Image Mirroring

Proiect Aplicații Web cu suport Java

Proiect realizat de: Boboc Talida-Ștefania

Facultatea de Automatică și Calculatoare

Grupa 332AB

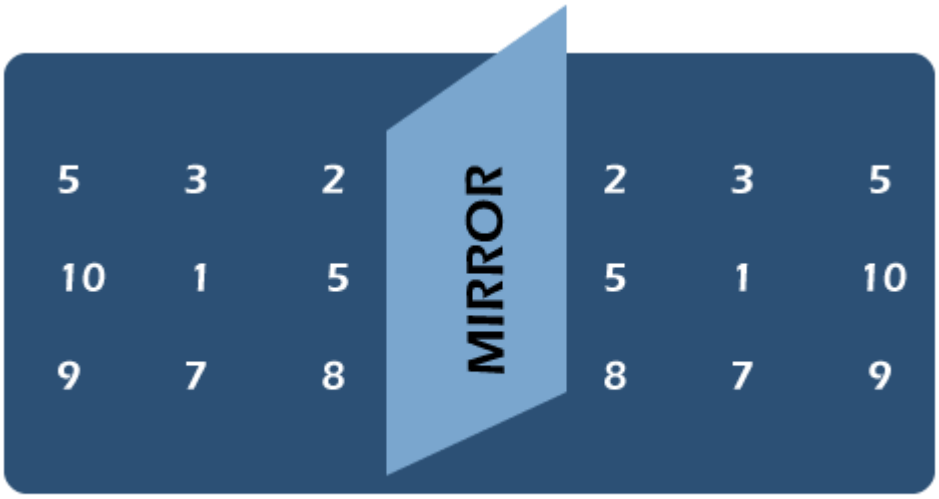
Cuprins

1. Introducere. Descrierea aplicației

Obiectivul general al proiectului este dezvoltarea unei aplicații Java pentru procesare de imagini. Mai exact, aplicația pe care am ales să o implementez propune o soluție pentru oglindirea imaginilor. Pentru a putea rula aplicația trebuie specificată cu exactitate locația imaginii ce urmează a fi modificată precum și denumirea acesteia. Rezultatul final poate fi salvat sub forma unei noi imagini, specificând locul și numele acesteia, sau poate fi suprascris peste imaginea inițială.

2. Partea teoretică

Pentru a putea procesa imaginea dată, am considerat-o ca fiind o matrice de pixeli. Așadar, problema oglindirii imaginii s-a transformat într-o problemă de interschimbare a coloanelor acestei matrici. Astfel, prima și ultima coloană se vor interschimba, respectiv a doua și penultima coloană etc. În cazul în care există număr impar de coloane, coloana din mijloc va rămâne neschimbată.



Mirror Image of a 2D Array

3. Descrierea implementării

Pașii prin care imaginea este procesată sunt următorii:

- Citirea imaginii sursă: imaginea sursă este citită de la sursa specificată în linia de comandă de către utilizator și transformată într-o matrice de pixeli. De asemenea sunt salvate dimensiunile acestei imagini.

- Prelucrarea imaginii: Se parcurge matricea coloana cu coloana și sunt interschimbate coloanele între ele conform algoritmului prezentat la partea teoretică.
- Scrierea imaginii modificate: Noua matrice de pixeli este scrisă ca imagine la adresa specificată în linia de comandă de către utilizator. Această imagine are dimensiunile calculate inițial.

Imaginea sursă este dată sub formă de fișier BMP, adică bitmap pe 24 de biți.

Aplicația permite operații de lucru cu fișiere, odată cu operații de intrare de la tastatura prin paramterii liniei de comandă pentru asignarea fișierelor de intrare și ieșire. Scriem în linia de comandă locația de unde este preluată imaginea și locația unde ne dorim sa ajungă. În implementare am folosit 3 niveluri de moștenire Read -> Write -> ProcessImage. De asemenea am utilizat și o interfață.

4. Descrierea structurală – arhitecturală și funcțională a aplicației

Aplicația conține 2 pachete și anume: packTest unde se găseste clasa Main și packWork unde se găsesc clasele Read, Write, Interface, ProcessImage, ImageMirror.

Funcționarea aplicației urmărește următoarele etape: citirea unui fișier cu extensia .bmp, calcularea timpului de citire a acestui fișier, prelucrarea acestuia și salvarea rezultatului în fișierul specificat de către utilizator, totodată se calulează și timpii aferenți fiecărei etape.

Am implementat un sistem de forma Producer-Consumer care utilizează thread-uri și elemente de sincronizare. Producătorul citește informația din fișierul sursă și o furnizează consumatorului prin intermediul pipe-urilor și a comunicării multithread. Elementele de sincronizare sunt utilizate pentru a proteja împotriva interferenței cu alte thread-uri posibile. După ce toată informația din imaginea sursă este consumată, începe procesarea.

De asemenea, am utilizat pipe-uri pentru a comunica între thread-uri. Acestea permit unui thread să scrie date într-un canal și altui thread sa citească date din același canal. Am folosit clasele oferite de Java (PipedInputStream și PipedOutputStream) pentru a crea și utiliza pipe-uri. Am creat o instanță a clasei PipedOutStream și am conectat-o la o instanță a clasei PipedInputStream prin apelarea metodei connect(), după care am utilizat metodele write() și read() pentru a scrie și citit date din pipe.

5. Descrierea modulelor

- Read: este o clasă abstractă în care este menționată metoda abstractă readImage de tip BufferedImage cu parametrul path, reprezentând calea fișierului de intrare în care se află imaginea. Pentru această clasă, nivelul de moștenire este 1.
- Write: este o clasă abstractă care moștenește clasa Read și implementează metoda readImage (pentru citirea imaginii). De asemenea, ea menționează și metoda abstractă writeImage de tip void, cu parametrii path și image (calea către

fișierul de ieșire și imaginea care trebuie oglindita). Pentru acestă clasă, nivelul de moștenire este 2.

- ProcessImage: moștenește clasa Write și implementează metoda writeImage (pentru scrierea imaginii) și readImage (pentru citirea imaginii). Pentru această clasă, nivelul de moștenire este 3.
- Interface: este o interfață care va fi implementată de către clasa ImageMirror. Ea conține metodele care trebuie implementate de către această clasă.
- ImageMirror: clasa care implementează metodele din Interface si realizează agloritmul de oglindire.
- Main: clasa în care se preiau din linia de comandă calea fișierului de intrare împreună cu numele imaginii inițiale și calea fișierului de ieșire unde se va scrie imaginea modificată. Tot aici se calculeaza și timpii necesari fiecărei etape: citire, prelucrare și scriere.
- Producer: este un obiect de tip Thread care va apela metoda readImage din clasa Read sau din clasa derivata Write. Înainte de citirea fiecărui sfert de imagine, thread-ul va intra în starea Not Runnable folosind metoda Read.
- Consumer: este un obiect de tip Thread care va apela metoda writeImage din clasa Write și metoda mirror din clasa ImageMirror. Am utilizat metoda notify() pentru a mă asigura că Consumer așteaptă până când Producer citește imaginea.

6. Exemplificare

Pentru început, vom adăuga cei doi parametrii necesari, parametrul cu adresa fișierului de intrare și cel cu adresa fișierului de ieșire.

```
Incepe procesul de prelucrare a imaginii!
Introduceti calea unde se gaseste imaginea de intrare:
D:\\FACULTATE\\AN 3 sem 1\\AWJ\\tema proiect\\imagini\\tigru.bmp
Introduceti calea unde doriti sa se afiseze imaginea de iesire:
D:\\FACULTATE\\AN 3 sem 1\\AWJ\\tema proiect\\imagini\\tigru_out.bmp
```

Imaginea de intrare este următoarea:

În urma rulării aplicației, sunt afișate următoarele informații:

```
Citirea s-a efectuat cu succes!
Citirea imaginii a durat: 96.0ms
Algoritmul de transformare a durat: 57.0ms
Scrierea s-a efectuat cu succes!
Scrierea imaginii a durat: 200.0ms
```

Imaginea rezultată este următoarea:

În folderul sursă vom avea:

tigru

tigru_out

7. Cerințe implementare

În vederea realizării proiectului am respectat următoarele cerințe de implementare:

1. Imaginea sursă este un fișier de tip BMP.
2. Am utilizat secvențe de cod low-level.
3. Am inclus în totalitate conceptele POO – încapsulare (aplicația este structurată pe clase și fiecare clasă are metodele ei), moștenire (clasele Read, Write, ProcessImage), polimorfism, abstractizare (clasele Read, Write, ProcessImage).
4. Codul sursă respectă ”coding standars” și este comentat.
5. Am efectuat operații de lucru cu fișiere.
6. În Main sunt preluate de la tastatură datele de intrare în legătură cu fișierele de intrare și fișierele de ieșire.
7. Aplicația este împărțită în clase cu ierarhii și conține 3 niveluri de moștenire.
8. Aplicația include varargs (folosit la funcția Count care calculează timpul aferent unei etape).
9. Aplicația include constructori.
10. Aplicația include bloc static de inițializare.
11. Aplicația include o interfață Interface și clasa ImageMirror care o implementează.
12. Aplicația include clase abstracte: Read, Write, ProcessImage.
13. În clasele Read și Write au fost tratate excepțiile.
14. Aplicația conține 2 pachete: PackTest cu clasa Main și PackWork cu clasele Read, Write, ProcessImage, ImageMirror.
15. Aplicația conține Producer-Consumer.
16. Aplicația conține comunicație prin Pipes.

8. Concluzii

În urma rulării aplicației pe mai multe imagini, am constatat faptul că timpii de citire, prelucrare și scriere diferă în funcție de dimensiunea imaginilor. Totodată, aceștia pot să difere chiar și pentru aceeași imagine.

De asemenea, am constatat faptul ca limbajul Java oferă o modalitate facilă de a implementa algoritmi de prelucrare a imaginilor, deoarece cu ajutorul algoritmilor matemtatici se pot face cu ușurință modificări asupra structurii imaginilor. Am observat în urma realizării acestei aplicații faptul că o stuctura organizată de clase este eficientă și ușurează munca.

9. Bibliografie

În vederea realizării acestui proiect, am studiat următoarele surse:

https://www.tutorialspoint.com/how-to-create-a-mirror-image-using-java-opencv-library#

https://blog.idrsolutions.com/how-to-read-and-write-images-in-java/

https://nisal-pubudu.medium.com/understanding-threads-multi-threading-in-java-6e8c988d26af

https://stackoverflow.com/questions/35317789/java-multithreading-on-very-small-image-taking-a-long-time

https://www.itcsolutions.eu/ro/2011/07/31/tutorial-java-scjp-18-blocuri-de-initializare/

https://www.baeldung.com/java-varargs