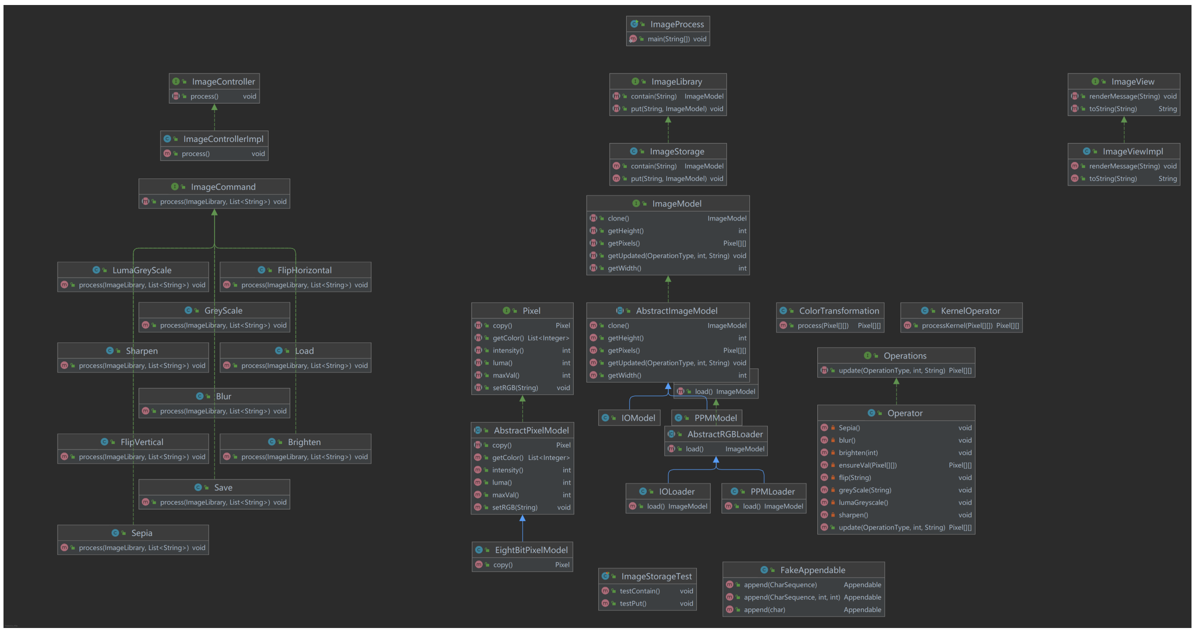
Image Processing V3.0

Overview of the Program:

This program allows the user to process different image types with some basic commands. The currently supported image type is .ppm. And the program allows the user to change the brightness, flipping vertically or horizontally and changing to greyscale different component ways.

Program Structure



ImageProcess

This is the main class that we can run this program in two different ways.

```
1. Using a script by calling -file / -f

2. Using terminal command lines by leaving it blank. (Strar the GUI)

3. Using a text-based input by callig -text
```

The imageprocess takes in a model, view and controller.

```
Error handled:
1. Invalid function call (calling things except: -help/-h, -file/-f or blank)
```

How an command is executed

1. Setting new Commands in HashMap

Put an command value and key in the commands HashMap that is storage all commands the controller supports. The key should be a String as the name of this command that user will use, and the value should be the specific class for commands that implements the ICommand interface, like

```
knownCommands.put("[Command Name]", (Scanner s) -> {
  return new [Command class]([Possbile arguments]);};
```

2. Creating the new Command class

This new created class has to implement the ICommand interface and implements the process method. The constructor of the class depends on the information it needs. If the input from process is null, IllegalArgumentException will be throwed.

```
// Template
public class [ClassName] implements ImageCommand {
    String fileNameIn;  // This is for the search of the model.
    String fileNameOut; // This is for the storage in the library.
    [Possible variables for constructor]

  public [ClassName](String fileNameIn, String fileNameOut, [Possible variables
for constructor]) {
      this.filaNameIn = fileNameIn;
      this.filaNameOut = fileNameOut;
      [Possible variables for constructor]
  }

  @Override
  public void process(ImageLibrary library, List<String> stringCommands)
          throws IllegalArgumentException {

    ImageModel model = library.contain(fileNameIn);
    if (model == null) {
      System.out.println("Cannot find image");
      throw new IllegalArgumentException("Cannot find image");
    }
    model.getUpdated(Operations.OperationType.[Operation type from command],
[Possible needed Integer], [Possible needs String]);
    library.put(this.fileNameOut, model);
    System.out.println(fileNameIn + " has been greyscale as " + fileNameOut);
  }
}
```

3. Add new enum in Opeations interface

```
enum OperationType {
  Save,GreyScale, Flip, Brighten, Blur, Sharpen, LumaGreyScale, Sepia
  // Current supports commands type.
}
```

Every new added command types should add a specific enum in the OperationType and also implement in the Operator class.

How to implement:

a. Add one case in the switch part.

```
switch (type) {
  case [New added enum name with command]:
    this.[Implement method names for this specific command]([Possbiel passed in
arguments]);
    break;
```

b. Implement the command method.

```
private void [Implement method names for this specific command]([Possbile passed
in Arguments]]) {
    [Implementation code];
  ensureVal(this.pixels); //Ensure the value does not exceed the 0-255 boundary.
}
```

Structure

Controller

ImageController

```
This is the controller interface that contains the method of starting the whole
program as process().
```

-ImageControllerImpl

> This class is the implementation of the controller interface. It will takes in a model, a view
>
> and a readable input (from user). Command lines will be read and executed here.

```
Error handled:
1. The command fails. Every failed commands will return "Action failed" and
specific information about the failuremessage rendered to the view. .
2. Null input from the Main. Null input for constructor will throw Exception and
message rendered to the view.
3. Rendering message fails. If the rendering message fails, it will throw error.
```

Model

storage:

ImageLibrary:

This is the interface for image storage. It allows to put a model with name into the library and make copy of an existing image model in the library.

-ImageStorage

This class implements the ImageLibrary as HashMap for library and allows users to store different types of image

models and make a copy of the given name.

image:

ImageModel:

This is the interface for different images models. It allows different operations to the model and give back copies of the information about the image.

-AbstractImageModel:

This is the abstracted class that implements the ImageModel interface. The abstract class

stores the image as an 2-D array of Pixel. Each pixel will represent each pixel from the image.

This class allows changes on brightness, greyscale by components, flip, clone and giving back

copies of some basic information. This model is compatible for different types of images.

It allows the alpha channel input with the extension of pixel.

```
Error handled:
1. The prohibited height and width and a null pixel[][] input.
2. Wrong input for fliping.
```

-PPMModel

PPModel is specific for the PPM image. It extends from the AbstractImageModel. There is

no change over the abstract class.

pixel

Pixel

This is the interface for representing different types of pixels (3-channels, 4-channels pixels). It allows to return different value component, set a new value for itself and make a hard copy of itself.

-AbstractPixelModel

This is an abstract class that implements the Pixel interface. This class stores the rgb

information as a arraylist, which is flexible on the length. The length is currently limited in the abstract class as 3 since the only allowed type is ppm. Its methods allow it to make a hard

copy of it self (this method is abstract), make changes on the value-component.

-EightBitPixelModel

This class extends from the AbstractPixelModel and override the abstract method copy().

```
Error handled:
1. The length of pixel array input of ppm file is longer than 3 intergers.
```

Operations

This is the interface that contains an enumeration of all supported commands and the inherited Operator class will implement all those supported commands as void method that mutate the pixels. Commands class will create operator and pass into this interface.

- Operator

This is a class that implements the Operation. This class contains all the command's implementation that will mutate the copy of the model's pixels as new pixels.

```
Handled Error:
1. The passed in arguments from update method is empty or invalid.
2. The updation fails.
```

- ColorTransformation

This is a class that uses for applying a filter to the pixels that can alter the color on the pixels as the projection in the given filter (2-D arrays) (as two matrix's multiplication).

- KernelOperator

This is a class that uses for applying a kernel to the pixels that can achieve some blur, sharpen or more image effect changes that can be processed by kernel projection.

```
Handled Error:
1. The passed in kernel or filter is empty.
2. The passed in arguments are empty.
3. The updation fails.
```

View

ImageView:

This is an interface for view. It allows to change a image with name to a writable string and render messages to the output.

-ImageViewImpl:

This class implements the ImageView. This is a generalized view class that can accept different types of image model. It can only output the ppm now. But just by adding information about head lines of png or jpg file, it will be adaptable.

```
Error handled:
1. The library or the appendable output is null.
2. The failure of rendering message.
3. The failure of finding the given name image in the library.
```

Important Updates for ImageProces V2.0

1. More functions added

```
a. Supports more image types(JPEG, PNG, BMP, WEBMP, GIF, PPM, ...)
b. Supports the tranformation between different image types.
(IMPORTANT: Transfering a png file with transparency to other types will cause
the pixel that has 0 on alpha channel as complete transparent becomes a white
pixel)
c. Supports functions as blur (Gaussian blur), sharpen, speia color and greyscale
color.
```

2. Justification on structure changes (v2.0)

```
In order to achieve better functionality that allows the publisher to add new
functions after the publishment of this application without touching the model, a
new interface Operations and its class Operator that implements it are added.
They functions as the main pixels processor in the whole function. As a result,
all new added functions will be added toward operations instead of the model. The
model becomes purely the storage of a image's basic information and its pixels.
As a result, in the future, if any new functions are needed, the model will not
be alerted.
The basic logic of Operations begins with the change in the controller. Instead
of calling the model's functions, the controller's commands will create a new
Operator and call the update method that updates the copyed model's pixels. The
update method called in the controller's commands will give the Operation type as
what the command needs and also possible needed information for updating. Then,
the command will go to the update method. With the given operation type, the
switch function will lead to the needed method in the operator. Eventually, the
update will return a updated pixels to the controller. And the controller will
put the copyed model with updated pixels into the library with given name.
```

3. Justification on New files

```
The part that is changed is in the View package since we implement GUI. The GUI
enables the user to click different buttions and bars to choose the action they
want instead of using text input in the terminal. The new ImageHistogram file is
responsible for creating a histogram panel that takes in image information on
pixels and create a histogram for it basing on the frequency of color values'
appearance. The new GUIImpl still extends the ImageView and implements it. This
class is for constructing the whole JFrame GUI.
```

Citation for the Image used:

Clipart Library. “Frowny Face Pictures #1269551.” *Clipart Library*, Clipart Library, clipart-library.com/clipart/8ixMR94ip.htm.

Han, Steven. “Aerial View Of Shanghai Skyline At Dusk.” *Photos.com*, Getty Images, 16 Oct. 2020, photos.com/featured/aerial-view-of-shanghai-skyline-at-dusk-steven-han.html.]

Clipart Library. “Leopard PNG Transparent Images #2387070.” *Clipart Library*, Clipart Library, clipart-library.com/clipart/Leopard-Download-PNG.htm.