Proiect AWJ 2022

Tema 14: Rotirea unei imagini (90, 180, 270)

Rădoi Constantin-Iulian
Grupa 331AAb

1. Introducere

Pentru această temă am folosit cunoștiințele acumulate la curs şi la laborator, dar şi câteva surse menționate în secţia Bibliografie. Am inclus în totalitate conceptele POO – încapsulare, moștenire, polimorfism, abstractizare; am învățat să lucrez cu fișiere (File), varargs, dar și cu imagini (Buffered Image și ImageIO) și Thread-uri (Consumer, Producer, Buffer).

2. Descrierea aplicației

Partea teoretică

Ce am folosit pentru acest proiect:

- o interfață: Rotation
- o clasă ce implementează interfața: RotateImage
- o clasă abstractă: ReadWriteImage
- o clasă concretă ce extinde clasa abstractă: Image
- o clasă ce moștenește clasa Image și conține metoda main: Execute
- pachetele:
    - java.io.File
    - java.io.IOException
    - java.imageio.ImageIO
    - java.awt.image.BufferedImage
    - java.util.Scanner,
- varargs pentru degrees (numărul de grade cu care rotim imaginea)
- clase Consumer, Producer, Buffer pentru lucrul cu threaduri

Descrierea implementării

Clasa principala este Execute, clasa ce moștenește clasa Image.
Pașii sunt următorii:

- Se inițializează Scanner-ul de imagine (Java.utils.Scanner) pentru a lua input
- Se verifica existenta args-urilor. Daca exista, se va adauga in newName args[0].
- Se citesc de la tastatura, în ordine, numele imaginii citite, locatia de scriere, noul nume al imaginii rotite (in caz ca nu a fost dat prin args), numarul de rotatii aplicate, gradele rotatiilor)
- Se initializeaza Buffer-ul, Consumer-ul si Producer-ul - clase ce mostenesc ThreadClass, ce mosteneste la randul ei Thread
- In producer se citeste imaginea, si se trimite pe Buffer
- Imaginea este rotita prin metoda rotate() a clasei Image, ce mosteneste ReadWriteImage(clasa abstracta). rotate() instantiaza un obiect RotateImage ce contine methodele rotate90(), rotate270(), rotate180().

- Pentru metodele rotate90( ), rotate180( ) și rotate270( ) am creat o nouă imagine numită *rotated*, alocându-i măsurile potrivite cu constructorul BufferedImage() și setRGB( ) din pachetul Buffered Image pentru a-i asigna valorile potrivite din imaginea originală.
- Metodele *readImage( )* și *writeImage( )* primesc ca parametri locațiile fișierelor sursă și destinație sub forma unui String, acestea returnând fișierele propriu-zise: File input și File output.
- Vom folosi methoda writeImage() pentru a scrie imaginea rotita la locația dată.
- Vom afișa timpii de rulare

3. **Evaluare performanțe**

Folosind metoda java.lang.System.currentTimeMillis( ) am evaluat durata etapelor de execuție a programului, prin numărul de milisecunde dintre start și stop etapă.

După cum se observă și în program, etapele de execuție evaluate sunt:

```
================================================================================
Proces terminat.
Etapa de citire a datelor dureaza: 16117 milisecunde (timp total insertie date de catre utilizator)
Etapa de citire a fisierului dureaza: 58 milisecunde
Etapa de procesare a imaginii dureaza: 23 milisecunde
Etapa de scriere a fisierului destinatie dureaza: 11 milisecunde
======= DONE =======
================================================================================
```

4. **Concluzii cu privire la performanță**

Etapele ce nu țin de utilizator durează, în medie , 90 de milisecunde.

5. **Bibliografie**

https://stackoverflow.com/questions/11951646/setrgb-in-java
https://stackoverflow.com/questions/9707938/calculating-time-difference-in-milliseconds
https://www.tutorialspoint.com/how-many-ways-can-we-read-data-from-the-keyboard-in-java
https://docs.oracle.com/javase/7/docs/api/java/io/File.html
https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html
https://stackoverflow.com/questions/4871051/how-to-get-the-current-working-directory-in-java