Logo

Description automatically generated

**APLICAȚIE PENTRU GESTIUNEA FIȘIERELOR MULTIMEDIA**

**Profesor coordonator: Student:**

**Ciurea Cristian Cupii Cezin**

Cuprins

[Introducere 3](#_Toc87015230)

[1. Descrierea problemei 3](#_Toc87015231)

[1.1. Detalierea claselor si enumeratiei din cadrul aplicatiei 3](#_Toc87015232)

[2. Implementarea solutiei 4](#_Toc87015233)

[2.1. Meniul aplicatiei 4](#_Toc87015234)

[3. Concluzie 8](#_Toc87015235)

[Bibliography 9](#_Toc87015236)

# Introducere

Scopul proiectului este realizarea unei aplicatii pentru gestionarea fisierelor multimedia din cadrul unui sistem de operare, oferind utilizatorilor posibilitatea de a crea si de a sterge fisiere sau foldere.

Pentru dezvoltarea aplicatiei am folosit limbajul de programare Java, limbaj orientat-obiect, lansat in 1995 [1] unul din cele mai folosite limbaje din ziua de astazi datorita puterii pe care o are prin portabilitate (Cross-Platform), simplitate si prin faptul ca este printre putinele limbaje de programare 100% orientate obiect. [2]

Meniul din cadrul aplicatiei permite interactiunea utilizatorului cu solutia prezentata unde datele sunt validate cu ajutorul unui *regex* ce blocheaza datele introduse incorect atat de la tastatura cat si din fisierele de intrare preluate.

La nivelul aplicatiei a fost definita o clasa ce reprezinta o exceptie ridicate in unele situatii: CustomException.

# Descrierea problemei

Problema rezolvata de aplicatia noastra este cea a gestiunii fisierelor multimedia prin salvarea lor in foldere intr-un mod cat mai optim.

Datele din cadrul aplicatiei au fost manipulate cu ajutorul claselor din cadrul Java.

Clasele prezente in cadrul aplicatiei sunt urmatoarele: FileP, Folder, FolderCollection si enum-ul FileType, toate fiind incapsulate in pachetul „classes”, clasa Menu care contine toata functionalitea interactiunii dintre utilizator si aplicatie se afla in pachetul „menu” iar clasa Main se afla in pachetul main.

## Detalierea claselor si enumeratiei din cadrul aplicatiei

* **Clasa FileP**  - are in componenta sa atributele name (String), extension (FileType), location (String). In aceasta clasa se regasesc urmatoarele:
  + Constructor cu si fara parametri
  + Getteri si setteri pentru fiecare atribut in parte
  + Metode de tipul toString() pentru afisarea datelor in consola si de scriere in fisier text
* **Clasa Folder** – are in componenta atributele folderName (String) si o lista de fisiere (ArrayList<FileP>). In aceasta clasa se regasesc urmatoarele:
  + Constructorul cu si fara parametri
  + Getteri si setteri pentru fiecare atribut in parte
  + Metode de tipul toString() pentru afisarea datelor in consola si de tipul toStringInFile() pentru scriere in fisier text
* **Clasa FolderCollection** – are in componenta atributele name (String) si o lista de foldere (Set<Folder>). In aceasta clasa se regasesc urmatoarele:
  + Constructorul cu si fara parametri
  + Getteri si setteri pentru fiecare atribut in parte
  + Metode de tipul toString() pentru afisarea datelor in consola si de tipul toStringInFile pentru scriere in fisier text
  + Metoda readFromFile() pentru citirea din fisier text
  + Metoda writeInFile() pentru scrierea in fisier text
* **Enum-ul FileType –** are in componenta sa patru tipuri de fisiere, avand rolul de a defini o serie de constante pentru a putea simplifica modul de lucru din cadrul aplicatiei.
* **Clasa Menu** – contine metoda statica openMenu prin care se genereaza meniul cu optiuni cu ajutorul caruia utilizatorul interactioneaza cu aplicatia.In cadrul meniului sunt prezente toate optiunile pe care le poate avea utilizatorul, vizualizarea unui folder specific, adaugarea si stergerea de foldere, iar la iesirea din aplicatie, toate datele vor fi salvate intr-un fisier text „input.txt”

# Implementarea solutiei

Pentru ca aplicatia sa fie usor reutilizabila si de catre al programatori am impartit clasele in pachete astfel:

* **classes** – clasele utilizate in cadrul aplicatiei
* **exceptions** – unde sunt definite exceptiile
* **menu** – clasa ce contine tot ce tine de operatiile utilizatorului din meniu
* **main** – clasa ce contine programul de baza ce ruleaza aplicatia

Graphical user interface, text, application

Description automatically generated

*Figura 2.1. Pachetele aplicatiei*

## Meniul aplicatiei

Modul in care a fost realizat meniul aplicatiei a fost prin implementarea unui switch, o structura conditionala des folosita in astfel de cazuri. Inputul utilizatorului este preluat cu ajutorul unui Scanner, astfel acesta este prelucrat si efectuat.

**Meniu principal**

1. Vizualizare foldere
2. Iesire

8. Generare raport

In momentul in care utilizatorul alege prima optiune din meniu, este afisata structura folderelor si continutul acestora si este deschis urmatorul submeniu

1. Vizualizare folder
2. Creare folder
3. Stergere folder
4. Meniu principal

Daca se alege optiunea 0, utilizatorul este redirectionat catre meniul anterior.

Daca se alege optiunea 3, utilizatorul va trebui sa introduca numele folderului pe care doreste sa il vizualizeze, urmand ca numele sa fie validat, iar daca acesta exista este redirectionat catre submeniul urmator:

1. Adaugare fisier unde se adauga un fisier nou in director
2. Stergere fisier unde se sterge un anumit fisier din director

Text

Description automatically generated

*Figura 2.2. Stergerea unui fisier*

Daca se alege optiunea 4 sau 5, utilizatorul trece prin aproape acelease etape ca in cazul etapelor 6 si 7.

Alegerea optiunii 2 din meniul principal va inchide aplicatia si va salva datele cu toate schimbarile efectuate de catre utilizator cu ajutorul functiei *writeInFile* din cadrul clasei FolderCollecion

Text

Description automatically generated with medium confidence

*Figura 2.3. Inchiderea aplicatiei*

*Text

Description automatically generated*

*Figura 2.4. Functia writeInFile*

Alegerea optiunii 8 va duce la generarea unui raport text care contine detalii despre locatia noastra, cum ar fi numarul de foldere si continutul fiecaruia, fiind afisat in consola si scris in fisierul „raportGeneral.txt”

Text

Description automatically generated

**Lucrul cu fisiere**

Un aspect foarte important al aplicatiei este lucrul cu fisiere. La deschiderea aplicatiei, in cazul in care fisierul exista, se citesc datele din interiorul lui, iar in caz ca nu, se genereaza un fisier nou. La inchiderea aplicatiei, se scriu datele in cadrul aceluiasi fisier creat mai devreme, el fiind gestionat printr-un obiect de tipul File, creandu-se un obiect ce permite citirea fluxurilor de fisiere (FileReader). In continuare, datele sunt preluate linie cu linie cu ajutorul unei instante din cadrul clasei BufferedReader.

Colectia de foldere este citita linie cu linie, pe prima linie aflandu-se numele colectiei iar pe urmatoarele linii numarul de foldere si fiecare in parte.

Text

Description automatically generated

Text

Description automatically generated

*Figura 2.5. Functia readFromFile*

# Concluzie

In concluzie, proiectul are rolul de a gestiona fisierele multimedia prin stocarea acestora in foldere. Pentru a putea creste performanta aplicatiei, am utilizat limbajul de programare Java, un limbaj compilat al carui compilator genereaza cod de octeti, un plus fata de celelalte limbaje. [3]

Managementul datelor a fost asigurat prin utilizarea colectiilor, prin organizarea claselor in pachete, salvarea datelor in fisiere text si denumirea variabilelor dupa conventiile Clean Code.

Utilizatorul poate efectua, in cadrul aplicatiei operatii de adaugare, stergere si citire din/in fisiere text, toate acestea fiind realizate prin meniul cu optiuni.

In ultimul rand, aplicatia poate fi extinsa si actualizata cu usurinta prin adaugarea de functionalitati noi, cum ar fi asocierea altor proprietati pentru toate fisierele salvate.

# Bibliografie

|  |  |
| --- | --- |
| [1] | Conf. Univ. Cristian Ciurea, Programare si Proiectare Orientate Obiect. |
| [2] | E. Bruce, Thinking in Java (ed 4), Prentice Hall, 2006. |
| [3] | A. B. Downew, Think Java (How to Think Like a Computer Scientist), 2011. |