Image Brightness modification

Modificarea luminozitatii unei imagini

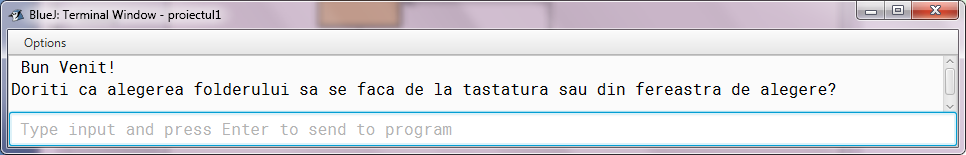
1. Introducere

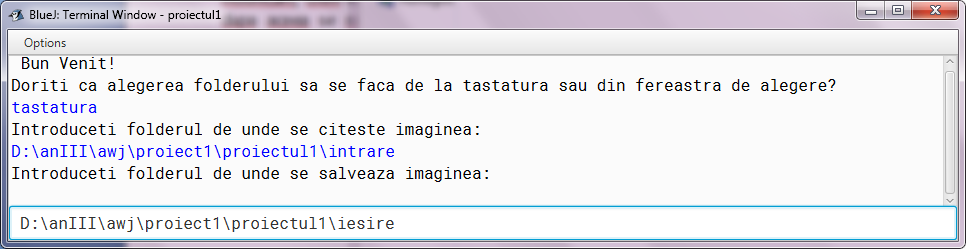
Aceasta este documentatia pentru realizarea proiectului de la materia Aplicatii Web cu Suport Java. Dupa cum se poate vedea si din titlu, proiectul are ca functionalitate modificarea luminozitatii unei imagini folosind limbajul de programare Java. Eu am folosit BlueJ pentru compilare si scrierea programului. Mai jos voi descrie cum am realizat acest proiect si cum functioneaza acesta.

1. Descrierea aplicatiei cerute

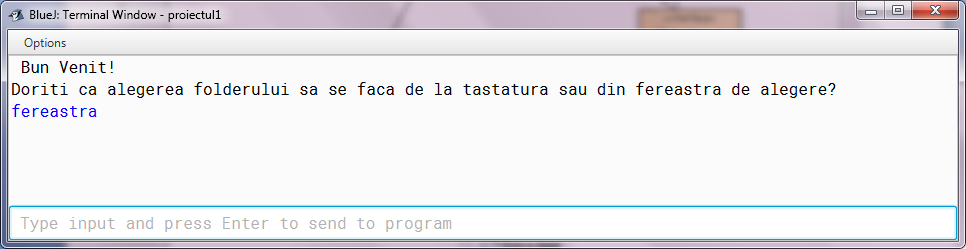
Pasii executiei sunt acestia: citeste folderul din care se ia imaginea ce va fi prelucrata, citeste folderul unde se va salva imaginea rezultata, alege din folderul anterior mentionat imaginea ce vrem sa o modificam, citim de la tastatura factorul de luminozitate cu care modifica imaginea si dupa aceea se realizeaza modificarea propriuzisa. La fiecare pas de executie se calculeaza timpul de executie si se afiseaza la sfarsitul executarii pasului.

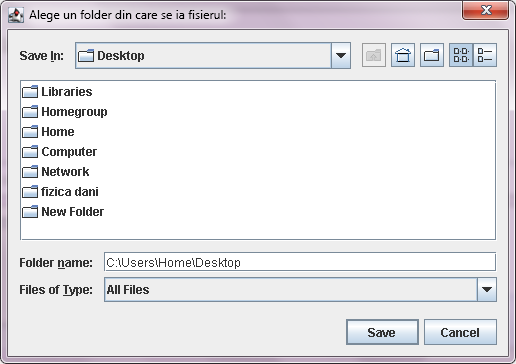
Utilizatorul la inceput are posibilitatea de a alege daca citirea folderelor de intrare si iesire se face dand calea directa de la tastatura sau folosind fereastra de alegere astfel:



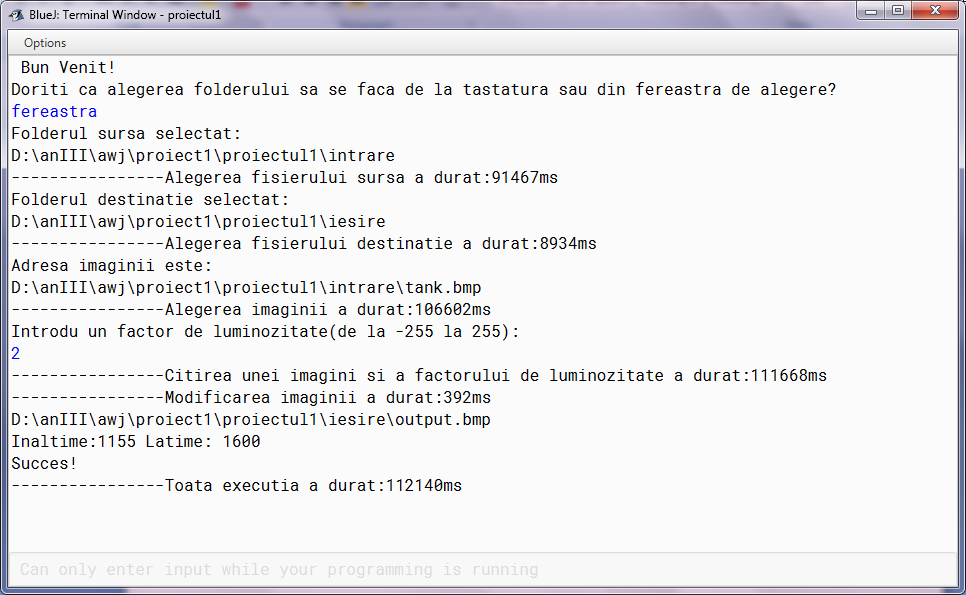


Aceasta a fost citirea de la tastatura sau folosim fereastra de alegere:





Asa se alege si imaginea din folderul de intrare, prin fereastra de alegere. Iar asa arata fereastra dupa ce am introdus factorul de luminozitate de la tastatura si dupa terminarea executiei:



1. Partea teoretica

Pentru realizarea acestei aplicatii s-au utilizat cunostinte de programare orientata pe obiecte, mai exact cunostinte de programare Java. Printre conceptele folosite se numara:

-Incapsulare = modalitate de ascundere a informatiilor in interiorul instantelor cu scopul de a oferi accesul la acestea in mod indirect, prin intermediul unei metode; Acest concept implica utilizarea de setteri pentru a modifica valoarea atributelor si utilizarea de getteri pentru a accesa valorile atributelor.

-Mostenire = procedeu specific programarii orientate pe obiect. O clasa ce mosteneste o alta clasa preia atributele și comportamentele neprivate de la aceasta;

-Polimorfism = posibilitatea unei instanțe de a lua forma oricăruia dintre prototipurile superioare celui cu care a fost creată;

-Abstractizare = **un prototip abstract** reprezinta un prototip folosit doar cu rol de generalizare. Acesta nu poate fi instantiat ci doar cumuleaza caracteristici comune altor prototipuri folosite;

-Suprascriere = redefinirea comportamentul unei metode (setul de instructiuni), pastrand insa identificatorul;

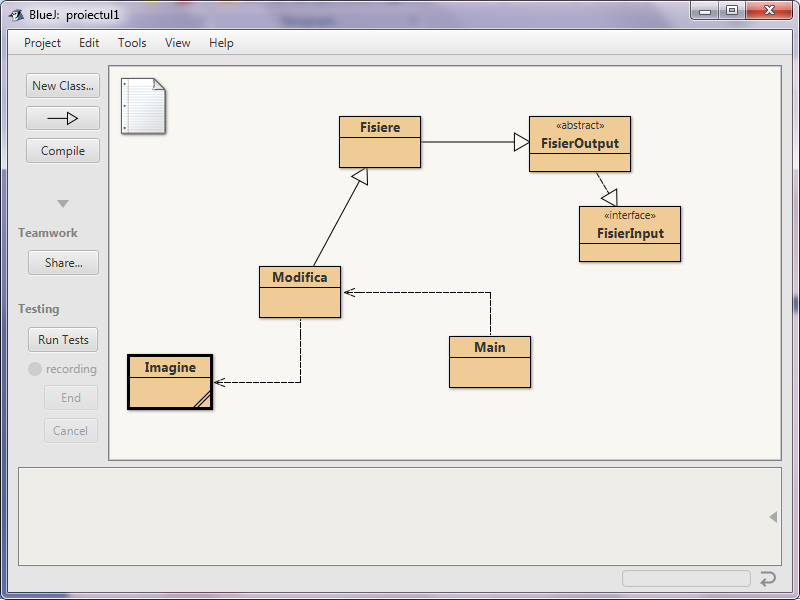
-Alte concepte de baza ale programarii orientate pe obiecte;

1. Descrierea implementarii

Aplicatia este alcatuita din cinci clase si o interfata. Am inceput cu interfata FisierInput unde doar am declarat doua metode. Am continuat cu clasa abstracta FisierOutput ce implementeaza interfata, unde am suprascris metodele din interfata, adica metodele de citire a folderelor de intrare iesire prin fereastra de alegere. Tot in aceasta clasa abstracta am definit si setterii si getterii. Urmeaza clasa Fisiere, ce mosteneste clasa FisierOutput, unde introduc un nou atribut: adresa imaginii, suprascriu metodele declarate in clasa parinte si am implementez functionalitatea ca utilizatorul sa aleaga modul de citire al folderului. Clasa Modifica mosteneste clasa Fisiere si implementeaza functia de modificare a luminozitatii imaginii, citind de la tastatura factorul de luminozitate. In aceasta clasa am un atribut Imagine, ce este o instanta a clasei Imagine. Aceasta clasa transpune imaginea bmp intr-o matrice de pixeli si modifica pixel cu pixel in functie de factorul de luminozitate. Dupa aceea in Main am creat o instanta a clasei Modifica si am apelat metodele acesteia. La fiecare pas de executie se calculeaza timpul de executie si se afiseaza la sfarsitul executarii pasului.

1. Descrierea modulelor

Asa arata structura de clase:



Pe scurt cate ceva despre fiecare:

FisierInput = aceasta interfata imi pregateste operatiile pentru a citi adresa folderului sursa si pentru a citi adresa folderului destinatie, folosind fereastra de alegere implementata cu JFileChooser.

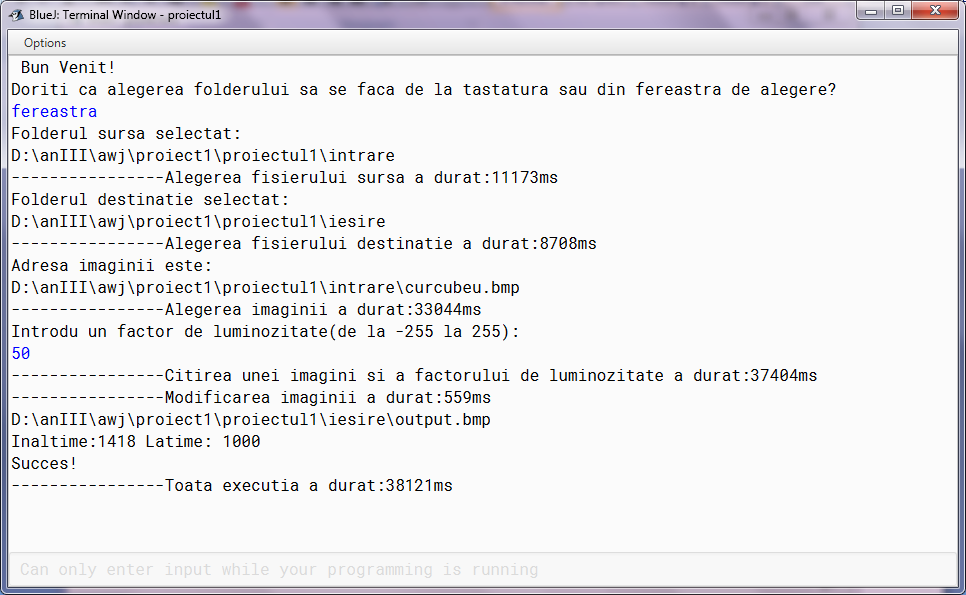
FisierOutput = aceasta clasa abstracta suprascrie operatiile mentionate in interfata, introduce o noua metoda ce obtine adresa imaginii ce va fi modificata prin fereastra de alegere si introduce setterii si getterii pentru adresa\_input si adresa\_output;

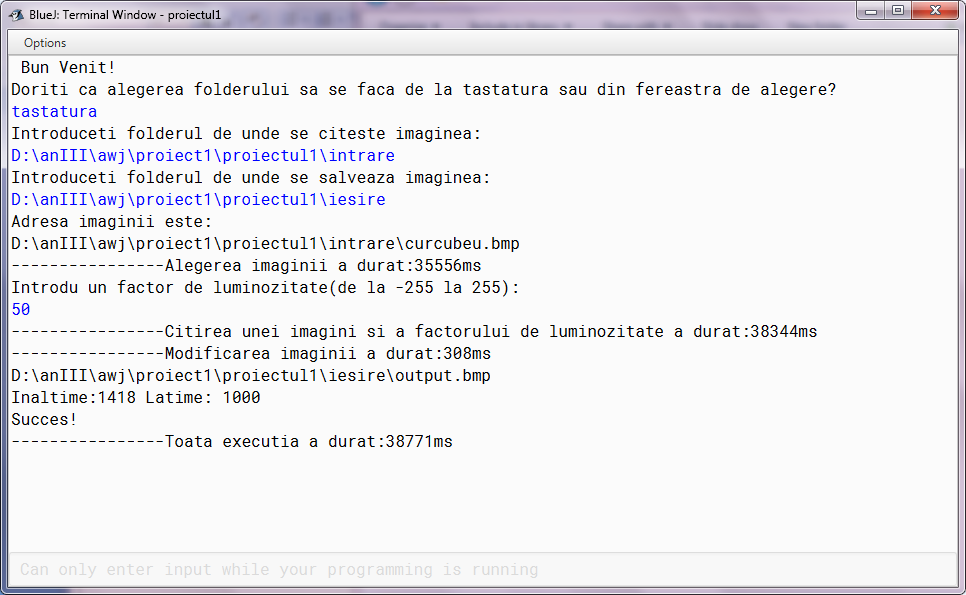
Fisiere = aceasta clasa afla adresa imaginii a carei luminozitate va fi modificata am rescris metoda setInput pentru a obtine adresa imaginii; De asemenea implementeaza setterii si getterii;

Modifica = aceasta clasa citeste de la tastatura factorul de luminozitate si creeaza o instanta a clasei imagine, dupa aceea apeleaza functia de modificare a clasei imagine. Dupa ce fac modificarea luminozitatii se scrie in folderul de iesire imaginea rezultata.

Imagine = aceasta clasa contine setterii si getterii specifici si implementeaza metoda de modificare a imaginii pixel cu pixel. Se imparte fiecare pixel in culorile de baza: rosu, verde si albastru si se aduna aceste valori cu factorul de luminozitate. In functia de citire tratez o exceptie in caz ca imaginea nu exista.

Main = creez o instanta a clasei Modifica si apelez functia de modificare a luminozitatii.

1. Evaluare performante



Conform rularii programului de doua ori, asupra aceleiasi imagini, dar folosind cele doua metode ce citire a folderelor se observa ca: timpul total de executie este aproape acelasi, calitatea pozei rezultate fiind aceeasi in ambele cazuri. Cu alte cuvinte nu exista diferente majore in exectutie, timpul de executie fiind in medie de: 38446 ms.

1. Concluzii

Concluzionand cele mentionate mai sus, aceasta aplicatie a fost realizata prin utilizarea conceptelor de baza ale programarii orientate pe obiecte cum ar fi: incapsularea, mostenirea, suprascrierea, abstractizarea etc. Am facut intai interfata FisierInput, care este implementata de catre clasa abstracta FisierOutput. Aceasta clasa este mostenita de catre clasa Fisiere, iar impreuna cu interfata si clasa parinte se ocupa de obtinerea adresei fisierului de intrare si folderului destinatie. Mai departe imaginea este prelucrata in clasa care mosteneste clasa Fisiere si anume clasa Modifica. In clasa modifica avem o instanta a clasei Imagine, care duce pana la capat modificarea pixel cu pixel a imaginii. In main se creeaza o instanta a clasei Modifica, unde ii este aplicata metoda de setare a luminozitatii.

1. Bibliografie

* <https://cursuri.telacad.ro/courses/course-v1:TelecomAcademy+Java1A+2018_T1/info>
* Curs AWJ: Java01.pdf pana la Java08.pdf
* <https://docs.oracle.com/javase/7/docs/api>