$ semafoare: un semafor comun pe post de bariera, pe care firele elfilor il elibereaza cand ajung la punctul de sincronizare si un vector de semafoare, cate unul pentru fiecare elf (sunt M elfi), pe care firele elfilor trebuie sa-l treaca pentru a-si continua executia dincolo de bariera. Sincronizarea la bariera se va realiza prin achizitionarea repetata a semaforului bariera de M ori, urmata de eliberarea semafoarlor de continuare a executiei.

3. Folosirea clasei CyclicBarrier

Pachetul *java.util.concurrent* contine clasa *CyclicBarrier*, ce furnizeaza o metoda mai convenabila de implementare a sincronizarii la bariera. Rescrieti programul cu odihnirea elfilor folosind aceasta clasa.

4. Implementarea proprie a barierei

Se cere realizarea propriei implementari a unei versiuni simplificate a clasei *CyclicBarrier* cu specificatia urmatoare:

```
public class CyclicBarrier {

   public CyclicBarrier(int parties);

   public void await();
}
```

Parametrul *parties* reprezinta numarul de fire care trebuie sa ajunga la bariera inainite de a fi lasate sa continue.

Indicatie: Nu putem folosi un vector de semafoare, intrucat ar trebui ca *await* sa aiba ca parametru valoarea i ce indica indexul semaforului pe care sa realizam blocarea. In schimb, putem folosi un singur semafor pe care asteapta toate procesele si un numarator care contorizeaza cate procese au ajuns la bariera. Acest numarator este partajat, asa ca trebuie sa protejam accesul la el cu un zavor / semafor separate. Nu putem folosi metoda “synchronized” din Java in locul acestui semafor. De ce?