

Ala-Too International University

**Color changer with JavaFX**

**HueU**

<https://github.com/DOSLAN/HueU>

**1 author:**

Surname and Name: Zheksheev Doslan

Faculty: Faculty of Engineering and Computer Science

Department: Computer Science Department

Group Name: COM – 18

E-MAIL: doslan.zheksheev@iaau.edu.kg

Mobile: +996 703 291 036

GitHub: <https://github.com/DOSLAN?tab=repositories>

Contents

[ACKNOWLEDGEMENT 3](#_Toc29383447)

[INTRODUCTION 3](#_Toc29383448)

[**First Things First:** 3](#_Toc29383449)

[**Execution Procedures:** 4](#_Toc29383450)

[**Nodes** 4](#_Toc29383451)

[**Image nodes (Input Image node and Output node):** 6](#_Toc29383452)

[**Input Image:** 6](#_Toc29383453)

[**Output:** 6](#_Toc29383454)

[**Converter Nodes (Separate RGB and Combine RGB):** 8](#_Toc29383455)

[**Separate RGB:** 8](#_Toc29383456)

[**Combine RGB:** 8](#_Toc29383457)

[**Evaluating Nodes (Add, Subtract, Multiply, Divide, Distance)** 8](#_Toc29383458)

[**Logic Nodes (Max, Min, Equals, Greater than)** 9](#_Toc29383459)

[**Examples:** 10](#_Toc29383460)

[**OBJECT ORIENTTED EXPLANATION** 11](#_Toc29383461)

[**Package Main:** 12](#_Toc29383462)

[**Controller.java:** 12](#_Toc29383463)

[**Package window:** 12](#_Toc29383464)

[**Window.java:** 12](#_Toc29383465)

[**NodeEditor.java:** 12](#_Toc29383466)

[**Package Node:** 13](#_Toc29383467)

[**MyNode.java:** 14](#_Toc29383468)

[**Parameter.java:** 18](#_Toc29383469)

[**Converter.java:** 21](#_Toc29383470)

[**Math.java:** 21](#_Toc29383471)

[**Getter\_Image.java:** 21](#_Toc29383472)

[**Setter\_Image.java** 22](#_Toc29383473)

[**Value.java:** 23](#_Toc29383474)

[**Connector.java:** 23](#_Toc29383475)

[**Package mouseGestures:** 23](#_Toc29383476)

[**MyCirclesMakeDraggable.java:** 23](#_Toc29383477)

[**MyNodesMakeDraggable.java:** 25](#_Toc29383478)

[**REFERENCE** 26](#_Toc29383479)

# ACKNOWLEDGEMENT

First of all, I would like to thank my lecturer Mr. Nurlan Shaidullaev for helping me to acquire some basic knowledge of “Java Programming Language”. At the same time, he gave me the opportunity to learn something new related to our module like constructors, methods, arrays, JavaFX etc.

Beside from my lecturer, I like to thank my other classmates for helping to understand the assignment related questions more clearly. They gave their best for completing this report on time. I thank them for their efforts.

# INTRODUCTION

First time when I use program named Blender I loved the user interface of Node Editor (where you can edit textures). It was convenient to invent and construct. It can be used to every type of editors, constructors. I want to develop such user interface and I decide to develop Color editor.

You can divide png, jpg files to red, green and blue colors and with them make some mathematical expressions to make new image file. It is comfortable to make mono chrome, contrast, brightness and other filters, and understand filter algorithms.

This assignment is based on developing an Color changer using “Java Programming Language”. For that I used JavaFX in this development so that it will become more users friendly to interact.

EXPLANATIONS

In this documentation we have given explanations of how to interact successfully with this Color changer. We have explained here step by step so that it will surely help users to become more user friendly with it. Below are our explanations:

## **First Things First:**

Before execute this program users need to do some works so that it will run properly into their system. First they need to make sure their system is having “JDK”. If they don’t have it then they can download from this below link:

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

Depending on their system (Windows 64bit/32bit) they need to download and install. Then they need to add the “JAVA” files to their system “PATH” so that the system can run the program from CMD (Command Prompt). The path will show something like this “C:\Program Files (x86)\Java\jre1.8.0\_25\bin;”. Now just add the address besides the current path directory and save it.

The other way they can execute this program in to download the IDE (Integrated Development Environment) on their system. They can download IntelIJ, ECLIPSE or NETBEANS depending on the windows (32bit/64bit). I highly recommend IntelIJ, because I developed it by IntelIJ, and files are hard to be accustomed to other IDE’s. Below is the links:

IntelIJ:

<https://www.jetbrains.com/idea/download/>

## **Execution Procedures:**

When user executes this program it will show area (figure 1) where you can edit your photo.

You can scroll by dragging scroll bars or with mouse-scroll (vertically) and shift + mouse-scroll (horizontally).

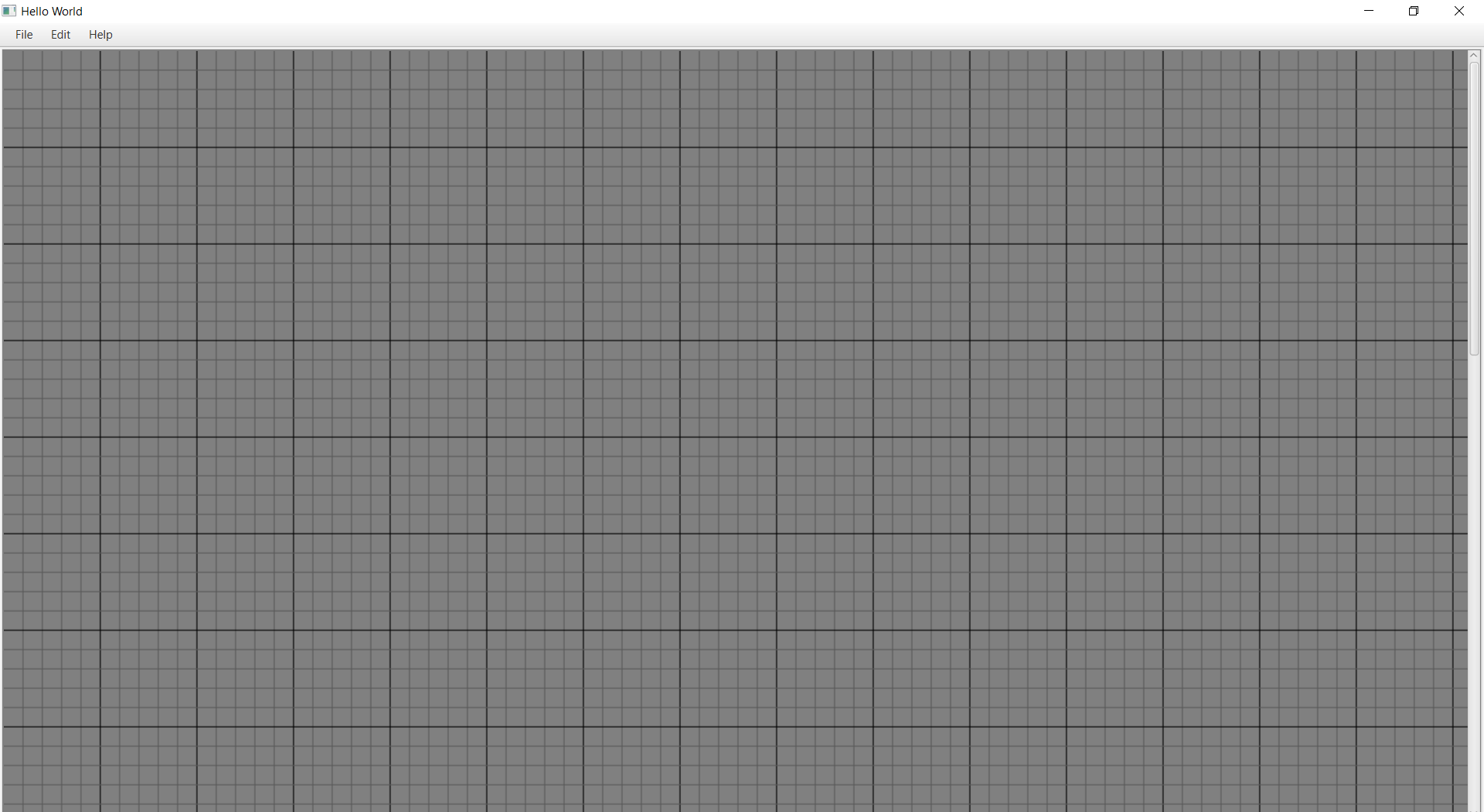


Figure 1

## **Nodes**

(figure 2) is the example of node. I named it “node”. Nodes are what user will work. Nodes in this program is everything. Nodes helps to user add image, edit colors, make some manipulations of values of colors and save it.

You can drag it, connect it to another node (by red circles), text into text fields which it has.

Nodes have points (red circles). Via them user can connect to another node. For connecting user needs to drag this point to the point of another node which user wants. Points are in two types input and output. Inputs are on the left of a node; outputs are on the right. Values which node have (it would create by text filed, or choosing image) will go out from output point and picked up by input point to another node. Points is input can connected to the only one point, if user will try first connected line would be deleted.

WARNING!!!: Be sure that color values connected to only color values and images connected to only images.

WARNING!!!: Be sure that color values are between 0 and 255

WARNING!!!: Be sure that images have same resolution.

WARNING!!!: Do not drag your node out of range of editor’s area. You would not able to get it again. Sorry I will fix it later.

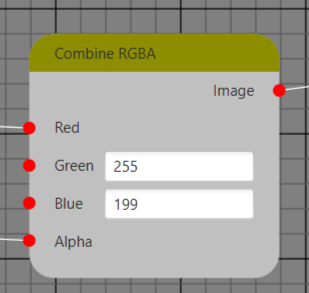


Figure 2

You can create node by going to Edit >> Add >> and choose node which you want (figure 3).

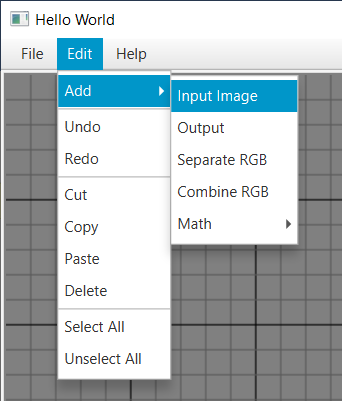


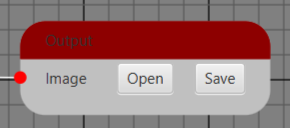
Figure 3

All types of nodes:

* + Input Image
  + Output
  + Separate RGB
  + Combine RGB
  + Add
  + Subtract
  + Multiply
  + Divide
  + Distance
  + Max
  + Min
  + Equals
  + Greater than

## **Image nodes (Input Image node and Output node):**

Nodes which will deal with images are Input image and Output (figure 4 and 5).



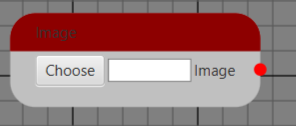


Figure 4 Figure 5

### **Input Image:**

It is in Edit>>Add>>Input Image

It has button “choose”.

When user click the button “choose” there would appear new window (figure 6) where user can choose image.

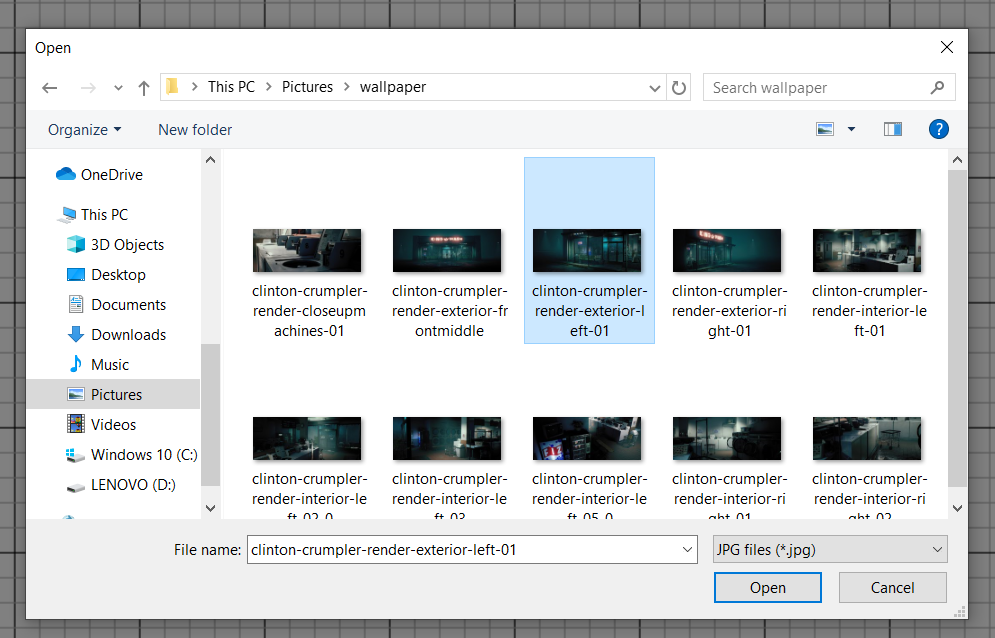


Figure 6

### **Output:**

If user want to use this node, he must go to Edit>>Add>>Output.

It has two buttons “Open” and “Save”. After completing work user can see the result by clicking the button “Open” (figure 7) and save it by clicking the button “Save” (figure 8) and showing the path where he wants to save:

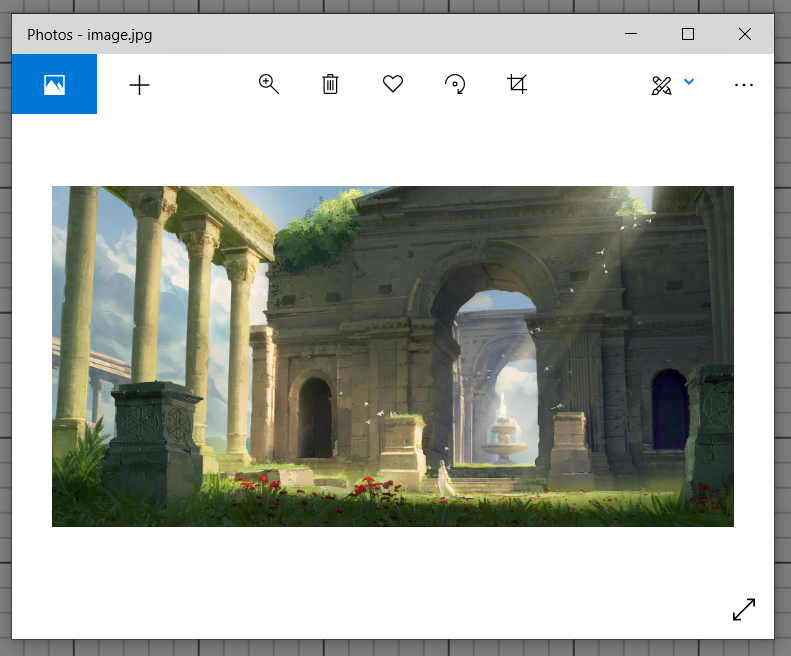


Figure 7

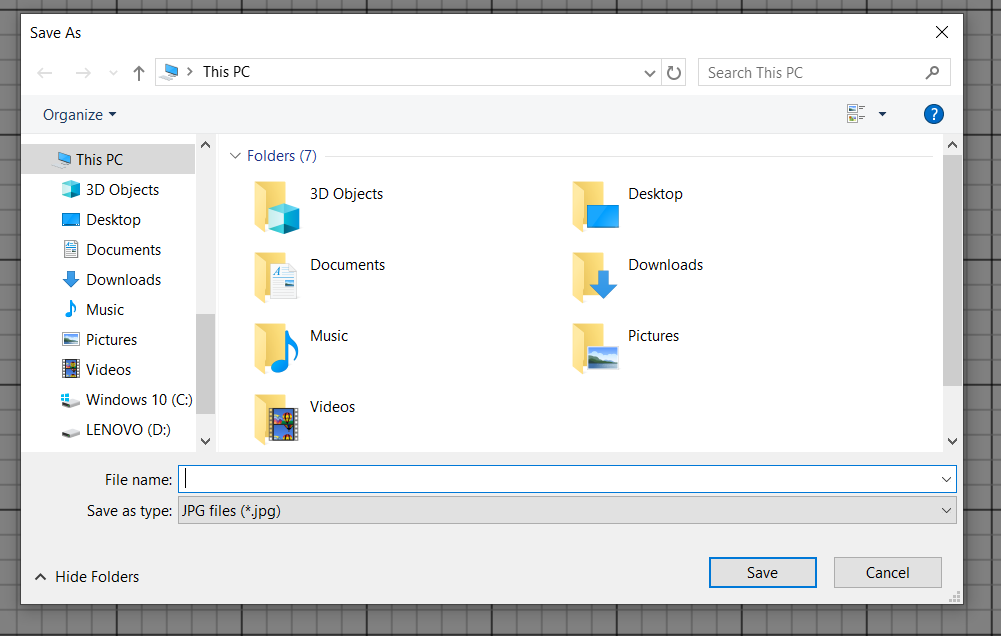


Figure 8

## **Converter Nodes (Separate RGB and Combine RGB):**

They are Separate RGB (figure 9) and Combine RGB (figure 10).

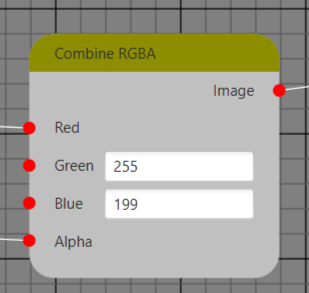


Figure 9

Figure 10

Figure 9

### **Separate RGB:**

If user want to use this node, he must go to Edit>>Add>>Separate RGB.

By this node you can separate image into color values.

WARNING!!!: Be sure that colors connected to only colors and images connected to only images.

### **Combine RGB:**

If user want to use this node, he must go to Edit>>Add>>Combine RGB.

There are text fields where you can write only integers. For every color value (Red, Green, Blue, Alpha) would be set the value which is connected by point if there is no connected value color would get the entered value to the text field. If point is connected text field would disappear.

## **Evaluating Nodes (Add, Subtract, Multiply, Divide, Distance)**

If user want to use this nodes, he must go to Edit>>Add>>Math>> and the name of the node. Than will appear a node (figure 11).

These nodes help to user add, subtract, multiply and divide colors values each other or by entered value. Distance node will evaluate absolute value of difference of values. These evaluations will be evaluated to every pixel value separately.

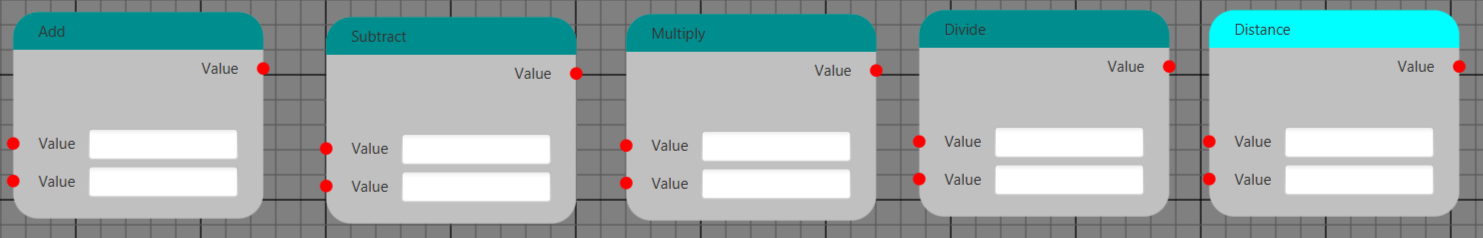


Figure 11

## **Logic Nodes (Max, Min, Equals, Greater than)**

If user want to use this nodes, he must go to Edit>>Add>>Math>> and the name of the node. Than will appear a node (figure 12).

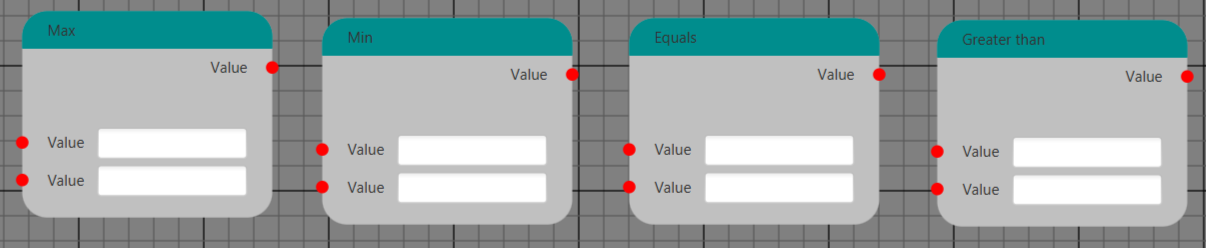
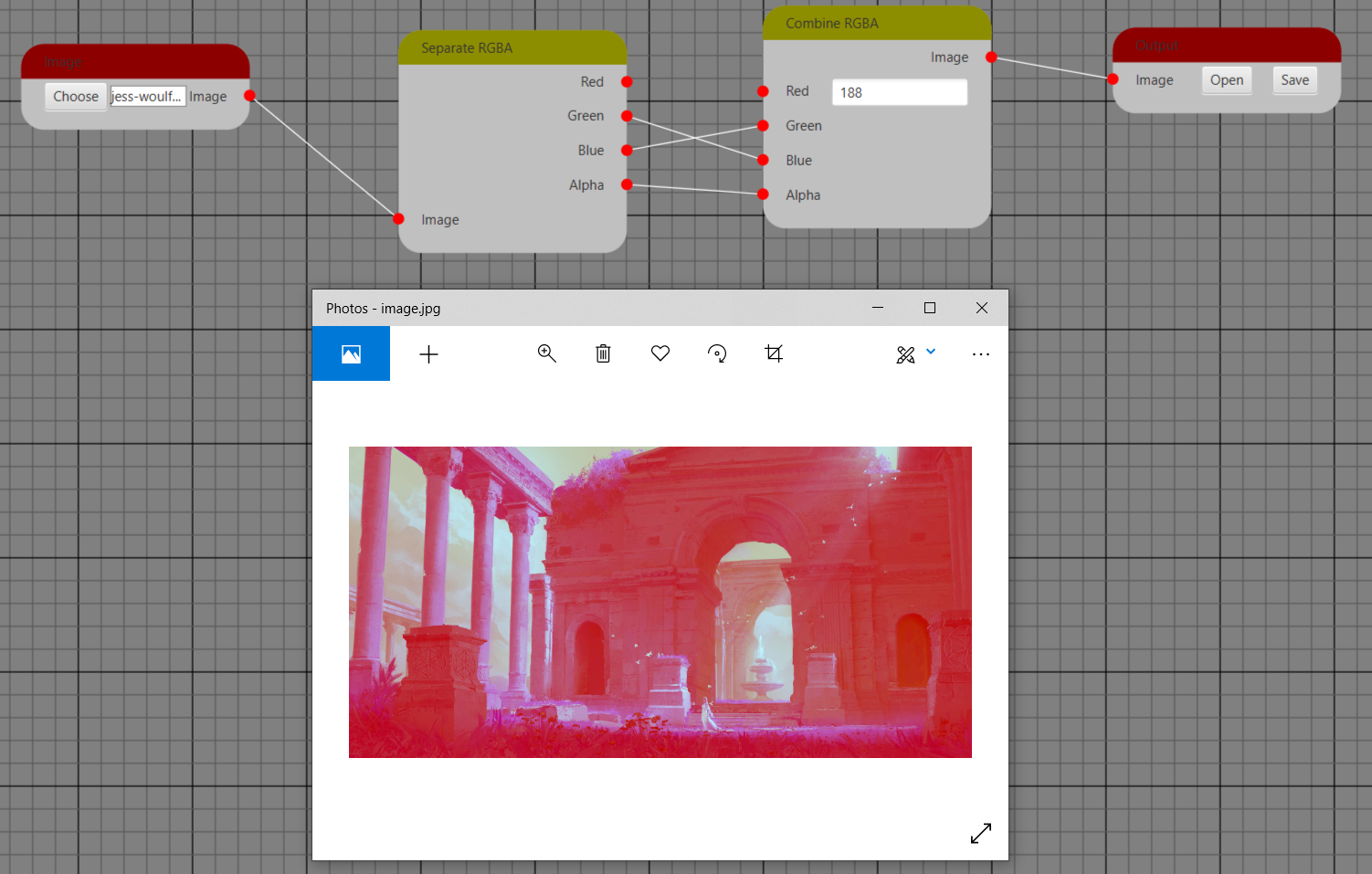


Figure 12

Max and Min nodes returns maximum and minimum of two entered values. Equals node returns 1 if they are equal, else 0. Greater than node returns 1 if first value is bigger than or equal to the second value, else 0.

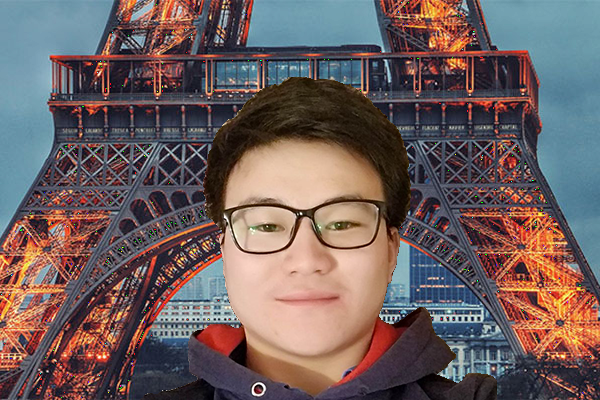
# **Examples:**



**Advanced Example:**

paris.jpg

****



**+ =**

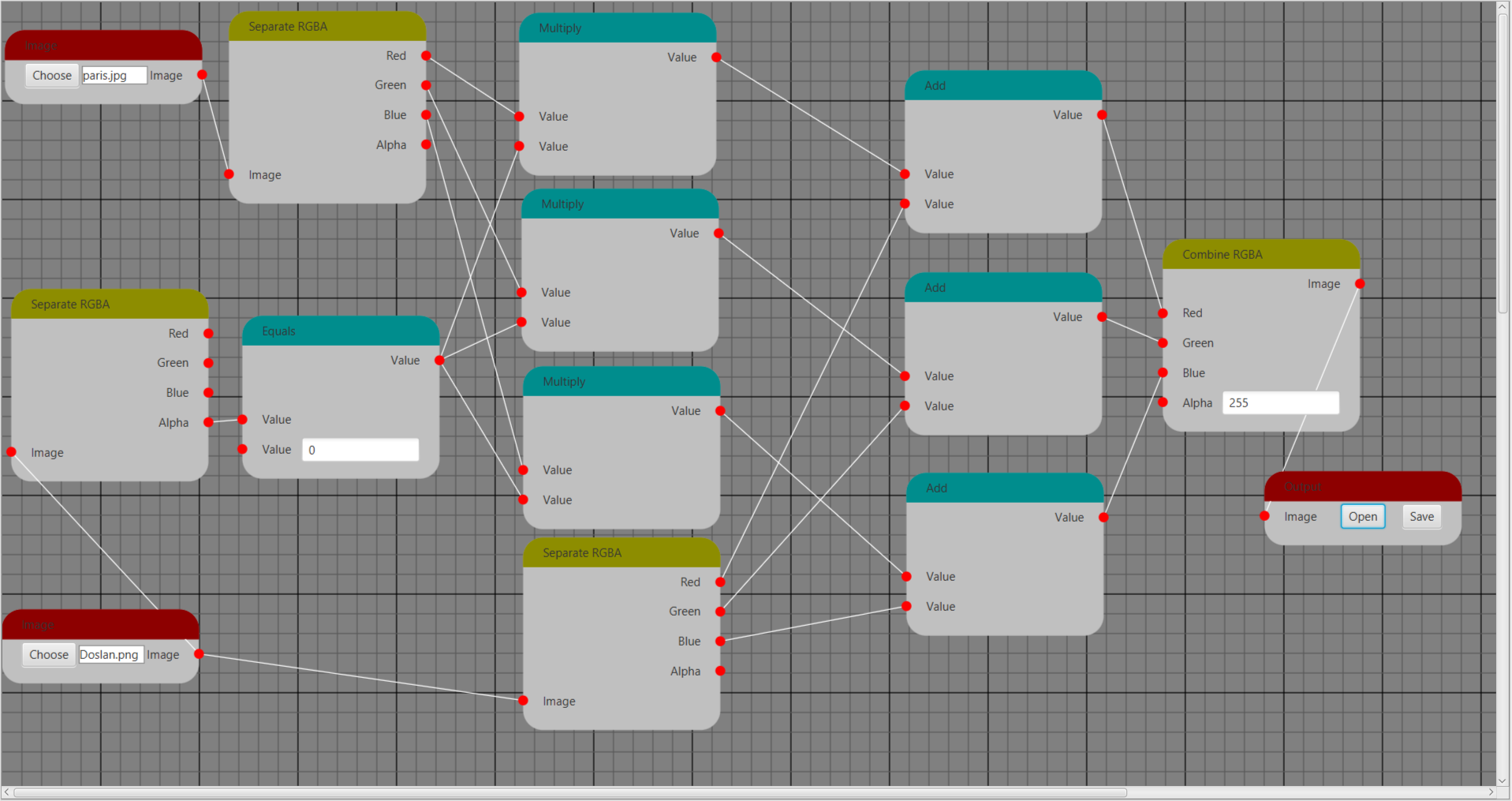
Output

Doslan.png

We can add two pictures: Two picture must have same resolution. For example in our case these images’ resolutions are 400X600. Second one must be transperant background or background must fixed color.

By Equals node we we looking is it transparent? If it’s transparent we have 1, else 0. We multiple it to Paris’s color and add to Doslan’s color.

* If Doslan’s pixel was transperant by the result we have Paris’s pixel: Paris’s pixel’s color multiplied by one and added to Doslan’s pixel’s color which is 0 (It’s transperant).
* If was not transperant by the resut we have Doslan’s pixel: Paris’s pixel’s color multiplied by zero and added to Doslan’s pixel’s color is equals Doslan’s pixel’s color.



# **OBJECT ORIENTTED EXPLANATION**

In object-oriented programming, for example, an object is a self-contained entity that consists of both data and procedures to manipulate the data. In other way, object oriented is the software engineering concept where it is represented using the “OBJECTS”. Below are the objected oriented samples we used in this “Java Programming Language”:

In my program there is four packages Main, mouseGestures, node and window (figure 13).

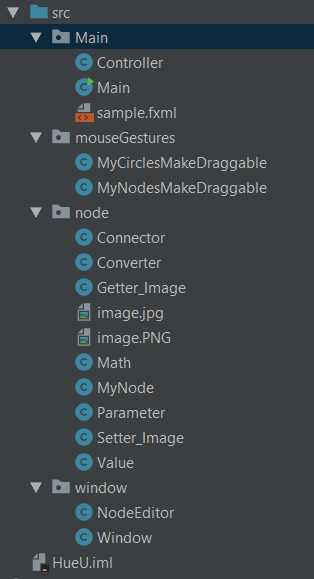


Figure 13

## **Package Main:**

There is three files in it: sample.fxml , Controller.java , Main.java. sample.fxml is created by Scene Builder. Main.java is simple main class.

### **Controller.java:**

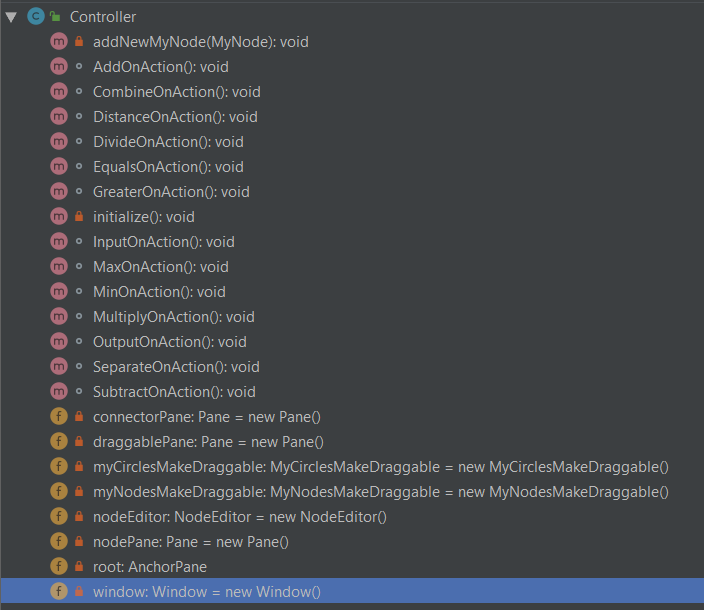


Figure 14

All methods (figure 14) in this class:

window is childe of root, nodeEditor is childe of window. Panes nodePane, connectorPane and draggablePane are added into nodeEditor by calling its method add(Pane).

*nodePane:* it is on the back. All MyNode nodes will be added into it.

*connectorPane:* it is on the middle. All connecting lines will be added into it.

*draggablePane:* it is on the front. All active nodes (draggable or pressable nodes) will be added into it. Active nodes would not be work if some other node will be front of it , because of it this pane is on the front.

*InputOnAction(); OutputOnAction(); SeparateOnAction(); CombineOnAction(); AddOnAction(); SubtractOnAction(); MultiplyOnAction(); DivideOnAction(); DistanceOnAction(); MaxOnAction(); MinOnAction(); EqualsOnAction(); GreaterOnAction():*

This methods are called when Edit>>Add>>name of node was pressed. It creates this node and call method addNewMyNode(this node).

*addNewMyNode(myNode):*It styling the myNode and adding every part of myNode into needed pane (nodePane, connectorPane and draggablePane).

## **Package window:**

It has two java classes: NodeEditor and Window

### **Window.java:**

It extends SplitPane, it is not necessary , it can extends just Pane. I make it extending SplitPane for my future plans. It has one method add(Node)

*add(node):* it makes node childe of window.

### **NodeEditor.java:**

UML Diagram of this class shown in figure 15.

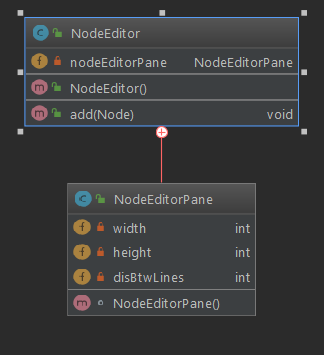


Figure 15

NodeEditor has class NodeEditorPane. NodeEditor estends ScrollPane, NodeEditorPane Extends Pane. In constructor NodeEditor() new NodeEditorPane() was setted to it’s content.

NodeEditor have only one method add(Node) which will add Node into childrens of the nodeEditorPane which is content of NodeEditor.

NodeEditorPane have 3 parameters: width = 2000, height = 2000, disBtwLines = 20.

In contructor of this class painted grid with distances between grids equal to disBtwLines, evert line is gray and every fifth line is black.

## **Package Node:**

UML Diagram of this package is shown in the figure 16.

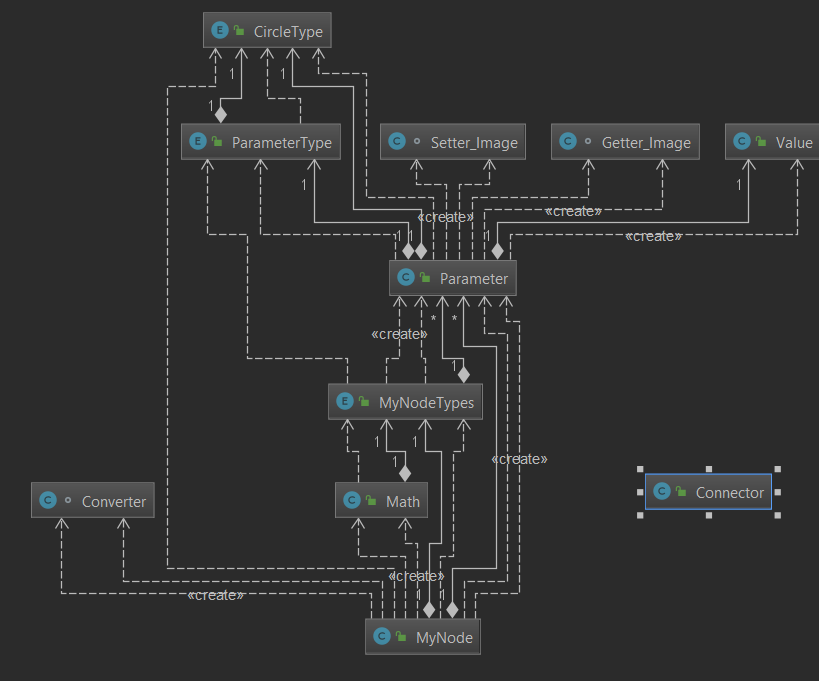


Figure 16

Node package has 8 classes and an image:

* MyNode.java
* Parameter.java
* Converter.java
* Math.java
* Getter\_Image.java
* Setter\_Image.java
* Value.java
* Connector.java
* image.PNG

### **MyNode.java:**

UML Diagram and file structure of this class is shown if the figure 17 and 18.

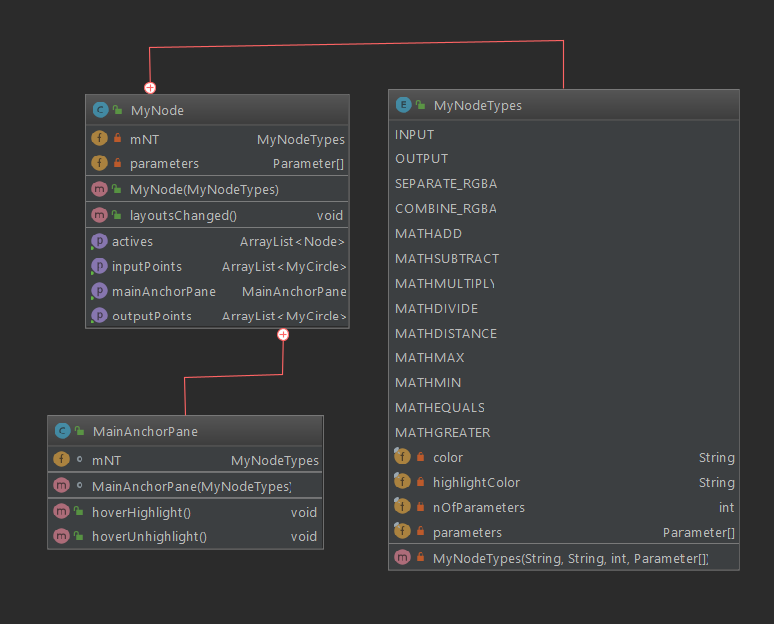


Figure 17

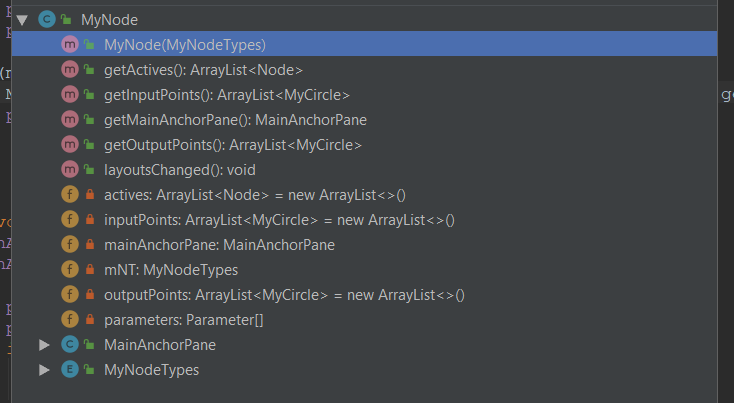


Figure 18

It has enum “MyNodeTypes” and inner class “MainAnchorPane”.

MyNodeTypes: It is a enum which describes a type of a Node.

Every enum we say type of the node have four attributes:

* String color: Color of node’s title bar.
* String hightlightColor: Color of node’s title bar when highlighted (it will highlight when hovered by mouse pointer)
* int nOfParameters: number of parameters.
* Parameter[] parameters: parameters.

public enum MyNodeTypes {  
 *INPUT*("#8d0000" ,"#f00" ,2, new Parameter[]{  
 new Parameter("Image" , ParameterType.*NAME*),  
 new Parameter("Image" , ParameterType.*IMAGE\_OUTPUT*)}),  
 *OUTPUT*( "#8d0000","#f00", 2, new Parameter[]{  
 new Parameter("Output" , ParameterType.*NAME*),  
 new Parameter("Image" , ParameterType.*IMAGE\_INPUT*)}) ,  
 *SEPARATE\_RGBA*( "#8d8d00" ,"#ff0" , 6, new Parameter[]{  
 new Parameter("Separate RGBA" , ParameterType.*NAME*),  
 new Parameter("Red" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("Green" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("Blue" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("Alpha" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("Image" , ParameterType.*IMAGE\_GETTER*)}) ,  
 *COMBINE\_RGBA*("#8d8d00" ,"#ff0" ,6, new Parameter[]{  
 new Parameter("Combine RGBA" , ParameterType.*NAME*),  
 new Parameter("Image" , ParameterType.*IMAGE\_SETTER*) ,  
 new Parameter("Red" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Green" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Blue" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Alpha" , ParameterType.*VALUE\_GETTER*)}) ,  
 *MATHADD*("#008d8d" ,"#0ff" , 5 , new Parameter[]{  
 new Parameter("Add" , ParameterType.*NAME*),  
 new Parameter("Value" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("" , ParameterType.*MATH*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) }),  
 *MATHSUBTRACT*("#008d8d" ,"#0ff" , 5 , new Parameter[]{  
 new Parameter("Subtract" , ParameterType.*NAME*),  
 new Parameter("Value" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("" , ParameterType.*MATH*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) }),  
 *MATHMULTIPLY*("#008d8d" ,"#0ff" , 5 , new Parameter[]{  
 new Parameter("Multiply" , ParameterType.*NAME*),  
 new Parameter("Value" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("" , ParameterType.*MATH*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) }),  
 *MATHDIVIDE*("#008d8d" ,"#0ff" , 5 , new Parameter[]{  
 new Parameter("Divide" , ParameterType.*NAME*),  
 new Parameter("Value" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("" , ParameterType.*MATH*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) }),  
 *MATHDISTANCE*("#008d8d" ,"#0ff" , 5 , new Parameter[]{  
 new Parameter("Distance" , ParameterType.*NAME*),  
 new Parameter("Value" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("" , ParameterType.*MATH*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) }),  
 *MATHMAX*("#008d8d" ,"#0ff" , 5 , new Parameter[]{  
 new Parameter("Max" , ParameterType.*NAME*),  
 new Parameter("Value" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("" , ParameterType.*MATH*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) }),  
 *MATHMIN*("#008d8d" ,"#0ff" , 5 , new Parameter[]{  
 new Parameter("Min" , ParameterType.*NAME*),  
 new Parameter("Value" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("" , ParameterType.*MATH*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) }),  
 *MATHEQUALS*("#008d8d" ,"#0ff" , 5 , new Parameter[]{  
 new Parameter("Equals" , ParameterType.*NAME*),  
 new Parameter("Value" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("" , ParameterType.*MATH*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) }),  
 *MATHGREATER*("#008d8d" ,"#0ff" , 5 , new Parameter[]{  
 new Parameter("Greater than" , ParameterType.*NAME*),  
 new Parameter("Value" , ParameterType.*VALUE\_SETTER*) ,  
 new Parameter("" , ParameterType.*MATH*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) ,  
 new Parameter("Value" , ParameterType.*VALUE\_GETTER*) });  
  
  
 private final String color;  
 private final String highlightColor;  
 private final int nOfParameters;  
 private final Parameter[] parameters;  
 MyNodeTypes(String color , String highlightColor , int nOfParameters, Parameter[] parameters){  
 this.nOfParameters = nOfParameters;  
 this.parameters = parameters;  
 this.color = color;  
 this.highlightColor = highlightColor;  
 }  
}

MainAnchorPane: It extends AnchorPane. It is a title bar of the node.

It has two methods:

* MainAnchorPane(MyNodeTypes mNT): it’s a constructor.
* hoverHighlight(): changes a color according to mNT
* hoverUnhighlight(): changes a color according to mNT

MyNode.java has attributes:

* MainAnchorPane mainAnchorPane: title bar
* ArrayList<MyCircle> inputPoints: value would input into the node via this point (red circle)
* ArrayList<MyCircle> outputPoints: value would output into the node via this point (red circle)
* ArrayList<Node> actives: array of points which are active (hoverable and draggable). It will be changed when some point dragged. See MyCirclesMakeDraggable.java
* MyNodeTypes mNT: type of the node.
* Parameter[] parameters: parameters of the node.

MyNode.java has methods:

* MyNode (MyNodeTypes mNT): constructor of MyNode. Create and design node according to mNT. It creates converter and math and sets parameters into it according to mNT:
  + For INPUT and OUTPUT there is no either converter either math
  + For SEPARATE and COMBINE it creates converter
  + For MyNodeTypes beginning with MATH it creates math
* layoutsChanged() : it refreshes coordinates of the node. It would be called when node dragged.
* getMainAnchorPane(): return mainAnchorPane
* getInputPoints() : return inputPoints
* getOutputPoints() : return outputPoints
* getActives() : return actives

### **Parameter.java:**

UML Diagram of it in the figure 20. File structure in the figure 19

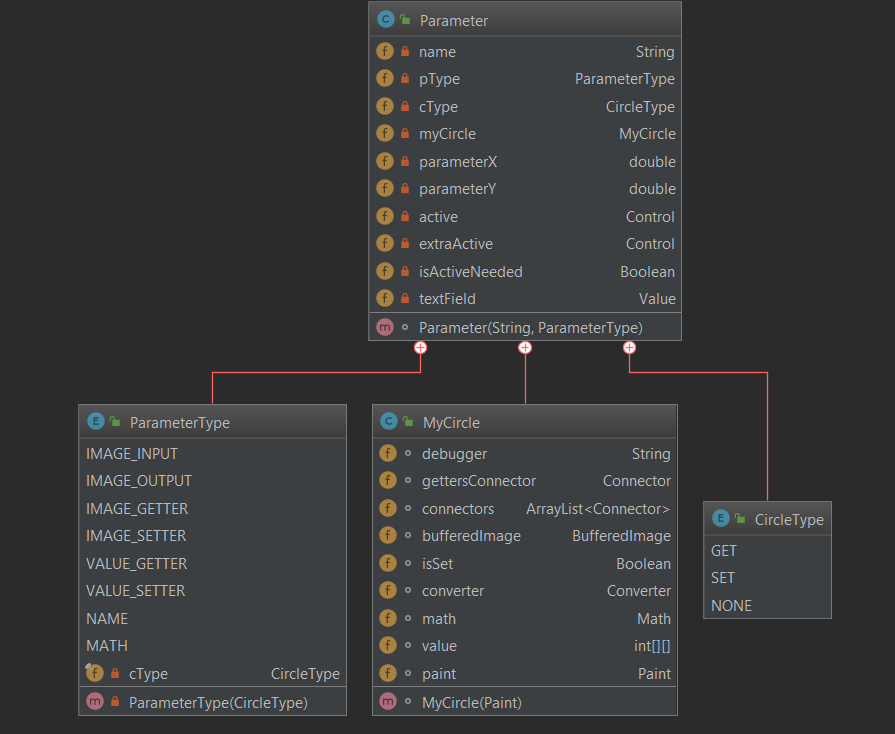
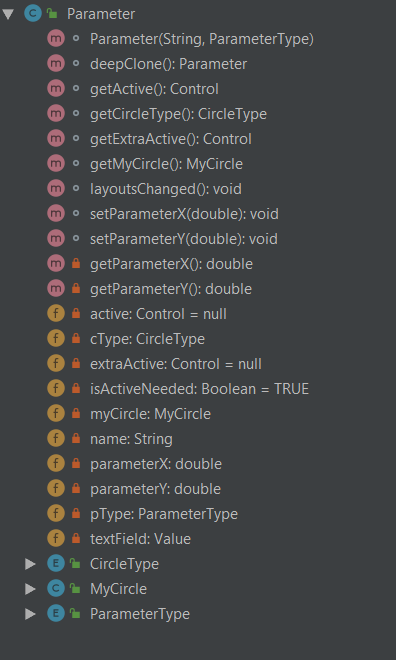


Figure 20

Figure 19

Parameter.java has inner class MyCircle and enums ParameterType and CircleType.

ParameterType: It is type of a parameter.

It has one attribute CircleType cType: type of a point of the parameter.

public enum ParameterType {  
 *IMAGE\_INPUT*(CircleType.*GET*),  
 *IMAGE\_OUTPUT*(CircleType.*SET*),  
 *IMAGE\_GETTER*(CircleType.*GET*),  
 *IMAGE\_SETTER*(CircleType.*SET*),  
 *VALUE\_GETTER*(CircleType.*GET*),  
 *VALUE\_SETTER*(CircleType.*SET*),  
 *NAME*(CircleType.*NONE*),  
 *MATH*(CircleType.*NONE*);  
 private final CircleType cType;  
  
 ParameterType(CircleType cType) {  
 this.cType = cType;  
 }  
}

CircleType:

public enum CircleType {  
 *GET*, *SET*, *NONE*}

MyCircle:

MyCircle has attributes:

* Connector gettersConnector: If it’s type is GET it must be have only one connector(It can input only one information). So it’s it. gettersConnector will used if MyCircle’s type is GET.
* ArrayList<Connector> connectors: array of connectors connected to this MyCircle
* BufferedImage bufferedImage: would be used if its Parameter’s type begins with IMAGE.
* Boolean isSet: Used for type GET
* Converter converter: It’s null. It was setted when converter was created in the constructor of MyNode.
* Math math: It’s null. It was setted when converter was created in the constructor of MyNode.
* int[][] value: would be used if its Parameter’s type begins with VALUE.
* Paint paint: Color of it. It will be changed when hovered.

MyCircle has methods:

* MyCirle(Paint): constructor
* Getters and settrers:
* setConverter(Converter converter): was called on the constructor of the MyNode.
* setMath(Math math): was called on the constructor of the MyNode.
* getBufferedImage(): returns image according to cType, pType and converter

if (converter == null) {  
 if (pType == ParameterType.*IMAGE\_INPUT*) {  
 if (gettersConnector.getSetterPoint() != null) {  
 System.*out*.println(debugger + "input");  
 return gettersConnector.getSetterPoint().getBufferedImage();  
 } else {  
 System.*out*.println(debugger + "ERROR not Connected");  
 return bufferedImage;  
 }  
 } else if (pType == ParameterType.*IMAGE\_GETTER*) {  
 if (gettersConnector.getSetterPoint() != null) {  
 System.*out*.println(debugger + "setter");  
 return gettersConnector.getSetterPoint().getBufferedImage();  
 } else {  
 System.*out*.println(debugger + "ERROR not Connected");  
 return bufferedImage;  
 }  
 } else {  
 System.*out*.println(debugger + "it must be image\_output");  
 return bufferedImage;  
 }  
} else {  
 System.*out*.println(debugger + "converter!=null");  
 BufferedImage result = converter.getImage();  
 System.*out*.println("returned");  
 return result;  
}

* getValue(): returns value according to converter, math, pType, cType, isSet.

if (converter == null) {  
 if (pType == ParameterType.*IMAGE\_INPUT*) {  
 if (gettersConnector.getSetterPoint() != null) {  
 System.*out*.println(debugger + "input");  
 return gettersConnector.getSetterPoint().getBufferedImage();  
 } else {  
 System.*out*.println(debugger + "ERROR not Connected");  
 return bufferedImage;  
 }  
 } else if (pType == ParameterType.*IMAGE\_GETTER*) {  
 if (gettersConnector.getSetterPoint() != null) {  
 System.*out*.println(debugger + "setter");  
 return gettersConnector.getSetterPoint().getBufferedImage();  
 } else {  
 System.*out*.println(debugger + "ERROR not Connected");  
 return bufferedImage;  
 }  
 } else {  
 System.*out*.println(debugger + "it must be image\_output");  
 return bufferedImage;  
 }  
} else {  
 System.*out*.println(debugger + "converter!=null");  
 BufferedImage result = converter.getImage();  
 System.*out*.println("returned");  
 return result;  
}

* setBufferedImage(BufferedImage bufferedImage)
* Connector getGettersConnector()
* Boolean isGetter()
* void setIsSet(Boolean set): if set is true a.e some connector setted, text field will be deleted(I just moved it into non visible side of NodeEditorPane). It must called if type is GET.
* List<Connector> getConnectors()
* layoutsChanged(): refreshes the position of MyCircle. Called when myCircle pressed or dragged.
* hoverHighlight(): changes color to color. Called when drag ended or hover exited.
* hoverUnhighlight() changes color to hightlightColor. Called when hovered.

Parameter.java has attributes:

* private String name;
* private ParameterType pType;
* private CircleType cType;
* private MyCircle myCircle;
* private double parameterX: position of parameter on the NodeEditorPane. Its original positron is on the MyNode. parameterX will be used for positioning active or extractive on to draggablePane and other.
* private double parameterY: like parameterX.
* private Control active: active part of parameter
* private Control extrActive: if there is second active part (Parameter.IMAGE\_OUTPUT)
* private Boolean isActiveNeeded: true if active needed.
* private Value textField. Text field.

Parameter.java has methods:

* Parameter(String name, ParameterType pType) : creates all parts of parameter according to pType. There setted active, extrActive, isActiveNeeded.
* Getters and setters:
* getActive()
* getExtraActive()
* getParameterX()
* getParameterY()
* setParameterX
* setParameterY
* getMyCircle()
* getCircleType()
* layoutsChanged(): called when layouts changed. Refreshes position of parameter and its parts.
* deepClone() : return new Parameter(name, pType). Clones from MyNodeTypes.parameteres. If you would not clone it only one parameter will be setted to nodes with same type.

### **Converter.java:**

It has methods:

* Converter(MyCircle imageMyCircle, MyCircle redMyCircle, MyCircle greenMyCircle, MyCircle blueMyCircle, MyCircle alphaMyCircle): contructor.
* getImage: combines redMyCircle greenMyCircle blueMyCircle alphaMyCircle and returns resulting image.
* getRedArray(): returns arrayList[width][height] of red values of imageMyCircle.
* getGreenArray(): returns arrayList[width][height] of green values of imageMyCircle.
* getBlueArray(): returns arrayList[width][height] of blue values of imageMyCircle.
* getAlphaArray(): returns arrayList[width][height] of alpha values of imageMyCircle.

### **Math.java:**

It has methods:

* Math(MyCircle IValueMyCircle, MyCircle IIValueMyCircle, MyNode.MyNodeTypes mNT): constructor.
* Int[][] getResultArray(): combining IValueMyCircle and IIValueMyCircle and applying some calculation according to mNT returns resulting array.

### **Getter\_Image.java:**

It esxtends Button. It is “Save” button. Also it has attributes “Open” button.

When “Save” buttons pressed creates javafx.stage.FileChooser and save image into chosen file.

EventHandler<ActionEvent> saveEventHandler = event -> {  
 BufferedImage bufferedImage = superMyCircle.getBufferedImage();  
  
 FileChooser fileChooser = new FileChooser();  
  
 //Set extension filter  
 FileChooser.ExtensionFilter extFilterJPG = new FileChooser.ExtensionFilter("JPG files (\*.jpg)", "\*.JPG");  
 FileChooser.ExtensionFilter extFilterPNG = new FileChooser.ExtensionFilter("PNG files (\*.png)", "\*.PNG");  
 fileChooser.getExtensionFilters().addAll(extFilterPNG, extFilterJPG);  
  
 //Show open file dialog  
 File file = fileChooser.showSaveDialog(null);  
  
 if (file != null) {  
 try {  
 ImageIO.*write*(bufferedImage, "png", file);  
 } catch (IOException ex) {  
 System.*out*.println("IOException" + ex.toString());  
 }  
 }  
};

When “Open” button pressed it save image into image.PNG and opens it:

EventHandler<ActionEvent> openEventHandler = event -> {  
 BufferedImage bufferedImage = superMyCircle.getBufferedImage();  
 System.*out*.println(bufferedImage.getWidth());  
  
 try {  
 File defaultFile = new File(getClass().getResource("image.PNG").getFile());  
 ImageIO.*write*(bufferedImage, "png", defaultFile);  
 Desktop.*getDesktop*().open(defaultFile);  
 } catch (IOException e) {  
 System.*out*.println("IOException" + e.toString());  
 }  
};

### **Setter\_Image.java**

It extends Button. It is button with name “Choose”. When it is pressed creates javafx.stage.FileChooser it gets choosen file.

It has attributes:

* Label imageName: it will set when “Choose” button pressed. It is used to set textField in the InputImage node’s text field where will be showed name of the choosen file.
* BufferedImage bufferedImage: it will set when “Choose” button pressed
* Parameter.MyCircle superMyCircle: it will set when “Choose” button pressed

It has methods:

* getIamgeName()
* setSuperMycircle()

When “Choose” button pressed:

EventHandler<ActionEvent> onAction = event -> {  
 FileChooser fileChooser = new FileChooser();  
  
 //Set extension filter  
 FileChooser.ExtensionFilter extFilterJPG = new FileChooser.ExtensionFilter("JPG files (\*.jpg)", "\*.JPG");  
 FileChooser.ExtensionFilter extFilterPNG = new FileChooser.ExtensionFilter("PNG files (\*.png)", "\*.PNG");  
 fileChooser.getExtensionFilters().addAll(extFilterJPG, extFilterPNG);  
  
 //Show open file dialog  
 File file = fileChooser.showOpenDialog(null);  
  
 try {  
 bufferedImage = ImageIO.*read*(file);  
 imageName.setText(file.getName());  
 superMyCircle.setBufferedImage(bufferedImage);  
 } catch (IOException ignored) {  
 }  
};

### **Value.java:**

It extends TextField. There can be written only integers. It has only one method getValue() which returns written value.

### **Connector.java:**

It extends Line.

It has two attributes MyCircle getterPoint and setterPoint.

It has methods:

* setGetterPoint
* setSetterPoint
* getSetterPoint
* layoutsChanged: It positions the line’s ends to getterPoint and setterPoint.
* MyCircle dragStart;
* MyCircle dragEnd;
* Line line;

## **Package mouseGestures:**

### **MyCirclesMakeDraggable.java:**

It has inner class MyCircleMakeDraggable (without ‘s’). It is used to make one MyCircle draggable and hoverable.

It collects MyCircles and makes they draggable and hoverable.

It has attributes:

* Pane drawingPane: where connectors and lines will be drawed. It was setted by connectorPane in thenController.java.
* List<MyCircle> inputPoints: MyCircle with type SET
* List<MyCircle> outputPoints: MyCircle with type GET
* List<MyCircle> allPoints: All MyCircles.
* List<MyCircle> activePoints: first it’s allPoints. When MyCircle with type SET was dragged it’s outputPoints, when MyCircle with type GET was dragged it’s inputPoints.

Make hoverable:

private void hoverHighlight(MyCircle myCircle){  
 myCircle.hoverProperty().addListener(new ChangeListener<Boolean>() {  
 @Override  
 public void changed(ObservableValue<? extends Boolean> observable, Boolean oldValue, Boolean newValue) {  
 if (newValue) {  
 myCircle.hoverHighlight();  
 } else {  
 myCircle.hoverUnhighlight();  
 }  
 }  
 });  
}

It used eventhandlers:

* Mouse pressed: prepares for dragging. Line drawed.

private EventHandler<MouseEvent> onMousePressed = event -> {  
 dragStart = draggableMyCircle;  
 if (dragStart.isGetter()){  
 drawingPane.getChildren().remove(dragStart.getGettersConnector());  
 dragStart.setIsSet(Boolean.*FALSE*);  
 }  
 line = new Line();  
 line.setStroke(Color.*WHITE*);  
 drawingPane.getChildren().add(line);  
};

* Mouse dragged: change line according to mouse’s position

private EventHandler<MouseEvent> onMouseDragged = event -> {  
 dragStart.hoverHighlight();  
  
 line.setStartX(dragStart.getCenterX());  
 line.setStartY(dragStart.getCenterY());  
 if (dragEnd==null){  
 line.setEndX(event.getX());  
 line.setEndY(event.getY());  
 }  
 else if (activePoints.contains(dragEnd)){  
 line.setEndX(dragEnd.getCenterX());  
 line.setEndY(dragEnd.getCenterY());  
 }  
};

* Mouse released: if released on MyCircle with opposite type line becomes converter. If not just delete it.

private EventHandler<MouseEvent> onMouseReleased = event -> {  
 drawingPane.getChildren().remove(line);  
 dragStart.hoverUnhighlight();  
 if (dragEnd != null){  
 Connector connector;  
 if (dragStart.isGetter()){  
 connector = dragStart.getGettersConnector();  
 connector.setGetterPoint(dragStart);  
 connector.setSetterPoint(dragEnd);  
 dragStart.setIsSet(Boolean.*TRUE*);  
 }  
 else {  
 drawingPane.getChildren().remove(dragEnd.getGettersConnector());  
 connector = dragEnd.getGettersConnector();  
 connector.setGetterPoint(dragEnd);  
 connector.setSetterPoint(dragStart);  
 dragEnd.setIsSet(Boolean.*TRUE*);  
 }  
 drawingPane.getChildren().add(connector);  
 }  
 activePoints = allPoints;  
};

* Drag detected: to make myCircles starting full drag (startFullDrag())

private EventHandler<MouseEvent> onDragDetected = event -> {  
 MyCircle cell = (MyCircle) event.getSource();  
 cell.startFullDrag();  
  
 if (inputPoints.contains(dragStart)){  
 activePoints = outputPoints;  
 }  
 else if (outputPoints.contains(dragStart)){  
 activePoints = inputPoints;  
 }  
  
};

* Mouse drag entered: to make hoverability work when mouse dragged

private EventHandler<MouseEvent> onDragEntered = event -> {  
 if (activePoints.contains(draggableMyCircle)){  
 dragEnd = draggableMyCircle;  
 dragEnd.hoverHighlight();  
  
 line.setEndX(dragEnd.getCenterX());  
 line.setEndY(dragEnd.getCenterY());  
 }  
};

* Mouse drag exited: to make hoverability work when mouse dragged

private EventHandler<MouseEvent> onDragExited = event -> {  
 if (activePoints.contains(draggableMyCircle)){  
 if (dragEnd!=null)dragEnd.hoverUnhighlight();  
 dragEnd = null;  
 }  
};

### **MyNodesMakeDraggable.java:**

It has inner class MyNodeMakeDraggable (without ‘s’). It is used to make one MyNode draggable and hoverable.

It collects MyNodes and makes they draggable and hoverable.

MyNodeMakeDraggable:

It has two attributes:

* MyNode draggableMyNode: whole node will move when draggableMainPane dragged
* MainAnchorPane draggableMainPane; it will become hoverable.
* double mouseLayoutX\_OnMyNode
* double mouseLayoutY\_OnMyNode;

make hoverable:

draggableMainPane.hoverProperty().addListener(new ChangeListener<Boolean>() {  
 @Override  
 public void changed(ObservableValue<? extends Boolean> observable, Boolean oldValue, Boolean newValue) {  
 if (newValue) {  
 draggableMainPane.hoverHighlight();  
 } else {  
 draggableMainPane.hoverUnhighlight();  
 }  
 }  
  
});

It has eventhandlers:

* Mouse pressed: prepare for dragging.

private EventHandler<MouseEvent> onMousePressedEventHandler = event -> {  
 mouseLayoutX\_OnMyNode = event.getSceneX() - draggableMyNode.getLayoutX();  
 mouseLayoutY\_OnMyNode = event.getSceneY() - draggableMyNode.getLayoutY();  
 draggableMyNode.layoutsChanged();  
};

* Mouse dragged: change position of node according to mouse position.

private EventHandler<MouseEvent> onMouseDraggedEventHandler = event -> {  
 draggableMainPane.hoverHighlight();  
 draggableMyNode.setLayoutX(event.getSceneX() - mouseLayoutX\_OnMyNode);  
 draggableMyNode.setLayoutY(event.getSceneY() - mouseLayoutY\_OnMyNode);  
 draggableMyNode.layoutsChanged();  
};

* Mouse released:

private EventHandler<MouseEvent> onMouseReleasedEventHandler = event -> {  
 draggableMyNode.layoutsChanged();  
 draggableMainPane.hoverUnhighlight();  
};

# **REFERENCE**

JavaFx Tutorials from oracle:

<https://docs.oracle.com/javafx/2/>

Javafx Oracle Documantations:

<https://docs.oracle.com/javase/8/javafx/api/toc.htm>

StackOverFlow:

<https://stackoverflow.com/>

GeeksforGeeks Image processing:

<https://www.geeksforgeeks.org/image-processing-java-set-1-read-write/>