IMAGE RESIZING – ZERO ORDER HOLD METHOD

-proiect Java-

STUDENT : ORBISOR BIANCA-ALEXANDRA

INTRODUCERE

Prin intermediul aplicatiei Zoom un utilizator poate sa mareasca o imagine pe care o introduce.

DESCRIEREA APLICATIEI CERUTE

Utilizatorul introduce numele pozei pe care doreste sa o mareasca si numele noii poze marite. Cele doua poze se vor gasi in folder-ul Image. Imaginea introdusa de utilizator are format bmp, pe cand poza rezultata are format png. In folder-ul Image se afla poza originala, iar dupa ce poza este redimensionata, in acest folder apare, pe langa poza initiala, si poza marita. Daca, dupa ce utilizatorul a marit o imagine, acesta doreste sa o micsoreze trebuie doar sa merga in folder-ul Image unde gaseste poza originala (echivalent cu a micsora poza marita).

PARTEA TEORETICA

Zero order hold method este o metode de a mari o imagine. In zero order hold method, alegem doua elemente adiacente de pe linia respective si apoi le adunam si impartim rezultatul la doi, si plasam rezultatul intre cele doua elemente.

Prima data marim pe linii, apoi pe coloane.

FOR EXAMPLE

Luam o imagine cu dimensiunea 2 linii si 2 coloane si o marim, utilizand metoda zero order hold.

1	2
3	4

ROW WISE ZOOMING

1	1	2
3	3	4

Luam primele doua numere : (2 + 1) = 3 si le impartim cu 2, obtinem 1,5 pe care il aproximam cu 1.

COLUMN WISE ZOOMING

1	1	2
2	2	3
3	3	4

Luam valoarile a doi pixeli adiacenti de pe coloana – 1 si 3. Le adunam si obtinem 4 si apoi impartim prin doi si obtinem doi, pe care il plasam intre ei.

NEW IMAGE SIZE

Dimensiunea imaginii noi este 3 x 3, pe cand dimensiunea imaginii originale este 2x2. Formula pentru calculul dimensiunii imaginii noi :

(2(numar linii) minus 1) X (2(numar coloane) minus 1)

DESCRIEREA APLICATIEI (Structurala, Arhitecturala si Functionala)

Aplicatia este formata dintr-o interfata (Interfata), trei clase (Zoom, Informatii, Detalii) si o clasa abstracta MyClass.

1. INTERFATA :
   - Interfata
2. CLASE :
   - Zoom – implementeaza interfata Interfata
   - Informatii – subclasa a clase Zoom
   - Detalii – subclasa a clasei Informatii
3. CLASA ABSTRACTA :
   - MyClass – subclasa a clasei Detalii

Interfata are o singura metoda abstracta getInformatii. In interfata apare doar declararea metodei fara implementare. Clasa Zoom implementeaza interfata Interfata (detine metoda getInformatii din interfata si apare in cadrul acestei clase si o implementare a acestei metode).

In cadrul clasei Zoom se efectueaza cea mai mare parte a proiectului. Ca si campuri avem width(private), height(private), newWidth, newHeight, o imagine si un fisier. Incapsulare – crearea de metode care sa returneze valoarea unui camp private si metode pentru setarea acestor campuri. Polimorfism – crearea metodei afisare cu lista de parametrii care difera de la o metoda “afisare” la alta. Citirea unei imagini se face cu ajutorul metodei readImage unde se citeste calea spre fisierul ce contine imaginea, crearea unei variabile care sa stocheze imaginea respectiva. Citirea cu succes este urmata de mesajul “Reading complete”. Daca citirea nu s-a efectuat cu succes se apeleaza o exceptie. Pentru a afisa elementele componente ale unei imagini se foloseste metoda afis in cadrul careia am folosit varargs (nu are importanta ce ii dau ca parametrii, metoda imi afiseaza indiferent de numarul de parametrii).

In main se citeste numele pozei ce se doreste sa se mareasca prin intermediul metodei readImage. Se citeste numele pozei rezultate in urma maririi. Se calculeaza noua dimensiune a imaginii. Prin intermediul matricei inter construim o copie a imaginii originale si in care vom retine valoarea fiecarui pixel al imaginii. Construim o noua imagine care are noua dimensiune calculata.

Urmeaza partea de prelucrare a imaginii initiale. Initializam poza cea noua astfel : pe fiecare linie, din doi in doi, si pe fiecare coloanal, din doi in doi, setam RGB – ul noii imagini cu valoarea din vectorul valoare corespondent pozitiei.

Prin intermediul unei matrice intermediare inter calculam valoarea fiecarui pixel in urma prelucrarii. Metoda studiata se aplica mai intai pe linii, iar apoi pe coloane.

Dupa ce am calculate fiecare pozitie scriem imaginea in folder-ul introdus de utilizator la inceputul aplicatiei.

Clasa Informatii este subclasa clasei Zoom. Are ca si camp proiect de tip string in care se retine numele proiectului. Metoda getProiect returneaza numele proiectului, setProiect seteaza numele proiectului, iar getInformatii apeleaza metoda getInformatii din clasa parinte (Zoom) care afiseaza informatii despre poza la care adauga numele proiectului.

Clasa Detalii este subclasa clasei Informatii. Are ca si camp materie de tip string in care se retine numele proiectului. Metoda getMaterie returneaza numele materiei, setMaterie seteaza numele materiei, iar getInformatii apeleaza metoda getInformatii din clasa parinte (Informatii) care afiseaza informatii despre poza la care adauga numele materiei.

Clasa abstracta MyClass este subclasa clasei Informatii. Are un camp nume de tip string in care se retine numele. Metoda getNume returneaza numele, setNume seteaza numele, iar ca si metoda abstracta avem afisareNume.

BIBLIOGRAFIE

http://www.tutorialspoint.com/dip/Zooming_Methods.htm

https://www.geeksforgeeks.org/variable-arguments-varargs-in-java/

https://study.com/academy/lesson/java-verargs-use-examples.html

https://www.dyclassroom.com/image-processing-project/how-to-create-a-random-pixel-image-in-java

https://www.tutorialspoint.com/java_dip/understand_image_pixels.htm

https://www.dyclassroom.com/image-processing-project/how-to-get-and-set-pixel-value-in-java

DOCUMENTATIE COD SURSA

```
package temaJava;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.awt.Color;
import java.awt.image.BufferedImage;
import java.awt.image.RenderedImage;
import javax.imageio.ImageIO;

public class Zoom implements Interfata{

	//get image width and height
	int width;    //width of the image
	int height;  //height of the image

	//new image dimension
	int newWidth;       //width of the image
	int newHeight;      //height of the image

	private int nrLinii;
	private int nrColoane;

	public BufferedImage image = null;
	public static File f = null;

	//incapsulare
	public int getNrLinii() {
		return nrLinii;
	}

	public void setNrLinii(int nrLinii) {
		if(nrLinii > 0)
			this.nrLinii = nrLinii;
	}

	public int getNrColoane() {
		return nrColoane;
	}

	public void setNrColoane(int nrColoane) {
		if(nrColoane > 0)
			this.nrColoane = nrColoane;
	}

	//polimorfism
	//afisare numar de linii
	public void afisare(int nrLinii)
	{
		System.out.println("Numar linii : " + this.nrLinii);
	}
```

```
	//afisare numar linii si numar coloane
	public void afisare(int nrLinii, int nrColoane)
	{
		System.out.println("Numar linii : " + this.nrLinii + "\n" + "Numar
coloane : " + this.nrColoane);
	}

	//citirea unei imagini
	public BufferedImage readImage(String nume)
	{
		//read image
		 try{

		   //imaginea care se citeste
		    f = new File("D:\\Image\\"+ nume + ".bmp"); //image file path
		    image = new BufferedImage(225, 225, BufferedImage.TYPE_INT_ARGB);
		    image = ImageIO.read(f);

		    System.out.println("Reading complete.");

		  }catch(IOException e){
		    System.out.println("Error: "+e);
		  }
		return image;

	}

	//se retuneaza width-ul imaginii
	public int getWidth(BufferedImage image2) {
		return ((RenderedImage) image2).getWidth();    //width of the image
	}

	//se seteaza width-ul imaginii
	public void setWidth(int width) {
		if(width > 0)
			this.width = width;
	}

	//se retuneaza height-ul imaginii
	public int getHeight(BufferedImage image2) {
		return ((RenderedImage) image2).getHeight();  //height of the image
	}

	//se seteaza height-ul imaginii
	public void setHeight(int height) {
		if(height > 0)
			this.height = height;
	}

	//se retuneaza width-ul imaginii redimensionate
	public static int getnewWidth(int width) {
		return 2 * width - 1;     //width of the image
	}

	//se seteaza width-ul imaginii redimensionate
```

```
	public void setnewWidth(int newWidth) {
		if(newWidth > 0)
			this.newWidth = newWidth;
	}

	//se retuneaza height-ul imaginii redimensionate
	public static int getnewHeight(int height) {
		return 2 * height - 1;	//height of the image
	}

	//se seteaza height-ul imaginii redimensionate
	public void setnewHeight(int newHeight) {
		if(newHeight > 0)
			this.newHeight = newHeight;
	}

	//folosire varargs
	public void afis(int ...in )
	{
		for(int i : in)
			System.out.print(i + " ");
	}



	public static void main(String[] args) throws IOException {
		// TODO Auto-generated method stub
		Zoom z = new Zoom();
		BufferedImage image;

		//get image width and height
		int width;	//width of the image
		int height;  //height of the image

		//new image dimension
		int newWidth;		//width of the image
		int newHeight;		//height of the image

		z.setNrLinii(200);
		z.afisare(z.getNrLinii());

		//citirea de la tastatura a numelui pozei ce se doreste sa se mareasa si
numele pozei rezultate

		BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

		System.out.println("Introduceti numele pozei ce doriti sa o mariti:");
		String numeFisierIn = br.readLine();
		System.out.println("Introduceti un nume pentru poza marita:");
		String numeFisierOut = br.readLine();

		br.close();

		System.out.println("Numele pozei " + numeFisierIn);
```

```
            System.out.println("Numele pozei marite " + numeFisierOut);

            image = z.readImage(numeFisierIn);

            //get image width and height
            System.out.println("Width " +z.getWidth(image));
            height = z.getWidth(image);
            System.out.println("Height " +z.getHeight(image));
            width = z.getWidth(image);

            //folosire varargs
            System.out.println("Varargs");
            z.afis(height);
            System.out.println();
            z.afis(height,width);
            System.out.println();

            //matricea valoare contine valoarea fiecarui pixel
            int[][] valoare = new int [height][width];

            //new image dimension
            newWidth = getnewWidth(width);
            newHeight = getnewHeight(height);
            System.out.println("newWidth " +  newWidth);
            System.out.println("newHeight " + newHeight);

            //cream o matrice ce este o copie intermediara o imaginii finale
            int[][] inter = new int [newHeight][newWidth];

            //create buffered image object img
            BufferedImage newimg = new BufferedImage(newWidth, newHeight,
BufferedImage.TYPE_INT_ARGB);
            //file object
            File newf = null;

            //am luat valorile lui alpha,read,green si blue pentru fiecare pixel al
imaginii
            //am retinut aceste valori in vectorul valoare
          for(int i = 0; i < height; i++) {
            for(int j = 0; j < width; j++) {

                  Color c = new Color(image.getRGB(i, j));
                  int p = (c.getAlpha()<<24) | (c.getRed()<<16) | (c.getGreen()<<8)
| c.getBlue(); //pixel
                  valoare[i][j] = p;
            }
          }

          int l = -1;
            int c = -1;
            //initializam poza cea noua
            for(int i = 0; i < newHeight; i = i + 2) {
                  l++;
                  c = -1;
                  for(int j = 0; j < newWidth; j = j + 2) {
```

```
                        c++;
                        newimg.setRGB(i, j, valoare[l][c]);
                }
        }

        int newValoare = 0;

        //cream o matrice ce este o copie intermediara o imaginii finale
        for(int i = 0; i < newHeight; i++) {
                for(int j = 0; j < newWidth; j++) {

                        Color c2 = new Color(newimg.getRGB(i, j));
                        int p = (c2.getAlpha()<<24) | (c2.getRed()<<16) |
(c2.getGreen()<<8) | c2.getBlue(); //pixel
                        inter[i][j] = p;
                        newimg.setRGB(i, j, p);
                }
        }

        //lucram intai pe linii
        for(int i = 0; i < newHeight; i = i + 2) {
                for(int j = 0; j < newWidth-2; j = j + 2) {
                        newValoare = (int)((inter[i][j] + inter[i][j+2])/2);
                        newimg.setRGB(i, j + 1, newValoare);
                }
        }

        //apoi lucram pe coloane
        for(int j = 0; j < newWidth; j = j + 2) {
                for(int i = 0; i < newHeight-2; i = i + 2) {
                        newValoare = (int)((inter[i][j] + inter[i+2][j])/2);
                        newimg.setRGB(i + 1, j, newValoare);
                }
        }

        //write image
        try{
                newf = new File("D:\\Image\\" + numeFisierOut + ".png");
                ImageIO.write(newimg, "png", newf);

                f = new File("D:\\Image\\initial.png");
                ImageIO.write(image, "png", f);

                System.out.println("Writing complete.");
                }catch(IOException e){
                        System.out.println("Error: " + e);
                }

        }

    @Override
    public void getInformatii() {
        // TODO Auto-generated method stub
        System.out.println("Width = " + width);
        System.out.println("Height = " + height);
```

```
	}

}
```

```
package temaJava;

public class Informatii extends Zoom{

	public String proiect;

	//returneaza numele proiectului
	public String getProiect() {
		return proiect;
	}

	//seteaza numele proiectului
	public void setProiect(String proiect) {
		this.proiect = proiect;
	}

	//ofera informatii despre poza
	public void getInformatii() {
		// TODO Auto-generated method stub
		super.getInformatii();
		System.out.println("Proiectul este : " + proiect);
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Zoom z = new Zoom();
		Informatii i = new Informatii();
		String nume;
		z.setNrLinii(200);
		z.afisare(z.getNrLinii());
		i.setProiect("Zoom using zero order hold method");
		nume = i.getProiect();
		System.out.println(nume);

	}

}
```

```
package temaJava;

public class Detalii  extends Informatii{

	public String materie;

	//returneaza numele materiei
	public String getMaterie() {
		return materie;
	}
```

```
	//seteaza numele materiei
	public void setMaterie(String materie) {
		this.materie = materie;
	}

	//ofera informatii despre poza
	public void getInformatii() {
		// TODO Auto-generated method stub
		super.getInformatii();
		System.out.println("Materia este : " + materie);
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Zoom z = new Zoom();
		Informatii i = new Informatii();
		Detalii d = new Detalii();
		String nume;
		String materie;
		z.setNrLinii(200);
		z.afisare(z.getNrLinii());
		i.setProiect("Zoom using zero order hold method");
		nume = i.getProiect();
		System.out.println(nume);
		d.setMaterie("AWJ");
		materie = d.getMaterie();
		System.out.println("Materie : " + materie);
	}
}

package temaJava;

public interface Interfata {

	//metoda abstracta - este doar declarata, nu si implementata
	abstract void getInformatii();

}

package temaJava;

//clasa abstracta
abstract class MyClass extends Detalii{

	String nume;

	//returneaza numele
	public String getNume() {
		return nume;
	}

	//seteaza numele
	public void setNume(String nume) {
		this.nume = nume;
	}
```

```
	//metoda abstracta
	abstract void afisareNume();

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Detalii d = new Detalii();
		String materie;
		d.setMaterie("AWJ");
		materie = d.getMaterie();
		System.out.println("Materie : " + materie);
	}
}
```