APLICATII WEB CU SUPORT JAVA

***Image resizing (Zooming +/-) with keeping aspect ratio. Zooming K times method***

-BARBU LUCIAN STEFAN 333AA-

Cuprins

[1. Cerinte 3](#_Toc124250618)

[*1.1 Cerinte de implementare* 3](#_Toc124250619)

[*1.2 Cerinta tema* 4](#_Toc124250620)

[2. Descrierea algoritmului 4](#_Toc124250621)

[3.Descrierea structurala si functionala a codului 7](#_Toc124250622)

[*3.1 Pachete* 7](#_Toc124250623)

[*3.2 Clase* 7](#_Toc124250624)

[4.Exemplificarea functionalitatii programului 14](#_Toc124250625)

[4.1 Marirea imaginii 14](#_Toc124250626)

[4.2 Micsorarea imaginii 16](#_Toc124250627)

# 1. Cerinte

## *1.1 Cerinte de implementare*

1. Imaginea sursa este BMP (fisier) – 24bit BMP – RGB.
2. Pentru procesare se folosesc doar algoritmi si/ sau secvente de cod low-level (nu se accepta utilizare de metode de procesare altele decat cele scrise in tema).
3. Include in totalitate conceptele POO – incapsulare, mostenire, polimorphism, abstractizare.
4. Codul sursa respecta absolut toate “Coding standards”. Codul sursa este comentat.
5. Operatii de lucru cu fisiere.
6. Operatii de intrare de la tastatura si prin parametri liniei de comanda pentru asignarea fisierelor de intrare, parametri / setarile / optiunile de executie si pentru asignarea fisierelor de iesire.
7. Aplicatia trebuie sa fie multimodulara (impartirea in clase cu ierarhii – chiar cu cost in timp de procesare). Cel putin 3 niveluri de mostenire.
8. Include varargs.
9. Include constructor.
10. Include cel putin un bloc de initializare si un bloc static de initializare.
11. Include Interface (cu o clasa care o implementeaza).
12. Include Clase Abstracte cu metode abstracte si clase concrete care extind clasele abstracte.
13. Include tratarea exceptiilor.
14. Aplicatia contine 2 pachete: Pachetul 1 sa contina aplicatia de test, pachetul 2 sa contina restul claselor.
15. Aplicatia contine Producer-Consumer cu urmatoarele cerinte:

* Un nou thread este alocat citirii din fisier a imaginii sursa – Producer Thread. Intra in Not Runnable dupa citirea a fiecarui sfert (1/4) de informatie.
* Un nou thread (Consumer Thread) este alocat consumului informatiei furnizate de Producer Thread. Se utilizeaza “multithread communication” (notify).
* Se insereaza output la consola si sleep (1000) pentru a evidentia etapele comunicarii.
* Se folosesc elementele de sincronizare pentru protectia la o eventuala interferenta cu alte posibile threaduri.
* Dupa terminarea consumului intregii informatii de imagine sursa, se incepe procesarea.

1. Aplicatia contine comunicatie prin Pipes cu urmatoarele cerinte:

* Consumer utilizeaza un Pipe pentru a transmite imaginea procesata catre un obiect de tipul WriterResult.
* Transmiterea prin pipe se face partitionand informatia in 4 segmente.
* La transmiterea fiecarui segment segment se trimite la consola un mesaj.
* La receptia fiecarui segment segment se trimite la consola un mesaj.
* Rezutatul se depune intr-un fisier.

## *1.2 Cerinta tema*

Tema consta in implementarea in limbajul Java a unui program de redimensionare a imaginilor folosing algoritmul “Zooming k times”.

# 2. Descrierea algoritmului

Algoritmul are la baza lucrul cu pixelii unei imagini si depinde de un anumit factor (numit factor de zoom) notat cu “k”. Etapele algoritmului sunt urmatoarele:

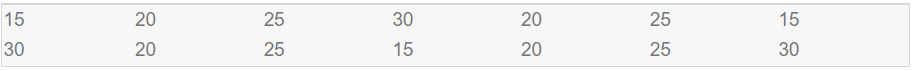
1. Row wise zooming (Zoom pe linii) :

* Se aleg primii doi pixeli vecini (15 si 30) :

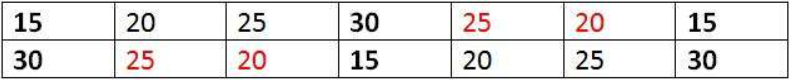


Daca presupunem k = 3 => trebuie introduse k – 1 = 3 – 1 = 2 valori noi intre acestia.

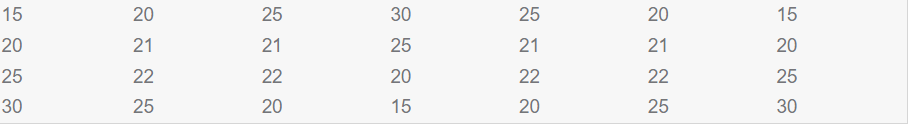
* Se scade cel mai mic din cel mai mare => 30 – 15 = 15
* Aceasta valoare se imparte la k => 15 / 3 = 5(numit aux)
* Se adauga aceasta valore la numarul cel mai mic => 15 + 5 = 20.Acesta va fi primul pixel nou introdus in imagine(imediat dupa 15).
* Se mai adauga inca o data valoarea aux => 20 + 5 = 25.Acesta va fi urmatorul pixel ce va fi adaugat in imagine.
* Adaugam 2 pixeli noi deoarece k = 3 (3 – 1 = 2).
* Se repeta operatia pentru urmatorii doi pixeli vecini (30 si 15) pana la sfarsitul liniei.
* Se repeta operatia pe toate liniile imaginii si se obtine :



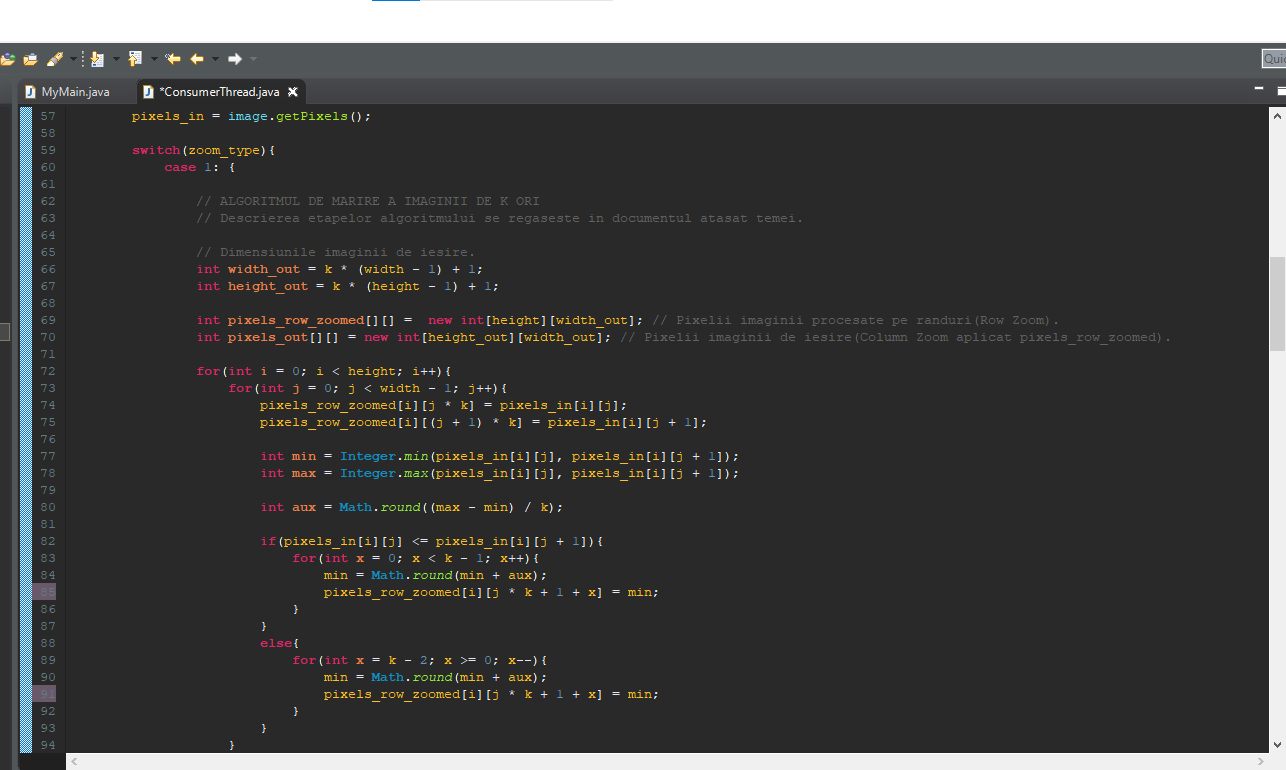
* Intre cei doi pixeli din imaginea initiala pixelii adaugati trebuie ordonati (crescator de la cel mai mic la cel mai mare),astfel ca imaginea finala va arata astfel:

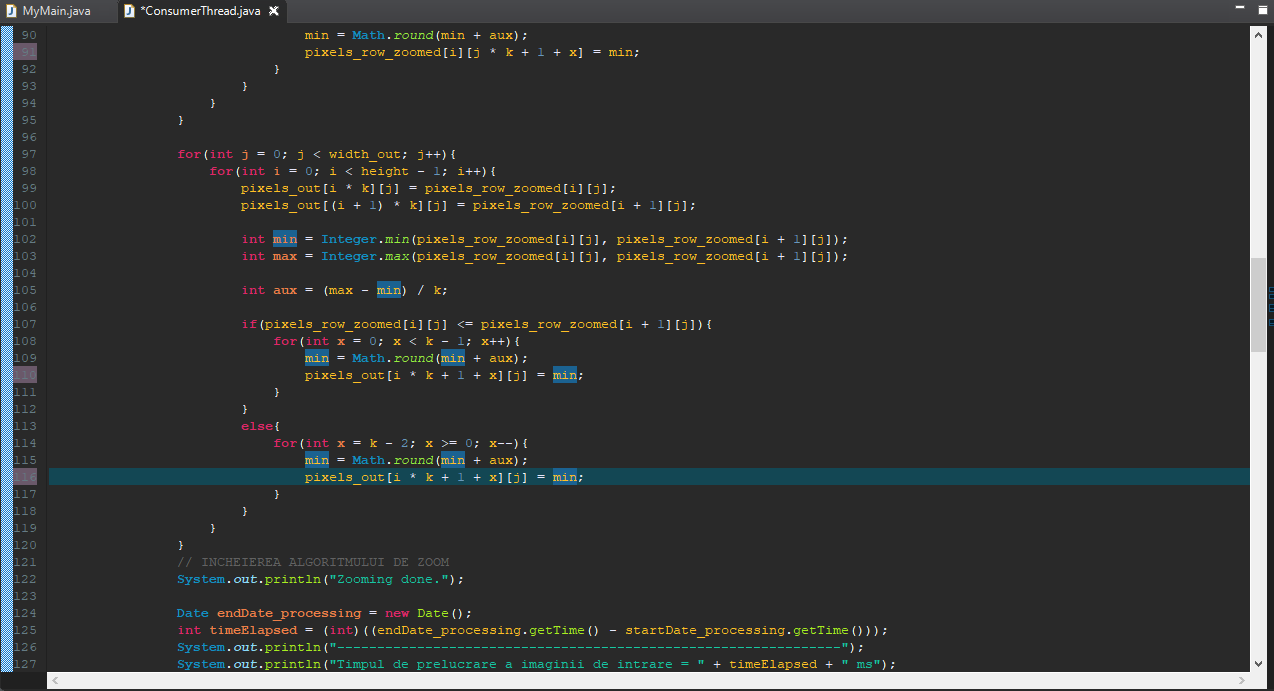


1. Column wise zooming (Zoom pe coloane) :
   * Se aplica aceeasi procedura ca pe linii,cu diferenta ca acum pixelii vecini se iau pe coloana (15 si 30, 20 si 25 etc.)
   * Astfel imaginea finala va avea urmatorea matrice de pixeli:



Daca imaginea initiala avea dimensiunile ***height*** x ***width*** atunci imaginea finala va dimensiunile ***k \* (height – 1) + 1*** x ***k \* (width – 1) + 1***.



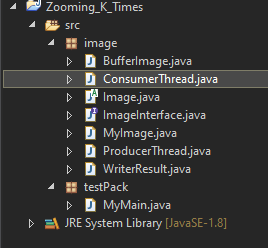


Implementarea algoritmului in cod

Algoritmul este implementat in metoda run() a clasei ConsumerThread.

# 3.Descrierea structurala si functionala a codului

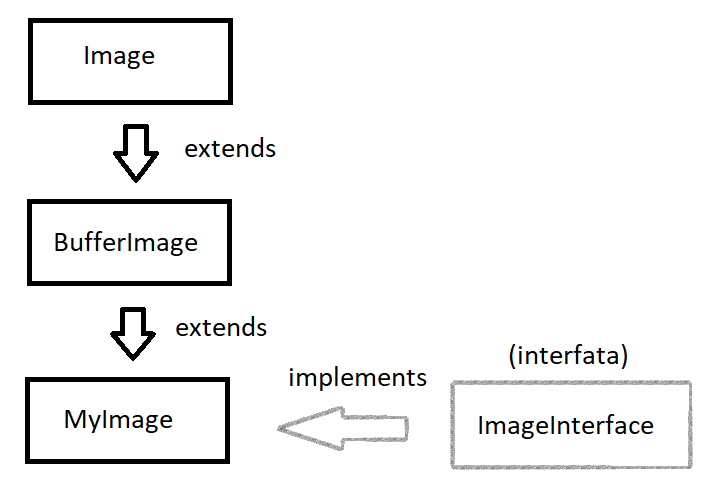
## *3.1 Pachete*

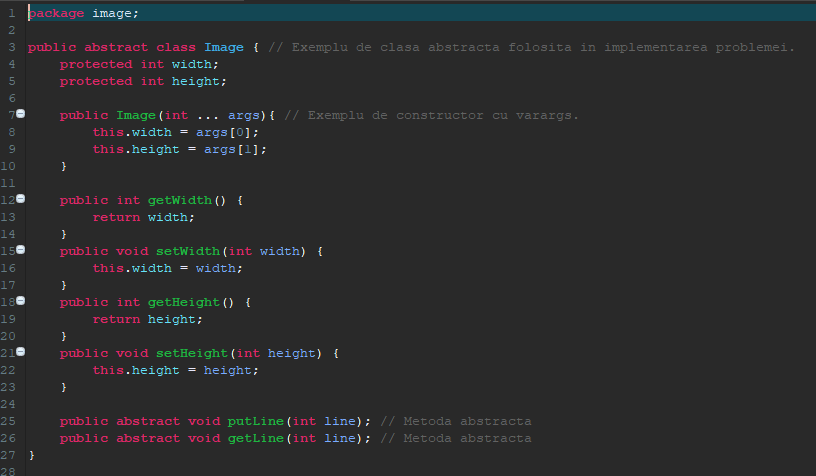


Aplicatia contine 2 pachete : pachetul image (care contine clasele ce definesc imaginea si threadurile folosite) si pachetul testPack (care contine aplicatia de test).

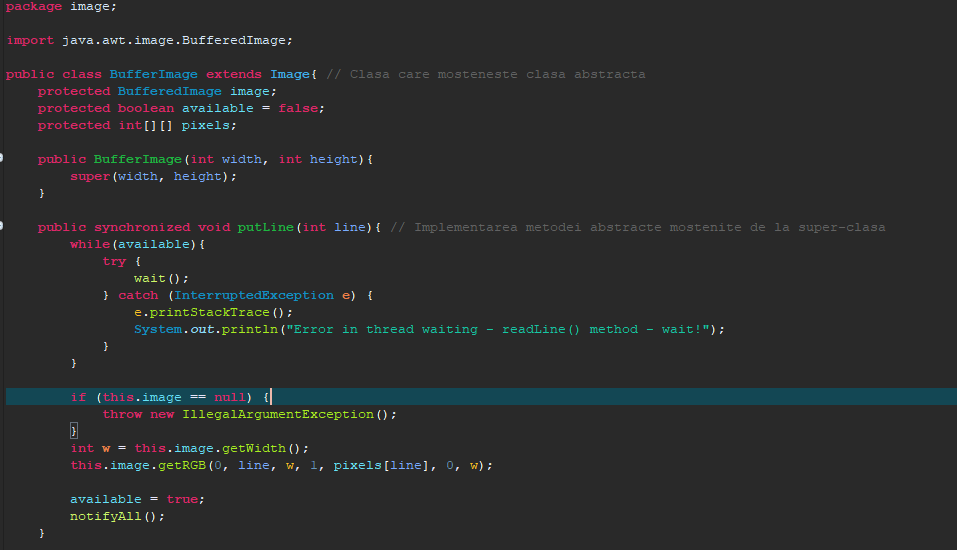
## *3.2 Clase*

Pachetul **image :**



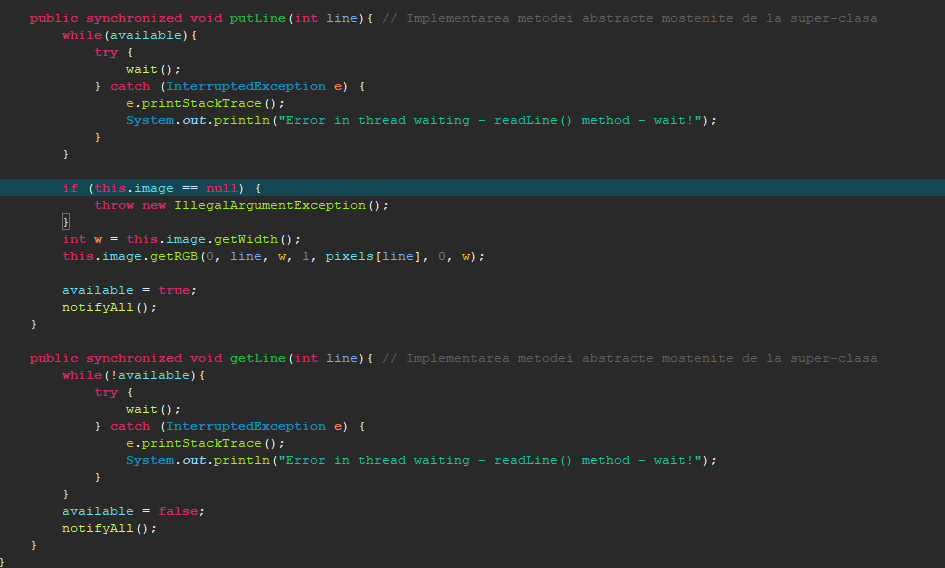


Clasa abstracta Image



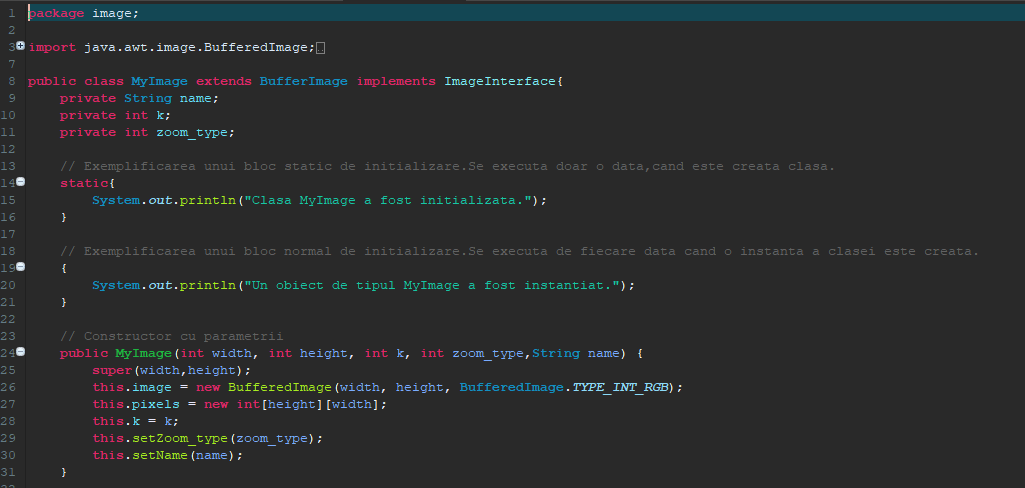
Clasa BufferImage

Deoarece clasa BufferImage mosteneste clasa abstract Image aceasta implementeaza toate metodele abstracte ale clasei parinte.

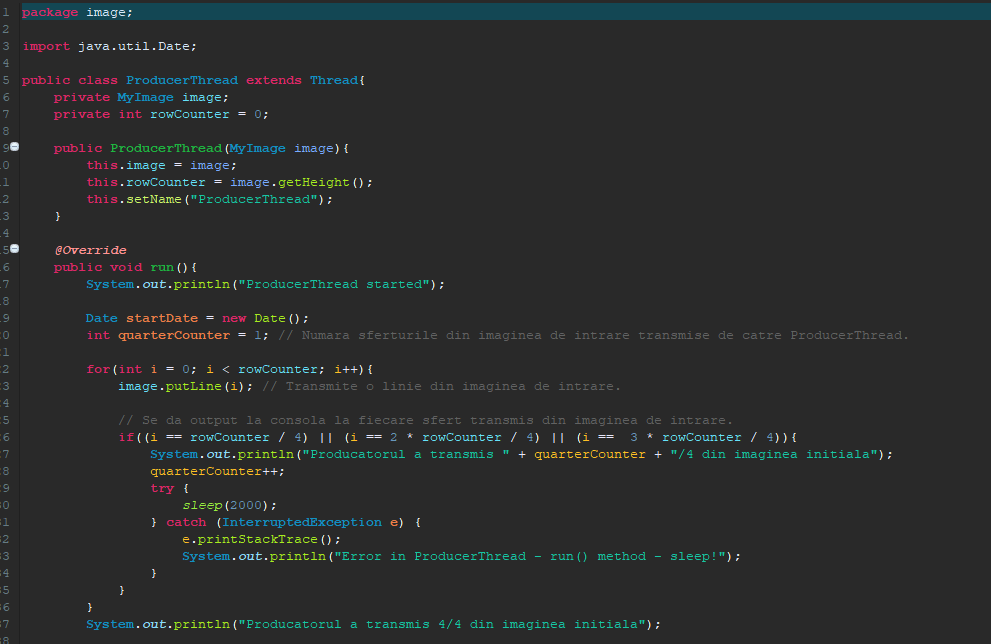


Metodele abstracte implementate in clasa BufferImage

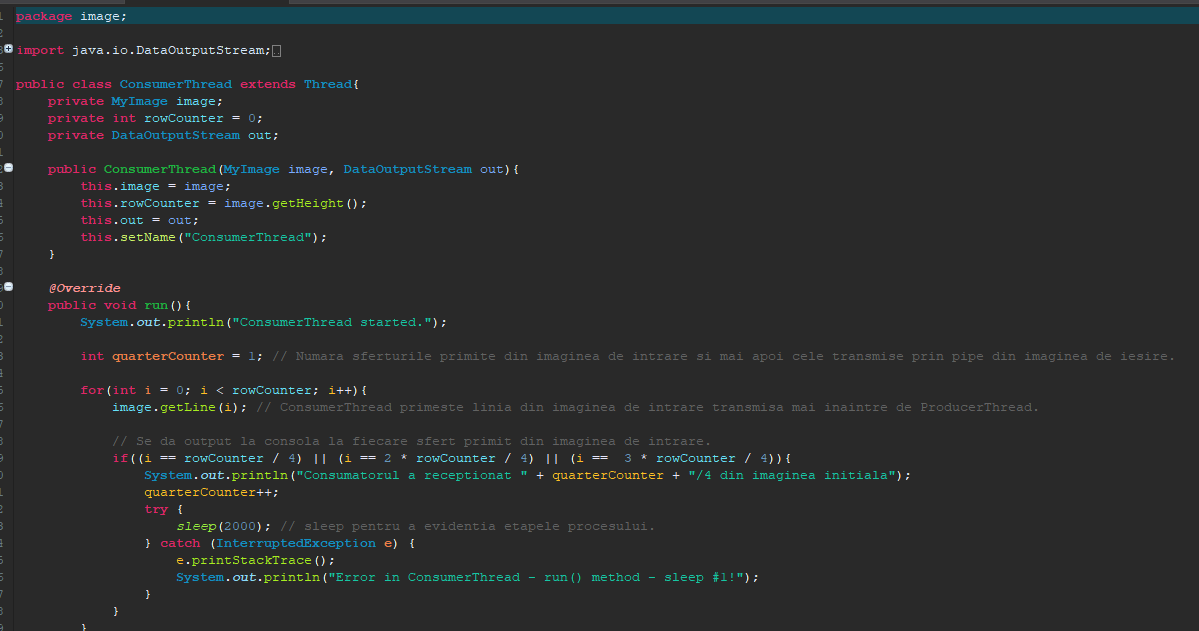
Clasa de interes este MyImage.Aceasta clasa este folosita ca un buffer unde salvata imaginea iniatiala care mai apoi este procesata si suprascrisa de catre imaginea finala.



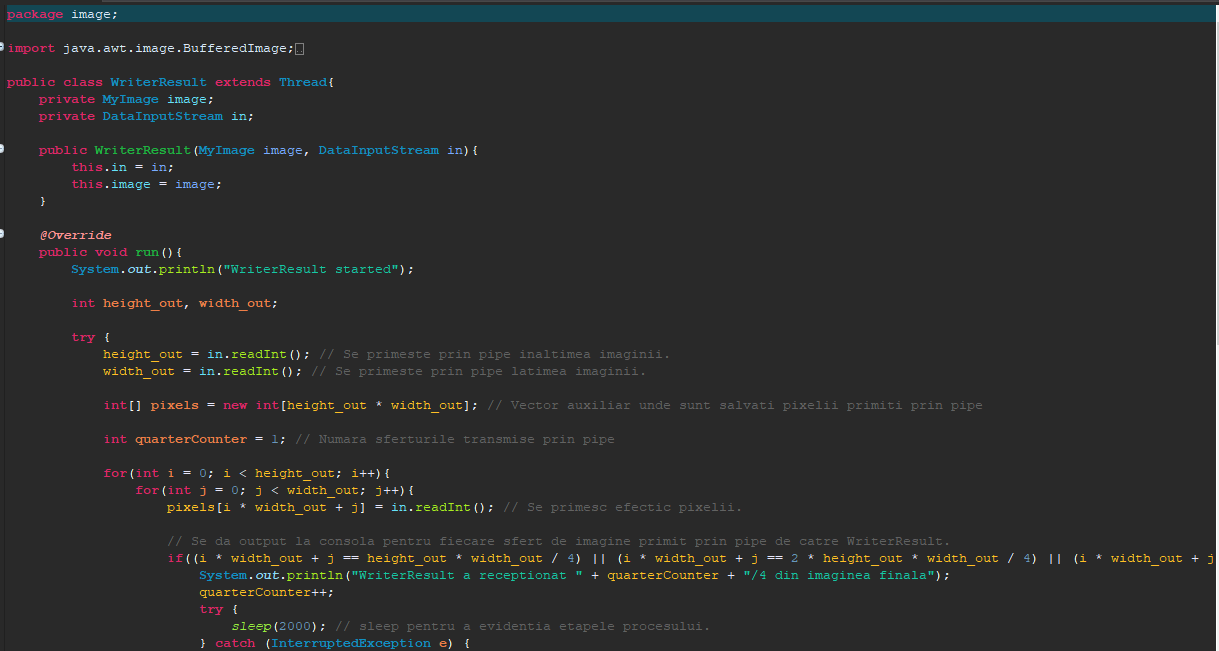
Clasele responsabile pentru citirea,procesarea si scrierea imaginii sunt : ProducerThread,ConsumerThread si WriterResult.

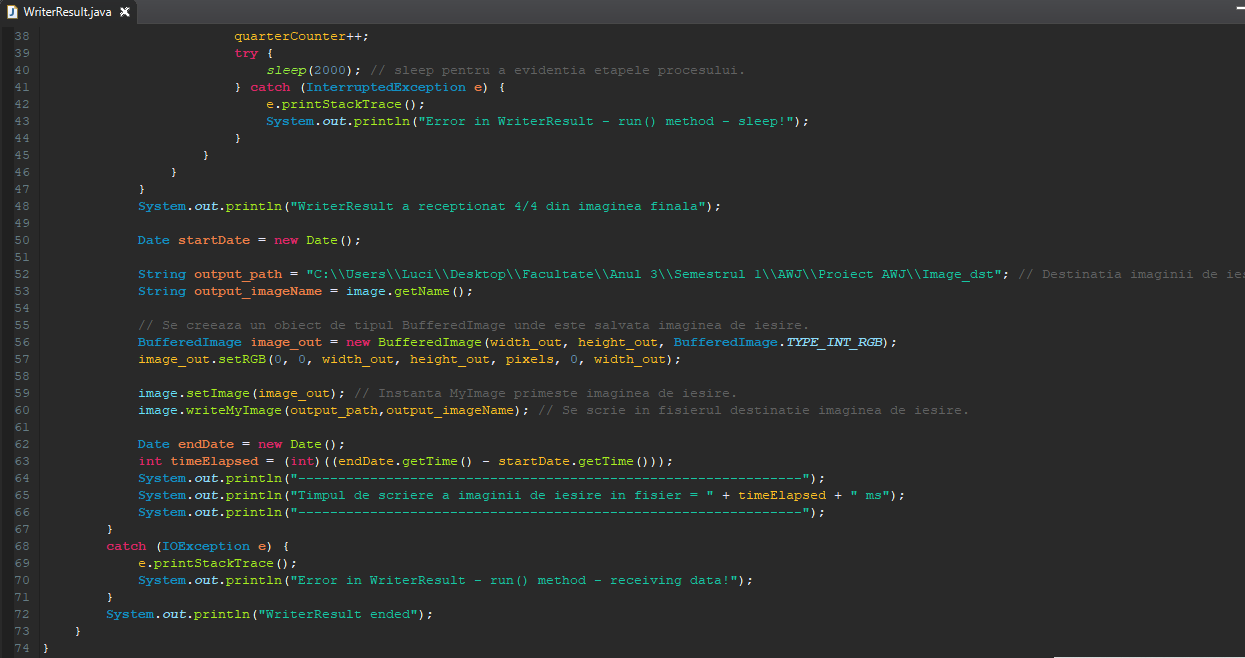


Clasa ProducerThread



Clasa ConsumerThread





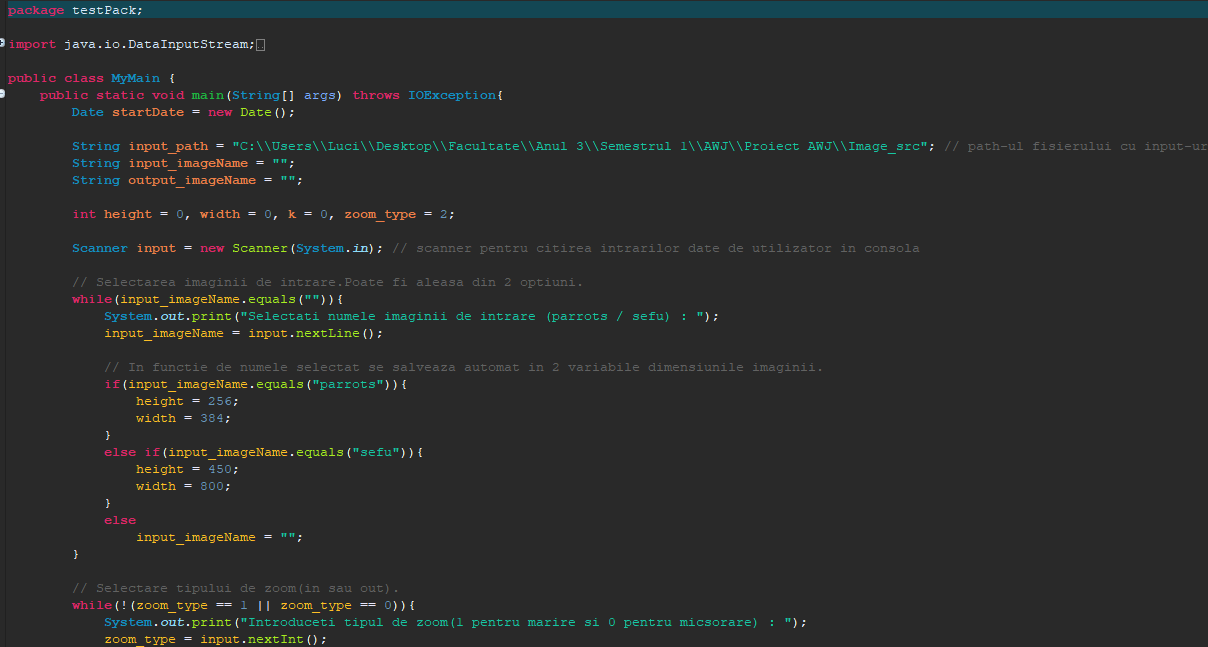
Clasa WriterResult

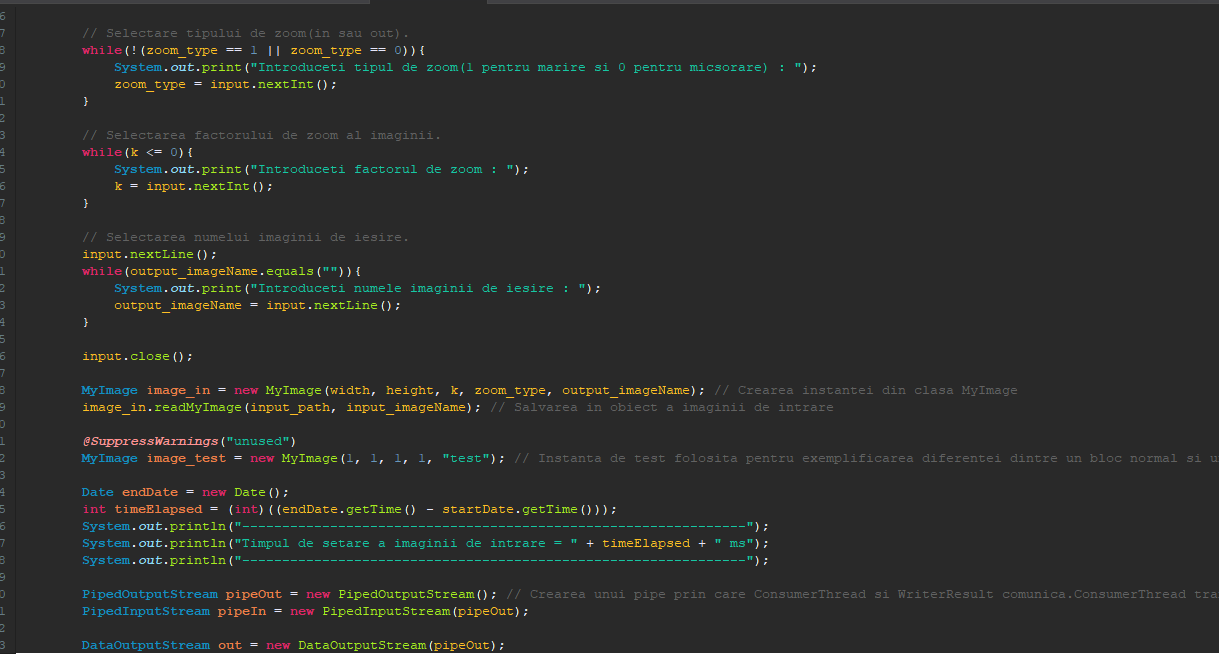
**Functionalitate**

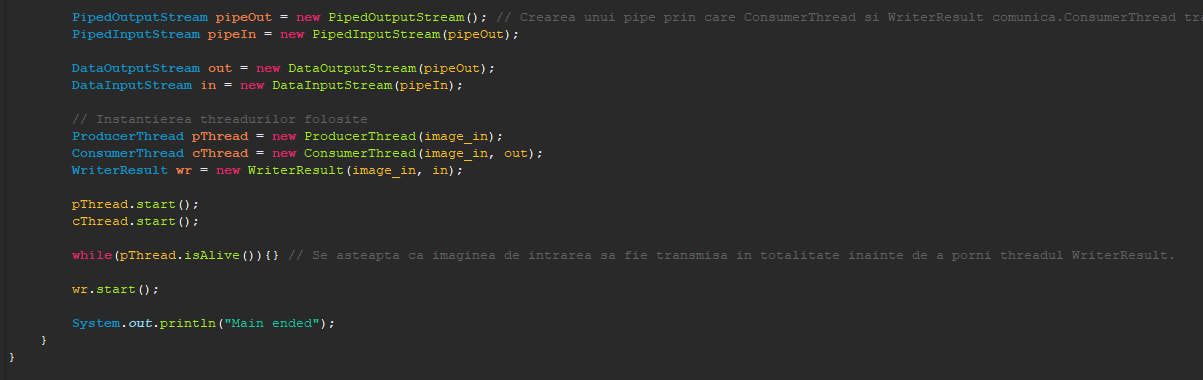
Toate cele 3 clase impart accesul la bufferul MyImage.Deoarece ProducerThread si ConsumerThread scriu(respectiv citesc) alternativ informatii din buffer, metodele de scriere(putLine()) si citire(getLine()) din clasa BufferImage sunt sincronizate,astfel asigurand ca ordinea de acces la buffer este cea dorita(ProducerThread scrie o linie de pixeli in imagine si apoi asteapta ca ConsumerThread sa o citeasca.ConsumerThread asteapta semnalul available care spune ca exista o noua linie de citit in buffer).

Pachetul **testPack**:

Pachetul contine o singura clasa,mai exact clasa de test denumita MyMain.







**Functionalitate**

Se seteaza path-ul fisierului de intrare.Apoi se asteapta input-ul utilizatorului cu privire la imaginea care va fi modicata (parrots sau sefu), tipul de zoom(1 pentru marire si 0 pentru micsorare), factorul de zoom(k), si numele imaginii finale care va fi scrisa in fisierul de iesire.

Dupa setarea intrarilor se copiaza in buffer imaginea de intrare,se creaza pipe-ul prin care ConsumerThread si WriterResult vor comunica si se instantiaza 3 threaduri : pThread(de tipul ProducerThread), cThread(de tipul ConsumerThread) si wr(de tipul WriterResult).

Threadurile pThread si cThread pornesc in acelasi timp,iar wr asteapta terminarea threadului pThread(care marcheaza incheierea transmiterii imaginii de intrare) pentru a porni.

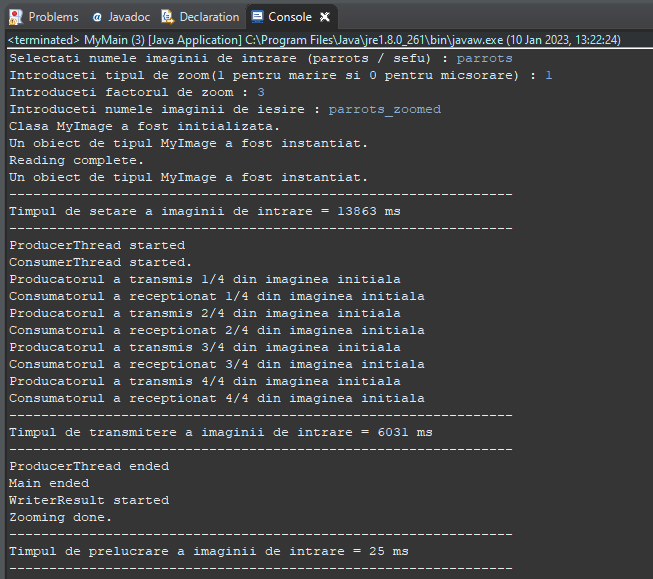
Apoi threadul wr va astepta procesarea imaginii de intrare in cThread si transmiterea primelor informatii prin pipe(mai exact dimensiunile imaginii : inaltimea si latimea),mai apoi fiind trimisi pixelii imaginii de iesire.

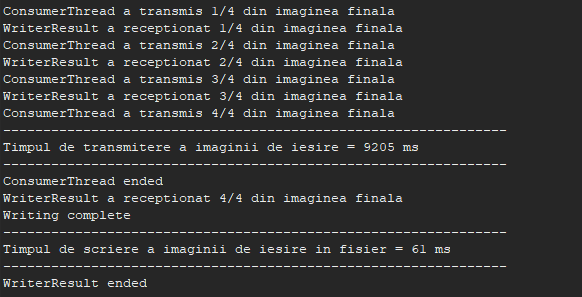
Dupa ce imaginea de iesire a fost transmisa in totalitate prin pipe aceasta suprascrie imaginea de intrare din bufferul MyImage si se apeleaza metoda writeMyImage pentru a scrie imaginea in fisierul de iesire.

De asemenea fiecare dintre urmatoarele procese este cronometrat(in milisecunde) : scrierea intrarilor de catre utilizator, transmiterea imaginii de intrare, prelucrarea imaginii, transmiterea prin pipe a imaginii de iesire, scriere imaginii in fisierul de iesire.

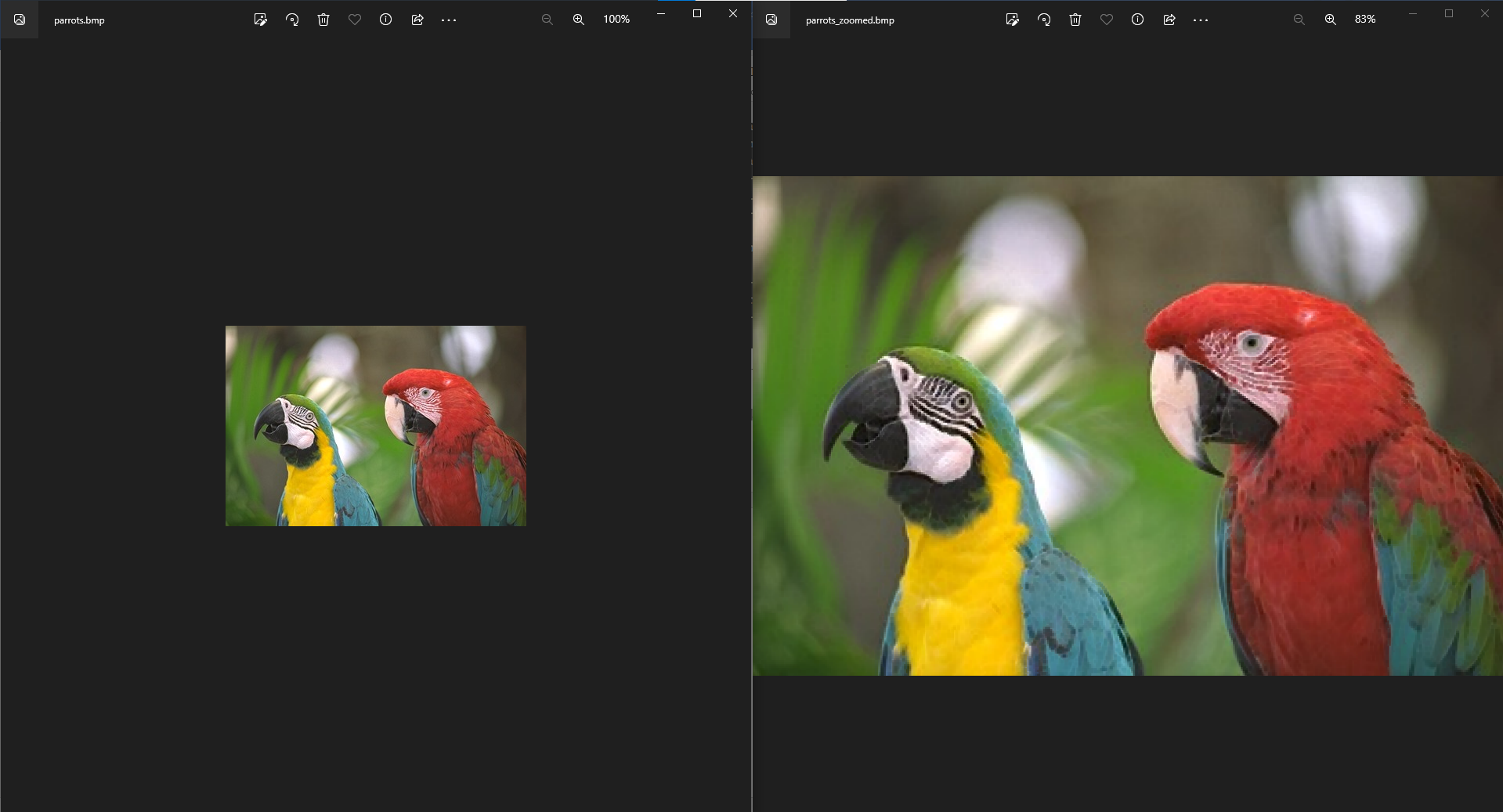
# 4.Exemplificarea functionalitatii programului

## 4.1 Marirea imaginii

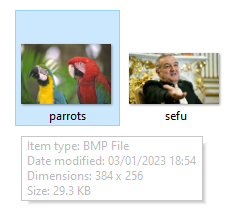
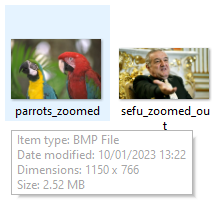




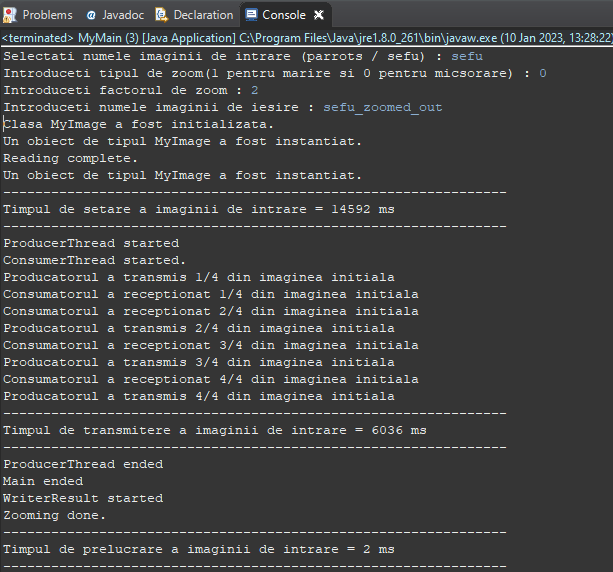
Consola dupa executarea programului

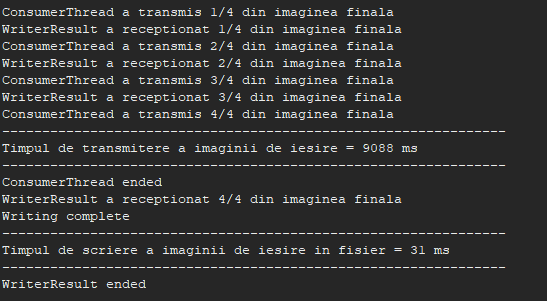


Imaginea de intrare Imaginea de iesire

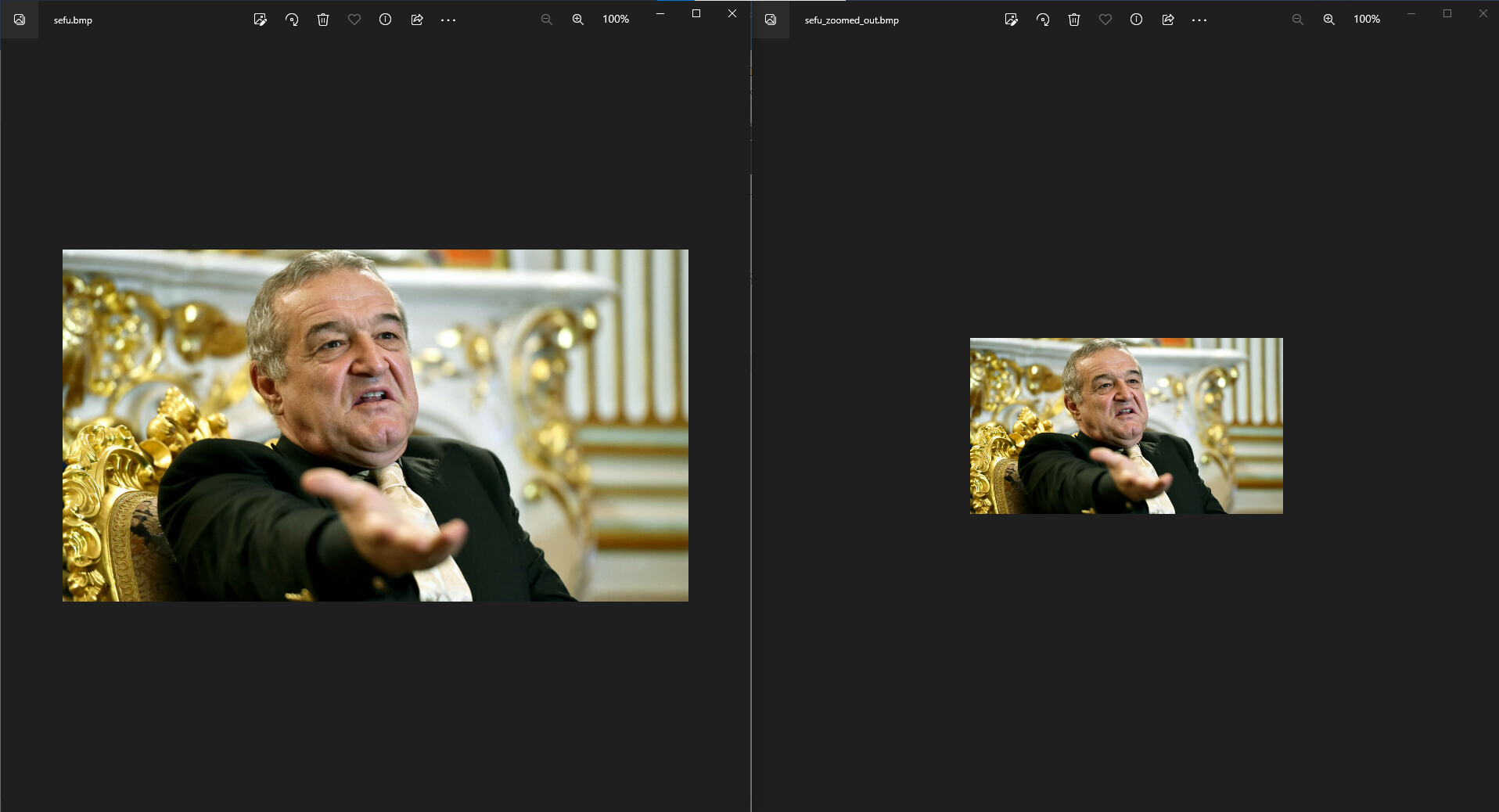


## 4.2 Micsorarea imaginii





Consola dupa executarea programului



Imaginea de intrare Imaginea de iesire

