

**Proiect Sisteme Distribuite**

**Coordonator:**

**Cristinel Costea Eduard**

**Student,**

**Mate Lenard Istvan**

**Grupa Calculatoare Anul IV**

Cuprins

[**Introducere:** 3](#_Toc57648610)

[Titlu: 3](#_Toc57648611)

[Context: 3](#_Toc57648612)

[Cerințe: 3](#_Toc57648613)

[- ANEXE: 4](#_Toc57648614)

[REFERINȚE: 4](#_Toc57648615)

[Aplicația: 4](#_Toc57648616)

[Procesarea comenzilor: 8](#_Toc57648617)

[Rolul Masterului: 8](#_Toc57648618)

[Rulare: 10](#_Toc57648619)

[Noduri Server: 11](#_Toc57648620)

[Concluzie: 11](#_Toc57648621)

[Sitografie: 11](#_Toc57648622)

# **Introducee:**

## Titlu:

Titlul aplicației “­ Sistem de fișiere distribuit” descrie în mod sugestiv aplicația ce a fost proiectat în limbajul de programare Java cu ajutorul editorului de text Netbeans.

## Context:

In rețeaua locală câteva noduri client partajează un sistem de fișiere

prin intermediul unui nod master și câteva noduri pentru stocare.

Nodul master doar inregistrează numele și locația fișierelor.

Un nod client poate afișa lista fișierelor, poate deschide

un fișier in mod Read-Only sau Read-Write.

Utilizatorul (prin interfața CLI a clientului) va obține lista de fișiere

printr-o comandă similară comenzilor dir (windows) sau ls(linux),

conținutul unui fișier similar comenzilor type (windows) sau cat(linux),

ștergerea unui fișier cu del sau rm.

Se poate considera un director unic. Fișierele vor fi de tip text,

pentru scriere se va considera o comandă de genul: append <string>.

Un fișier poate fi stocat integral pe un singur nod sau distribuit (chunks).

## Cerințe:

- aplicație java (command line interface)

- documentație scrisă + cod sursă (github)

- scurtă descriere a conceptului Distributed File Systems, particularități.

- obiective propuse

- descrierea implementării

- concluzii (obiective atinse)

- bibliografie

### - ANEXE:

- min 2 imagini (screen captures)

- fișiere log: nod client, nod master, nod stocare

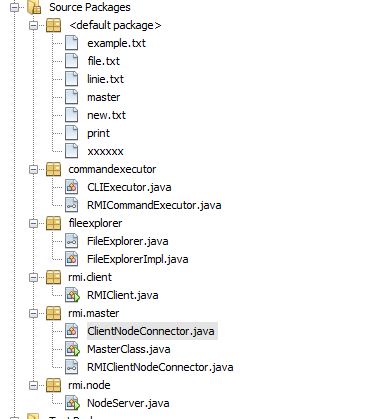
- fragment cod sursă comentat

## REFERINȚE:

<https://github.com/Lenard-Mate/CommandLineSD>

# Aplicația:

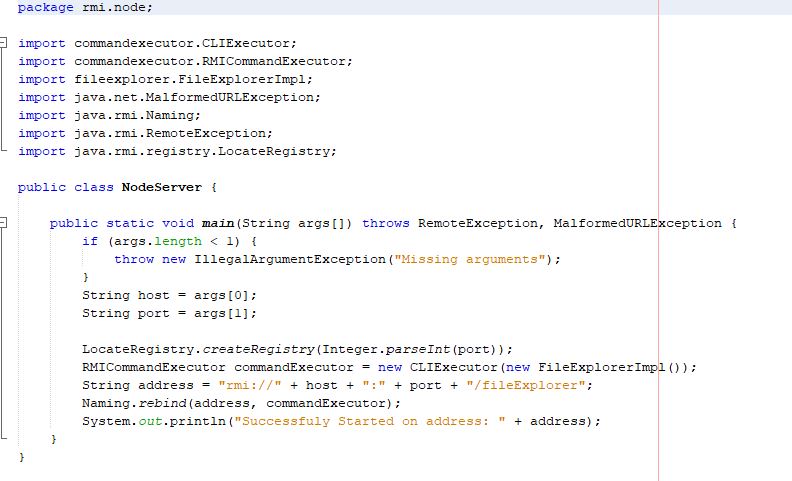
În următoarea imagine putem observa distribuția pachetelor unde se regăsește nodul pentru server,nodul Master și nodul Client.



Nodul server este doar o componentă care primește date și returnează rezultate, cu o descriere mai amplă NodeServer prin RMI primeste o comanda de la master care este transim mai departe la interfața RMICommandExecutor care apelează lcasa CLIExecutor iar aceasta la rândul lui apelează interfața FileExplorer care este conectat la clasa FileExplorerImpl. Modivul din care aplicația este atât de înpățit este că fiecare clasă are responsabilitațile sale și este mult mai ușor de urmărit codul.

CLIExecutor selectează comenzile și le verifică înainte de a apela vreo metodă din FileExplorer în cazul în care comanda a fost corectă atunci este apelată metoda potrivită pentru comandă iar metodele se regăsesc în clasa FileExplorerImpl. La execuția comenzii va fi returnat un răspuns de către NodeServer la nodul Master unde se va returna la client.

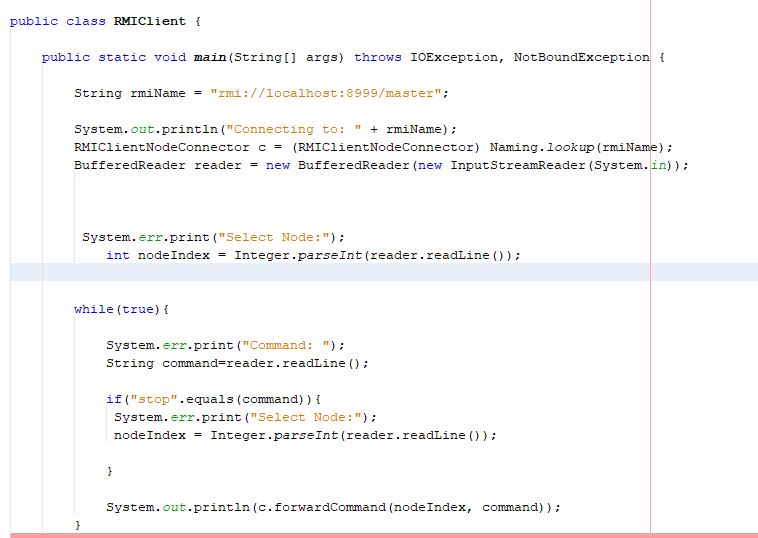
Următoarea imagine ne prezintă nodul Server



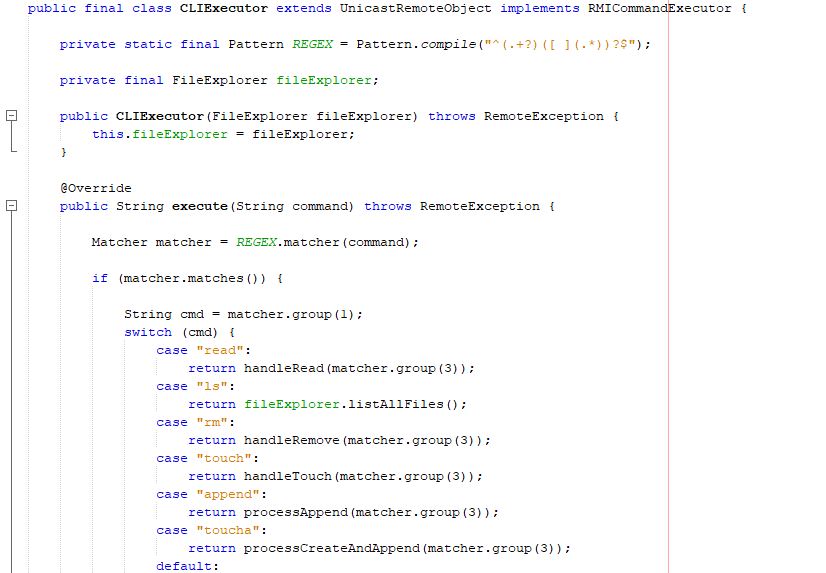
După care este prezentat nodul Master



Și nu în ultimul rând Clientul



# Procesarea comenzilor:



Pentru prelucrarea și verificarea comenzilor am folosit un REGEX care separă comanda de numele fișierului, sau numele fișierului de conținutul acesteia în cazul în care adăugam text în fișier.

# Rolul Masterului:

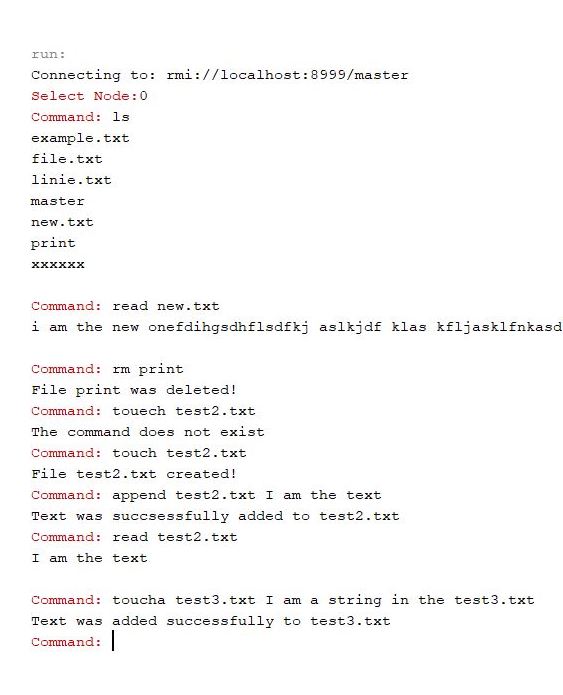
Masterul poate fi văzut ca un proxy care transimte datele primate mai departe dar are un rol specific și anume selectarea nodului care a fost cerut de client pe lângă asta să știe la care nod se va trimite comanda care va fi executată de nodulServer.

La master se pot conecta mai mulți client simultan la fel ca și noduir de Servere, comenzile lansate simultan nu reprezintă o problem din motiv că Java are o funcție numit IOException care reușește să evidențieze comenzile primite și le execută după în ordinea sosirii.



# Rulare:

În imaginea următoare putem observa cum rulează aplicație și cum sunt utilizate comenzile acestora.



Clientul după ce a selectat nodul intră într-o buclă unde poate efectua comenzi de screiere citire creare ștergere a fișierelor.

Este important de specificat pentru rulare că ordinea în care se pornesc nodurile este important, ordinea corectă este NodeServer, Master, Client și în această ordine se fac și conexiunile.

# Noduri Server:

Momentan există 2 noduri care se conectează la master prin argumentele setate în NetBeans dar bineînțeles se pot crea atâtea noduri câte vrem noi cu condiția ca dispozitivul fizic să suporte. Într-o etapă mai avansată se vor putea crea noduri dinamice folosind comenzile setate prin interfața clientului.

# Concluzie:

Această aplicație are toate caracteristicile enumerate de către cerință și corespunde normelor de așteptare în cea ce privește contextual definit la începutul documentației.

# Sitografie:

<http://cti.ubm.ro/sdi/proiect/2020/tema.html>

http://cti.ubm.ro/sdi/laboratoare/05/