Marele vrăjitor - Salvează lumea

I. Salvarea lumii de strigoi!

Halloween a trecut încă o dată! Marele Vrăjitor și personalul său (vrăjitoare și demoni) se pregătesc pentru încă o tură în salvarea lumii și învingerea strigoiilor care în fiecare an încearcă să cucerească lumea.

Marele vrăjitor

El deține mai multe cuiburi raspandite în întreaga lume în locuri ascunse (în fiecare an încearcă din nou să salveze lumea). El are demoni în fiecare cuib care creează ingrediente pentru poțiuni, are vrăjitoare în așteptarea ca ingredientele pentru poțiune să fie făcute pentru a crea poțiuni și are nevoie disperată ca planul său să funcționeze. Cu poțiuni poate învinge toți strigoii care se vor ridica.

El a aflat de la noi că sunteți cu toții pregătiți să-l ajutați și să implementați un plan de cuiburi prin utilizarea concurenței în Java (știe că sunteți deja toți experți în domeniu), care ii vor oferi ajutorul necesar.

Mai jos puteți găsi câteva informații pe care le știm și le aflăm de la Marele Vrăjitor despre planurile sale și lucrurile care trebuie făcute:

II. Lucruri despre care ne-a spus Marele Vrăjitor

El ne-a instruit să vă spunem următoarele lucruri pe care le are în minte:

1. Există mai multe **Cuiburi** de vrăjitoare în întreaga lume:
    - Număr aleator de Cuiburi (între 3 și 20) care creează diferite ingrediente pentru poțiune
    - Fiecare Cuib creează un număr nelimitat de ingrediente pentru poțiuni de diferite tipuri [există 10 posibile ingrediente diferite]
    - Toate Cuiburile sunt sub formă matricială (NxN aleatoriu, unde N este cuprins între 100-500)
    - Fiecare cuib conține o listă cu demoni
    - Nu poate fi mai mult decât N / 2 număr de demoni într-un cuib
    - Fiecare cuib la fiecare 10 secunde își va anunța demonii să se oprească timp de 1 secundă, speriind toți demonii și făcându-i să-și piardă ingredientele

2. Există Demoni. Demonii sunt cu adevărat importanți; creează toate ingredientele pentru poțiuni. Iată ce se știe despre ei:

(Demon la munca)

- Demonii sunt creați aleatoriu în fiecare cuib într-un loc aleatoriu în cuib (timp aleatoriu între 500-1000 milisecunde)
- Când fiecare demon este creat la întâmplare, trebuie să-i spună cuibului ca au fost adăugați cuibului
- Nu pot fi doi demoni pe aceeași poziție
- Fiecare demon lucrează independent de orice alt demon
- Demonii creează ingrediente pentru poțiuni prin:
    - Deplasarea într-o direcție (stânga, dreapta, sus, jos)
    - Când ajung la un perete al cuibului, ei se mișcă în orice altă direcție decât cea a peretelui
        - De asemenea, pentru 2 ture în plus, nu vor putea crea niciun ingredient în acest caz
        - După 10 lovituri in zid, demonii vor fi retrogradați cu 100 de nivele pe abilitățile lor sociale (a se vedea mai jos ce înseamnă abilități sociale)
    - Odată ce se mișcă, creează, de asemenea, un ingredient de un tip aleatoriu din lista tipurilor de ingrediente posibile
    - Dacă un demon este înconjurat de alți demoni și nu se poate mișca, acesta se oprește pentru o perioada de timp aleatorie (între 10-50 milisecunde)
        - Când un demon întâlnește un alt demon în timp ce încearcă să se miște, amândoi își cresc nivelul de abilități sociale
    - Cu fiecare 100 de niveluri de calificări de abilitate socială, demonii sunt promovați și astfel pot crea 1 ingredient suplimentar la fiecare mișcare
        - Numărul maxim de ingrediente posibil să fie creat la fiecare mișcare de un demon este de 10
    - Când ingredientul este creat, își notifică cuibul pentru a ști ce ingredient(e) au creat
    - După ce demonul creează un ingredient, trebuie să se odihnească o perioadă scurtă de timp (30 de milisecunde)

      *) Sunt foarte rapizi
    - Demonii vor funcționa pentru totdeauna. Ca atare, nu trebuie să se oprească.

3. Are și **vrăjitoare**:

(Vrăjitoare în așteptare, întrebându-se dacă primește toate ingredientele pentru a crea poțiuni)

- Vrăjitoarele așteaptă să primească ingrediente din cuiburi.
- Ele pot citi întotdeauna din cuiburi numărul de ingrediente (și care ingrediente), dar nu vor avea acces atunci când demonii notifică cuibul despre noile ingrediente
- Un maxim de 10 vrăjitoare pot citi dintr-un cuib la un moment dat (acesta este și numărul maxim de vrăjitoare pe care Marele Vrăjitor îl are)
- Un timp aleatoriu trebuie să treacă între două lecturi de cuib consecutive

(Vrăjitoare care vizitează cuiburi pe mătură)

- Când o vrăjitoare vizitează un cuib, va primi ingrediente disponibile, le va pune în geanta sa magică și va crea poțiuni din ele
    - Există 20 de poțiuni diferite, fiecare necesitând între 4-8 ingrediente de tipuri specifice, astfel încât să fie creată
    - Fiecare poțiune poate fi creată într-un număr aleatoriu de timp (între 10-30 de milisecunde)
    - Pentru ca vrăjitoarele să nu fie aglomerate cu ingredientele, găsiți un mecanism adecvat care să citească doar cât mai multe ingrediente posibile pentru a crea poțiuni, având în vedere, de asemenea, să nu aveți prea multe ingrediente per cuib (a se vedea corectitudinea)
    - Este la latitudinea Dvs. să vă gândiți la potențialele poțiuni și la lista lor de ingrediente
- Vrăjitoarele vor pune toate poțiunile printr-un portal magic și le vor trimite Marelui Vrăjitor
- Mai multe vrăjitoare pot pune în același timp poțiuni prin portalul magic, în timp ce Marele Vrăjitor citește singur toate poțiunile trimise prin portalul magic

- Vrăjitoarele sunt cunoscute de la început.

4. Există strigoi:
   - La început este un număr fix de strigoi, un număr aleatoriu între 20-50
   - De fiecare dată la întâmplare (între 500-1000 de milisecunde), ei vizitează la întâmplare cuiburi
   - Fiecare vizită la un cuib, pe strigoi, înseamnă următoarele:
     - Un număr aleatoriu de demoni, între 5-10 este retras, fiind prea speriati să mai lucreze
     - Ingredientele actuale disponibile in cuib se pierd
   - Dacă există vrăjitoare care vizitează cuibul, atunci:
     - O vrajitoare se va lupta cu un strigoi
     - Va cere 2-5 poțiuni de la Marele Vrăjitor pentru a lupta cu strigoiul
     - Dacă există poțiuni disponibile, strigoiul va fi speriat
     - Dacă nu, pentru fiecare strigoi neînvins, 10% din ingredientele disponibile în legământ se vor pierde, dar nu se va pierde niciun demon

5. Cercul Marelui Vrăjitor va face următoarele:
   - acest cerc conține toate cuiburile
   - creeaza cuiburile
   - creează ajutorul Marelui Vrăjitor, care creează demoni și le dă un cuib aleatoriu în care să lucreze (nu uitați că fiecare demon este responsabil să se înregistreze în cuib)

III. Lucruri care sunt necesare

Vă rugăm să rețineți că fluxul de control (concurent) din acest exemplu este destul de subtil. Mutarea demonilor și raportarea la cuib este inițiată independent de fiecare dintre demoni, deoarece fiecare dintre ei rulează un fir de executie separat.

Din această cauză, diferiți demoni pot executa acțiunea de mișcare și de a raporta catre cuib simultan. Raportarile simultane catre cuib trebuie, de asemenea, să fie sincronizate.

Dar chemarea apelului de raportare declanșează, de asemenea, apelarea la metoda de raportare individuală a tuturor demonilor înregistrați în cuib. Acest lucru trebuie din nou să fie sincronizat în mod corespunzător, deoarece ambele metode de raportare și mișcare sunt sincronizate, ceea ce înseamnă că nu vor fi chemate simultan pentru un anumit obiect.

IV. Câteva indicii

Marele Vrăjitor știe că îl poți ajuta. El știe că aveți ca instrumente următoarele noțiuni legate de concurenta cu care îl puteți ajuta:

- Semafoare, Monitoare și Zavoare (știe că le veți folosi în mod corect pe toate)
- Firele de executie sunt cu adevărat importante pentru el. El a spus chiar că fiecare dintre demonii săi poate fi un fir separat în implementarea planului sau de cuiburi
- El vrea ca ei să comunice: demonii creează ingredientele pentru poțiune și vrăjitoarele le primesc, le transformă în poțiuni pentru a da Marelui Vrăjitor pentru a salva lumea

- Și Marele Vrăjitor vrea ca vrăjitoarele să-i livreze poțiunile în locația sa printr-un portal magic TCP/IP (a aflat că este bine). Dacă primește toate poțiunile, atunci poate ști daca poate să salveze lumea.

V. Sarcini importante de îndeplinit (de asemenea)

1. Puteți retrage un demon

Prima voastră sarcină este să adăugați încă un fir la programul dvs. care retrage demonii la momente aleatorii (adică cu întârzieri aleatorii scurte între ele). Dar demonii ar trebui retrași în ordine aleatorie. Retragerea unui demon înseamnă că metoda de rulare a acesteia trebuie să se încheie. Acest lucru ar trebui obținut făcând bucla să se încheie normal și NU prin apelarea metodei de a opri un fir.

Mai mult, demonul trebuie îndepărtat din cuib (într-un mod sigur similar cu adăugarea unui demon într-un cuib). După ce s-a făcut acest lucru, sistemul Java va recupera în cele din urmă spațiul alocat pentru obiectul demon.

Pentru a rezolva problema, puteți folosi un semafor pe care demonul încearcă să-l achiziționeze pentru a „obține permisiunea de a ieși la pensie”. Semaforul este inițial zero și apoi a fost lansat de mai multe ori într-un fir început în principal. Rețineți că este foarte util să încercați să achiziționați semaforul, folosind metoda tryAcquire.

Puneți în aplicare acest lucru și testați-vă programul. Asigurați-vă că înțelegeți modul în care designul face ca ordinea de retragere să fie imprevizibilă. Dacă doriți, puteți schimba comportamentul programului în continuare, astfel încât demonii pensionari să fie reîncărcați după un timp (aleatoriu).

2. Puteți acorda timp demonului să doarmă (pentru că ați observat că este foarte obosit)

Acum reveniți la versiunea originală a programului (faceți un nou director și reutilizați setul inițial de clase pe care l-ați implementat)

Acum trebuie să modificați programul pentru a obține următorul comportament: Când un demon după una dintre mișcările sale se găsește în zona diagonală a lumii (adică unde x este foarte aproape de y), acesta se va „culca”, adică se va opri in miscare. Rețineți că un demon poate sări peste zona diagonală într-o singură mișcare; acest lucru nu-l va determina să doarmă. Când toți demonii au fost inghetati la diagonală, toți se vor trezi și vor continua să se miște până când vor dormi din nou pe diagonală. Această mișcare / somn continuă pentru totdeauna.

Ar trebui să recunoașteți acest lucru ca o formă de sincronizare a barierei care poate fi obținută folosind semafoarele N + 1: un semafor de barieră comun, care este eliberate de catre firele de demoni când ajung la punctul de sincronizare și o serie de semafoare „continuă”, indexate prin fir, pe care firele il achiziționează pentru a continua dincolo de barieră.

De asemenea, este necesar un proces special de sincronizare a barierei, care dobândește în mod repetat semaforul de barieră de N ori, urmat de eliberarea tuturor semaforilor continue.

3. Demonii adormiți revizuită

Pachetul java.util.concurrent include clasa CyclicBarrier, care oferă mijloace mai convenabile pentru a realiza sincronizarea barierei. Rescrie programul din exercițiul anterior folosind această clasă în loc de semafoare.

4. Propria ta barieră ciclică

Acum pentru un exercițiu mai greu: implementați corpul propriei clase CyclicBarrier care oferă funcții similare cu clasa Java cu același nume. Dar ne mulțumim cu o versiune mai simplă cu următoarele specificații:

```
public class CyclicBarrier {

   public CyclicBarrier(int parties);

   public void await();
}
```

Parametrul parties reprezintă numărul de fire care trebuie să ajungă la barieră înainte de a le permite tuturor să continue. Scrie întreaga clasă și apoi folosește-o pentru a rezolva din nou problema demonilor adormiți.

Sugestie: nu putem urmări abordarea de tip sir-de-semafoare direct, deoarece asta ar necesita așteptarea de a avea un parametru i care indică indexul semaforului pe care să îl blocheze. În schimb, s-ar putea încerca să se folosească un singur semafor pe care toate procesele il folosesc pentru a fiblocate și o variabilă întreagă, numărând câte procese au ajuns la barieră. Dar această variabilă întreagă este partajată, de aceea trebuie să protejăm actualizările acesteia folosind un al doilea semafor mutex. Nu putem folosi blocarea sincronizată Java în locul semaforului mutex. De ce?