Documentație Proiect

-Tema #4 – Level Adjustment (contrast and black/white adjustment)–

Proiectul conține 8 fisiere .java sortate in 2 pachete:

-Pachetul **packWork** conține fișierele de bază ale programului, care realizează citirea și scrierea imaginilor și a timpilor în fișiere:

- **Producer** – Thread-ul care citește imaginea câte ¼ la un moment dat.
- **Consumer** – Thread care primește sferturile de imagine, dar este și locul unde are loc prelucrarea imaginii citite și salvarea ei pentru a putea fi vizualizată din file explorer.
- **Buffer** – fișierul intermediar între Producer si Consumer care face posibilă transmiterea sferturilor de imagine.
- **Interface** – interfața proiectului, implementează o singură metoda care va fi moștenită într-o clasa abstractă și folosită, în final, într-o clasă concretă care moștenește clasa abstractă.
- **Abstract** – clasa abstractă care moștenește Interface si pune bazele constructorilor folosiți în clasa de bază pentru afișarea imaginilor.
- **ShowImage** – clasa concretă care moștenește clasa Abstract și realizează salvarea imaginilor în folder-ul proiectului.
- **PrintTimes** – preia timpul de citire de la Producer și timpii de prelucrare și, respectiv, scriere din Consumer și îi salvează in 4 fișiere separate (3 pentru timpii individuali și 1 fișier pentru timpul total de rulare)

-Pachetul **packTest** conține 1 singur fișier, aplicația de bază care se folosește de toate clasele din **packWork**:

- **TestClass** – conține main-ul rulabil din program și realizează funcționarea corectă a acestuia facând multiple verificări atât în consolă cât și pentru buna funcționare a claselor

Scopul proiectului este citirea corecă a unei imagini și prelucrarea contrastului și a nivelului de alb/negru din aceasta.

Voi explica funcționalitatea proiectului din cadrul **TestClass** deoarece acolo are loc programul în sine.

Primul lucru întâlnit în fișier este o funcție care verifică dacă un String este sau nu un număr pozitiv sau negativ, funcție pe care o voi folosi mai târziu. Apoi începe metoda main a programului. Dau un mesaj utilizatorului pentru a introduce numele fișierului dorit a fi modificat și preiau input-ul. Fac verificarea dacă input-ul introdus este un nume valid de fișier, dacă exista și să nu fie nume de director. Dacă datele introduse ca input sunt greșite sau nu se găsește fișierul dorit, după 5 încercări programul se închide automat.

Dacă fișierul există și sunt îndeplinite condițiile inpuse, programul trece la următoarea etapă. Acum îi cer utilizatorului sa introducă valori pentru contrast și/sau pentru nivelul de alb/negru. Nivelul contrastului am cerut să fie pozitiv, oricât de mare. Prelucrarea din **Consumer** va folosi contrastul în procente. Adică pentru valoarea 5 introdusă de utilizator, deci valoarea contrastului inițial din imagine va fi redus la 5% din valoarea inițială. Pentru valoarea 120 contrastul va deveni 120% din valoarea inițială, pentru 500 va fi 500% șamd. Am făcut acest lucru prin înmulțirea valorilor RGB din imaginea inițială cu [valoarea introdusă de utilizator / 100]. Deci pentru contrast=200 pixelii vor avea valorile RGB înmulțite cu 2. Nu am impus o limită asupra contrastului deoarece pentru pixelii RGB din imagine fac verificări și nu se depășește valoarea de 255.

După contrast, îi cer utilizatorului să introducă valoarea pentru nivelul de alb/negru. Aici am impus condițiile ca valorile să fie între -100 (valoare pentru negru) și 100 (valoare pentru alb). Am realizat acest lucru adunând în **Consumer** pentru valorile de RGB valoarea introdusă de utilizator. Limitele sunt modificabile, puteam alege -500 și 500, -300 și 125, -inf și +inf, eu am decis să fie între -100 și 100.

Atât pentru contrast cât și pentru nivelul de alb/negru am făcut verificări în cadrul codului pentru a preveni valori greșit introduse. De exemplu pentru contrast nu se pot introduce valori negative și pentru nivelul de alb/negru nu este permisă ieșirea din intervalul -100 ~ 100. Pentru ambele, de asemenea, am făcut verificări să nu se introducă valori non-numerice (de exemplu “25T” sau “negru” sunt valori rejectate).

Totodată, dacă utilizatorul nu dorește să inițializeze contrast sau să modifice nivelul de alb/negru, când îi este cerut să introducă datele, la apăsarea tastei ENTER valorile pentru acestora iau valori predefinite. Eu le-am ales să fie implicit 0 amândouă, dar această valoare poate fi modificată în cod. Cu valorile predefinite, practic, codul creează o copie a imaginii originale.

Mai departe, am construit cate o variabilă de tip **Buffer**, **Producer** și **Consumer** și le-am inițializat cu imaginea inițială și valorile stabilite de utilizator pentru contrast și nivelul de alb/negru.

La finalul codului, folosind funcțiile create în **Producer** și **Consumer** *: getTimeRead(), getTimeWork()* și *getTimeWrite()*. Aceste funcții returnează timpii de citire, prelucrare și scriere, respectiv. Acești timpi sunt pasați clasei **PrintTimes** care creează fișierele cu timpii programului.

Am respectat cele 4 principii de POO: **Abstractizarea** (prin clasa abstractă **Abstract**), **Încapsularea**(prin multiplii parametrii privați din cadrul claselor), **Polimorfismul**(constructorii din **ShowImage** care pot avea multiplii parametrii posibili) și **Moștenirea** (tripla moștenire realizată prin lanțul **Interface** --> **Abstract** --> **ShowImage**).

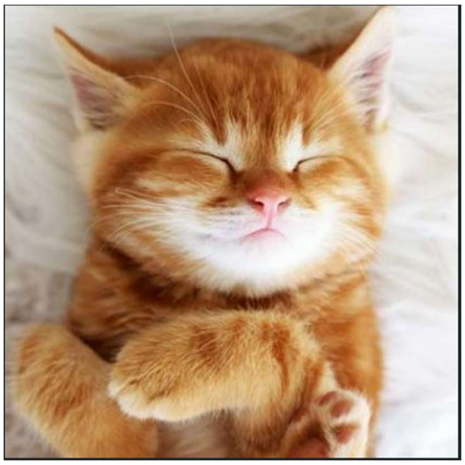
În clasa **PrintTimes** m-am folosit și de *varargs* în cadrul constructorului prin sintaxa “`public PrintTimes(long ...numbers)`”.

Mai jos voi atașa două imagini de “Before” și “After” cu un contrast dat ca input de 200 și un nivel de alb/negru de -15:

Conținutul consolei la rularea programului

```
<terminated> TestClass [Java Application] F:\Eclipse Java\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win3
Introduceti numele fisierului cu extensia .bmp: input2.bmp


Introduceti valorile pentru contrast(>0) si black/white(-100 ~ 100)
Apasati ENTER pentru a seta valoarea default 0 la fiecare!

Contrast: 200
Contrast va avea valoarea: 200.0


Black/White: -15
Black/White va avea valoarea: -15.0


S-a apelat constructorul din Buffer
S-a apelat constructorul din Producer: Thread-0
S-a apelat constructorul din Consumer: Thread-1
Inceput Thread in Producer: Thread-0
Inceput Thread in Consumer: Thread-1
Inceput run in Producer
Inceput run in Consumer
Citim 1/4 din imagine!
Consumer primeste datele din buffer
Citim 2/4 din imagine!
Consumer primeste datele din buffer
Citim 3/4 din imagine!
Consumer primeste datele din buffer
Citim 4/4 din imagine!
Consumer primeste datele din buffer
Incepere prelucrare date in Consumer
Sfarsit prelucrare in Consumer


Salvare in fisier imagini before si after
Sfarsit citire in Producer

Time Successfully wrote to the file.

Time Successfully wrote to the file.

Time Successfully wrote to the file.

Total Time Successfully wrote to the file.
5144
```

Before



After

În realizarea proiectului, m-am inspirat din următoarele surse:

-https://www.programiz.com/java-programming/online-compiler/

(pentru teste rapide ale unor funcții)

-https://docs.oracle.com/javase/7/docs/api/java/awt/image/BufferedImage.html

(pentru a vedea metodele pe care le pot folosi pentru a lucre cu imaginile)

-https://docs.oracle.com/javase/8/docs/index.html

(în general pentru mai multe metode și clase (de ex String))

-https://www.geeksforgeeks.org/opencv-understanding-contrast-in-an-image/

(pentru a înțelege cum pot lucra cu contrastul unei imagini)