Tema14:Rotate Image(90,180,270)

1. *Introducere:*

Pentru această temă am inclus în totalitate conceptele POO – încapsulare, moștenire, polimorfism, abstractizare; am învățat să lucrez cu fișiere (File) și cu varargs, cu imagini (Buffered Image și ImageIO), dar si cu thread-uri aplicatia fiind una de tip multithreading(asingnata pe principiul Producator-Consumator), cunostiinte acumulate la curs si la laborator.

2. *Descrierea aplicației*:

Partea teoretică:

Schelet proiect:

- o interfață: RotationAngle;
- o clasă ce implementează interfața: ImageToRotate;
- o clasă abstractă: FileOperations;
- o clasă concretă ce extinde clasa abstractă: Image;
- o clasa ce simuleaza thread-ul care se ocupa cu „producerea" a ¼ din informatia procesului de citire a imaginii:Producer;
- o clasa ce simuleaza „consumarea" informatiei produse de catre primul thread:Consumer;
- o clasa ce permite comunicarea celor doua fire de executie:Buffer;
- o clasă ce moștenește clasa Image și conține metoda main: Test;
- pachetele: java.util.Scanner , java.imageio.ImageIO, java.awt.image.BufferedImage, java.io.IOException, java.io.File;
- varargs pentru grade (numărul de grade cu care trebuie rotita imaginea);

Descrierea implementării:

Interfața *RotationAngle am folosit-o* împreună cu clasa care o implementează,*ImageToRotate*, pentru a crea metodele de rotire a imaginii: *rotates_with_90( ), rotates_with_180( )* și *rotates_with_270( ).*

Clasa abstractă *FileOperations* , care este moștenită de clasa *Image*, are ca scop implementarea metodei de rotatie și scrierea fișierului destinație .

Clasa *Image* este și ea moștenită de către clasa *Test*, unde se află metoda *main( ),* folosită pentru citirea datelor de la tastatură și de a apela metodele moștenite din *Image*.

Descrierea modulelor:

Pentru metodele *rotates_with _90( ), rotates_with_180( )* și *rotates_with_270( )* am creat o nouă imagine numită *rotated_img,*cu ajutorul constructorului *BufferedImage()* și *setRGB( )* din pachetul Buffered Image pentru a-i asigna valorile potrivite din imaginea originală.

Metoda *writeFile( )* are ca parametru locatia fișierului destinație sub forma unui String, acestea returnând fișierul de iesire propriu-zis.In mod asemenator se intampla si cu metoda clasei Image *readFile()* (care se ocupa de citirea imaginii utilizand thread-uri) ce primeste ca parametru locatia fisierului sursa.

Pentru metoda *rotateImage( )* am folosit varargs, pentru a permite introducerea de la tastatură a numarului de rotații cu care dorim sa rotim imaginea.Variabila *var* imi permite sa calculez unghiul final de rotire și de a apela metoda de rotire corespunzatoare.

În clasa Test ce include metoda main(),efectuam citirea datelor de la tastatură (locația fișierului sursă, locația fișierului destinație, numele fișierului destinație, numărul de rotații, gradele cu care dorim să rotim imaginea ,apelarea funcțiilor writeFile( ) ,readFile() și rotateImage( ), dar și analiza duratei de execuție a fiecărei etape.

Pentru implemetarea principiului Producator-Consumator m-am folosit de o clasa ajutatoare numita *Buffer* care imi retine informatia produsa de Producator si consumata de Consumator in felul urmator:imaginea(in format bytes) este citita pe bucati;dupa ce se citeste ¼ din informatia citirii sursei thread-ul 1 intra in Not Runnable si asteapta ca cel de-al doilea thread „sa se trezeasca” si sa consume informatia furnizata de Producator, urmand ca acesta sa instiinteze(notifyALL()) Producatorul ca acesta si-a indeplinit functia dupa care se reia procesul de citire a urmatoarei parti din informatie.Ambele clase(Producer si Consumer) mostenesc clasa Thread.

2. *Evaluare performanțe*

Am folosit metoda java.lang.System.currentTimeMillis( ) pentru evaluarea timpului de executie al etapelor prin numărul de milisecunde.

```
"Etapa de citire a informatiei despre imagine dureaza: " + (stop_read - start_read) + " milisecunde");
"Etapa de citire a fisierului sursa dureaza: " + (stop_read_source - start_read_source) + " milisecunde");
"Etapa de procesare a imaginii dureaza: " + (stop_rotation - start_rotation) + " milisecunde");
"Etapa de scriere a fisierului destinatie dureaza: " + (stop_destination - start_destination) + " milisecunde");
```

```
Etapa de citire a informatiei despre imagine dureaza: 35890 milisecunde
Etapa de citire a fisierului sursa dureaza: 2222 milisecunde
Etapa de procesare a imaginii dureaza: 118 milisecunde
Etapa de scriere a fisierului destinatie dureaza: 46 milisecunde
```

4.Concluzii

Observam ca citirea datelor durează cel mai mult, în funcție de cât de rapid sunt introduse datele de catre utilizator de la tastatură. Restul etapelor au un timp scurt de executie.

5.Bibliografie:

https://www.codespeedy.com/how-to-convert-image-to-byte-array-in-java/
https://stackoverflow.com/questions/12705385/how-to-convert-a-byte-to-a-bufferedimage-in-java
https://www.programiz.com/java-programming/inputstream
https://www.geeksforgeeks.org/variable-arguments-varargs-in-java/
https://javaconceptoftheday.com/read-and-write-images-in-java/
https://stackoverflow.com/questions/9707938/calculating-time-difference-in-milliseconds