Raport Tehnic
Sisteme Concurente si Distribuite

Nedelcu Robert-Ionut
Grupa CR3.2B

January 2023

1 Uncle John's Eggs Farm

1.1 Enunt

Uncle John heard that selling organic eggs has become a great business nowadays. Therefore, he decided to breed hens and create some farms for producing organic eggs. The hens lay eggs which are picked up by his employees and sent to the farm headquarter. Uncle John relies on you to help him implement the farm plan by using concurrency in Java.

Unchiul John a auzit că vânzarea de ouă ecologice a devenit o afacere grozavă în zilele noastre. Prin urmare, el a decis să crească găini și să creeze niște ferme pentru producerea de ouă ecologice. Găinile depun ouă care sunt ridicate de angajaţii săi şi trimise la sediul fermei. Unchiul John se bazează pe tine pentru a-l ajuta să implementeze planul fermei folosind concurența în Java.

1.2 Instructiuni Ferma

Există mai multe ferme:

- Găinile sunt crescute în ferme. Numărul de ferme este un număr aleator între 2 și 5
- Fiecare fermă este proiectată ca o matrice cu N x N unde $100 \leq N \leq 500$
- Fiecare fermă are o listă de găini
- La fiecare câteva secunde sistemul de monitorizare a fermei va solicita poziția găinilor in interiorul fermei
- Nu pot exista mai mult de N / 2 gaini pentru o ferma

Există găini care produc ouă:

- Găinile depun oua aleatoriu în fiecare fermă într-un loc aleator, la un moment aleator t, unde $500 \leq t \leq 1000$ (milisecunde)
- Două găini nu pot fi în același loc
- Găinile depun ouă prin:
  - Deplasarea într-o direcție (stânga, dreapta, sus, jos)
  - Când ajung la gardul fermei, se deplasează în orice altă direcție decât în gard
  - La fiecare mișcare, depun un ou
  - Dacă găina este înconjurată de alte găini și nu se poate mișca în nicio direcție, se oprește pentru un timp aleatoriu t, unde $10 \leq t \leq 50$ (milisecunde)
  - De fiecare dată când se produce un ou, sistemul de monitorizare a fermei informează angajații despre poziția în care se află oul
  - După ce se creează un ou, găina trebuie să se odihnească până când este capabilă să producă un alt ou (timpul de odihnă este de 30 milisecunde)
  - Găinile vor produce ouă cât sunt în viață
- La fiecare câteva secunde sistemul de monitorizare a fermei va solicita poziția găinilor in interiorul fermei
- Nu pot exista mai mult de N / 2 gaini pentru o ferma

Există și angajați ai fermei:

- Angajații așteaptă să primească ouă de la fermă
- Ei pot citi oricând informații din sistemul de monitorizare a fermei cu privire la numărul de ouă depuse, dar nu li se va permite accesul atunci când senzorii notifică sistem de monitorizare că un ou a fost depus sau când sistemul de monitorizare a fermei însuși solicită senzorii despre noile ouă depuse.
- Maxim 10 angajați ai fermei pot citi simultan din sistemul de monitorizare
- Trebuie să treacă un timp aleatoriu între două citiri consecutive ale sistemului de monitorizare
- Toți angajații vor livra ouăle la sediul fermei
- Numărul angajaților este cunoscut de la început. Sunt mai mult de 8

Sediul fermei va face următoarele:

- Sediul cuprinde toate fermele
- El creează fermele
- Acesta creează găinile și le atribuie la ferme aleatorii (rețineți că fiecare găină trebuie să o facă se înregistrează singur la fermă)

1.3 Lucruri care sunt necesare

Vă rugăm să rețineți că fluxul de control (concurent) din exemplu este destul de subtil. Mișcarea găinilor iar raportul de ouă este inițiat independent de fiecare găină, deoarece fiecare dintre ele rulează separat fir de control. Din acest motiv, diferite găini pot executa acțiunea de mutare și raportul de ouă simultan. Dar raportul de apel declanșează și apelarea metodei de raportare individuală a tuturor găinilor înregistrate în cadrul fermei. Acest lucru din nou trebuie îngrijit corespunzător, deoarece atât raportează, cât și se mută metodele sunt sincronizate, ceea ce înseamnă că nu vor fi apelate simultan pentru un anumit obiect.

1.4 Câteva indicii

Cerințe preliminare de concurență:

- Semafoare, monitoare și încuietori
- Firele sunt importante. De exemplu, fiecare găină poate fi modelată ca un fir separat în implementarea managementului fermei
- Colaborarea este cheia: găinile depun ouăle pe care angajații le ridică și le trimit la sediul fermei
- Managerul fermei dorește ca angajații să trimită ouăle prin TCP/IP

2 Intelegere Problema

2.1 Abordare

Deci stim ca unchiul John vrea sa isi deschida o afacere in care are mai multe ferme de gaini pentru a produce oua.Deci fiecare ferma va contine gaini care vor face oua cand se muta intr-una din directiile sus , jos , dreapta , stanga.Gainile sunt create random si ocupa de asemenea o pozitie random in ferma creata la un timp aleatoriu.Ferma avand o dispunere matriciala NxN si nu pot exista mai mult de N/2 gaini.Ouale trebuie sa fie transmise prin angajati, unchiului John Deci vom avea clasa mare si anume sediul fermelor sau afacerea propriu zisa Mai departe vom avea clasa pentru ferma ca fiind ferma propriu zisa, clasa unchiului John care il evidentiaza pe unchiul John si de asemenea vom avea clase pentru pozitia si directia gainilor, o clasa separata pentru generarea lor ,o clasa pentru angajati si inca o clasa pentru trimiterea oualor ,plus inca una pentru a rula aplicatia

2.2 Construirea Claselor

SaleEggs - reprezinta afacerea propriu zisa sau sediul fermelor, ea cuprinzand numarul de ferme , fermele propriu zise, gainile si angajatii fermelor, avand si metoda de creare ferme.

Aceasta functie ne ajuta sa cream fermele , cu ajutorul functiei Random, vom genera un numar intre 2 si 5 ferme, un numar de angajati care se stie de la inceput ca este mai mare ca si 8, apoi vom genera gainile in functie de cate ferme vom avea, ca la final sa rulam executarea acestora.

```
public void createFarm() {
        Random rand = new Random();
        number_farms = rand.nextInt(4) + 2; // Numar random intre 2 si 5
        // Numar random , dar se stie ca sunt
        // mai multi de 8 de aceea vom adauga 8
        int number_employees = rand.nextInt(10) + 8;
        farms = new Farm[number_farms]; // Cream fermele
        hens_generators = new HensGenerator[number_farms]; // Cream gainile
        employees = new Employee[number_employees]; // Cream angajatii

        List<String> asList = Arrays.asList(
        MessageFormat.format("Au fost create {0} ferme", number_farms),
                MessageFormat.format(
                "Au fost creati {0} angajati", number_employees));
        for (String s : asList) {
            System.out.println(s);
        }
        // Aici se incepe rularea lor
        IntStream.range(0, number_farms).forEachOrdered(i -> {
            int number = rand.nextInt(500) + 100;
            farms[i] = new Farm(number, i + 1);
            hens_generators[i] = new HensGenerator(farms[i]);
            System.out.println(MessageFormat.format("Ferma {0} are {1} gaini",
                    i + 1,
                    number));
        });
        IntStream.range(0, number_employees).forEachOrdered(i -> {
            employees[i] = new Employee(farms, i + 1, transferQueueEggs);
        });
        IntStream.range(0, number_farms).forEachOrdered(i -> {
            hens_generators[i].start();
            farms[i].start();
        });

    }
```

Farm - reprezinta o ferma respectiva, avand ca date numarul fermei curente, dimensiunea acesteia, zavoarele si lista de gaini cat si de oua si un semafor pentru odihnirea gainilor respective.Aceasta clasa prezinta functii de adaugare a gainilor , raportarea pozitiei si directiei cat si crearea de oua, plus functii de

get.

Aceasta functie ne va ajuta sa adaugam gaini impreuna cu pozitia acesora in ferma noastra.

```
public boolean addHen(HenPosition henPosition) {
        var result = false;
        hensListLock.lock(); // Blocam lista de gaini

        var X = henPosition.getXposition();
        var Y = henPosition.getYposition();
        // Daca pozitia acesteia nu este deja ocupata , o adaugam in lista

        if (!farm[X][Y]) {
            farm[X][Y] = true;
            henArrayList.add(henPosition); // Adaugam gaina
            henPosition.start();
            // Raportam pozitia actuala a gainii
            henPosition.reportHenPosition();
            hensListLock.unlock(); // Deblocam lista de gaini
            result = true;
        }
        return result; // Returnam rezultatul
    }
```

Cu ajutorul acestei functii vom aveam pozitia gainii si vom putea schimba dupa cum spune si enuntul problemei, in timp ce o gaina isi raporteaza pozitia , aceasta isi poate modifica pozitia curenta , deci dupa ce raportam vom putea schimba pozitia acesteia si vom putea vedea si directia ei de mers , ca apoi sa verificam in ce parte va merge pentru a putea vedea daca aceasta va putea crea oua.In acest caz am verificat daca o ia in jos,sus,stanga sau dreapta , ea va face oua , altfel nu.

```
private void positionOfHen() {
        try {
            Random rand = new Random();
            // Blocam listele
            farmLock.lock();
            hensListLock.lock();
            eggsLock.lock();
            Iterator<HenPosition> iterator = henArrayList.iterator();
            if (iterator.hasNext()) {
                do {
                    HenPosition hen = iterator.next();
                    hen.reportHenPosition(); // Raportam pozitia gainii
                    // Generam random coordonata X pentru ca gaina
```

```
                // se poate muta
                // dupa ce raporteaza pozitia
                var X = rand.nextInt(size);
                // Generam random coordonata Y pentru ca gaina
                // se poate muta
                // dupa ce raporteaza pozitia
                var Y = rand.nextInt(size);
                // Schimbam pozitia gainii , deoarece dupa ce
                // isi raporteaza
                // ea isi poate schimba directia de mers
                hen.changeHenPosition(X, Y);
                // Instantiem directia acesteia dupa
                // ce am adaugat noile coordonate dupa schimbare
                HenDirection hen_direction = new HenDirection(size);
                // Verificam daca gaina se muta in jos,atunci
                // va crea ou
                if(hen_direction.moveDown(hen.getXposition(),
                hen.getYposition())){
                    createEgg(hen.getNumber(), count_egg);
                }
                // Verificam daca gaina se muta in stanga,atunci
                // va crea ou
                else if(hen_direction.moveLeft(hen.getXposition(),
                hen.getYposition())){
                    createEgg(hen.getNumber(), count_egg);
                }
                // Verificam daca gaina se muta in dreapta,atunci
                // va crea ou
                else if(hen_direction.moveRight(hen.getXposition(),
                hen.getYposition())){
                    createEgg(hen.getNumber(), count_egg);
                }
                // Verificam daca gaina se muta in sus,atunci
                // va crea ou
                else if(hen_direction.moveUp(hen.getXposition(),
                hen.getYposition())){
                    createEgg(hen.getNumber(), count_egg);
                }
                // Altfel gaina nu va crea ou
            } while (iterator.hasNext());
        }
    } catch (Exception e) {
        System.out.println(MessageFormat.format("Exceptie din ferma: {0}", e
        e.printStackTrace();
    } finally {
        // Dupa raportarea pozitiei vom debloca listele.
```

```
            hensListLock.unlock();
            farmLock.unlock();
            eggsLock.unlock();

        }
    }
```

Aici avem functia pentru care gaina va scoate un ou.

```
    private void createEgg(int hen_number, int egg) {
        try {
            eggsLock.lock(); // Blocam lista de oua
            eggs.add(egg); // Adaugam oul
            count_egg++; // Daca oul s-a creat
            // inseamna ca oul viitor scos de aceeasi
            // gaina o sa fie incrementat cu 1
            System.out.println(MessageFormat.format("Gaina cu numarul: {0}
            a scos oul: {1}", hen_number, egg));
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            eggsLock.unlock(); // Deblocam lista de oua
        }
    }
```

Aceasta functie reprezinta functia de get a unui ou.De fiecare data cand un ou este luat , dimensiunea va scadea cu 1.

```
    public int getEgg() throws Exception {
        int egg;
        try {
            employeeSemaphore.acquire();
            // Maxim 10 angajati ai fermei pot
            // citi simultan din sistemul de monitorizare
            eggsLock.lock(); // Blocam lista de oua
            egg = eggs.get(eggs.size() - 1); // Luam oul
            eggs.remove(eggs.size() - 1);
            // Apoi decrementam deoarece am luat un ou ,
            // deci numarul de ou va fi mai mic cu 1
        } finally {
            eggsLock.unlock(); // Deblocheaza lista de oua
            employeeSemaphore.release();
        }
        return egg; // Returnam oul
    }
```

Employee - reprezinta angajatul propriu zis, acesta avand ca date numarul acestuia curent,fermele existente de unde trebuie sa ia oua, si o lista in care va

culege ouale de la gaini.Ca metode avem 2 functii , una fiind de transfer a oualor de catre angajat de la ferme si una de transfer a oualor catre unchiul John.

Aceasta functie ne va ajuta sa transportam ouale de la angajati catre unchiul John.

```
private void transferEggToUncleJohn(int egg) {
    // Dam oul
    eggs.giveEgg(egg);
    System.out.println(MessageFormat.format("Angajatul cu numarul
    {0} i-a trimis lui unchiul John
    oul cu numarul {1} ", number, egg));
}
```

Aceasta functie reprezinta metoda ca un ou sa fie luat de la ferma de catre angajati.

```
private int transferEggFromFarm() throws Exception {
    Random rand = new Random();
    int farm = rand.nextInt(SaleEggs.number_farms);
    // Luam oul
    return farms[farm].getEgg();
}
```

UncleJohn - reprezinta clasa care il evidentiaza pe unchiul John si se comporta ca un fir de executie, aici avem si metoda prin care John va primi ouale de la angajati.

Afacerea da roade asadar unchiul John primeste oua.

```
public void run() {
    do {
        // Unchiul John primeste oua de la angajatii lui
        int eggs = received_eggs.receiveEggs();
        System.out.println(MessageFormat.format("Unchiul John a
        primit oul cu numarul {0}", eggs));
    } while (true);
}
```

TransferEggs - reprezinta clasa unde se va realiza transportul de oua , ca date avem un contor care tine evidenta oualor transportate si o coada in care vor fi puse, ca si metode avem functii pentru trimiterea si primirea oualor.

Aceasta metoda ne ajuta pentru transferul de oua catre unchiul John,dupa ce oul s-a trimis cu succes , vom incrementa la contor si vom notifica procesul.

```
public synchronized void giveEgg(int egg) {
    if (tail - head == eggsCount.length) {
        do {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        } while (tail - head == eggsCount.length);
    }
    // Se trimite oul deci se adauga la contor
    eggsCount[tail % eggsCount.length] = egg;
    tail += 1;
    notifyAll(); // Dam de stire ca oul s-a trimis cu succes
}
```

Iar aici avem fix invers, metoda pentru care unchiul John va primi ouale de la angajatii lui, exact la fel dupa ce va primi oul , se va adauga la contor si se va notifica procesul.

```
public synchronized int receiveEggs() {
    if (tail == head) {
        do {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        } while (tail == head);
    }
    // Se primeste oul deci se adauga la contor
    int egg = eggsCount[head % eggsCount.length];
    head += 1;
    notifyAll(); // Dam de stire ca oul s-a primit cu succes
    return egg; // Returnam oul
}
```

HensGenerator - aceasta clasa va genera random gainile , ca date de intrare avem ferma pentru care firul va genera gainile , iar ca functii avem metoda de run si o functie pentru crearea gainilor.

Functie pentru creare de gaini, cu ajutorul functiei random , vom da lock la ferma curenta , apoi vom crea random si coordonatele pentru pozitia acesteia si apoi gaina impreun cu pozitia ei va fi adaugata la ferma, crescand astfel si dimensiunea totala a gainilor din ferma , ca la final sa dam unlock.

```
public void createHen() {
```

```
        Random rand = new Random();
        ReentrantLock farmLock = farm.getFarmLock();
        // Gainile nu se pot misca in timp ce in ferma
        // este adugata o noua gaina
        farmLock.lock(); // Vom bloca ferma
        var farmsCount = farm.getSize();
        if (farm.getNumberOfHens() == (farmsCount / 2)) {
            farmLock.unlock(); // Deblocam ferma
        } else {
            var X = rand.nextInt(farmsCount); // Generam random
            // coordonata X pentru pozitie gainii noi
            var Y = rand.nextInt(farmsCount); // Generam random
            // coordonata Y pentru pozitia gainii noi
            ReentrantLock hensCounterLock = SaleEggs.getHensCounterLock();
            // Niciun alt fir nu poate accesa numarul de gaini din ferma
            hensCounterLock.lock();
            // Cream noua gaina
            HenPosition hen = new HenPosition(SaleEggs.number_total_hens
            , X, Y, farm); // Generam random si pozitia acesteia
            // Adaugam gaina in ferma noastra
            if (farm.addHen(hen)) {
                SaleEggs.number_total_hens += 1;
                // Numarul total de gaini va i incrementat cu 1
                System.out.println(MessageFormat.format("Gaina cu numarul
                {0} a fost adaugata in ferma {1} cu succes ",
                hen.getNumber(), farm.getNumber()));
                hensCounterLock.unlock(); // Deblocam counterLock-ul
            } else {
                hensCounterLock.unlock(); // Deblocam counterLock-ul
            }
            farmLock.unlock(); // Deblocam ferma
        }
    }
```

Metoda run

```
    public void run() {
        do {
            Random rand = new Random();
            long millis = rand.nextInt(1000) + 500;
            // Sleeping a random time between 500 and 1000 milliseconds
            try {
                Thread.sleep(millis);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
```

```
            // Crearea unei gaini noi
            createHen();
        } while (true);
    }
```

HenDirection - aceasta clasa reprezinta directa in care gainile vor merge, de asemenea avem functii pentru a returna in ce parte o vor lua gainile (jos,stanga,sus,dreapta).

Aici vom avea metodele pentru a verifica in ce directie o vor lua gainile : sus,jos,dreapta sau stanga.

```
// Functie care verifica daca gaina s-a mutat in stanga
public boolean moveLeft(int X, int Y) {
    return Y - 1 > 0 && !farm[X][Y - 1];
}

// Functie care verifica daca gaina s-a mutat in jos
public boolean moveDown(int X, int Y) {
    return X + 1 < count && !farm[X + 1][Y];
}

// Functie care verifica daca gaina s-a mutat in sus
public boolean moveUp(int X, int Y) {
    return X - 1 > 0 && !farm[X - 1][Y];
}

// Functie care verifica daca gaina s-a mutat in dreapta
public boolean moveRight(int X, int Y) {
    return Y + 1 < count && !farm[X][Y + 1];
}
```

HenPosition - reprezinta clasa in care vom avea pozitia gainilor, ca date avem numarul gainii curente,cat si coordonatele unde se afla in pozitie,precum si ferma de care apartine. Avem de asemenea metode pentru a raporta pozitia curenta cat si pentru a schimba pozitia, deoarece gainile dupa ce isi raporteaza pozitia curenta isi pot schimba cu usurinta pozitia.

Functie pentru a raporta pozitia curenta a gainilor.

```
public void reportHenPosition() {
    System.out.println(MessageFormat.format("Gaina cu numarul {0} se
    afla la ({1},{2}) in ferma {3}",
            number, x, y, henFarm.getNumber()));
}
```

Functie pentru a schimba pozitia unei gaini, dupa ce schimbam pozitia vom raport si directiei.

```
public void changeHenPosition(int newX, int newY) {
    x = newX;
    y = newY;
    // Dupa ce am schimbat pozitia gainii anuntam si directia
    new HenDirection(x);
    new HenDirection(y);
}
```

Metoda de run Semafor pentru odihnirea gainilor

```
public void run() {
    do {
        // Semafor pentru odihnirea gainilor
        SaleEggs.getHenRestSemaphore().release();
        // Sleeping 30 milliseconds
        try {
            Thread.sleep(30);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    } while (true);
}
```

Demo - reprezinta clasa in care vom instantia obiectele si vom porni aplicatia

In demo vom avea functia main pentru instantierea claselor obiect si pentru rularea aplicatiei propriu zise.

```
public static void main(String[] args) {
    // Transferam ouale
    TransferEggs eggs = new TransferEggs();
    // Instantiem pe unchiul John
    UncleJohn uncleJohn = new UncleJohn(eggs);
    // Cream afacerea unchiului John , cea de vanzare de oua
    SaleEggs business = new SaleEggs(eggs);
    // Afacerea se extinde prin crearea de noi ferme
    business.createFarm();
    // Unchiul John va incepe sa primeasca oua
    uncleJohn.start();
}
```

3 Date de intrare

Datele de intrare sunt generate random cu ajutorul functiei Random(). Fermele au o dispunere matriceala N x N , N fiind random intre 100 si 500, angajatii

precum si gainile sunt la fel creati random.

4 Date de iesire

Iesirea este reprezentata prin numarul de ferme create, numarul de angajati creati , precum si de pozitia gainilor,adaugarea gainilor la ferme ,si crearea oualor de catre gaini.

5 Rezultate dupa rularea aplicatiei

La prima rulare observam cum numarul de ferme , angajati cat si numarul de gaini pentru fiecare ferma a fost creat random , apoi va urma pozitia gainilor cat si scoaterea oualor de catre acestea.

```
C:\Users\rober\.jdks\openjdk-18\bin\java.exe "-javaagent:D:\Faculta
Au fost create 4 ferme
Au fost creati 8 angajati
Ferma 1 are 302 gaini
Ferma 2 are 227 gaini
Ferma 3 are 501 gaini
Ferma 4 are 220 gaini
Gaina cu numarul 1 se afla la (153,95) in ferma 4
Gaina cu numarul 1 a fost adaugata in ferma 4 cu succes
Gaina cu numarul 2 se afla la (391,420) in ferma 3
Gaina cu numarul 2 a fost adaugata in ferma 3 cu succes
Gaina cu numarul 3 se afla la (38,34) in ferma 2
Gaina cu numarul 3 a fost adaugata in ferma 2 cu succes
Gaina cu numarul 4 se afla la (233,95) in ferma 1
Gaina cu numarul 4 a fost adaugata in ferma 1 cu succes
Gaina cu numarul 5 se afla la (198,26) in ferma 4
Gaina cu numarul 5 a fost adaugata in ferma 4 cu succes
Gaina cu numarul 6 se afla la (405,52) in ferma 3
Gaina cu numarul 6 a fost adaugata in ferma 3 cu succes
Gaina cu numarul 7 se afla la (184,215) in ferma 1
Gaina cu numarul 7 a fost adaugata in ferma 1 cu succes
Gaina cu numarul 8 se afla la (15,21) in ferma 4
Gaina cu numarul 8 a fost adaugata in ferma 4 cu succes
Gaina cu numarul 9 se afla la (86,212) in ferma 2
Gaina cu numarul 9 a fost adaugata in ferma 2 cu succes
Gaina cu numarul 10 se afla la (373,453) in ferma 3
Gaina cu numarul 10 a fost adaugata in ferma 3 cu succes
Gaina cu numarul 11 se afla la (47,17) in ferma 1
```

```
Gaina cu numarul: 3 a scos oul: 8
Gaina cu numarul 9 se afla la (119,204) in ferma 2
Gaina cu numarul: 9 a scos oul: 9
Gaina cu numarul 13 se afla la (60,24) in ferma 2
Gaina cu numarul: 13 a scos oul: 10
Gaina cu numarul 17 se afla la (225,22) in ferma 2
Gaina cu numarul: 17 a scos oul: 11
Gaina cu numarul 21 se afla la (34,81) in ferma 2
Gaina cu numarul: 21 a scos oul: 12
Gaina cu numarul 27 se afla la (181,61) in ferma 2
Gaina cu numarul: 27 a scos oul: 13
Gaina cu numarul 32 se afla la (126,151) in ferma 2
Gaina cu numarul: 32 a scos oul: 14
Gaina cu numarul 38 se afla la (142,21) in ferma 2
Gaina cu numarul 1 se afla la (180,156) in ferma 4
Gaina cu numarul 4 se afla la (18,238) in ferma 1
Gaina cu numarul 2 se afla la (256,269) in ferma 3
Gaina cu numarul: 38 a scos oul: 15
```

```
Gaina cu numarul: 178 a scos oul: 264
Gaina cu numarul 183 se afla la (214,501) in ferma 4
Gaina cu numarul: 183 a scos oul: 265
Gaina cu numarul 188 se afla la (103,316) in ferma 4
Gaina cu numarul: 188 a scos oul: 266
Gaina cu numarul 193 se afla la (57,370) in ferma 4
Gaina cu numarul: 193 a scos oul: 267
Gaina cu numarul 197 se afla la (238,101) in ferma 3
Gaina cu numarul 197 a fost adaugata in ferma 3 cu succes
Gaina cu numarul 198 se afla la (10,151) in ferma 1
Gaina cu numarul 198 a fost adaugata in ferma 1 cu succes
Gaina cu numarul 199 se afla la (130,206) in ferma 2
Gaina cu numarul 199 a fost adaugata in ferma 2 cu succes
Gaina cu numarul 200 se afla la (367,375) in ferma 5
Gaina cu numarul 200 a fost adaugata in ferma 5 cu succes
Gaina cu numarul 201 se afla la (388,76) in ferma 4
Gaina cu numarul 201 a fost adaugata in ferma 4 cu succes
Gaina cu numarul 202 se afla la (203,43) in ferma 3
Gaina cu numarul 202 a fost adaugata in ferma 3 cu succes
Gaina cu numarul 203 se afla la (10,77) in ferma 2
Gaina cu numarul 203 a fost adaugata in ferma 2 cu succes
Gaina cu numarul 204 se afla la (19,272) in ferma 5
Gaina cu numarul 204 a fost adaugata in ferma 5 cu succes
Gaina cu numarul 205 se afla la (322,132) in ferma 4
Gaina cu numarul 205 a fost adaugata in ferma 4 cu succes
Gaina cu numarul 206 se afla la (137,177) in ferma 1
Gaina cu numarul 206 a fost adaugata in ferma 1 cu succes
Gaina cu numarul 207 se afla la (184,259) in ferma 3
```

6 Concluzie

Am ramas placut surprins de intelegerea si implementarea acestei teme si consider ca a fost o provocare pe care sper ca am trecut-o cu brio. Dupa rezolvarea acestei teme mi-am imbunatatit skill-urile in Latex , cat si in Java Concurrency pot spune . A fost o provocare super interesanta.

7 Referinte

https://www.baeldung.com/java-semaphore

https://www.geeksforgeeks.org/difference-between-lock-and-monitor-in-java-concurrency/

https://www.tutorialspoint.com/java/java_multithreading.htm

https://www.geeksforgeeks.org/multithreading-in-java/