pictureinOOP

```
package OOPtry;


import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.util.Scanner;

import javax.imageio.ImageIO;


public class PictureinOOP {


    public static void main(String[] args) {



        while(true) { //For continueing the program or shutting it
down


        //Scanner

        Scanner sc = new Scanner(System.in);


        String fileName;

        BufferedImage cImage = null;


        while(true){

            System.out.println("Write picture file name:");

            fileName = sc.nextLine();


            try {

                cImage = ImageIO.read(new File(fileName)); //
Placeringen af filen er relativ til java filen.
```

```
                break;
            } catch (IOException | IllegalArgumentException e) {

            }
        }



        //Multiarray initializing and creating.
        byte[][] pictureArray = new byte[cImage.getWidth()][];



        //Image initializing
        int h = cImage.getHeight();
        int w = cImage.getWidth();
        Image photo = new Image(h,w);

        //Multiarray filling
        photo.assempleImage(pictureArray, cImage);

        //Amount of pixels
        System.out.println("Limit: " + 257*257 + ". Amount of pixels:
" + cImage.getWidth()*cImage.getHeight() + "." );

        //Full or partly drawn?
        String fullorPart = "";
        boolean fullScaleNeeded = false;
        boolean partlyScaleNeeded = false;
        boolean fullPicture = false;
        boolean picturePart = false;

        while(true) {
            if(cImage.getHeight()*cImage.getWidth() < 257*257) {
                System.out.println("Full picture or a specific part? 
Answer FP or SP");
                fullorPart = sc.nextLine();


                if (fullorPart.compareTo("FP") == 0) { // compare 
giver 0 hvis de to Strings er ens.
                    fullPicture = true;
                    break;
                } else if(fullorPart.compareTo("SP") == 0) {
                    picturePart = true;
                    break;
                } else {
                    System.out.print("Wrong answer, please try again: 
");
                }
            } else {
                System.out.println("Full picture or a specific part? 
Answer FP or SP");
                fullorPart = sc.nextLine();


                if (fullorPart.compareTo("FP") == 0) { // compare 
giver 0 hvis de to Strings er ens.
                    fullScaleNeeded = true;
                    break;
                } else if(fullorPart.compareTo("SP") == 0) {
                    partlyScaleNeeded = true;
                    break;
                } else {
                    System.out.print("Wrong answer, please try again: 
");
                }
            }
        }

        // Setup roboClient
        String hostName = "192.168.0.103";
        int port = 12345;
        RobotClient roboC = new  RobotClient(hostName, port);

        Scale scaledImage = new Scale(cImage);

        if(fullPicture) {

            Message code = new Message();
            String message = code.convertToMessage(pictureArray,
cImage);
            //Draw full image
            photo.drawImage(pictureArray);
            //Write full message
            System.out.println(message);

            //Connect and send to PLC
            roboC.connect();
            if(roboC.isConnected()){
                roboC.write(message);
                System.out.println("Message send");

                while(true) {
                    System.out.println("Disconnect?");
                    String disconnectOrNot = sc.nextLine();
                    if (disconnectOrNot.compareTo("YES") == 0 ) {
                        roboC.disconnect();
                        break;
                    }
                }
            }








        } else if(picturePart) {
            System.out.println("Specific part? Alright, please type in
two x-values and y-values");
            System.out.println("values for " + "x, has to value
within: " + cImage.getWidth() + " and y, within: " +
cImage.getHeight());
            System.out.println("x1 has to be smaller than x2 and so
goes for y1 and y2");

            System.out.println("x1: ");
            int x1 = sc.nextInt();
            System.out.println("x2: ");
            int x2 = sc.nextInt();
            System.out.println("y1: ");
            int y1 = sc.nextInt();
            System.out.println("y2: ");
            int y2 = sc.nextInt();

            PartImage part = new PartImage();
            String partMessage = part.messagePart(x1, y1, x2, y2,
pictureArray);
            //Draw part of immage
            part.drawImage(x1, y1, x2, y2,pictureArray);
```

```
    //Write part of message
    System.out.println(partMessage);

    //Connect and send to PLC
    roboC.connect();
    if(roboC.isConnected()){
        roboC.write(partMessage);
        System.out.println("Message send");

        while(true) {
            System.out.println("Disconnect?");
            String disconnectOrNot = sc.nextLine();
            if (disconnectOrNot.compareTo("YES") == 0 ) {
               roboC.disconnect();
               break;
            }
        }
    }




}else if(fullScaleNeeded) {

    //Draw scaled image
    scaledImage.amountOfScaling(pictureArray);

    int n = scaledImage.getValueN();
```

```
            scaledImage.drawImage(pictureArray, n);
            String scaledMessage = 
scaledImage.convertToMessage(pictureArray, cImage, n);
            System.out.println(scaledMessage);









            //Connect and send to PLC
            roboC.connect();
            if(roboC.isConnected()){
                roboC.write(scaledMessage);
                System.out.println("Message send");
            }

            while(true) {
                System.out.println("Disconnect?");
                String disconnectOrNot = sc.nextLine();
                    if (disconnectOrNot.compareTo("YES") == 0 ) {
                        roboC.disconnect();
                        break;
                    }
            }



        }else if(partlyScaleNeeded) {
            System.out.println("Specific part? Alright, please type in
two x-values and y-values");

            System.out.println("values for " + "x, has to value
within: " + cImage.getWidth() + "and y, within: " +
cImage.getHeight()); //scaller billedet først. og få denne string til
at afhænge af det nye billede

            System.out.println("x1 has to be smaller than x2 and so
goes for y1 and y2");


            System.out.println("x1: ");

            int x1 = sc.nextInt();

            System.out.println("x2: ");

            int x2 = sc.nextInt();

            System.out.println("y1: ");

            int y1 = sc.nextInt();

            System.out.println("y2: ");

            int y2 = sc.nextInt();




            //Draw part of scaled image

            scaledImage.amountOfScaling(pictureArray);

            int n = scaledImage.getValueN();

            scaledImage.drawImage(x1, y1, x2, y2, pictureArray, n);


            //Write part of scaled message

            String scaledMessagePart =
scaledImage.scaledMessagePart(x1, y1, x2, y2, pictureArray, n);

            System.out.println(scaledMessagePart);


            //Connect and send to PLC

            roboC.connect();
            if(roboC.isConnected()){
                roboC.write(scaledMessagePart);
                System.out.println("Message send");
            }

            while(true) {
                System.out.println("Disconnect?");
                String disconnectOrNot = sc.nextLine();
                if (disconnectOrNot.compareTo("YES") == 0 ) {
                    roboC.disconnect();
                    break;
                }
            }



        }

        //Continue?
        System.out.println("Continue?");
        System.out.println(" YES/NO");
        String stopOrNot = sc.nextLine();
            if (stopOrNot.compareTo("NO") == 0 ) {
                break;
            }
        }
    }
}
```

Image

```
package OOPtry;


import java.awt.image.BufferedImage;


/**
 *
 * @author cubey
 */
public class Image {
    private int iHeight;
    private int iWidth;



    Image (int newIheight, int newIwidth) {
        this.iHeight = newIheight;
        this.iWidth = newIwidth;
    }





    public void assempleImage(byte pictureArray[][], BufferedImage
image) {
        Color ofPicture = new Color();
        for(int x = 0; x < this.iWidth; x++) {
            pictureArray[x] = new byte[this.iHeight]; // Indsætning af
array i multiarrayet
            for(int y = 0; y < this.iHeight; y++) {


                ofPicture.setCrgb(image.getRGB(x, y));
                ofPicture.setRed((ofPicture.getCrgb()>>16)&0xff);
                ofPicture.setGreen((ofPicture.getCrgb()>>8)&0xff);
                ofPicture.setBlue(ofPicture.getCrgb()&0xff);

                int average = ofPicture.getAverage();

                image.setRGB(x, y, average);



                if(average >= 160) {
                    pictureArray[x][y] = (byte)(0);
                } else {
                    pictureArray[x][y] = (byte)(1);
                }
            }
        }
    }



    public void drawImage(byte pictureArray[][]) {
        for(int y = 0; y < this.iHeight ; y++) {
            for(int x = 0; x < this.iWidth; x++) {
                if(pictureArray[y][x] == 0) {
                    System.out.print("0 ");
                } else {
                    System.out.print("1 ");
                }
            }
            System.out.println();
        }
    }
}
```

Message

```
package OOPtry;


import java.awt.image.BufferedImage;


public class Message {
    private char upOrDown;
    private String message;


    Message() {
        message = "";
    }


    public String convertToMessage(byte pictureArray[][],
BufferedImage image) {
        for(int y = 0; y < image.getHeight(); y++) {
            for(int x = 0; x < image.getWidth(); x++) {
                if(x < image.getHeight() && y < image.getWidth()) {


                    if(pictureArray[y][x] == 0) {
                       upOrDown = 'U';
                    } else {
                       upOrDown = 'D';
                    }
                    message = message + upOrDown;
                }
            }
            message = message + 'N';
        }
        message = message + 'Q';

        return  message;
    }



}
```

PartImage

```
package OOPtry;




public class PartImage {
    private String message;
    private char upOrDown;

    PartImage() {
        message = "";
    }

    public String messagePart(int x1,int y1, int x2,int y2, byte
pictureArray[][]) {

        for(int y = y1; y < y2; y++) {
            for(int x = x1; x < x2; x++) {
                if (pictureArray[y][x] == 0) {
                   upOrDown = 'U';
                } else {
                   upOrDown = 'D';
                }
                message = message + upOrDown;
            }
            message = message + 'N';
        }
        message = message + 'Q';
        return message;
    }



   public void drawImage(int x1,int y1, int x2,int y2,byte
pictureArray[][]) {
        for(int y = y1; y < y2; y++) {
            for(int x = x1; x < x2; x++) {
                if(pictureArray[y][x] == 0) {
                    System.out.print("0 ");
                } else {
                    System.out.print("1 ");
                }
            }
            System.out.println();
        }
    }


}
```

Scale

```
package OOPtry;

import java.awt.image.BufferedImage;

public class Scale {
    private int iHeight;
    private int iWidth;
    private long pAmount;
    private long spAmount;
    private int n;


    private char upOrDown;
    private String message;


    Scale(BufferedImage image) {
        message = "";
        this.iHeight = image.getHeight();
        this.iWidth = image.getWidth();
        this.pAmount = this.iHeight*this.iWidth;
    }


    public int getValueN() {
        return this.n;
    }
```

```
public long getValueSPAmount() {
    return this.spAmount;
}

public long getValuePAmount() {
    return this.pAmount;
}

// Full scaled picture

public void amountOfScaling(byte pictureArray[][]) {
    int limit = 257*257;
    int n = 1000;
    while(true) {
            long divided = this.pAmount*1/n;
            this.spAmount = this.pAmount-divided;

            if (this.spAmount < limit || n == 2) {
                this.n = n;
                break;
            } else {
                if (n > 2) {
                    n--;
                }
            }
    }
}

public void drawImage(byte pictureArray[][], int n) {
        for(int y = 0; y < this.iHeight; y++) {
            if (y%n != 0) {

                for(int x = 0; x < this.iWidth; x++) {

                    if(x%n != 0) {
                        if(pictureArray[y][x] == 0) {
                            System.out.print("0 ");
                        } else {
                            System.out.print("1 ");
                        }
                    }


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


    public String convertToMessage(byte pictureArray[][],
BufferedImage image, int n) {


        for(int y = 0; y < image.getHeight(); y++) {
            if(y%n != 0) {
                for(int x = 0; x < image.getWidth(); x++) {
                    if(x%n != 0) {
                        if(x < image.getHeight() && y <
image.getWidth()) {
                            if(pictureArray[y][x] == 0) {
                               upOrDown = 'U';
                            } else {
                               upOrDown = 'D';
                            }
                            message = message + upOrDown;
                        }
                    }
                }
                message = message + 'N';
            }
        }
        message = message + 'Q';

        return  message;
    }



    // part of scaled picture

    public String scaledMessagePart(int x1,int y1, int x2,int y2, byte
pictureArray[][], int n) {

        for(int y = y1; y < y2; y++) {
            if(y%n != 0) {
                for(int x = x1; x < x2; x++) {
                    if(x%n != 0) {
                        if (pictureArray[y][x] == 0) {
                           upOrDown = 'U';
                        } else {
                           upOrDown = 'D';
                        }
                        message = message + upOrDown;
                    }
```

```
                    }
                    message = message + 'N';
                }
            }
            message = message + 'Q';
            return message;
        }



        public void drawImage(int x1,int y1, int x2,int y2,byte
pictureArray[][], int n) {
            for(int y = y1; y < y2; y++) {
                if(y%n != 0) {
                    for(int x = x1; x < x2; x++) {
                        if(x%n != 0) {
                            if(pictureArray[y][x] == 0) {
                                System.out.print("0 ");
                            } else {
                                System.out.print("1 ");
                            }
                        }
                    System.out.println();
                    }
                }
            }
        }
    }
```

Color

```
package OOPtry;

public class Color {
    private int rgb;
    private int red;
    private int green;
    private int blue;
    private int average;

    Color() {
    }

    public int getAverage() {
        average = this.average = (this.red + this.green +
this.blue)/3;
        return this.average;
    }

    public int getCrgb() {
        return this.rgb;
    }

    public void setRed(int newRed) {
        this.red = newRed;
    }

    public void setGreen(int newGreen) {
        this.green = newGreen;
    }

    public void setBlue(int newBlue) {
        this.blue = newBlue;
    }

    public void setCrgb(int newRGB) {
        rgb = newRGB ;
    }



}
```