Ferma unchiului John

ŢOCA Sebastian - Bogdan

Sesiunea de *Iarnă 2023*

Pentru rezolvarea problemei, am împărţit aplicaţia în două părţi:

- o parte ce cuprinde crearea fermelor de găini şi a server-ului (**Ferma_John.java**);
- o parte ce cuprinde activitatea angajatilor fermei (**AngajatClient.java**).

1 Partea 1 - Ferma_John.java

Au fost create următoarele clase:

1. clasa Ferma ce extinde clasa Thread pentru monitorizarea fermei din 5 în 5 secunde;
2. clasa Gaina ce extinde clasa Thread pentru monitorizarea fiecărei găini;
3. clasa Multi ce extinde clasa Thread pentru tratarea solictărilor venite de la angajaţi;
4. clasa Ferma_John ce creează vectorul de ferme cât şi server-ul.

1.1 Clasa Ferma

În cadrul acestei clase am declarat următoarele atribute şi obiecte:

1. **N** ce reprezintă dimensiunea matricei din fermă
2. **mat** matricea unde sunt pozitionate gainile din fermă
3. **oua** matricea unde sunt pozitionate ouale in ferme
4. **nr_gaini** ce reţine numarul gainilor din fermă
5. **gaini** vectorul de obiecte (gainile din fermă)

6. **timp_monitorizare** numărul de milisecunde pentru monitorizare fermă
7. **ferma_nr** numărul fermei
8. **ou_depus** arată dacă s-a depus cel puţin un ou în fermă
9. **angajat_nr** reţine numărul angajatul care adună ouăle

Iar ca metode folosite:

1. Comstrutorul **Ferma(numarul_fermei, dim_N, vectorul_gaini, ng1, ng2)**, unde **ng1**,**ng2** partea de început şi sfarşit din vectorul_gaini;
2. **get_Ferma()** returnează un obiect tip Ferma;
3. **Initializare_Ferma()** la început nu avem găini poziţionate în fermă şi nici ouă în fermă;
4. **Aranjare_Gaini(vector_găini)** aşează aleator găinile în fermă;
5. **afisare_Ferma()** permite afişarea fermei la fiecare fir de execuţie;
6. **afisare_vector_oua()** crează un fişier pentru fiecare angajat cu poziţia ouălelor din fermă;
7. **verifica_poz(x,y)** returnează TRUE dacă este liberă casuţa unde se deplasează gaina sau nu a atins limtele fermei;
8. **deplasare_gaina()** permite deplasarea fiecărei găini prin fermă. Aici se aşteaptă un timp aleator **t**, unde 10 ¡= t ¡= 50 milisecunde până la următoarea mutare posibilă a găinii, dacă este înconjurată de alte găini;
9. metoda **run()** ce execută fiecare fir în parte;
10. **set_ou_depus(boolean)** specifică că a fost depus cel puţin un ou în fermă;
11. **get_ou_depus()** returnează dacă a fost depus cel puţin un ou în fermă;
12. **set_angajat_nr(int)** setează numărul angajatului care va colecta ouăle din fermă.

1.2 Clasa Gaina

În cadrul acestei clase am declarat următoarele atribute şi obiecte:

1. **timp_ou** repreyintă timpul de depunere a ouălelor aleator;
2. **pozitie_x**,**pozitie_y** reprezintă poziţia unde se află gaina;
3. **ou_depus** indică dacă a fost depus oul sau nu;
4. **gaina_nr** numărul găinii din ferme (Fiecare găină are un număr unic);

Iar ca metode folosite:

1. **Gaina(int x, int y, int ng)** constructorul de iniţializare găina, aşezarea găinii cu numărul *ng* pe coordonate (x,y) în fermă;
2. **get_x()** returnează poziţia găinii pe linie;
3. **get_y()** returnează poziţia găinii pe coloană;
4. **get_gaina_nr()** returnează numărul găinii din fermă
5. **set_x(int)** setează poziţia găinii pe linie;
6. **set_y(int)** setează poziţia găinii pe coloană;
7. **depune_ou()** metoda în care găina depune ou, apoi se odihneşte aproximativ 30 milisecunde;
8. **deplasare_gaina()** generează o posibilă deplasare a găinii;
9. metoda **run()** ce execută fiecare fir în parte;

1.3 Clasa Multi

Aceasta trateză fire multiple de execuţie a solictărilor venite la server; În cadrul acestei clase am declarat următoarele atribute şi obiecte:

1. **s** obiect al clasei **Socket**;
2. **infromClient** obiect al clasei **DataInputStream**;
3. **out** obiect al clasei **OutputStream**;
4. **writer**cobiect al clasei **BufferedWriter**;

5. **solicitare_angajat** reprezintă angajatul care solicită informaţii despre fermă;

6. **fj** obiect al clasei **Ferma_John**;

7. **sunt_oua_la_ferma** arată dacă sunt ouă depuse de găini la fermă

Iar ca metode folosite:

1. Constructorul **Multi(Socket s,Ferma_John fj)** unde se iniţializează fluxurile de intrare/iesire între server şi angajaţi;

2. metoda **run()** tratează fiecare solicitare a angajaţilor, verifică dacă sunt depuse ouă în fermă şi răspunde clientului (angajatului);

1.4 Clasa Ferma_John

Aceasta trateză execuţia simultană a fermelor.

1. **vector_gaini** se creează vectorul de găini (fiecare găină are alocat un număr unic;

2. **nr_gaini** numărul găinilor din fermă;

3. **nr_ferme** numărul de ferme;

4. **f** vectorul de obiecte tip Ferma.

Iar ca metode folosite:

1. constructorul **Ferma_John(int n)** iniţializeaă fermele, unde n reprezintă numărul de găini/ În cadrul contructorului se distribuie găinile aleator în ferme;

2. **StartFerma(** porneşte fiecare fir (fermă);

3. **afisare_vector_gaini()** tipăreşte vectorul de găini

4. **afisare_ferma_oua()** returnează numărul fermei de unde se pot colecta ouă;

5. metoda principală **main** unde se creează ferma şi server-ul.

2 Partea 2 - AngajatClient.java

Au fost create următoarele clase:

1. clasa ReceiverClient ce extinde clasa Thread pentru tratarea răspunsului favorabil primit de la fermă (de la server);

2. clasa AngajatClient ce extinde clasa Thread pentru tratarea răspunsului primit de la fermă (de la server). Aici se întreabă ferma dacă sunt ouă de adunat;

2.1 Clasa ReceiverClient

Este o extensie a clasei Thread, pentru a executa fire independente. În cadrul acestei clase am declarat atribute şi obiecte pentru menţinerea fluxurilor de comunicare între client şi server, cât şi

1. **angajat_nr** pentru angajatul ce solicită informaţiil;

2. **v** matricea cu ouă ale fermei;

3. **n** dimensiunea matricei fermei NxN

4. **ferma_nr** numărul fermei de unde adună angajatul ouăle.

Iar ca metode folosite:

1. Constructorul **ReceiverClient(AngajatClient chat, InputStream in, int angajat_nr)** unde se iniţializează fluxurile de intrare de la server şi numărul angajatului;

2. **citire_fisier()** citeşte din fişierul alocat fiecărui angajat datele despre ferma de unde adună aouăle;

3. **afisare_oua()** afişează matricea fermei de ouă şi numărul de ouă colectate;

4. metoda **run()** tratează fiecare răspuns al server-ului, dacă sunt ouă de colectat, iar în caz afirmativ afişează matricea fermei;

5. **stopThread()** opreşte din execuţie firul curent.

2.2 Clasa AngajatClient

În cadrul acestei clase am declarat atribute şi obiecte pentru menţinerea fluxurilor de comunicare între client şi server. Metode folosite:

1. **AngajatClient(String address, int port, int angajat_nr)** constructorul care stabileşte adresa IP cât şi portul pe care ascultă clientul;
2. **connect()** returnează TRUE sau FALSE dacă s-a stabilit conexiunea cu server-ul sau nu;
3. metoda **run()** tratează solicitarea către server pentru a colecta ouă;
4. **stopThread()** opreşte din execuţie firul curent;
5. metoda principală **main** unde se creează lista de angajaţi, a conxiunii şi porneşte fiecare fir (angajat în parte, alocându-i un timp de aşteptare de 2000 milisecunde.

3 Compilare şi executare

3.1 Compilare

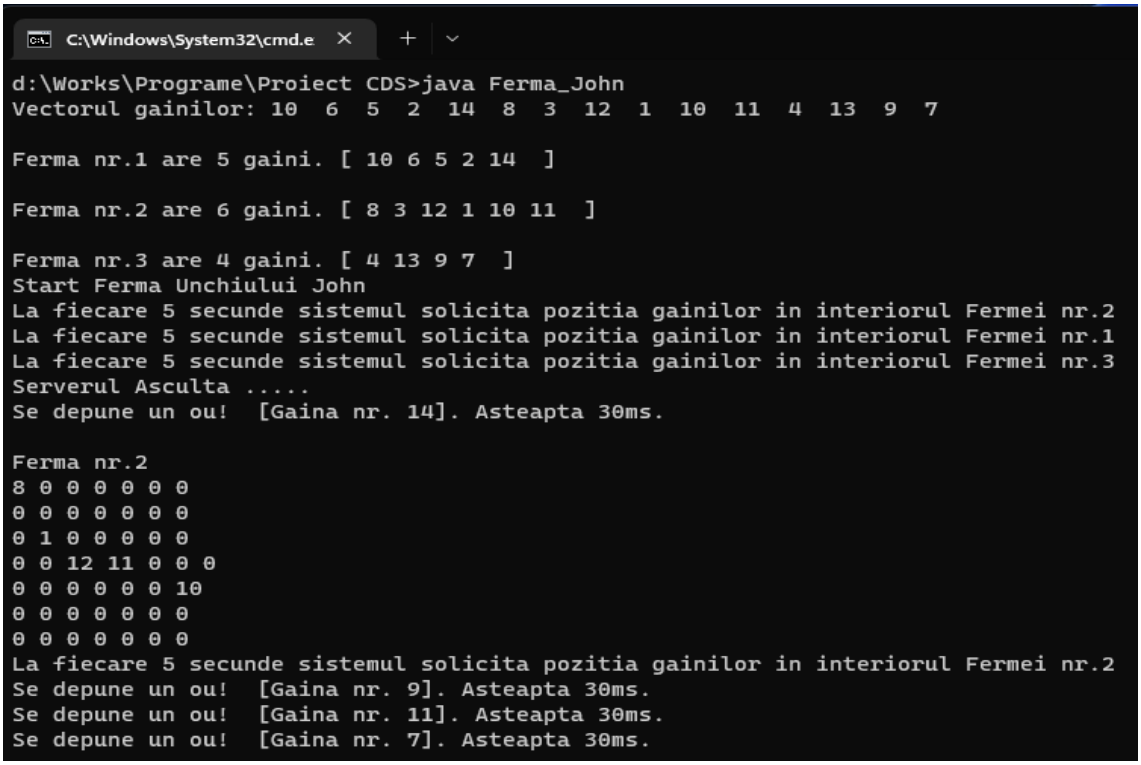
În folderul unde se află cele două coduri sursă JAVA am deschis două ferestre **Command Prompt**, apoi în una dintre acestea am lansat comenzile:

1. **javac Ferma_John.java**
2. **javac AngajatClient.java**

3.2 Lansarea în execuţie

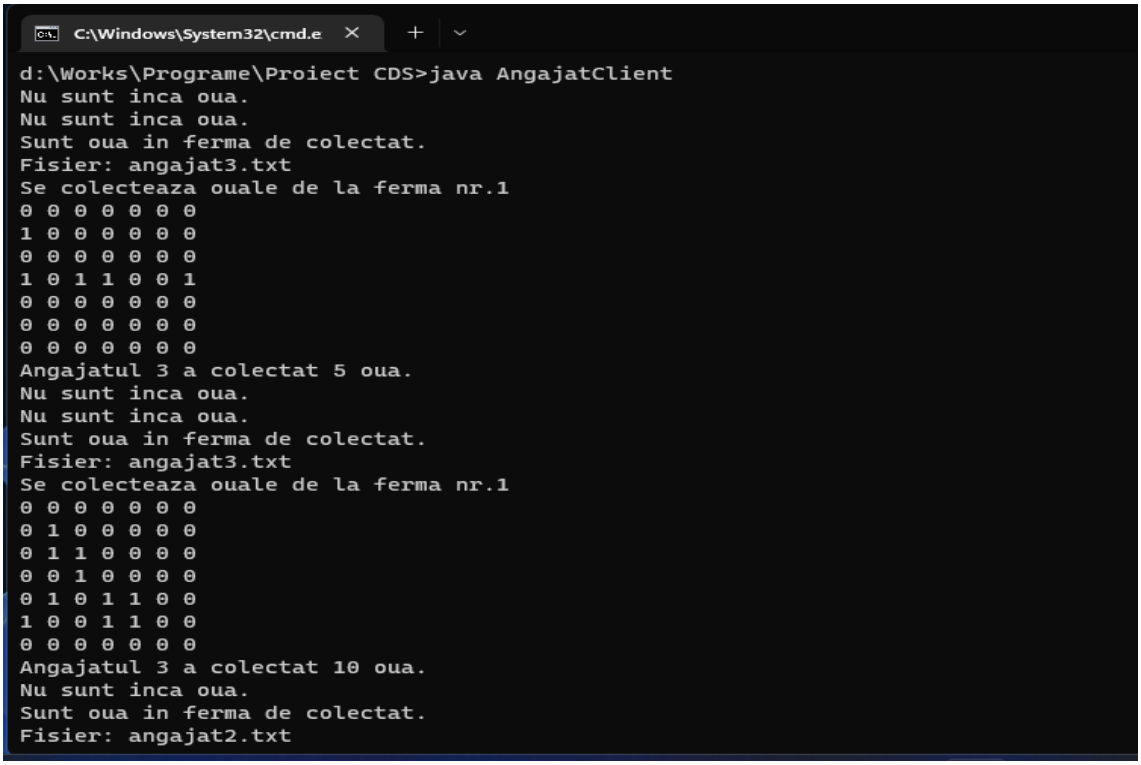
În prima fereastră de **Command Prompt** vom lansa serverul, tastând:

java Ferma_John



Apoi, în a doua fereastră de **Command Prompt** vom lansa aplicaţia de tip client, cea a angajaţilor, tastând:

java AngajatClient


C:\Windows\System32\cmd.e
d:\Works\Programe\Proiect CDS>java AngajatClient
Nu sunt inca oua.
Nu sunt inca oua.
Sunt oua in ferma de colectat.
Fisier: angajat3.txt
Se colecteaza ouale de la ferma nr.1
0 0 0 0 0 0 0
1 0 0 0 0 0 0
0 0 0 0 0 0 0
1 0 1 1 0 0 1
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
Angajatul 3 a colectat 5 oua.
Nu sunt inca oua.
Nu sunt inca oua.
Sunt oua in ferma de colectat.
Fisier: angajat3.txt
Se colecteaza ouale de la ferma nr.1
0 0 0 0 0 0 0
0 1 0 0 0 0 0
0 1 1 0 0 0 0
0 0 1 0 0 0 0
0 1 0 1 1 0 0
1 0 0 1 1 0 0
0 0 0 0 0 0 0
Angajatul 3 a colectat 10 oua.
Nu sunt inca oua.
Sunt oua in ferma de colectat.
Fisier: angajat2.txt