Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

Departamentul *Informatica și Ingineria Sistemelor*

# RAPORT

la lucrarea de laborator nr. 2

la Programarea în Rețea

**Tema:** PROGRAMARE MULTI-THREADING

A efectuat: st. gr. TI-183 Bortă Sergiu

A verificat: Cebotari Daria

Chişinău – 2020

**Obiectivele lucrării:** înțelegerea modelelor de execuție concurentă și cunoașterea tehnicilor esențiale de sincronizare ale activităților bazate pe operațiile atomare ale semaforului; obiectivul specific constând în crearea unei aplicații Java ce ar utiliza sigur diverse structuri într-un context de execuție concurentă

Realizarea firelor de execuţie în Java/C#. Proprietăţile firelor. Stările unui fir de execuţie. Lansarea, suspendarea şi oprirea unui fir de execuţie. Grupuri de Thread-uri. Elemente pentru realizarea comunicării şi sincronizării.

1. Ce este un proces?

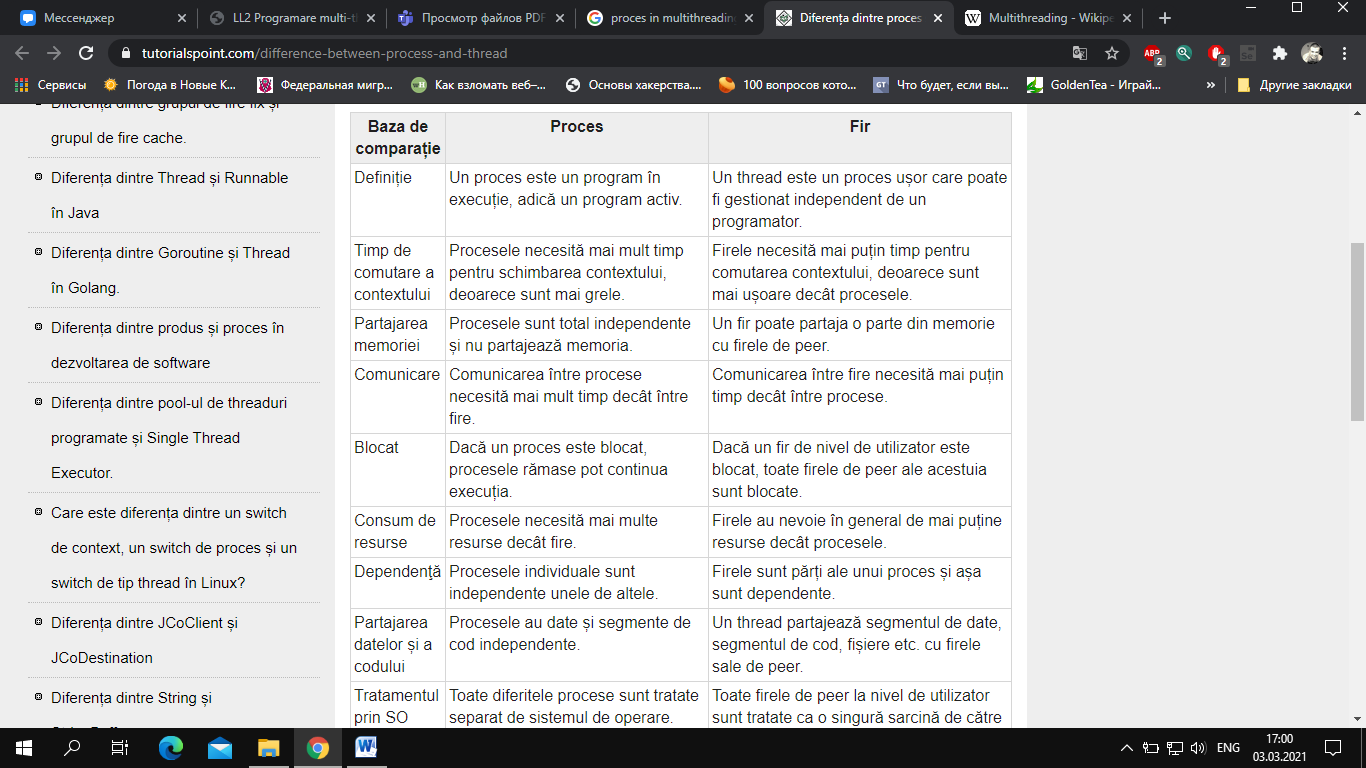
Un proces este un program activ, adică un program aflat în execuție. Este mai mult decât codul programului, deoarece include contorul de programe, stiva de proces, registre, codul programului etc.

1. Ce este un fir de executie?

Firul de execuție sau [thread-ul](https://ro.wikipedia.org/wiki/Thread) este un mic proces sau task, având propriile instrucțiuni și date.

1. Diferenta dintre proces si fir de executie?

Un thread este un proces ușor care poate fi gestionat independent de un programator. Îmbunătățește performanța aplicației folosind paralelismul. Un fir partajează informații precum segmentul de date, segmentul de cod, fișiere etc. cu firele de peer, în timp ce conține propriile registre, stivă, contor etc.



1. Definitia la Programarea concurenta?

**Programarea concurenta**este activitatea de construire a unui program continand procese multiple care se executa in paralel. Aceste procese sunt in competitie pentru accesarea resurselor critice si coopereaza pentru realizarea anumitor task-uri.

**Program care in timpul executarii sale creeaza mai multe procese care se executa intr-un paralelism abstract,adica nu neaparat pe procesoare distincte.**

1. Cind apare ,,deadlock”?

 poate apărea într-o situație în care un fir de așteptare aștepta un blocaj de obiect, care este dobândit de un alt fir de așteptare, iar al doilea fir de așteptare aștepta un blocaj de obiect care este dobândit de primul fir de executare. Deoarece ambele fire se așteaptă unul pe celălalt pentru a elibera blocarea, starea se numește blocare.

1. Ce este Semafor?

Un semafor controlează accesul la o resursă partajată prin utilizarea unui contor. Dacă contorul este mai mare decât zero, atunci accesul este permis. Dacă este zero, atunci accesul este refuzat. Contorul contorizează permise care permit accesul la resursa partajată. Astfel, pentru a accesa resursa, unui fir trebuie să i se acorde o autorizație din semafor.

Java oferă clasa **Semaphore** în pachetul java.util.concurrent care implementează acest mecanism, deci nu trebuie să implementați propriile semafore.

1. Ce este Mutex?

este cel mai simplu tip de sincronizator - aceasta **asigură că doar un singur fir poate executa secțiunea critică a unui program de calculator la un moment dat** .

1. Synchronization?

e capacitatea de a controla accesul mai multor fire la orice resursă partajată

Sincronizarea este folosită în principal

1. Pentru a preveni interferența firului.
2. Pentru a preveni problema consistenței.

### Tipuri de sincronizare

Există două tipuri de sincronizare

1. Sincronizare proces
2. Sincronizarea firului