Tema de Casa

Filip Alina-Andreea
Calculatoare Romana
Anul III
Grupa 2

January 17, 2021

1 Problem statement

Companiile farmaceutice din lume au finalizat insfarsit producerea unui vaccin impotriva covid-19, dar au nevoie de ajutorul nostru pentru distribuirea vaccinului in lume. Am primit un set de instructiuni de la companie ce trebuie implementate folosind concurenta in Java.

Am considerat ca o metoda care ar respecta cerintele problemei referitoare la accesul la vaccine ar fi folosirea unui zavor pentru resursele din clada Fabrica.

De asemenea clasa fabrica este un thread la randul sau deoarece trebuie sa citeasca pozitiile robotilor producatori la randul sau.

Roboti producatori produc vaccine atata timp cat au unde sa se miste, iar dupa producerea unui vaccin se odihneste.

2 Pseudocode of algorithms

Pentru realizarea acestui program am utilizat mai multi algoritmi impartiti in mai multe fisiere sursa.

```
1: function MAIN                                          ▷ In Companie.java
2:     doze ← Rand()
3:     NrF ← Rand
4:     for i ← 1 to NrF do
5:         N ← Rand()
6:         NrRP ← Rand()
7:         for i ← 1 to NrRP do
8:             robotiP ← robotiP.add
9:         end for
10:        fabrici ← fabrici.add
11:    end for
12:    while current< doze do
13:        request ← "doza"
14:        if receive == "DA" then
15:            dozeCompanie ← dozeCompanie+1;
16:        end if
17:    end while
18: end function
```

```
function RUN                                                    ▷ In RobotT.java
    while true do
        client ← listen.accept()
        c ← connection
        c ← c.start()
    end while
end function
```

```
function RUN                                                    ▷ In Connection.java
    if daca == "Doza" then
        for i ← 1 to NrF do
            if takevaccine() == true then
                primit ← true
                send ← "DA"
            end if
        end for
        if primit == false then
            send ← "NU"
        end if
    end if
end function
```

```
function RUN                                                    ▷ In Fabrica.java
    while RobotiActivi do
        sleep ← 10
        for i ← 1 to NrRP do
            r ← robot.row
            c ← robot.col
        end for
    end while
end function
```

```
function APPEND
    lock ← lock
    while count == Ncitire do
        stackFull ← await
    end while
    vaccin[newest] ← v
    count ← count+1
    stackEmpty ← signalAll
    lock ← unlock
end function
```

```
1: function TAKE
2:     lock ← lock
3:     while count == 0 do
4:         stackEmpty ← await
5:     end while
6:     vaccin[oldest] ← v
7:     count ← count-1
8:     stackFull ← signalAll
9:     lock ← unlock
10: end function
```

```
1: function RUN                                          ▷ In RobotP.java
2:     while i <= max do
3:         active ← 1
4:         if checkmove==true then FABRICA.APPEND(i)
5:             sleep ← 3/1000
6:         end if
7:         i ← i+q
8:     end while
9:     active ← 0
10: end function
```

```
1: function CHECKMOVE
2:     if col+1<N and matrice[row][col+1]==0 then
3:         matrice[row][col] ← 0
4:         col ← col+1
5:         matrice[row][col] ← 1
6:     end if
7:     if row+1<N and matrice[row+1][col]==0 then
8:         matrice[row][col] ← 0
9:         row ← row+1
10:        matrice[row][col] ← 1
11:    end if
12:    if row-1>=0 and matrice[row-1][col]==0 then
13:        matrice[row][col] ← 0
14:        row ← row-1
15:        matrice[row][col] ← 1
16:    end if
17:    if col-1>=0 and matrice[row][col-1]==0 then
18:        matrice[row][col] ← 0
19:        col ← col-1
20:        matrice[row][col] ← 1
21:    end if
22:    if nomove then
23:        sleep ← rand()
24:    end if
25: end function
```

3 Application outline

In continuare voi vorbi despre componentele proiectului.

3.1 Architectural overview

Avem o clasa Companie care se ocupa cu creearea fabricilor/punctelor de lucru si a robotilor producatori care vor fi asignati fiecarei fabrici. De asemenea clasa se ocupa si cu creearea robotilor transportori si trimitereacererilor de transport de doze.

Clasa RobotT se ocupa cu primirea cererilor si stabilirea conexiunii cu fabricile.

Clasa RobotP se ocupa cu creearea dozelor la fiecare miscare in matricea fabricii asignate robotului. Are o functie care verifica daca robotul poate sa se miste iar in caz contrar robotul ia o pauza.

Clasa Connection reprezinta conexiunea intre roboti transportori si fabrici. Aici se apeleaza functia de extragere a unei doze din fabrica.

Clasa Fabrica porneste functionarea robotilor sai producatori, contine dozele produse de robot si o funtie cu ajutorul careia roboti transportori iau dozele din fabrica.

3.2 Input format

Input-ul consta intr-un intreg reprezentand numarul de doze din plan.

3.3 Output format

Output-ul consta in string-uri care afiseaza activitatile executate de thread-urile din program. Cum ar fi: creearea unei doze, transportarea ei etc.

3.4 Implementare

Am considerat fabrica ca fiind o resursa comuna cu zavoare. Cand un thread acceseaza resursa blocheaza zavorul si il elibereaza la terminarea prelucrarii sale in resursa.

De asemenea fabrica reprezinta un thread deoarece are nevoie sa verifice periodic pozitiile robotilor sau producatori in matrice.

Initiam am crezut ca functia care determina daca robotul producator se poate misca poate fi imbinata cu functia care adauga vaccinul in fabrica insa aceasta optiune nu prea reprezenta cerinta din problema in care verificarea posibilitatii unei miscari in matrice este atribuita robotului producator asadar am creeat mai multe functii de extragere a valorii din matrice si de setare a valorilor pentru ca robotul producator sa poata citi valoarea unei matrici de la pozitia de interes.

Robotul producator se aseamana cu producatorul facut la laboratorul 7 cu zavoare, el creeaza doze din q in q incepand cu id-ul astfel incat un robot sa nu produca acelasi vaccin din lista de 2 ori sau un vaccin oarecare.

Roboti transportori folosesc conexiunea pentru a extrage o doza din fabrica si a o primite la sediul principal/companie.

3.5 Solutie

Solutia aleasa de mine pentru implementarea acestei probleme consta in:

- Clasa companie joaca rolul de client pentru robotul transportor
- Clasa robot transportor joaca rolul de server pentru clasa companie si de consumator pentru clasa Fabrica
- Clasa fabrica joaca rolul de resursa comuna cu zavor intre tobotul transportor si robotul producator
- Clasa connection reprezinta conexiunea dintre fabrica si robotul transportor si companie

Toate clasele folosesc consurenta in Java, astfel roboti producatori functioneaza independent unii de alti si de restul thread-urilor din program, roboti transportori folosesc TCP/IP pentru comunicarea cu compania, iar fabrica funcioneaza independent atunci cand cere pozitiile orbotilor si nu le afecteaza modul de producere al vaccinelor.

Ca date de test am folosit o functie random care alege aleator un numar de doze ce trebuie produse. Datele rezultata reprezentand activitatile desfasurate de fire.

4 Observatii

Fiind destul de compicata si cu o lista stufoasa de instructiuni, problema a fost destul de incurcata si a reprezentat o provocare implementrarea sa.

Programul scris de mine nu are o functionare optima intrucat robotul transportor nu extrage dozele din fabrici, astfel avand nevoie de mai mult rafinament si poate o privire mai atenta asupra ordinii in care ar trebui sa se intampla lucrurile.

Mai multe explicatii se regasesc in comentariile din cod.

5 References

Codul din laboratorul 7 si laboratorul cu conexiunile TCP/IP.