FACULTY OF
AUTOMATIC CONTROL
AND COMPUTERS
University POLITEHNICA of Bucharest

Image Brightness Modification

Vasilache Dana-Maria 332AA

Februarie 2021

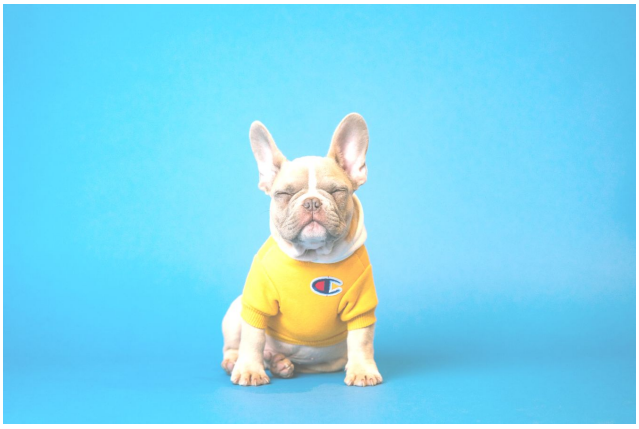
Cuprins

Introducere + Descrierea aplicației cerute

Am realizat o aplicație prin care se poate modifica (din cod) luminozitatea unei imagini în format .bmp. Pentru a realiza acest lucru, în linia de comandă introduc calea imaginii pe care doresc sa o modific. Printr-o metodă numită ‘transform()’ se face convertirea propriu-zisă. După ce aceasta s-a efectuat, o salvez oriunde doresc, în formatul ‘nume.bmp’, introducând în linia de comandă calea. Pot regla factorul de luminozitate dorit direct din cod.

Pentru fiecare pas afişez timpii necesari execuţiei (citirea din fişier, conversia, scrierea în fişier)

Imagine iniţială

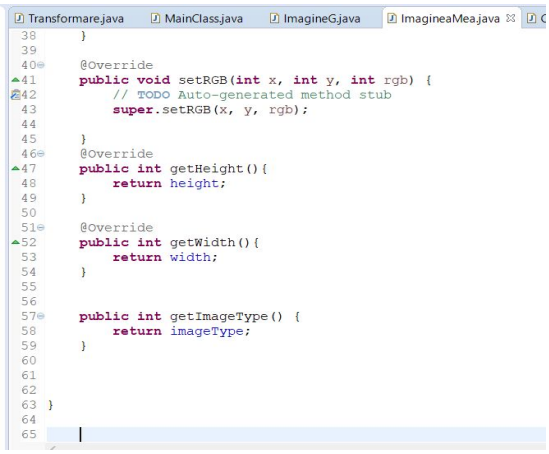


Imagine rezultată

Partea teoretică

Din punct de vedere POO, aplicaţia conţine toate principiile:

- Încapsulare: procesul prin care ţinem datele şi funcţiile separate de exterior



Exemplu: Metodele din clasa ‘ImagineaMea’ le-am folosit şi în clasa ‘Transformare’

- Moştenire - 3 niveluri

```
public abstract class ClasaAbs1
{

public abstract class ClasaAbs2 extends ClasaAbs1 {


public class ClasaAbs3 extends ClasaAbs2 {
```

- Abstractizare: oferă o definire precisă a graniţelor conceptuale din perspectiva unui privitor extern. Am implementat metodete din Interfaţa într-o altă clasă.

```
1 public interface Interfata {
2
3     public int getWidth();
4     public int getHeight();
5     public void setRGB(int x, int y, int rgb);
6
7 }
8
```

De aici am îndeplinit şi cerinţa: existenţa unei interfeţe cu o clasa care o implementează.

Transformare.java | ClasaAbs1.java | ClasaAbs2.java | ImagineaMea.java | MainClass.java | Consumer.java | Producer.java | Interfata.java

```java
import java.awt.image.BufferedImage;

public class ImagineaMea extends BufferedImage implements Interfata {

    int width;
    int height;
    int imageType;

    public ImagineaMea(ColorModel cm, WritableRaster raster, boolean isRasterPremultiplied, Hashtable<?, ?> propertie
        super(cm, raster, isRasterPremultiplied, properties);
    }

    public ImagineaMea(int width, int height, int imageType, IndexColorModel cm) {
        super(width, height, imageType, cm);
    }

    public ImagineaMea(int width, int height, int imageType) {
        super(width, height, imageType);
    }

    public void setImageType(int imageType) {
        this.imageType = imageType;
    }


    public void setWidth(int width) {
        this.width = width;
```

- Polimorfism - putem defini o metodă cu acelaşi nume în aceeaşi clasă, dar cu parametri diferiţi

```java
public ImagineaMea(ColorModel cm, WritableRaster raster, boolean isRasterPremultiplied, Hashtable<?, ?> properties) {
    super(cm, raster, isRasterPremultiplied, properties);
}

public ImagineaMea(int width, int height, int imageType, IndexColorModel cm) {
    super(width, height, imageType, cm);
}

public ImagineaMea(int width, int height, int imageType) {
    super(width, height, imageType);
}
```

Includerea varargs:

```java
@Override
// varargs -  numar variabil de argumente
public void fun(String str, int ...x)
{
    System.out.println("Sirul este : " + str);
    System.out.println("Numarul de argumente al sirului : "+ x.length);

    for (int i = 0; i < x.length; i++) {
        System.out.print(i + " ");
    }

    System.out.println();
}
```

Descrierea implementării

Codul respectă cerinţele temei.

Algoritmul pentru Image Brightness - modificarea luminozităţii în funcţie de ‘factorlumin’, pe care îl putem modifica direct din cod. Am lucrat cu fişiere, scrieri şi citiri din/în acestea.

```
        int factorlumin=100;

// Se obtine RGB-ul imaginii
        for (int y = 0; y < height; y++) {
            for (int x = 0; x < width; x++) {
                Color c=new Color(image.getRGB(x,y));
                //adding factor to rgb values
                int red=c.getRed()+factorlumin;
                int blue=c.getBlue()+factorlumin;
                int green=c.getGreen()+factorlumin;
                if (red >= 256) {
                 red = 255;
                }
                else if (red < 0) {
                red = 0;}
                if (green >= 256) {
                green = 255;
                }
                else if (green < 0) {
                green = 0;}
                if (blue >= 256) {
                blue = 255;
               }
                else if (blue < 0) {
                blue = 0;}
                image.setRGB(x, y,new Color(red,green,blue).getRGB());

            }
        }
```

Descrierea structurala architecturala si functionala a aplicatiei implementate

- Din punct de vedere a structurii arhitecturală

  Aplicaţia este formată din

  - 3 clase abstracte (clasa **ClassAbs1**, clasa **ClasaAbs2** (care extinde ClasaAbs1 şi care, totodată are număr variabil de argumente) şi **Clasa Abs3** (care extinde ClasaAbs2)
  - Clasa **‘Imaginea Mea**’, pentru care am realizat metode pentru a lua şi a seta lungimea şi lăţimea imaginii
  - Interfaţa numită **‘Interfaţă’**
  - Clasa **Transformare** - unde se realizează citirea din fişier, conversia şi scrierea în fişier
  - **Main**

- **Producer** şi **Consumer** - prin care am sincronizat thread-urile, cu ajutorul unui obiect din clasa Transformare

- Din punct de vedere funcţional
  - În momentul rulării, utilizatorul trebuie sa dea de la tastatură calea spre imaginea dorită, ca în exemplul: D:\DownloadD\Eclipse\VasilacheDana_332AA\model.bmp

```
Introduceti calea catre fisierul sursa SAU "exit" pentru a iesi
D:\DownloadD\Eclipse\VasilacheDana_332AA\model.bmp
```

  - Utilizatorul de asemenea, trebuie sa dea calea pentru scriere, ca în exemplul: D:\DownloadD\Eclipse\VasilacheDana_332AA\modelmodificat.bmp

```
Dati calea catre fisier destinatie
D:\DownloadD\Eclipse\VasilacheDana_332AA\modelmodificat.bmp
```

În imaginea de mai jos, este ceea ce îmi afişează în linia de comandă. Thread-urile sunt de asemenea sincronizate.

```
Introduceti calea catre fisierul sursa SAU "exit" pentru a iesi
D:\DownloadD\Eclipse\VasilacheDana_332AA\model.bmp
Citirea din fisier a durat 0.662 secunde
Transformarea imaginii a durat 1.19 secunde
Dati calea catre fisier destinatie
D:\DownloadD\Eclipse\VasilacheDana_332AA\wjewjeijw.bmp
Scrirea in fisier a durat 0.569 secunde
Introduceti calea catre fisierul sursa SAU "exit" pentru a iesi
S-a incarcat 0/4 din imagine.
Parte din imagine 0/4  procesata si adaugata cu succes!
S-a incarcat 1/4 din imagine.
Parte din imagine 1/4  procesata si adaugata cu succes!
S-a incarcat 2/4 din imagine.
Parte din imagine 2/4  procesata si adaugata cu succes!
S-a incarcat 3/4 din imagine.
Parte din imagine 3/4  procesata si adaugata cu succes!
S-a incarcat 4/4 din imagine.
Imaginea a fost citita in intregime
Parte din imagine 4/4  procesata si adaugata cu succes!
```

Descrierea modulelor

Clasa ‘Transformare’ realizează conversia, citirea din fişier, scrierea în fişier. Cu un factor de luminozitate care poate fi schimbat din cod, putem alege cât de mult să deschidem o fotografie.

Main- Printr-un scanner am făcut legătura cu citirea din consola, putând să dau calea pentru imaginea dorită, şi calea pentru scrierea imaginii, redenumind direct în consola noua imagine.

Consumer, Producer - Sunt sincronizate, având un obiect de tip Transformare. Sincronizarea am făcut-o în clasa Transformare

ClasaAbs1, ClasaAbs2, ClasaAbs3 - în care se definesc metode abstracte, creează un model minim de metode obligatorii care trebuie definite în sub-clase normale

Evaluare performanţe

```
Citirea din fisier a durat 0.675 secunde
Transformarea imaginii a durat 1.169 secunde

Scrirea in fisier a durat 0.577 secunde
```

Concluzii

Aplicaţia realizată în cadrul proiectului atinge obiectivele trasate în cerinţă şi a fost o experienţă utilă pentru a îmi dezvolta cunoştinţele despre limbajul Java, a înţelege mai bine conceptele simple, cât şi cele mai complexe. Am învăţat lucruri noi şi mi-am exersat vechile cunoştinţe în cadrul acestei aplicaţii.

Bibliografie

https://curs.upb.ro

https://www.tutorialspoint.com/java/java_multithreading.htm

https://www.udemy.com/

https://stackoverflow.com/questions/11163578/algorithm-to-modify-brightness-for-rgb-image

Documentatie cod sursa (cod + comentariile din cod)

ClasaAbs1

```
public abstract class ClasaAbs1 //clasa abstracta
{

    public void descriere()
    {
       System.out.println("Aceasta este clasa abstracta numarul 1- Primul nivel de mostenire");
    }

    abstract public boolean readImageFromFile(String file);  //metoda abstracta

    abstract public void fun(String str, int ...x); //metoda abstracta cu varargs
}
```

ClasaAbs2

```
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

import javax.imageio.ImageIO;

public abstract class ClasaAbs2 extends ClasaAbs1 {

    @Override
    public void descriere() //metoda descriere, nu returnam nimic, doar afisam mesajul
    {
       System.out.println("Aceasta este clasa abstracta numarul 2 - Al doilea nivel de mostenire.");
```

```
    }

    @Override
    public void fun(String str, int ...x) // clasa care contine numar
variabile de argumente (varargs)
      {
      System.out.println("Sirul este: " + str); //dam sigur
      System.out.println("Numarul de argumente al sirului: "+ x.length);

      for (int i = 0; i < x.length; i++) {
            System.out.print(i + " "); // afisam fiecare argument
      }
      System.out.println();
      }

    @Override //subclass or child class to provide a specific
implementation of a method that is already provided by one of its
super-classes or parent classes.
    public boolean readImageFromFile(String file) { //metoda
       BufferedImage image = null; //image de tip Buffered image, fara
valoare (null)
       try (FileInputStream stream = new FileInputStream(file)) {
             image = ImageIO.read(stream); // citim din fisier in variabila
image de tip BuffereadImage
       } catch (FileNotFoundException e) {
             System.out.println("Nu am putut gasi fisierul " + file);//
tratam exceptiile in care nu se gaseste fisierul / apare o eroare in fisier
             return false;
       } catch (IOException e) {
             System.out.println("A aparut o eroare in timpul citirii din
fisier");
             return false;
       }
       return true;
    }

    abstract public void writeImageToFile(String file); // metoda de
scriere a imaginii in fisier
}
```

ClasaAbs3

```
public class ClasaAbs3 extends ClasaAbs2 {

    @Override
    public void writeImageToFile(String file) {
      // TODO Auto-generated method stub
    }

    public void descriere() { // metode de descriere; al treilea nivel de
                                        // mostenire, extinde ClasaAbs2
       System.out.println("Aceasta este clasa abstracta numarul 3 - Al treilea nivel de mostenire."); //afisam mesajul
    }
}
```

Consumer

```
class Consumer extends Thread { //Thread Consumer - am urmat structura din curs
    private Transformare obj; // Obiect de tip Transformare

    public Consumer(Transformare b) {
      obj = b;
    }

    @Override
    public void run() {
      int value = 0;
      for (int i = 0; i < 5; i++) { // i-ul merge de la 0-4, pentru a imparti procesarea in 4 parti.
            value = obj.get();
            System.out.println("Parte din imagine " + value + "/4 procesata si adaugata cu succes!");//afisam mesjaul imediat dupa ce producerul pune valoarea sa
       }
```

```
    }

}
```

Producer

```
class Producer extends Thread { // Thread producer - am urmat
structura din curs

    private Transformare obj;

    public Producer(Transformare b) {
       obj = b;
    }

    @Override
    public void run() {
       for (int i = 0; i < 5; i++) {

            obj.put(i);
            System.out.println("S-a incarcat " + i + "/4 din
imagine."); // pentru fiecare parte, afisam mesaj
                  System.out.println("Imaginea a fost citita in
intregime");
            } // in momentul in care se proceseaza ultima parte,
afisam un mesaj corespunzator

            try {
                  sleep((int) (Math.random() * 100));
            } catch (InterruptedException e) {
            }

       }

    }
```

Interfata

```
public interface Interfata { // interfata

    public int getWidth();

    public int getHeight();

    public void setRGB(int x, int y, int rgb);// metoda pentru a seta
rgb-ul unei imagini, x , y = lungimea si latimea

}
```

ImagineaMea

```
import java.awt.image.BufferedImage;
import java.awt.image.ColorModel;
import java.awt.image.IndexColorModel;
import java.awt.image.WritableRaster;

import java.util.Hashtable;

public class ImagineaMea extends BufferedImage implements Interfata
{// clasa care contine proprietatile imaginii, cu metodele ei
aferente

    int width; // latimea
    int height; //lungimea
    int imageType; //tipul

    public ImagineaMea(ColorModel cm, WritableRaster raster, boolean
isRasterPremultiplied,
            Hashtable<?, ?> properties) {
       super(cm, raster, isRasterPremultiplied, properties);
    } //hastable - mapeaza cheie cu valaorea

    public ImagineaMea(int width, int height, int imageType,
IndexColorModel cm) {
```

```
    super(width, height, imageType, cm);
}

public ImagineaMea(int width, int height, int imageType) {
    super(width, height, imageType);
}

public void setImageType(int imageType) { //setam tipul imaginii
    this.imageType = imageType;
}

public void setWidth(int width) { // setam latimea
    this.width = width;
}

public void setHeight(int height) { //setam inaltime
    this.height = height;
}

@Override
public void setRGB(int x, int y, int rgb) { //setam rgb
    // TODO Auto-generated method stub
    super.setRGB(x, y, rgb);

}

@Override
public int getHeight() { // luam inaltimea de la imagine
    return height;
}

@Override
public int getWidth() { //luam latimea
    return width;
}

public int getImageType() {  //luam tipul imaginii
    return imageType;
```

```
    }

}
```

Transformare

```
import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

import javax.imageio.ImageIO;

public class Transformare { // clasa transformare - se citeste
imaginea din fisier, se face conversia, se scrie imaginea in fisier

    private BufferedImage image = null;

    private int number = -1; // aici am facut Bufferul sincronizat,
pentru a sincroniza Producer si Consumer
    private boolean available = false;

    public synchronized int get() {
      while (!available) {
            try {
                  wait();// Asteapta producatorul sa puna o valoare
            } catch (InterruptedException e) {
                  e.printStackTrace();
            }
      }
      available = false;
      notifyAll();
      return number;
    }
```

```
    public synchronized void put(int number) {
      while (available) {
            try {
                  wait();// Asteapta consumatorul sa preia valoarea
            } catch (InterruptedException e) {
                  e.printStackTrace();
            }
      }

      this.number = number;
      available = true;
      notifyAll();
    }


    public boolean readImageFromFile(String file) { // Specificand
calea catre imagine, se citeste din fisier
      long time = System.currentTimeMillis(); // // Incepe
contorizeaza timpul actual al sistemului
      try (FileInputStream stream = new FileInputStream(file)) { //
se asteapta introducerea de la tastatura a imaginii
            //In 'Image' de tip BufferedImage vom stoca poza, fapt
pentru care i-am atribuit la inceput valoarea null
            image = ImageIO.read(stream);
      } catch (FileNotFoundException e) { //In caz ca nu este gasit
fisierul sau apare alta eroare, se arunca erori
            System.out.println("Nu am putut gasi fisierul " + file);
            return false;
      } catch (IOException e) {
            System.out.println("A aparut o eroare in timpul citirii
din fisier");
            return false;
      }

      time = System.currentTimeMillis() - time; // Timpului de
citire al imaginii citita de la tastatura
      System.out.println("Citirea din fisier a durat " + time /
1000.0f + " secunde"); // Afisam timpul de citire al imaginii
```

```
        return true;
    }

    // Aplicam algoritmul de conversie propriu zisa.

    public void transform() throws IOException {
        long time = System.currentTimeMillis(); //Incepe contorizeaza
timpul actual al sistemului

        int width = image.getWidth(); // se obtine latimea imaginii
        int height = image.getHeight(); // se obtine lunfimea imaginii

        int factorlumin = 100; // pentru a modifica luminozitatea
imayinii dupa preferinte, schimbam factorul

        // Se obtine RGB-ul imaginii
        for (int y = 0; y < height; y++) {
            for (int x = 0; x < width; x++) {
                Color c = new Color(image.getRGB(x, y));
                // adding factor to rgb values
                int red = c.getRed() + factorlumin;
                int blue = c.getBlue() + factorlumin;
                int green = c.getGreen() + factorlumin;
                if (red >= 256) {
                    red = 255;
                } else if (red < 0) {
                    red = 0;
                }
                if (green >= 256) {
                    green = 255;
                } else if (green < 0) {
                    green = 0;
                }
                if (blue >= 256) {
                    blue = 255;
                } else if (blue < 0) {
                    blue = 0;
                }
```

```
                    image.setRGB(x, y, new Color(red, green,
blue).getRGB());

               }
          }

          // Se calculeaza timpul necesar procesarii imaginii si se
afiseaza
          time = System.currentTimeMillis() - time;

          System.out.println("Transformarea imaginii a durat " + time /
1000.0f + " secunde");
     }

     public void writeImageToFile(String file) { // scrierea imaginii
in fisierul dat de la tastatura
          long time = System.currentTimeMillis();

          try (FileOutputStream stream = new FileOutputStream(file)) {
               ImageIO.write(image, "BMP", stream); // precizam ca
trebuie in format bmp
          } catch (FileNotFoundException e) { // aruncam exceptii in
cazul in care nu se gaseste calea sau nu se poate deschide fisierul
               System.out.println("Calea data " + file + " nu este
valida ! ");
               return;
          } catch (IOException e) {
               System.out.println("A aparut o eroare in timpul scrierii
in fisier");
               return;
          }


          time = System.currentTimeMillis() - time; // calculam timpul
necesar scrierii in fisierul destinatie
          System.out.println("Scrirea in fisier a durat " + time /
1000.0f + " secunde"); // il afisam
     }
```

```
}
```

Main

```
import java.io.IOException;
import java.util.*;

public class MainClass {

    public static void main(String[] args) throws IOException {
       Scanner scanner = new Scanner(System.in); // citim de la
tastatura

       try {
            while (true) {
                  Transformare reader = new Transformare(); // cream
un obiect de tip Transformare

                  boolean imageRead = false;
                  do {
                        System.out.println("Introduceti calea catre
fisierul sursa SAU \"exit\" pentru a iesi");
                        String line = scanner.nextLine(); // Se
citeste calea catre fisierul de intrare

                        if ("exit".equals(line)) { // in cazul in care
dam exit de la tastatura, se iese automat din program
                              System.out.println("S-a terminat
algoritmul.");
                              return;
                        }

                        imageRead = reader.readImageFromFile(line); //
citim imaginea data de la tastatura
                  } while (!imageRead);
```

```
                reader.transform(); // se aplica algoritmul de
Brightness Modification si se face conversia

                System.out.println("Dati calea catre fisier
destinatie");

                String line = scanner.nextLine();
                reader.writeImageToFile(line); // Luam calea spre
fisierul destinatie de la tasatura

                Producer t1 = new Producer(reader); // sincronizam
cele doua threaduri
                Consumer t2 = new Consumer(reader);
                t1.start();
                t2.start();

            }
        } finally {
            scanner.close();
        }
    }
}
```