Documentație Proiect AWJ

Rotate Image (90, 180, 270)

Stan Razvan-Octavian

332AA

Introducere

Prelucrarea imaginilor cât și analiza lor, a apărut din nevoia de a înlocui utilizatorul uman cu o mașină. Totuși, această idee a mers mai departe decât simpla înlocuire a observatorului

uman, deoarece au apărut soluții inovatoare pentru probleme cu care acesta nu mai fusese confruntat – ca in cazul imaginilor non-vizibile precum imaginile acustice, ultrasonore sau radar. Prin prelucrarea digitală a imaginilor se înțelege prelucrarea pe un calculator digital a unor date bidimensionale (imagini).

Acest proiect își propune transformarea unei imagini (primită ca input de la tastatura prin argument în linia de comandă) și anume rotirea acesteia cu 90, 180 sau 270 de grade în sens orar.

Proiectul este reprezentat de o aplicație Java care realizează o procesare a unei imagini. Aplicația nu are interfață grafică. Aceasta a fost realizată folosind Java 8, Eclipse Mars 2 și Windows x64.

Descrierea Aplicației

Aplicația are rolul de a roti o imagine primită ca input cu o valoare dată de la tastatură (90, 180, 270 de grade) în sens orar și de a o salva la o cale dată. Am folosit căi relative la folderul sursă (src).

Fișierul sursă este o imagine în format BMP RGB. Pentru procesare am folosit doar algoritmi și secvențe de cod low-level.

Aplicația își propune să respecte toate conceptele POO cât și principiile SOLID. Am restricționat accesul claselor la date declarându-le pe cele din urmă ‚private' și am generat setteri și getteri publici acolo unde a fost cazul (principiul de încapsulare). De asemenea m-am folosit și de alte principii precum moștenire, polimorfism sau abstractizare.

Coding style-ul urmat este cel de la Oracle cu spații înainte și după operatori. Am omis Javadoc deoarece metodele erau destul de explicite. În cadrul moștenirii am folosit tag-ul @Override.

Aplicația conține operații de lucru cu fișiere, fișierul input care este o imagine pe care vreau să o rotesc și fișierul output care este imaginea rotită. Ambele se află inițial în fișierul sursă însă pot trimite o cale absolută sau relativă către orice fișier din memorie.

Aplicația primește căile către fișiere, inițial în linie de comandă, însă acestea pot fi ulterior asignate folosind comanda setFiles pathInput pathOutput. Aplicața primeste comenzile de la tastatura direct în terminalul de run. De asemenea rotirea se face prin asignare de la tastatura a numarului de grade de rotire dorit.

Aplicația este împarțită în mai multe clase, scopul fiind multimodularizarea și încercarea de a face fiecare clasă să realizeze un singur obiectiv specific, clar.

Partea teoretică

Pentru realizarea rotirii m-am folosit de clasa AffineTransform care reprezintă o transformare afină 2D care realizează o mapare liniară de la coordonatele 2D la alte coordonate 2D care păstrează „rectitudinea” și „paralelitatea” liniilor. Transformările afine pot fi construite folosind secvențe de translatări, scalări, întoarceri (flip), rotații și shears.

În unele variante ale metodelor de rotație din clasa AffineTransform, un argument de precizie dublă specifică unghiul de rotație în radiani. Aceste metode au o manipulare specială pentru rotații de aproximativ 90 de grade (inclusiv multipli precum 180, 270 și 360 de grade), astfel încât cazul obișnuit de rotație a cadranului este tratat mai eficient. Această manipulare specială poate face ca unghiurile foarte apropiate de multiplii de 90 de grade să fie tratați ca și cum ar fi exact multiplii de 90 de grade. Pentru multipli mici de 90 de grade gama de unghiuri tratate ca o rotație a cadranului este de aproximativ 0,00000121 grade lățime. Această secțiune explică de ce este necesară o astfel de îngrijire specială și cum este implementată.

Deoarece 90 de grade este reprezentat ca PI / 2 în radiani și din moment ce PI este un număr transcendental (și, prin urmare, irațional), nu este posibil să se reprezinte exact un multiplu de 90 de grade ca o valoare exactă dublă de precizie măsurată în radiani. Ca urmare, este teoretic imposibil să se descrie rotațiile cadranului (90, 180, 270 sau 360 de grade) folosind aceste valori. Valorile cu virgulă dublă de precizie pot ajunge foarte aproape de multipli diferiți de zero de PI / 2, dar niciodată suficient de aproape pentru ca sinusul sau cosinusul să fie exact 0,0, 1,0 sau -1,0. Implementările Math.sin () și Math.cos () în mod corespunzător nu returnează niciodată 0,0 pentru niciun alt caz decât Math.sin (0,0). Cu toate acestea, aceleași implementări returnează exact 1,0 și -1,0 pentru o anumită gamă de numere în jurul fiecărui multiplu de 90 de grade, deoarece răspunsul corect este atât de apropiat de 1,0 sau -1,0 încât semnificația cu dublă precizie nu poate reprezenta diferența la fel de exact cât poate pentru numere care sunt aproape de 0,0.

Descrierea implementării

Aplicația este implementată în Java 8, folosind IDE-ul IntelliJ Idea pe un Windows 10 x64. Exportarea s-a facut ca jar pentru Eclipse Mars 2 și am verificat funcționalitatea acestui jar.

Descrierea arhitecturală – și funcțională a aplicației

Aplicația pornește imediat după apăsarea butonului run. Atenție, trebuie setate în IDE la Run -> Run Configurations -> argumentele din linia de comandă. In arhiva trimisă fișierul de

intrare se află în același folder cu folderul src deci am pus ca argumente input.bmp și output.bmp. Fișierul output.bmp nu trebuie neapărat să existe, el va fi creat sau în caz că există va fi suprascris.

După rulare ne apare meniul cu opțiuni. Acesta a fost realizat în clasa Main rulând continuu până la apariția comenzii stop. Citirea se face cu clasa Reader care implementează interfața ReaderInterface care are o singură metoda, readLine. Se citește fiecare comandă linie cu linie (citește cu BufferedReader până la întalnirea caracterului ‚n'). Pentru că citirea comenzilor se face o singură dată am optat pentru ultilizarea Design Pattern-ului Singleton în clasa Reader. Asta înseamnă că am setat constructorul ca private și am o metodă getInstance care returnează o singură instanță a clasei Reader.

```
Executia programului a inceput.
Introduceti comanda dorita:
Pentru start: 'start'
Pentru setarea sursei si a destinatiei: 'setFiles inputFile outputFile'
Pentru rotire: 'rotate degrees(90, 180, 270)'
Pentru oprire executie: 'stop'
```

În acest moment avem încarcate doar path-urile către fișiere. Pentru a începe primirea în clasa Producer a informației imaginii trebuie tastată comanda start.

Odată apăsată comanda start pornește execuția Threadului Producer care citește imaginea ca un vector de bytes și începe trimiterea informației către Consumer (care este în alt Thread). Consumer citește informația primită prin PipedInputStream. Consumer citește bytes până când primește -1 ca rezultat al comenzii in.read() unde in reprezintă DataInputStream-ul. Dupa ce aceste două Thread-uri își termină execuția intră în not Runnable. Înainte de asta consumer trimite șirul de bytes primit către instanța singleton a copilului clasei Transform, Rotate 90.

```
System.out.println("Incepe consumer...");
try {
    Thread.sleep( millis: 1000);
    int k = 0;
    for (int b = 0; ((b = in.read()) ≥ 0);) {
        fileContent[k] = (byte) b;
        k++;
    }

    fileContent2 = new byte[k];
    for (int i = 0; i < k; i++) {
        fileContent2[i] = fileContent[i];
    }
    System.out.println("Consumer primeste "+ fileContent2.length + " bytes.");
} catch (IOException e) {
    System.out.println("S-a terminat citirea");
} catch (InterruptedException e) {
    e.printStackTrace();
}
Rotate90 transform = Rotate90.getInstance(fileContent2);
System.out.println("Consumer a format imaginea.");
```

Acum imaginea este încărcată în aplicație. Se așteaptă comanda de rotire și anume rotate nGrade (90, 180, 270).

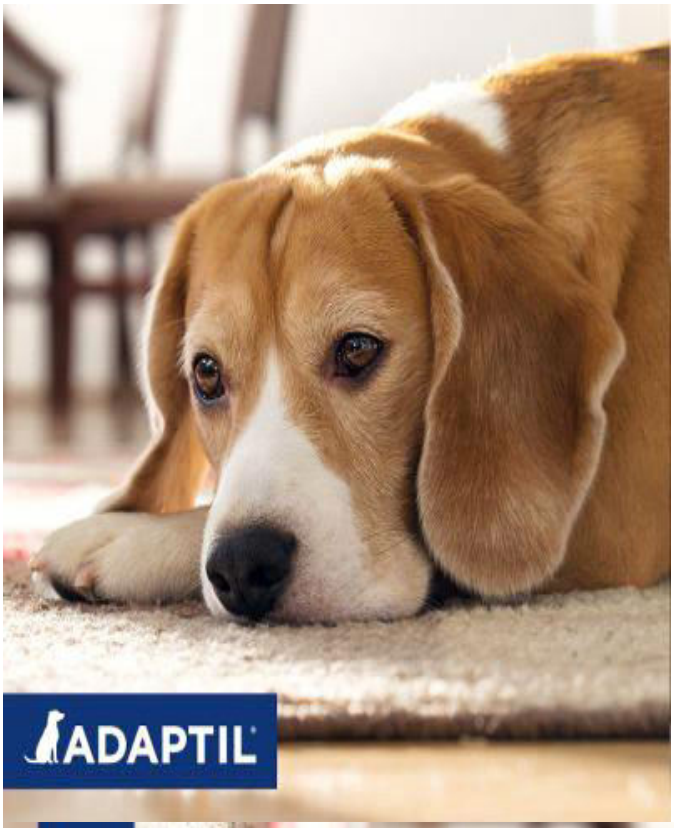
La primirea comenzii de rotire în instanța clasei Rotate90 începe transformarea șirului de bytes într-o BufferedImage. Aici este foarte importantă integritatea datelor transmise pentru că dacă lipsesc date, sau pe parcursul transmisiei informației a fost transmis si zgomot adică niște bytes care nu făceau parte din imaginea inițială atunci vom primi eroare și imaginea nu va putea fi formată corespunzător. Programul nostru primeste ca input poze cu dimensiunea de până la 100kB. În cazul în care dorim să folosim poze și mai mari putem să modificăm în cod la declararea variabilei fileContent.

```
ByteArrayInputStream bis = new ByteArrayInputStream(super.imageArr);
BufferedImage source = null;
try {
    source = ImageIO.read(bis);
} catch (IOException e) {
    e.printStackTrace();
}
```

Apoi se apelează, în funcție de valoarea dată la input, una dintre metodele de construire a transformării: rotateClockwise180, rotateClockwise90 sau rotateCounterClockwise90 (care este echivalentul rotirii cu 270 de grade în sens orar). Pentru rotire facem rotirea la centru facem transalție la origine. Rotația cu 180 de grade este echivalenta cu rotirea de doua ori cu 90 de grade.

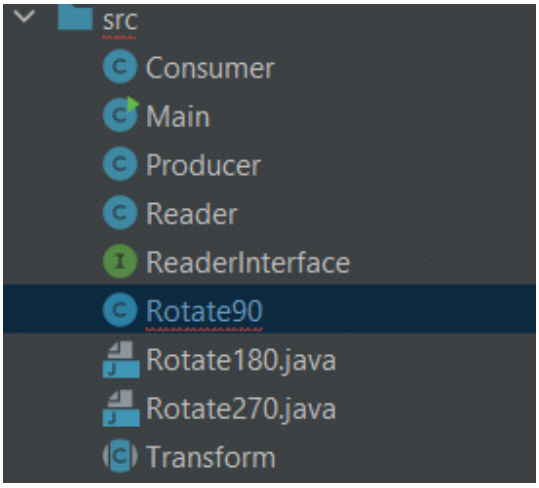
In continuare după rotire se face salvarea imaginii în fișierul setat. Dacă acesta nu există el va fi creat. Poate fi deschis si utilizat imediat.

Exemplu input si output pentru rotire cu 270 de grade. Observăm că nu s-a deteriorat calitatea imaginii.



Module

Am folosit următoarele fișiere:



De menționat este faptul că Transform este o clasă abstractă care este moștenită de clasa concretă Rotate90 și îi implementează metoda abstractă. Functionalitatea acestora a fost explicată în detaliu mai sus.

Concluzii

Operația de rotire este foarte folosită în cadrul procesării imaginilor. Rotirea unei imagini are loc printr-un proces în care se construiește o transformare care constă în rotire și translatare. Aplicația construită are rolul de a citi o imagine primită ca input și de a o roti cu un anumit unghi. Aplicația a avut ca scop urmarirea principiilor Java si SOLID.

Referinte

- https://www.quickprogrammingtips.com/, How to rotate an image using affine transform
- https://stackoverflow.com/, Java rotating image
- acs.curs.pub.ro, Cursul de AWJ
- Javadoc din clasa AffineTransform