

RAPORT TEHNIC

Universitatea din Craiova, Facultatea de Automatică, Calculatoare și Electronică



Sisteme Concurente si Distribuite

Student : Drăghici Andreea-Maria

Grupa : CR 3.1 B

Anul de studiu : III

Specializarea : Calculatoare Română

◀ Ianuarie 2022 ▶

Rezumat

Acest raport este o introducere în obiectivele de dezvoltare a temei de casă la disciplina Sisteme Concurente și Distribuie.

Introducere cuprins

1	CONTEXTUL PROBLEMEI	2
1.1	<i>Cerinta</i>	2
1.2	<i>Indicatii</i>	2
2	EXPLICATIA REZOLVARII ALESE	3
2.1	<i>Descrierea si intelegerea problemei</i>	3
2.2	<i>Abordarea problemei</i>	3
3	DATE EXPERIMENTALE	5
4	PROIECTAREA APLICATIEI	7
4.1	<i>Structura de nivel inalt a aplicatiei</i>	7
4.2	<i>Specificatia formatului datelor de intrare</i>	9
4.3	<i>Specificatia formatului datelor de iesire</i>	9
4.4	<i>Lista tuturor modulelor aplicatiei si descrierea lor</i>	9
5	Decizia implementarii sarcinilor suplimentare	10
5.1	<i>Task 1. Retragerea unui elf</i>	10
5.2	<i>Task 2. Odihnirea unui elf (utilizand semafoare)</i>	11
5.3	<i>Task 3. Folosirea clasei CyclicBarrier</i>	11
6	OBSERVATII SI REZULTATE	12
7	CONCLUZII SI BIBLIOGRAFIE	15

1 CONTEXTUL PROBLEMEI

1.1 Cerinta

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. 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).

1.2 Indicatii

Mos Craciun stie ca avem 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
- Mos Craciun vrea ca renii sa-l transmita cadourile prin TCP/IP, pentru a nu se pierde nici un cadou, care le va livra copiilor.

2 EXPLICATIA REZOLVARII ALESE

Programare concurenta

2.1 Descrierea si intelegerea problemei

Cunoastem ca Mos Craciun detine mai multe fabrici de jucarii. Fiecare fabrica contine elfi ce sunt creati aleator si ocupa o pozitie aleatoare in fabrica, la un moment aleator de timp, de aici deducem ca fabrica are o dispunere matriciala de forma $N \times N$, deci nu pot exista mai mult de $N / 2$ elfi.

Elfii creeaza cadouri mutandu-se intr-una din directiile: stanga, dreapta, sus, jos, iar la fiecare mutare ei creeaza un cadou. Deducem ca aceste cadouri trebuie sa fie transmise lui Mos Craciun cu ajutorul renilor, renii asteapta sa primeasca cadouri de la fabrici si sa le trimeata printr-o conducta catre Mos Craciun.

Concluzia: Atelierul lui Mos Craciun trebuie sa contina toate fabricile, sa le creeze si sa creeze elfii aleator in fabrici.

2.2 Abordarea problemei

In opinia mea, fiecare indicatie si instructiune enumerata atat mai sus, cat si in fisierul pdf de pe classroom, pot sa fie privite ca niste clase ce contin ca metode si date explicatiile insiruite.

In imaginea de mai jos se poate vedea ahitectura claselor. Mai multe detalii vor fi prezentate in sectiunea: Proiectarea Aplicatiei.

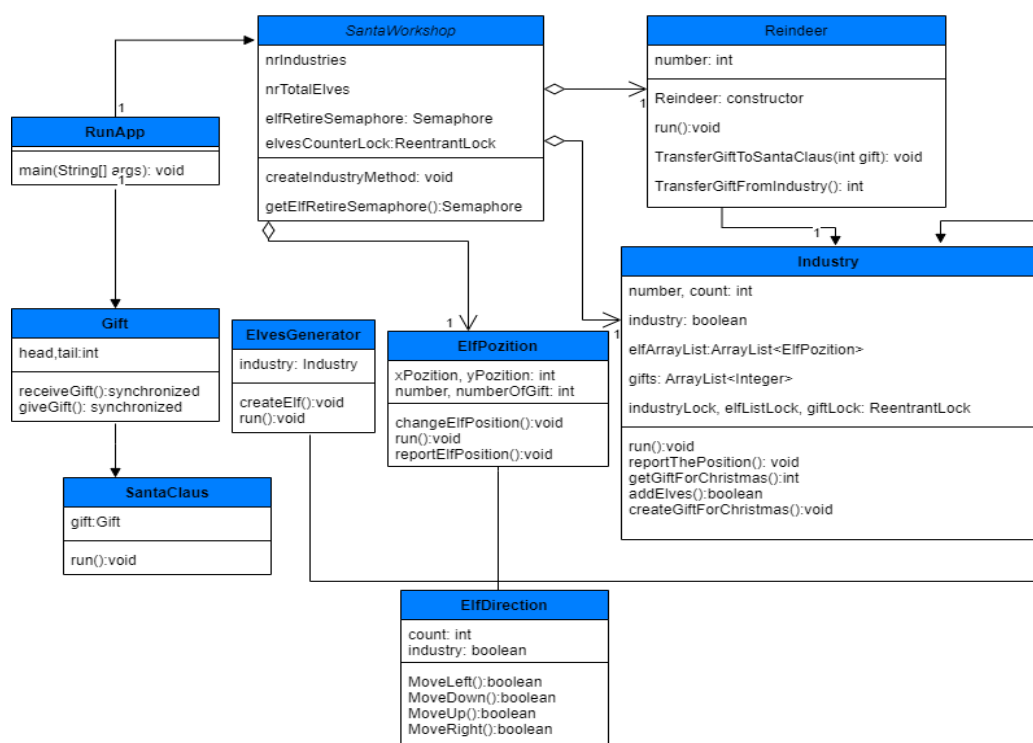


Figura 1: Arhitectura claselor

3 DATE EXPERIMENTALE

Data de intrare vor fi generate aleatoru. O instanta a clasei Random in Java va fi folosita pentru a genera un flux de numere pseudoeleatoare.

Vom genera aleatoru fabricile(intre 2 si 5 fabrici). O fabrica creaza jucarii, nu stim numarul jucariilor necesare pana in momentul in care planul fabricii este creat si executat.

Toate fabricile au o dispunere matriciala de forma N X N, unde N este un numar aleator intre 100-500. In cazul nostru N este redat de variabila number.

```
public void createIndustryMethod() {

    Random rand = new Random();
    nrIndustries = rand.nextInt(4) + 2;
    int nrReindeers = rand.nextInt(10) + 8;
    industries = new Industry(nrIndustries);
    generators = new ElvesGenerator(nrIndustries);
    reindeers = new Reindeer(nrReindeers);

    List<String> asList = Arrays.asList(MessageFormat.format("Au fost create {0} fabrici",
        nrIndustries), MessageFormat.format("Au fost creati {0} reni", nrReindeers));
    for (int i = 0; i < asList.size(); i++) {
        String s = asList.get(i);
        System.out.println(s);
    }
    IntStream.range(0, nrIndustries).forEachOrdered(i -> {
        int number = rand.nextInt(500) + 100;
        industries(i) = new Industry(number, i + 1);
        generators(i) = new ElvesGenerator(industries(i));
        System.out.println(MessageFormat.format("Fabrica {0} are {1} elfi",
            i + 1,
            number));
    });
    IntStream.range(0, nrReindeers).forEachOrdered(i -> {
        reindeers(i) = new Reindeer(industries, i + 1, giftQueue);
    });
    IntStream.range(0, nrIndustries).forEachOrdered(i -> {
        generators(i).start();
        industries(i).start();
    });
}
```

Elfii sunt creati si ei aleator in fiecare fabrica intr-o pozitie aleatoare, la un moment aleator de timp.

Imediat dupa create, fiecare elf creat trebuie sa informeze fabrica despre existenta lui.

Cunoastem ca nu pot exista mai mult de $N/2$ elfi in fabrica, deci numarul de elfi va fi mai mic decat dimensiunea fabricii.

```
public void createElf() {

    Random rand = new Random();
    ReentrantLock factoryLock = industry.getFactoryLock();
    // elfii nu se pot misca in timp ce in fabrica este adugat un nou elf
    factoryLock.lock();

    var industryCount = industry.getCount();
    if (industry.numberOfElves() == (industryCount / 2)) {
        factoryLock.unlock();

    } else {

        var X = rand.nextInt(industryCount);
        var Y = rand.nextInt(industryCount);
        ReentrantLock elvesCounterLock = getElvesCounterLock();

        //niciun alt fir nu poate accesa nr de elfi din fabrica
        //acesta este modificat acum
        elvesCounterLock.lock();
        // creaza un nou elf
        ElfPozition elf = new ElfPozition(nrTotalElves, X, Y, industry);

        // adauga elfii in fabrica
        if (industry.addElves(elf)) {
            nrTotalElves += 1;

            System.out.println(MessageFormat.format("Elful cu numarul {0} a fost creat in
                fabrica {1} cu succes ",
                    elf.getNumber(),
                    industry.getNumber()));

            elvesCounterLock.unlock();
        } else {
            elvesCounterLock.unlock();
        }
        factoryLock.unlock();
    }
}
```

4 PROIECTAREA APLICATIEI

4.1 Structura de nivel înalt a aplicației

Aplicația este alcătuită din pachete ce conțin subpachete, respectiv clase. Fiecare clasă este organizată având membrii și funcții membre. Pentru rezolvarea temei de casă am folosit noțiunile acumulate din platformele de laborator, cât și indicațiile din assignment.

Implementarea conține un singur pachet, acesta având mai multe subpachete:

- **com.tema.home** - pachetul principal;
- **initialimplementation** - subpachetul ce conține implementarea inițială;
- **cyclicbarrier** - subpachetul ce conține implementarea utilizând bariere;
- **semaphores** - subpachetul ce conține implementarea utilizând semafoare;

Primul subpachet reprezintă implementarea inițială, cea generică pentru tema noastră de casă.

Cel de al doilea subpachet reprezintă rezolvarea problemei utilizând Cyclic Barrier, unde am rescris implementarea cu odihnirea elfilor folosind această clasă.

Cel de al treilea subpachet cuprinde rezolvarea problemei utilizând semafoare. Am ales această abordare pentru a avea la final o structură cât mai organizată ce încearcă să atingă fiecare task din assignment.

Imaginea atașată constituie diagrama UML unde am încercat să scot în evidență structura claselor.

Fiecare clasă conține metode ce ajută la rezolvarea problemei. O parte din clase sunt instanțiate în fișierul RunApp, ce reprezintă fișierul principal ce rulează programul.

Cum spuneam și mai devreme, fiecare instrucțiune din assignment a fost transpusă în câte o clasă, respectiv metoda, mai jos se poate vedea acest lucru.

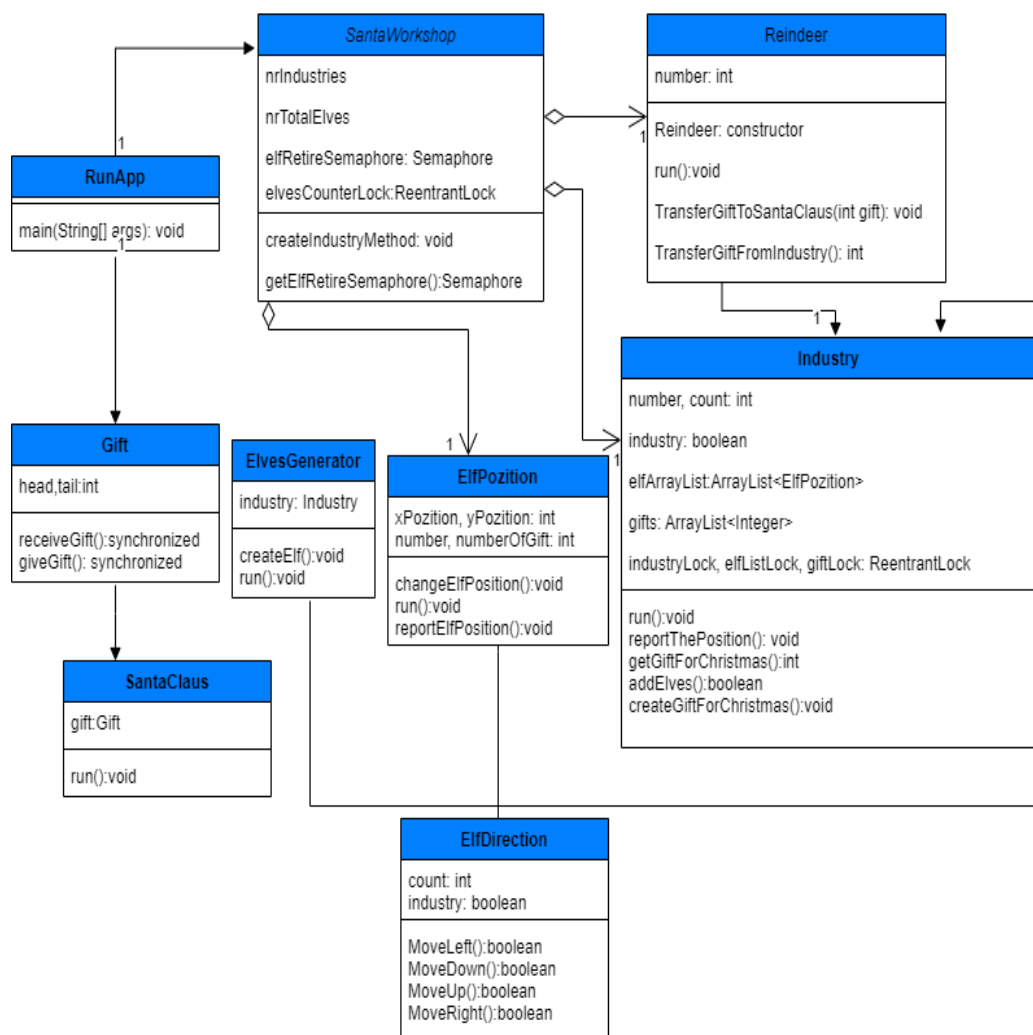


Figura 2: Arhitectura claselor

4.2 *Specificatia formatului datelor de intrare*

Datele de intrare sunt generate aleatoriu folosind o instanta a clasei Random pentru a genera un flux de numere pseudoaleatoare.

Generam aleatoriu fabricile, fabricile avand o dispunere matriceala $N \times N$, iar N este aleatoriu intre 100 si 500.

Elfii sunt creati si ei aleator in fiecare fabrica intr-o pozitie aleatoare, intr-un anumit timp.

4.3 *Specificatia formatului datelor de iesire*

Datele de iesire vor reprezenta numarul de fabrici, numarul de reni, numarul de elfi, cat si pozitia elfilor in fabrica.

4.4 *Lista tuturor modulelor aplicatiei si descrierea lor*

Clasa **ElfDirection** cuprinde implemenatarea pentru a verifica in care dintre directiile stanga, dreapta, sus, jos se poate muta un elf cand creaza cadouri. Pentru a verifica fiecare directie de mai sus, am implementat cate o metoda care ne returneaza rezultatul directiei in care elfii se muta.

Clasa **ElfPozition** implementeaza doua metode, o metoda care ne ajuta sa schimbam pozitia in care un elf se afla, deoarece doi elfi nu se pot afla in aceeasi pozitie si o metoda care ne afiseaza pozitia fiecarui elf in fabrica.

Tot in aceasta clasa intalnim si metoda `run()` care contine un semafor pe care elfii incearca sa-l achizitioneze pentru a primi permisiunea sa se retraga.

Clasa **ElvesGenerator** implementeaza o metoda numita **createElf** care ne ajuta sa generam elfii intr-o anumita fabrica. (aceasta metoda a fost deja explicata mai sus in descrierea datelor experimentale).

Mai implementeaza si metoda `run` unde firul nostru va realiza urmatoarele actiuni (elfii sunt creati aleator in fiecare fabrica intr-o pozitie aleatoare, la un moment aleator de timp) intr-o bucla infinita.

Clasa **Gift** implementeaza doua metode, `receiveGiff` si `giveGift` pentru a primi un cadou de la reni si de a transfera cadourile catre Mos Craciun.

Clasa **Industry** implementează un fir care acționează ca o fabrică. În această clasă mai găsim și o metodă care cere tuturor elfilor existenți poziția lor actuală, deoarece elfii nu se pot mișca în timp ce își raportează poziția, iar renii nu pot primi cadouri în timp ce fabrica le cere elfilor poziția și în timp ce aceștia creează un cadou.

Clasa **Reindeer** implementează un fir ce se comportă ca un ren. Această clasă mai implementează încă două metode, una care oferă cadourile Mosului, punându-le în coada de cadouri și o altă metodă care ne solicită un cadou existent de la o fabrică aleasă la întâmplare.

Clasa **SantaClaus** implementează un fir ce se comportă ca Mos Craciun. Tot aici găsim implementată și metoda care îl ajută pe Mos Craciun să primească cadouri la nesfârșit.

Clasa **SantaWorkshop** reprezintă fix atelierul lui Mos Craciun. Această clasă cuprinde numărul de fabrici existente, numărul total de elfi existenți, numărul de elfi și o metodă care ne ajută să generăm fabricile. Această metodă a fost explicată în paragraful Date experimentale.

Clasa **RunApp** cuprinde o parte din clase ce sunt instantiate aici, în această clasă rulează programul principal. Se creează coada de transfer a cadourilor, îl creează pe Mos Craciun, creează atelierul acestuia, apoi atelierul începe să creeze fabrici, iar Mos Craciun începe să primească cadouri de la reni.

5 Decizia implementării sarcinilor suplimentare

5.1 Task 1. Retragera unui elf

Pentru retragerea unui elf, folosim un semafor pe care elfii încearcă să-l achiziționeze pentru a primi permisiunea să se retragă. Retragera unui elf înseamnă că acesta va fi eliminat din fabrică și din lista elfilor existenți.

```
public void run() {
    do {
        /* semafor pe care elfii încearcă să-l achiziționeze pentru a primi permisiunea să
           se retragă. */
        getElfRetireSemaphore().release();
        // Sleeping 50 milliseconds
        try {
            Thread.sleep(50);
        } catch (InterruptedException e) {
            extracted(e);
        }
    } while (true);
}
```

```
/*
 * metoda pentru sarcina suplimentara" retragerea unui elf
 * blocheaza accesul la lista de elfi
 * blocheaza accesul la lista fabricii
 * elimina pozitia elfului din lista fabricii
 * deblocheaza accesul la lista cu elfii
 * deblocheaza accesul la lista fabricii
 */
public void retireElf(ElfPozition elfPozition) {

    try {
        /* modificarea listei elfilor si a pozitiei matricei fabricii */
        elfListLock.lock();//blocheaza accesul la lista de elfi
        industryLock.lock();//blocheaza accesul la lista fabricii
        elfArrayList.remove(elfPozition);//elimina pozitia elfului din lista fabricii
        int X = elfPozition.getXPozition();
        int Y = elfPozition.getYPozition();
        industry(X)(Y) = false;
        System.out.println(MessageFormat.format("Elful cu numarul {0} s-a retras din fabrica
        cu numarul {1}", elfPozition.getNumber(), number));
    } finally {
        elfListLock.unlock();//deblocheaza accesul la lista cu elfii
        industryLock.unlock();//deblocheaza accesul la lista fabricii
    }
}
```

5.2 Task 2. Odihnirea unui elf (utilizand semafoare)

Presupunem ca elful va ajunge la diagonala principala si va incerca sa dobandeasca un semafor pentru a modifica contorul pentru elfii ce asteapta la bariera, asteptand pana cand contorul este mai mic ca dimensiunea fabricii. Am modificat programul astfel incat atunci cand elful se afla in zona diagonalei principale acesta se va opri din miscare. Astfel daca toti elfii s-au oprit in zona diagonalei principale, ei vor fi treziti sa-si continue miscarea pana cand or ajunge sa se odihneaza in zona diagonalei principale.

5.3 Task 3. Folosirea clasei CyclicBarrier

Presupunem ca elful va ajunge la diagonala principala si va astepta la bariera pana cand N elfi o ating. Am utilizat pachetul `java.util.concurrent` contine clasa `CyclicBarrier`, ce furnizeaza o metoda mai convenabila de implementare a sincronizarii la bariera si am rescris programul cu odihnirea elfilor folosind aceasta clasa.

6 OBSERVATII SI REZULTATE

Pentru sincronizarea intrarii renilor in fabrica am folosit un semafor cu 10 permise pentru fiecare fabrica, adica maxim 10 reni pot accesa in acelasi timp fabrica.

//semafor pentru numarul maxim permis de reni in fabrica

private final Semaphore reindeerSemaphore = new Semaphore(10);

Pentru transferul cadourilor de la reni catre Mos Craciun a fost folosita o coada concomitenta care sincronizeaza metodele de adaugare a unui cadou in coada si scoaterea unui cadou din coada.

//metoda utilizata de Mos Craciun pentru a primi un cadou de la reni

```
public synchronized int receiveGift() {  
  
    /* asteptam pana cand buffer ul nu mai este gol */  
    if (tail == head) {  
        do {  
            try {  
                wait();  
            } catch (InterruptedException e) {  
                extracted(e);  
            }  
        } while (tail == head);  
    }  
    /* primire cadou */  
    int gift = giftsCount(head % giftsCount.length);  
    head += 1;  
  
    /* notificati ca buffer ul nu este plin */  
    notifyAll();  
  
    return gift;  
}
```

//metoda utilizata de un ren pentru a transmite un cadou lui Mos Craciun

```
public synchronized void giveGift(int gift) {  
    if (tail - head == giftsCount.length) {  
        do {  
            try {  
                wait();  
            } catch (InterruptedException e) {  
                extracted(e);  
            }  
        } while (tail - head == giftsCount.length);  
    }  
    giftsCount(tail % giftsCount.length) = gift;  
  
    /* adaugare cadou */  
    tail += 1;  
    notifyAll();  
}
```

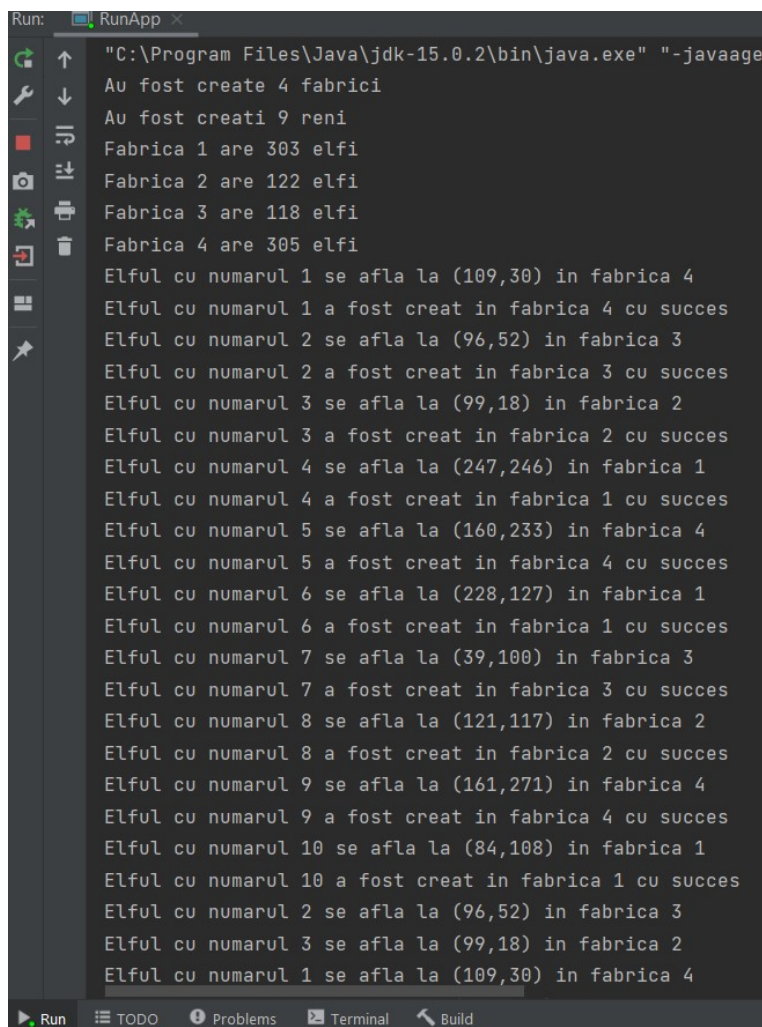
Pentru functionarea corecta a fabricilor am utilizat 3 incuietori.

```
//blocarea accesarii fabricii  
private final ReentrantLock industryLock = new ReentrantLock();  
  
//un lacat pentru accesarea listei de elfi  
private final ReentrantLock elfListLock = new ReentrantLock();  
  
//un lacat pentru accesarea listei de cadouri  
private final ReentrantLock giftLock = new ReentrantLock();
```

Stim ca doi elfi nu se pot deplasa in acelasi timp in fabrica sau doi elfi nu se pot misca in timp ce isi raporteaza pozitia.

Doi elfi nu pot fi adaugati in acelasi timp in lista de elfi, lista nu poate fi modificata cand doi elfi isi raporteaza pozitia.

Un ren nu poate primi un cadou in timp ce fabrica le cere elfilor sa-si raporteze pozitia. Lista de cadouri a unei fabrici poate fi accesata de un singur ren la un moment dat.

The image shows a screenshot of a Java application's output in an IDE terminal. The terminal window is titled "Run: RunApp x". The command executed is "C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-javaage". The output consists of several lines of text: "Au fost create 4 fabrici", "Au fost creati 9 reni", followed by the number of elves for each factory (Fabrica 1: 303, Fabrica 2: 122, Fabrica 3: 118, Fabrica 4: 305). Then, for each of the 9 cats, it shows the cat number, the factory it belongs to, and its coordinates (x, y). For example, "Elful cu numarul 1 se afla la (109,30) in fabrica 4". The cats are listed in order of their creation. The IDE interface includes a sidebar with icons for Run, Debug, Run and Debug, Test, Run and Test, and a bottom bar with Run, TODO, Problems, Terminal, and Build buttons.

```
Run: RunApp x
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-javaage
Au fost create 4 fabrici
Au fost creati 9 reni
Fabrica 1 are 303 elfi
Fabrica 2 are 122 elfi
Fabrica 3 are 118 elfi
Fabrica 4 are 305 elfi
Elful cu numarul 1 se afla la (109,30) in fabrica 4
Elful cu numarul 1 a fost creat in fabrica 4 cu succes
Elful cu numarul 2 se afla la (96,52) in fabrica 3
Elful cu numarul 2 a fost creat in fabrica 3 cu succes
Elful cu numarul 3 se afla la (99,18) in fabrica 2
Elful cu numarul 3 a fost creat in fabrica 2 cu succes
Elful cu numarul 4 se afla la (247,246) in fabrica 1
Elful cu numarul 4 a fost creat in fabrica 1 cu succes
Elful cu numarul 5 se afla la (160,233) in fabrica 4
Elful cu numarul 5 a fost creat in fabrica 4 cu succes
Elful cu numarul 6 se afla la (228,127) in fabrica 1
Elful cu numarul 6 a fost creat in fabrica 1 cu succes
Elful cu numarul 7 se afla la (39,100) in fabrica 3
Elful cu numarul 7 a fost creat in fabrica 3 cu succes
Elful cu numarul 8 se afla la (121,117) in fabrica 2
Elful cu numarul 8 a fost creat in fabrica 2 cu succes
Elful cu numarul 9 se afla la (161,271) in fabrica 4
Elful cu numarul 9 a fost creat in fabrica 4 cu succes
Elful cu numarul 10 se afla la (84,108) in fabrica 1
Elful cu numarul 10 a fost creat in fabrica 1 cu succes
Elful cu numarul 2 se afla la (96,52) in fabrica 3
Elful cu numarul 3 se afla la (99,18) in fabrica 2
Elful cu numarul 1 se afla la (109,30) in fabrica 4
```

Figura 3: Output

In aceasta sectiune voi descrie printr-un mic rezumat rezultatele obtinute in urma rularii programului. Rezultatele sunt obtinute aleatoriu in functie de generarea datelor de intrare.

Datele de iesire vor reprezenta numarul de fabrici, numarul de reni, numarul de elfi, catsi pozitia elfilor in fabrica. Rezultatele obtinute in urma rularii programului sunt cele de mai sus.

7 CONCLUZII SI BIBLIOGRAFIE

- Consider ca aceasta tema a reprezentat o provocare, atat intelegerea cerintei, abordarea task-urilor, cat si implementarea.
- In urma acestei teme, mi-am imbunatatit abilitatile de coding in limbajul Java, cat si notiunile de sintaxa in LaTeX.
- O alta concluzie este reprezentata de termenul limita al temei de casa. Acest parametru a avut un impact mai bun asupra organizarii timpului, cat si asupra unei coordonari mai responsabile a celorlalte aspecte realizate pentru a rezolva aceasta tema, ceea ce constituie un beneficiu mai amplu pentru dezvoltarea mea in acest domeniu.

Mai jos se pot observa cateva referinte bibliografice, capitole din cursuri, carti, site-uri web, referinte ce au ajutat in parcurgerea, documentarea si intelegerea mai ampla a temei de casa.

De pe site-urile respective m-am documentat, link-urile sunt atasate mai jos:

Referinte bibliografice

- (1) JAVA CONCURRENCY (MULTI-THREADING) - TUTORIAL,
[HTTPS://WWW.VOGELLA.COM/TUTORIALS/JAVACONCURRENCY/ARTICLE.HTML](https://www.vogella.com/tutorials/JAVACONCURRENCY/ARTICLE.HTML)
- (2) REENTRANT LOCK IN JAVA,
[HTTPS://WWW.GEEKSFORGEEEKS.ORG/REENTRANT-LOCK-JAVA/](https://www.geeksforgeeks.org/reentrant-lock-java/)
- (3) SEMAPHORE IN JAVA,
[HTTPS://WWW.GEEKSFORGEEEKS.ORG/SEMAPHORE-IN-JAVA/](https://www.geeksforgeeks.org/semaphore-in-java/)
[HTTPS://DOCS.ORACLE.COM/JAVASE/7/DOCS/API/JAVA/UTIL/CONCURRENT/SEMAPHORE.HTML](https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Semaphore.html)
- (4) JAVA 8 CONCURRENCY TUTORIAL: SYNCHRONIZATION AND LOCKS,
[HTTPS://WINTERBE.COM/POSTS/2015/04/30/JAVA8-CONCURRENCY-TUTORIAL-SYNCHRONIZED-LOCKS-EXAMPLES/](https://winterbe.com/posts/2015/04/30/java8-concurrency-tutorial-synchronized-locks-examples/)
- (5) JAVA.UTIL.CONCURRENT.CYCLICBARRIER IN JAVA,
[HTTPS://WWW.GEEKSFORGEEEKS.ORG/JAVA-UTIL-CONCURRENT-CYCLICBARRIER-JAVA/](https://www.geeksforgeeks.org/java-util-concurrent-cyclicbarrier-java/)
[HTTPS://DOCS.ORACLE.COM/JAVASE/7/DOCS/API/JAVA/UTIL/CONCURRENT/CYCLICBARRIER.HTML](https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/CyclicBarrier.html)
[HTTPS://WWW.JAVATPOINT.COM/LOCK-IN-JAVA](https://www.javatpoint.com/lock-in-java)

(6) INTRODUCING OVERLEAF AND L^AT_EX,

[HTTPS://WWW.OVERLEAF.COM/LEARN/LATEX/TUTORIALS](https://www.overleaf.com/learn/latex/tutorials)

[HTTPS://OEIS.ORG/WIKI/LIST_OF_LATEX_MATHEMATICAL_SYMBOLS](https://oeis.org/wiki/List_of_LaTeX_mathematical_symbols)

[HTTPS:](https://mirrors.nxthost.com/ctan/macros/latex/required/graphics/grfguide.pdf)

[//MIRRORS.NXTHOST.COM/CTAN/MACROS/LATEX/REQUIRED/GRAPHICS/GRFGUIDE.PDF](https://mirrors.nxthost.com/ctan/macros/latex/required/graphics/grfguide.pdf)

[HTTPS://SHARELATEX.PSI.CH/LEARN/XELATEX](https://sharelatex.psi.ch/learn/xelatex)

(7) CAPITOLUL 12 DIN CURS, CAT SI RESTUL CURSURILOR DE PE CLASSROOM.

(8) LABORATOR 12, CAT SI RESTUL LABORATOARELOR DE PE CLASSROOM.

© Ianuarie 2022.