Image resizing (Zooming +/-) – keeping aspect ratio

Pixel replication method

Proiect relizat de CIUCIU Anca-Maria

Grupă: 333 AA

➢ Introducere

Metoda de zooming Pixel Replication are, ca toate celelalte metode de zooming din lista de lucrări (Zero order hold method, Zooming K times method), avantajele și dezavantajele ei.

Aceasta metodă este, de asemenea, cunoscută și ca metoda de interpolare a celui mai apropiat vecin. Așa cum sugerează și numele, în această metodă replicăm pixelii vecini. Algoritmul lucrează pe principiul creșterii numărului pixelilor.

Așadar, pixelul este asignat pixelului cel mai apropiat adresei nou generate, ca și pixel de ieșire. Adresele fracționare calculate pentru pixelii sursă vor fi rotunjite la cel mai apropiat pixel, valid ca adresă.

Practic creăm pixeli noi be baza celor pe care deja ii avem. Fiecare pixel este replicat de n ori rând cu rând și coloană cu coloană și astfel obținem imaginea mărita de care avem nevoie.

➢ Exemplu de aplicare al algoritmului

Se consideră o imagine de 2 rânduri și 2 coloane și vrem să o mărim de două ori utilizând această metodă. Presupunem că imaginea poate fi înțeleasă ca urmatoarea matrice cu valorile pixelilor: **1 2**

3 4

- Prima dată se va aborda **row wise zooming** – mărire rând cu rând. Astfel, se copiază pixelii rândurilor la o celulă adiacentă ca in exemplul următor:

  1 1 2 2

  3 3 4 4

- În continuare, se va realiza **column wise zooming** – replicarea fiecărui pixel coloană cu coloană:

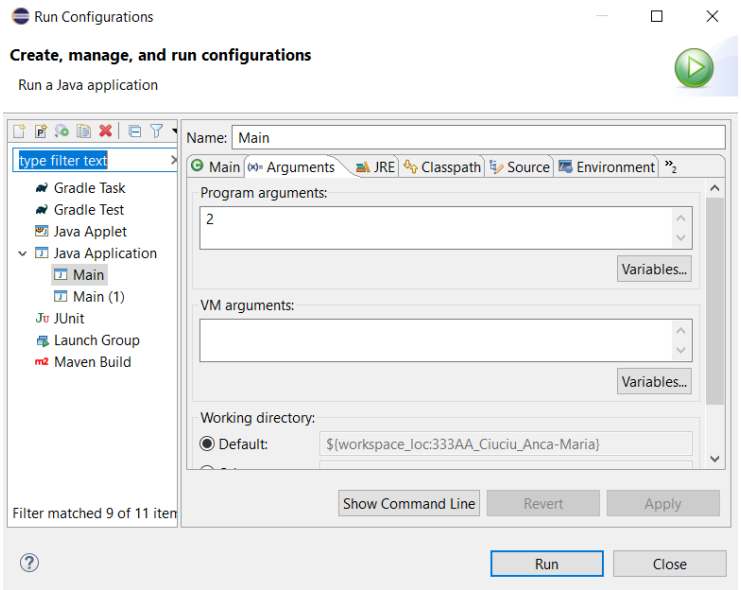
1 1 2 2
1 1 2 2
3 3 4 4
3 3 4 4

Prin urmare, imaginea originală de 2 x 2 a fost convertită într-una de 4 x 4 după mărire. Imaginea nouă are dimensiunile:
nr. rânduri imag. inițiala * factorul de mărire, nr. col. imag. inițiala * factor de mărire.

➢ Structura proiectului

✓ Parametri de intrare din linia de comandă (Run Configurations):

- Factorul de zooming: 2 (pe care îl vom varia pentru mai multe exemple)



✓ Parametri de la tastatură:

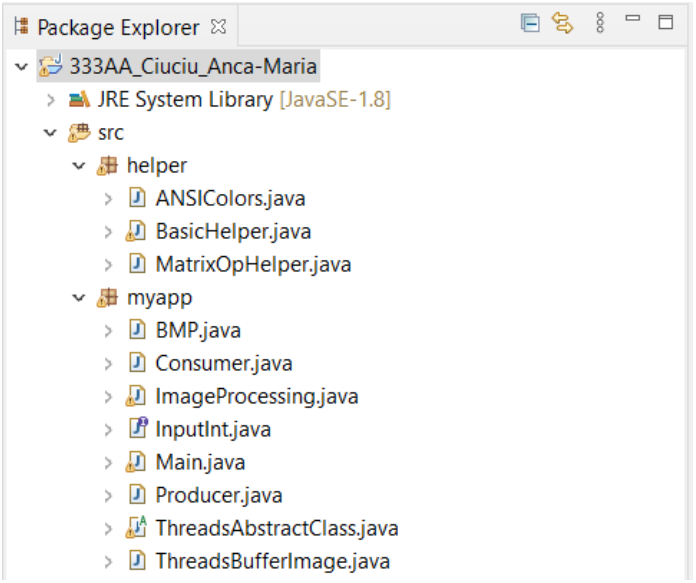
- Denumirea imaginii de intrare: flag.bmp
- Denumirea imaginii de ieșire: out.bmp

```
=== Let's start the image processing! ===

- Please, type the input file name:
flag.bmp

- Now, type the output file name:
out.bmp
```

✓ Ierarhie



➢ Arhitectura funcțională – clase

Proiectul conține doua pachete: *myapp* unde se află rezolvare efectivă a problemei și *helper*, unde se află mai multe utilitare dezvoltate cu scopul refolosirii codului.

- helper package conține următoarele clase:

  - MatrixOpHelper

  Aceasta clasa este compusă din metode extrem de utile pentru manipularea matricelor. Asadar, operatiile implementate in acest context sunt: `convertBytesToPixelMatrix` si `convertBytesToPixelMatrix. Dsd`

  Dupa cum reiese si din semnaturi, aceste doua operatii au rolul de a converti o matrice formata din Bytes intr-o matrice formata exclusive din pixeli si viceversa.

  - BasicHelper

  Aceasta clasa continue metode variate precum afisarea mesajelor cu scop de debug (*printDebugMessages*), afisarea unei matrice (*printMatrix*), afisarea proprietatilor ce tin de dimensiune a imaginii date spre input (*printInputFileProps*), setarea proprietatilor ce tin de dimensiune pentru I/O bmps, convertirea din bytes in int (*intToBytes*), precum si calcularea padding-ului folosindu-ne de bytes. (*computePaddingUsingBytes*).

  - ANSIColors

  Aceasta clasa a fost implementata cu scopul unui design mai atractiv, pentru a evidentia mesajele afisate in consola folosind mai multe culori. De asemenea, am utilizat un background color pentru fiecare element din matricea rezultat pentru a corespunde cu valorile RGB are imaginii date spre procesare, flag.bmp.

! Specificatie : pentru a obtine acelasi efect in consola, este necesara downloadarea plugin-ului ANSI console. Fara acest plugin, proiectul NU va emite erori, doar ca nu se va mai afisa in acelasi format.

- myapp package conține următoarele clase:

  - BMP

    Aceasta clasă conține caracteristicile fișierului de input (în cazul nostru, ireland.bmp) cât și caracteristicile trimise spre procesare – înălțime și lățime imagine, numărul de bytes, dimensiunea imaginii, și padding-ul.

    Astfel, s-au dezvoltat o serie de getters și setters pentru fiecare caracteristică în parte.

  - ThreadsAbstractClass

    Clasa aceasta moștenește clasa Thread, fiind implementată o metodă specifică thread-urilor, "run". Această metodă interoghează o variabilă bool "type". În cazul în care aceasta este 1, înseamnă că se inițializează procesul pentru Producer și odată cu el, procesul de citire, și prelucrare a fisierului BMP. In continuare, urmeaza citirea header-ului imaginii si setarea proprietatilor de input ale fisierului, obtinandu-se astfel datele transmise spre Consumer.

    Totodata, in final se va trimite output file-ul spre procesare cu:

```
// Initierea prelucrarii imaginii
BMP initOutputFile = ImageProcessing.processOutputFile(inFile, resultPixelMatrix, outputBMPFileName);
ImageProcessing.getFinalHeader(inFile, initOutputFile);
```

    Este de mentionat faptul ca, cat timp thread-ul Consumer este activ, thread-ul Producer va fi in starea de sleep iar la finalul intregului proces ambele thread-uri sunt distruse.

  - Producer

    Această clasă este folosită pentru pattern-ul Producer-Consumer.

Producer moștenește clasa ThreadsAbstractClass, implementând un constructor pentru Producator.

- Consumer

Această clasă este folosită pentru pattern-ul Producer-Consumer. Consumer moștenește clasa ThreadsAbstractClass, implementând un constructor pentru Consumator.

- ThreadsBufferImage

Acest buffer are rolul de a consolida implementarea multithreading -ului. Astfel, implementarea acesteia contine un getter si un setter astfel ca producatorul va transmite informatiile iar consumatorul le va prelua.

- ImageProcessing

Acestă metodă descrie întregul algoritm de procesare a imaginii, de manipulare a matricei de bytes și pixeli și de scriere a fisierului rezultat.

Prin urmare, metoda *imageProcessOnInputFile* primește ca input numele fisierului dat spre prelucrare. Se declara un obiect de tip BMP, verificandu-se totodata prin intermediul unei exceptii daca exista un fisier de intrare valid.

Urmeaza citirea datelor din fisierul de input, mai exact primii 54 de bytes si retinerea header-ului.

De asemenea, se vor specifica dimensiunile pentru obiectul declarat, se calculeaza dimensiunea unei linii (numarul de bytes) :

```
// Calcularea numarului de bytes de pe o linie a matricei
bmp.setRowLength((headerLen/BMPwidth) * bmp.getImageWidth());
```

dupa care se ve calcula padding-ul total si numarul total de bytes:

```
// Calcularea padding-ului total pentru fisierul de intrare
bmp.setImagePadding(bmp.getImageSize() - bmp.getImageNoBytes());

// Calcularea numarului total de bytes
bmp.setImageTotalBytes(bmp.getRowLength() * bmp.getImageHeight() + bmp.getImagePadding());
```

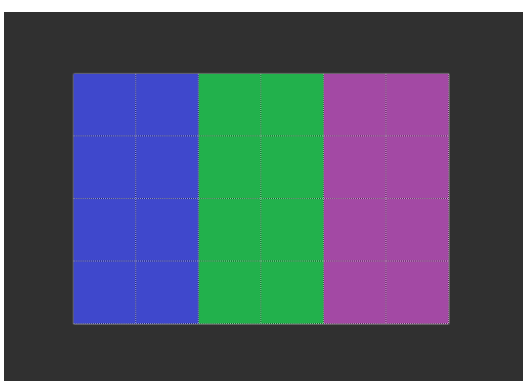
Clasa *processOutputFile* se ocupă cu calcularea dimensiunilor finale pentru matricea de bytes si paddingul care trebuie adaugat pe fiecare

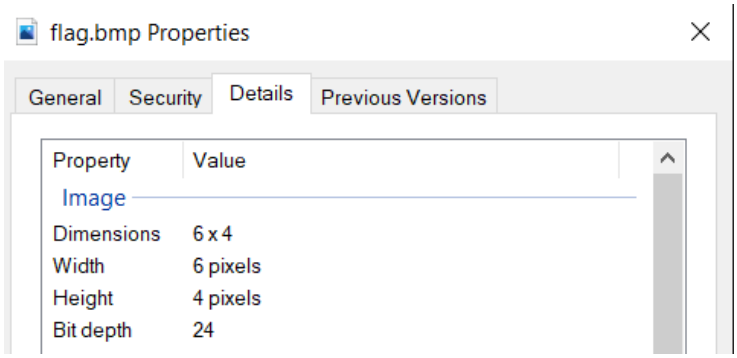
linie. Se vor seta valorile finale pentru fisierul de iesire, si se va afisa folosind ANSI colors matricea maximizata rezultata:

```
===  Processed Matrix  ===
204 72 63 204 72 63 204 72 63 204 72 63 76 177 34 76 177 34 76 177 34 76 177 34 164 73 163 164 73 163 164 73 163 164 73 163
204 72 63 204 72 63 204 72 63 204 72 63 76 177 34 76 177 34 76 177 34 76 177 34 164 73 163 164 73 163 164 73 163 164 73 163
204 72 63 204 72 63 204 72 63 204 72 63 76 177 34 76 177 34 76 177 34 76 177 34 164 73 163 164 73 163 164 73 163 164 73 163
204 72 63 204 72 63 204 72 63 204 72 63 76 177 34 76 177 34 76 177 34 76 177 34 164 73 163 164 73 163 164 73 163 164 73 163
204 72 63 204 72 63 204 72 63 204 72 63 76 177 34 76 177 34 76 177 34 76 177 34 164 73 163 164 73 163 164 73 163 164 73 163
204 72 63 204 72 63 204 72 63 204 72 63 76 177 34 76 177 34 76 177 34 76 177 34 164 73 163 164 73 163 164 73 163 164 73 163
204 72 63 204 72 63 204 72 63 204 72 63 76 177 34 76 177 34 76 177 34 76 177 34 164 73 163 164 73 163 164 73 163 164 73 163
204 72 63 204 72 63 204 72 63 204 72 63 76 177 34 76 177 34 76 177 34 76 177 34 164 73 163 164 73 163 164 73 163 164 73 163
```

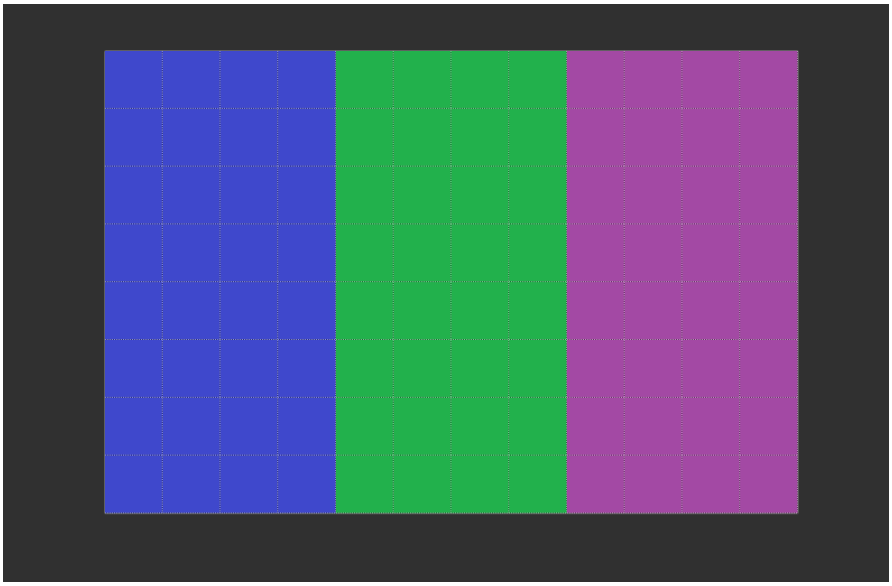
Am ales sa folosesc un ANSI color background pentru a evidentia campurile RGB are flag-ului.

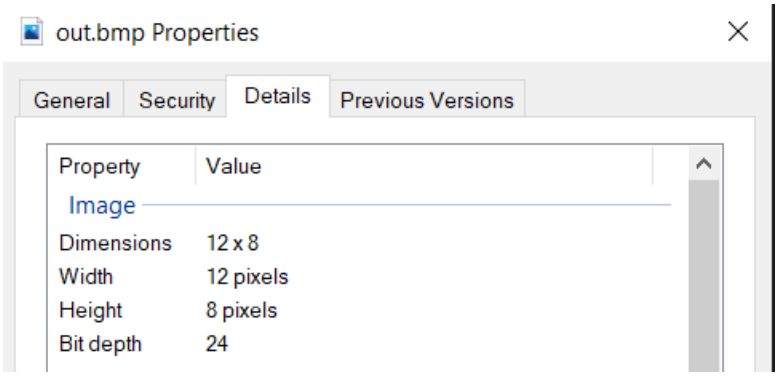
Dupa cum se poate si observa, imaginea initiala, neprelucrata afisata in mod reprezentativ pe pixeli este, observandu-se dimensiunea de inceput, de 6 x 4 pixels:





In urma aplicarii algoritmului de procesare a imaginii multithreading, am obtinut o imagine cu dimensiunile dublate dupa cum se poate observa si din numarul de pixeli:





- Main

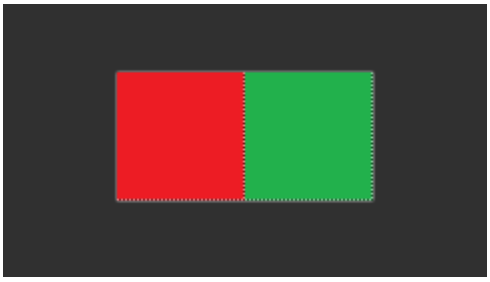
Clasa Main are rolul de a incepe procesul multithreading, apeland metodele corespunzatoare pentru initierea procesului de prelucrare de imagine. Totodata, in main se citesc de la tastatura numele fisierului de intrare si al celui de iesire. De asemenea, sunt calculati si timpii de executie:

```
Elapsed time for reading the image: 8362
```

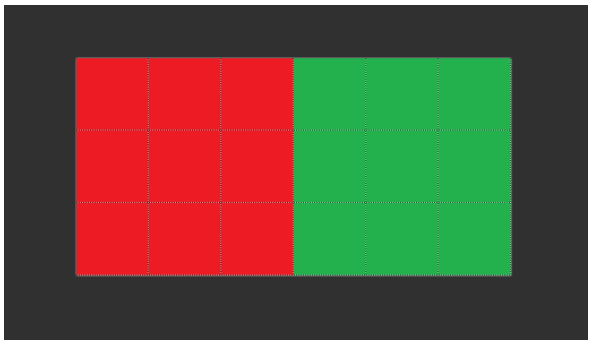
```
Elapsed time for processing the image: 2
```

➢ Alte rezultate obtinute cu aceasta aplicatie

- Mărirea unei imagini 2 x 1 px cu scaleFactor = 3



Property	Value
Image	
Dimensions	2 x 1
Width	2 pixels
Height	1 pixels
Bit depth	24



Property	Value
Image	
Dimensions	6 x 3
Width	6 pixels
Height	3 pixels
Bit depth	24

```
===  Processed Matrix  ===
36 28 237 36 28 237 36 28 237 76 177 34 76 177 34 76 177 34 0 0
36 28 237 36 28 237 36 28 237 76 177 34 76 177 34 76 177 34 0 0
36 28 237 36 28 237 36 28 237 76 177 34 76 177 34 76 177 34 0 0
```

➢ Bibliografie

https://github.com/

https://www.geeksforgeeks.org/image-processing-in-java-read-and-write/

https://techvidvan.com/tutorials/java-digital-image-processing/

https://www.tutorialspoint.com/dip/Zooming_Methods.htm

http://www.java2s.com/Tutorial/Java/0261__2D-Graphics/Enlarginganimagebypixelreplication.htm

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

https://stackoverflow.com/questions/36813921/zooming-an-image-in-java-by-repication-method

https://marketplace.eclipse.org/content/ansi-escape-console

https://www.youtube.com/watch?v=xLrK9PctVAk

https://en.wikipedia.org/wiki/Producer%E2%80%93consumer_problem

Vizualizarea si crearea imaginilor de test au fost realizate cu programul paint.net : http://paint.net/