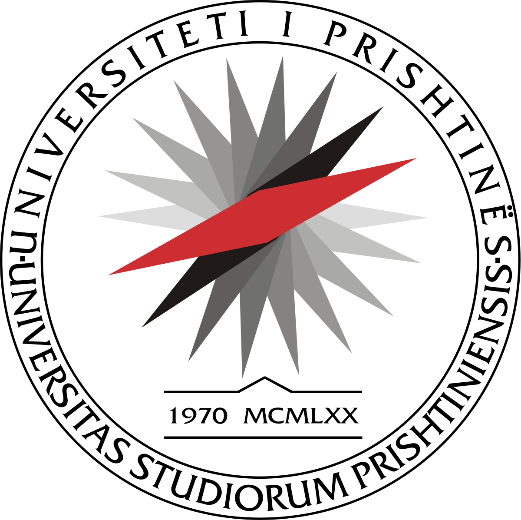
**Universiteti i Prishtinës**

Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike

Kodra e Diellit, p.n.

10000 - Prishtinë, Kosova



**SISTEMET OPERATIVE**

Threads & Concurrency

**Studentët:                                                                                       Profesor:**

Elirë Kamberi                                                                                   Artan Mazrekaj

Erëblina Zogjani

Gresa Salihu

Leonita Nika

*Janar 2021*

Përmbajtja

[1. Hyrje 3](#_Toc61782368)

[1.1 Çfarë është konkurenca(concurrency)? 3](#_Toc61782369)

[1.2 Procesi dhe thread-at 3](#_Toc61782370)

[1.3 Socket-at dhe protokollet 3](#_Toc61782371)

[2. Teknologjitë e përdorura 4](#_Toc61782372)

[3. Përshkrimi i problemit 4](#_Toc61782373)

[3.1 Problemi që duhet zgjidhur 5](#_Toc61782374)

[3.2 Zgjidhja e problemit 5](#_Toc61782375)

[4. Implementimi i kodit 9](#_Toc61782376)

[4.1 Klasa Client.java 9](#_Toc61782377)

[4.2 Klasa Server.java 11](#_Toc61782378)

[4.3 Klasa TCP.java 14](#_Toc61782379)

[5. Kontributet e studentëve 16](#_Toc61782380)

[6. Referencat 16](#_Toc61782381)

# **Hyrje**

Komunikimi Klient / Server përfshin dy komponente, klientin dhe serverin. Klientët dërgojnë kërkesa te serveri ndërsa serveri i përgjigjet kërkesave të klientit.

Në momentin kur kemi vetëm një klient i cili dëshiron të komunikojë është e thjeshtë, sepse në këtë rast lidhja e serverit nuk do të jetë e mbingarkuar për arsye se serveri për shkak të përdorimit të single-thread (procesimi i vetëm një të dhëne njëkohësisht) do të ketë në fokus vetëm atë klient.

Mirëpo në rastet kur më shumë se një klient dëshiron të komunikojë me serverin, atëherë serveri e ka për detyrë jo vetëm t’i pergjigjet kërkesave të një klienti, por duhet që njëkohësisht të jetë i gatshëm tu përgjigjet të gjithë klientëve që dërgojnë kerkesa. E kjo mundësohet me anë të proceseve multi-threaded pasi që duke i përdorur ato, një server **multi-threaded** pranon një lidhje nga një klient, fillon një thread për atë komunikim dhe vazhdon të dëgjojë kërkesa nga klientë të tjerë.

## **Çfarë është konkurenca(concurrency)?**

Konkurrenca është aftësia për të ekzekutuar disa programe ose disa pjesë të një programi paralelisht. Një detyrë që kërkon kohë duhet të kryhet në mënyrë asinkrone ose paralelisht, sepse kjo përmirëson shpejtësinë dhe ndërveprimin e programit.

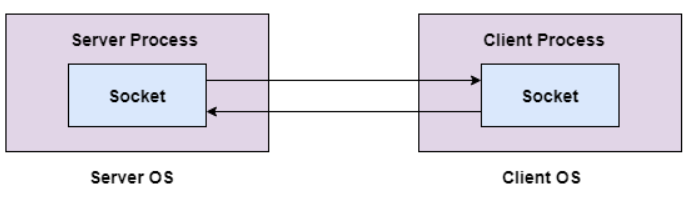
## **Procesi dhe thread-at**

Një proces zhvillohet në mënyrë të pavarur dhe i izoluar nga proceset e tjera. Ai nuk mund të përdorë drejtpërdrejt të dhënat e ndara në procese të tjera. Burimet e procesit si memoria dhe koha e CPU-së, i caktohen përmes sistemit operativ.

Ndërsa një thread është një proces i ashtuquajtur i lehtë. Ka grumbullin e vet të thirrjeve, por mund të ketë të dhëna të përbashkëta të thread-eve të tjera në të njëjtin proces. Çdo thread ka memorjen e vet cache. Nëse një thread lexon të dhëna të ndara, ajo i ruan këto të dhëna në memorjen e vet të memories. Përdorimi i thread-ave mundëson procesim paralel ose sjellje asinkrone.

## **Socket-at dhe protokollet**

Njëra nga tri metodat kryesore për komunikimin klient/server janë Socket-at.



**Socket-i** është një pikë fundore (endpoint) dhe paraqet një mënyrë virtuale të komunikimit të dy kompjuterëve në rrjet. Ata lehtësojnë komunikimin midis dy proceseve në të njëjtën makinë ose makina të ndryshme. Një socket indentifikohet në mënyrë unike nga dy komponente: IP Adresa dhe numri i portit. Shumë protokolle të aplikimit përdorin socketat për lidhjen dhe transferimin e të dhënave midis një klienti dhe një serveri.

Transferimi i sigurtë i këtyre kërkesave mundësohet në bazë të protokollit **TCP - Transimission Control Protocol**. Pasi në disa raste për shkak të mbingarkimit të rrjetës gjatë dërgimit të të dhënave mund të shfaqen gabime ose të humbin të dhënat, në këto raste me anë të TCP-së mundësohet ritransmetimi i paketave derisa të dhënat të merren plotësisht dhe saktë pa i humbur renditja.

Këto veçori e bëjnë këtë protokoll të jetë shumë më i përshtatshëm se protokolli **UDP - User Datagram Protocol** i cili nuk garanton dërgimin e të dhënave deri në destinacion pasi nuk vendos lidhje server-klient dhe nëse ndonjë paketë nuk ka arritur tek marrësi, ai nuk bën ritransmetimin e paketave të humbura. Kjo humbje në shumicën e rasteve bëhet për shkak të shpejtësisë së madhe që ofron ky protokoll.

# **Teknologjitë e përdorura**

Editori: Eclipse

Gjuha programuese: Java

Sistemi Operativ: Windows

Vegla për përshkrimin e detyrës: MS Word

# **Përshkrimi i problemit**

Problemi që do të shqyrtohet në vijim ka të bëjë kryesisht me implementimin e serverëve multi-threaded në Java.

## **Problemi që duhet zgjidhur**

Serverët single-threaded janë lloji më i thjeshtë i serverëve të cilët mund të përpunojnë (procesojnë) vetëm një kërkesë në të njëjtën kohë. Pra një echo server kthen të gjitha kërkesat që merr (pranon) nga një klient.

Për shembull, nëse një klient i dërgon serverit stringun **“Hello there!”**, atëherë serveri do të përgjigjet me **“Hello there!”**.

Ky server është single-threaded, që do të thotë se serveri nuk mund t'u përgjigjet echo klientëve njëkohësisht derisa të dalë (exit) klienti aktual. Për shkak të kësaj gjëje, serverët single-threaded mund ta degradojnë seriozisht performancën e sistemit.

Zgjidhja më e mirë është që të krijohet një server multi-threaded. Një server që ka më shumë se një thread njihet si server multi-threaded. Kur një klient dërgon kërkesën, gjenerohet një thread përmes së cilës një përdorues mund të komunikojë me serverin. Ne kemi nevojë për të gjeneruar shumë threads për të pranuar kërkesa të shumta nga shumë klientë në të njëjtën kohë, që nënkuptojmë se me anë të kësaj echo serveri do t’i shërbejë çdo klienti në një kërkesë të veçantë.

## **3.2 Zgjidhja e problemit**

Në editorin Eclipse krijojmë tri klasa të cilat janë:

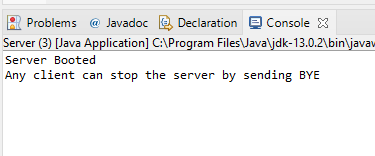
• Client

• Server

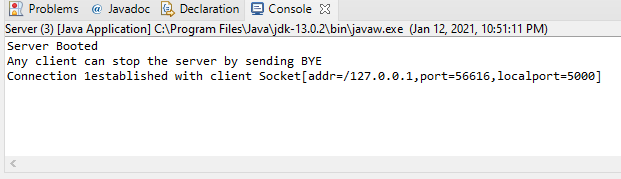
• TCP

Për transportimin e të dhënave ndërmjet klientit dhe serverit, përdorim protokollin TCP. Në vazhdim demonstrohet ekzekutimi i programit.

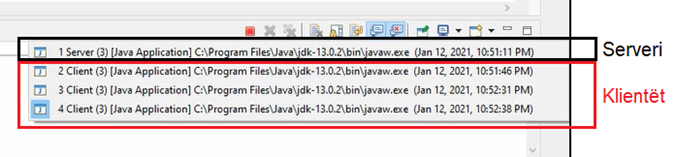
Së pari ekzekutohet pjesa e serverit:



Pastaj ekzekutohet pjesa e klientit:



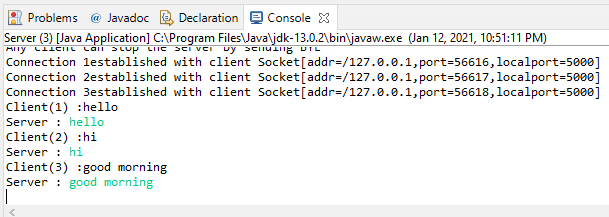
Ne mund të ekzekutojmë më shumë se një klient:



Pasi të bëhen këto ekzekutime, atëherë është e mundur që të dërgohen kërkesa ndërmjet klientëve dhe serverit. Në rastin tonë kemi ekzekutuar tre klientë dhe kemi shënuar mesazhet në vijim prej secilit klient tek serveri:



Pasi që këto mesazhe dërgohen tek serveri, atëherë mesazhet shihen në këtë lloj forme në *console* tek pjesa e serverit:



Klienti dhe Serveri nëse duan që ta mbyllin lidhjen, shtypin tekstin “***bye***” dhe paraqiten informatat si më poshtë:



Për detaje më të hollësishme vijojmë me implementimin e kodit.

# **Implementimi i kodit**

Në vazhdim të këtij dokumenti, është bërë analiza e kodit të implementuar ku përfshihen tri klasa të cilat janë:

* Klasa Client,
* Klasa Server dhe
* Klasa TCP.

## **4.1 Klasa Client.java**

**package** projekti;

//klasat qe do te importohen

**import** java.io.BufferedReader; //është klasë që lexon tekstin nga një Input Stream (njësoj si një file) duke i bërë buffer karakteret. Në përgjithësi, çdo kërkesë për lexim e bërë nga një Reader bën që një kërkesë përkatëse për lexim të bëhet nga karakteri themelor ose rrjedha e bajteve (byte stream).

**import** java.io.InputStreamReader; //Një InputStreamReader është një urë nga rrjedhat e bajtëve në rrjedhat e karakterit. Kjo i lexon bajtat dhe i dekodon ato në karaktere duke përdorur një charset të caktuar.

**import** java.io.PrintStream; //shton funksionalitetin në një rrjedhë tjetër të output-it, pra aftësinë për të shtypur reprezentime të vlerave të ndryshme të të dhënave në mënyrë të përshtatshme. Të gjithë karakteret e shtypura nga një PrintStream, shndërrohen në bajta duke përdorur enkodimin default të karakterit.

**import** java.net.Socket; //përfaqëson një socket dhe klasa java.net.ServerSocket ofron një mekanizëm për programin e serverit për të dëgjuar për klientët dhe për të vendosur lidhje me ta.

//Klasa Client

**public** **class** Client {

**public** **static** **void** main(String args[]) **throws** Exception

{

// Provojme te lidhemi me serverin – IP adresen dhe portin si parametra

Socket sk=**new** Socket("127.0.0.1",5000);

//Nese serveri nuk eshte duke degjuar – Do te marrim nje Exception

//Te klienti krijohen objektet BufferedReader dhe InputStreamReader. Krijojme InputStreamReader i cili kalon nepermjet getInputStream i cili eshte funksion i socket-it

BufferedReader sin=**new** BufferedReader(**new** InputStreamReader(sk.getInputStream()));

//Klasa PrintStream e kalon “string s” ne anen e serverit

PrintStream sout=**new** PrintStream(sk.getOutputStream());

//Merr input-in prej userit

BufferedReader stdin=**new** BufferedReader(**new** InputStreamReader(System.***in***));

String s; //Te dhenat tona do te ruhen ketu ne string s

//readLine() – Kjo metodë përdoret për të lexuar një rresht të vetëm të tekstit dhe nuk pranon asnjë parametër.

//equalsIgnore() – krahason dy stringje pavarësisht rastit se a është lower ose upper string. Kjo metodë kthen true nëse argumenti nuk është null dhe përfaqëson një string ekuivalent duke injoruar rastin (lower apo upper), përndryshe kthen false.

**while** ( **true** )

{

System.***out***.print("Client : "); //klienti shkruan nje mesazh

s=stdin.readLine(); //stdin e ben scan dhe e ruan ne String s;

sout.println(s); //e kthen stringun ne funskionin println duke perdorur objektin sout te klases PrintStream

//Nese klienti shtyp BYE, atehere loop do te behet break dhe mbyllet lidhja

**if** ( s.equalsIgnoreCase("BYE") ) //kushti

{

System.***out***.println("Connection ended by client");

**break**;

}

s=sin.readLine();

System.***out***.print("Server : "+s+"\n");

}

//Mbyllja e te gjitha lidhjeve qe jane krijuar

sk.close();

sin.close();

sout.close();

stdin.close();

}

}

## **4.2 Klasa Server.java**

**package** projekti;

//klasat qe do te importohen

**import** java.io.BufferedReader; //eshte klase qe perdoret per te lexuar tekstin nga nje rrjedhe hyrese e bazuar ne karakter.Mund te perdoret per te lexuar te dhena rresht pas rreshti me metoden readLine(). Kjo e ben performancen te shpejte

**import** java.io.IOException; //Kjo klase ka të bëjë me operacionet Input dhe Output në kodin Java. Perdoret ne rastet kur ndodh ndonjë dështim gjatë leximit, shkrimit dhe kërkimit të fajllave ose operacioneve të direktorisë.

**import** java.io.InputStreamReader; //është një urë nga rrjedhat e bajtëve në rrjedhat e karakterit: Lexon bajte dhe i deshifron ato në karaktere duke përdorur një karakter të caktuar. Karakteri që përdor mund të specifikohet me emër ose mund të jepet në mënyrë të qartë, ose karakteri i parazgjedhur i platformës mund të pranohet.

**import** java.io.PrintStream; //Një PrintStream shton funksionalitetin në një rrjedhë tjetër të daljes,domethënë aftësinë për të shtypur përfaqësime të vlerave të ndryshme të të dhënave në mënyrë të përshtatshme.

**import** java.net.ServerSocket;// Kjo klasë implementon bazat e serverit. Një server socket pret që kërkesat të vijnë në rrjet. Ai kryen disa operacione bazuar në atë kërkesë, dhe pastaj ndoshta i kthen një rezultat kërkuesit.

**import** java.net.Socket;// Kjo klase implenton socket te klienteve(ose qe ndryshe thirren vetem si sockets). Nje socket eshte nje pike perfundimtare per komunikimin ndermjet dy makinave.

**import** java.util.Scanner; //Një skanues i thjeshtë teksti i cili mund të analizojë llojet dhe vargjet primitive duke përdorur shprehje të rregullta.

**import** java.util.concurrent.ExecutorService; //Ndërfaqja ExecutorService trashegon Executor duke shtuar metoda që ndihmojnë në menaxhimin dhe kontrollimin e ekzekutimit të thread-ave. Gjendet në paketën java.util.concurrent. dhe përcakton metodat që ekzekutojnë nje grup thread-ave që kthejnë rezultatet dhe përcaktojnë statusin e mbylljes.

**import** java.util.concurrent.Executors;//Nderfaqja Executors siguron ekzekutimin e seciles detyrë, duke përfshirë detajet e përdorimit dhe krimit të thread-ave.

**public** **class** Server {

//Deklarimi i variables port

**int** port;

//Krijimi i server socket dhe client socket te cilet ne fillim i inicializojme me vleren null.

ServerSocket server=**null**;

Socket client=**null**;

ExecutorService pool = **null**; //inicilizimi i objektit te klases ExecutorService me vleren null.

//Kjo klase perdoret per krijimin dhe mirëmbajtjen e thredave.

**int** clientcount=0; //Numri i klienteve te lidhur si fillim merret 0.

**public** **static** **void** main(String[] args) **throws** IOException {

Server serverobj=**new** Server(5000);//Krijimi i nje objekti te klases Server me portin 5000

serverobj.startServer(); //Startimi i procesit te serverit

}

//Krijimi i nje konstruktori, të lidhur me portin specifik

Server(**int** port){

**this**.port=port;

pool = Executors.*newFixedThreadPool*(5); //Tregon se ne te njejten kohe vetem 5 klienta do te mund te lidhen.

}

//startServer mund te leshoj IOException(input output exception)

**public** **void** startServer() **throws** IOException {

server=**new** ServerSocket(5000);//krijimi i nje ServerSocket-i te ri me port 5000

System.***out***.println("Server Booted");

System.***out***.println("Any client can stop the server by sending BYE");

**while**(**true**)

{

//accept() e mban serverin ne pritje derisa klienti te konektohet

client=server.accept();

clientcount++;//Cdo here qe konektohet nje klient rritet vlera per 1.

//ekzekutimi i konstruktorit ServerThread i cili si parameter permban detajet e klientit ,numrin e klientave te konektuar dhe metoden ne te cilen po ekzekutohet.

ServerThread runnable= **new** ServerThread(client,clientcount,**this**);

pool.execute(runnable);

}

}

//Krijojme klases private statike ServerThread dhe implementojme nderfaqen Runnable

**private** **static** **class** ServerThread **implements** Runnable {

//Krijimi i server socket dhe client socket te cilet ne fillim i inicializojme me vleren null.

Server server=**null**;

Socket client=**null**;

BufferedReader cin;

PrintStream cout;

Scanner sc=**new** Scanner(System.***in***);//per input nga ana e klientit.

**int** id;

String s;

// ServerThread mund te leshoj IOException(input output exception)

ServerThread(Socket client, **int** count ,Server server ) **throws** IOException {

//vendosja e vlerave te parametrave per konstruktorin ServerThread.

**this**.client=client;

**this**.server=server;

**this**.id=count;

//u mundesua lidhja ne mes serverit dhe klientit.

System.***out***.println("Connection "+id+"established with client "+client);

//per te lexuar permbajtjen e socket-it client.

cin=**new** BufferedReader(**new** InputStreamReader(client.getInputStream()));

// per te mundesuar shkrimin e bajtave ne kete socket.

cout=**new** PrintStream(client.getOutputStream());

}

// ne momentin qe e implementojm nderfaqen Runnable atehere duhet ta mbishkruajm funksionin run()

@Override

**public** **void** run() {

//krijimi i variables x qe si fillim permban vleren 1

**int** x=1;

//perdorimi i bllokut try{} dhe catch{} per kapjen e errors

**try** {

**while**(**true**) {

//leximi rresht per rresht i permbajtjes se socket-it

s=cin.readLine();

//shfaqet ne console Client(id e klientit) dhe mesazhi qe i dergohet serverit

System. ***out***.print("Client("+id+") :"+s+"\n");

//Serveri i kthen klientit pergjigje

System.***out***.print("Server : ");

s=sc.nextLine();

// ne momentin kur klienti shenon mesazhin bye

**if** (s.equalsIgnoreCase("bye"))

{

cout.println("BYE"); //serveri e kthen te njetin

x=0; //ndryshohet vlera e variables x nga 1 ne 0

System.***out***.println("Connection ended by server"); //mbyllet lidhja ne mes klientit dhe serverit

**break**;

}

//Nese klienti nuk dergon mesazhin bye atehere do te shfaqen te dhenat e outputStream

cout.println(s);

}

//Mbyllja e te gjitha lidhjeve

cin.close();

client.close();

cout.close();

//ne rastin kur variabla x ka vleren 0.

**if**(x==0) {

System.***out***.println( "Server cleaning up." );

System.*exit*(0); //programi perfundoi me sukses

}

}

// ne rast te kapjes te ndonje errori

**catch**(IOException ex){

//shfaqim mesazhet qe ka ndodhur nje error dhe detajet rreth tij

System.***out***.println("Error : "+ex);

}

}

}

}

## **4.3 Klasa TCP.java**

**package** projekti;

//klasat qe do te importohen

**import** java.io.BufferedReader; //eshte klase qe perdoret per te lexuar tekstin nga nje rrjedhe hyrese e bazuar ne karakter. Mund te perdoret per te lexuar te dhena rresht pas rreshti me metoden readLine(). Kjo e ben performancen te shpejte.

**import** java.io.InputStreamReader;//është një urë nga rrjedhat e bajtëve në rrjedhat e karakterit: Lexon bajte dhe i deshifron ato në karaktere duke përdorur një karakter të caktuar. Karakteri që përdor mund të specifikohet me emër ose mund të jepet në mënyrë të qartë, ose karakteri i parazgjedhur i platformës mund të pranohet

**import** java.io.PrintStream;//Një PrintStream shton funksionalitetin në një rrjedhë tjetër të daljes, domethënë aftësinë për të shtypur përfaqësime të vlerave të ndryshme të të dhënave në mënyrë të përshtatshme

**import** java.net.ServerSocket;//Kjo klasë implementon bazat e serverit. Një server socket pret që kërkesat të vijnë në rrjet. Ai kryen disa operacione bazuar në atë kërkesë, dhe pastaj ndoshta i kthen një rezultat kërkuesit.

**import** java.net.Socket;// Kjo klase implenton socket te klienteve(ose qe ndryshe thirren vetem si sockets). Nje socket eshte nje pike perfundimtare per komunikimin ndermjet dy makinave

**import** java.util.Scanner;//Një skanues i thjeshtë teksti i cili mund të analizojë llojet dhe vargjet primitive duke përdorur shprehje të rregullta

**public** **class** TCP {

//Metoda main. Perdor perjashtime per te njoftuar per gjera qe nuk duhet te injorohen

**public** **static** **void** main(String[] args) **throws** Exception {

//Perdorim per te treguar serveri eshte i lidhur me portin 2000

ServerSocket ss = **new** ServerSocket(2000);

//Thirrja e ss.accept () bën që soketi i serverit të presë dhe të dëgjojë për lidhjet hyrëse në portin qe eshte i lidhur

Socket sk = ss.accept();

//Krijojme nje objekt te klases bufferedReader qe do te na mundesoje nje platforme per transferim te te dhenave. Ne kete objekt krijohet InputStreamReader i cili kalon nepermjet getInputStream qe eshte funksion i socket-it

BufferedReader cin = **new** BufferedReader(**new** InputStreamReader(sk.getInputStream()));

//Kemi krijuar PrintStream per getOutputStream

PrintStream cout = **new** PrintStream(sk.getOutputStream());

//Merr input nga perdoruesi

BufferedReader stdin = **new** BufferedReader(**new** InputStreamReader(System.***in***));

//Stringu s ku do te ruhen te dhenat

String s;

//Krijimi i nje objekti te klases Scanner

Scanner sc = **new** Scanner(System.***in***);

**while** (**true**)

{

//funksioni cin lexon inputin nga soketi permes inputStream

s = cin.readLine();

//shfaq outputin qe eshte i ruajtur ne s

System.***out***.print("Client : " +s+ "\n");

System.***out***.print("Server : ");

// s = stdin.readLine();

//pastaj perdor sc qe eshte objekt nga klasa Scanner dhe funksionin nextLine() qe do te e skanoj nga tastatura

s = sc.nextLine();

//Nese te dhenat ne s eshte bye pra nese klienti shty bye,atehere nderprehet loop dhe mbyllet lidhja

**if** (s.equalsIgnoreCase("bye"))

{

cout.println("BYE");

System.***out***.println("Connection ended by server");

**break**;

}

//Nese te dhenat ne s nuk eshte bye atehere objekti cout i PrintStream do te shfaq te dhenat e outputStream ne anen e serverit dhe pastaj do i vendos ato ne anen e klientit

cout.println(s);

}

//Mbyllja e lidhjeve

ss.close();

sk.close();

sc.close();

cin.close();

cout.close();

stdin.close();

}

}

# **Kontributet e studentëve**

|  |  |
| --- | --- |
| **Studentja** | **Kontributi** |
| Elirë Kamberi | Pikat 3, 2, 4.3, 6 |
| Erëblina Zogjani | Pikat 3, 2, 4.1, 6 |
| Gresa Salihu | Pikat 1, 2, 4.2, 6 |
| Leonita Nika | Pikat 1, 2, 4.2, 6 |

# **Referencat**

* Operating System Concepts (Tenth Edition) – Abraham Silberschats, Peter Baer Galvin, Greg Gagne
* <http://tutorials.jenkov.com/>
* <https://www.geeksforgeeks.org/multithreaded-servers-in-java/>
* <https://root.cern/TaligentDocs/TaligentOnline/DocumentRoot/1.0/Docs/books/DC/DC_99.html>
* <https://stackoverflow.com/>