MyPhotoshop — Subject

version **#1**

Yet Another Kind of Assistant 2017 <yaka@yaka.epita.fr>

Copyright

This document is for internal use at EPITA (website) only.

The use of this document must abide by the following rules:

- ▷ You downloaded it from the assistants' intranet.*
- ▷ This document is strictly personal and must **not** be passed onto someone else.
- ▷ Non-compliance with these rules can lead to severe sanctions.

Contents

* https://acu.epita.fr

Obligations

*Obligations are **fundamental** rules shared by all subjects. They are non-negotiable and to not apply them means to face sanctions. Therefore, do not hesitate to ask for explanations if you do not understand one of these rules.*

Obligation #0: Your submission's root must contain an `AUTHORS` file. Its format is described in *Project data*. If this file is missing, you get a zero.

Obligation #1: **Cheating**, as well as sharing source code, tests, test tools or coding-style correction tools is **strictly forbidden** and penalized by not being graded, being flagged as a cheater and reported to the academic staff.

Obligation #2: If you do not submit your work before the deadline, your grade will be replaced by F.

Obligation #3: Your submission repository must be **clean**. Except for special cases, which (if any) are **explicitely** mentioned in this document, an *unclean* repository may contain:

- binary files;[1]
- files with inappropriate privileges;
- forbidden files: `*~`, `*.swp`, `*.o`, `*.a`, `*.so`, `*.class`, `*.log`, `*.core`, etc.;
- a file tree that does not follow our specifications (see *Project data*).

Submitting an unclean repository lowers your final mark.

Obligation #4: All your files must be encoded in ASCII or UTF-8 without BOM.

Obligation #5: If a function, a command or a library is not *explicitly* authorized, it is **forbidden**. Abuses may lead to a sanction.

Obligation #6: When examples demonstrate the use of an output format, you must follow it scrupulously.

Advice

- ▷ Read the *whole* subject.
- ▷ If the root of your submission contains a `bootstrap` file, it will be executed first. This behavior will then be applied to an eventual `configure` file. Therefore, these files shall be executable.
- ▷ If the slightest project-related problem arise, you can get in touch with the assistants.

  Post in the `epita.cours.java` **newsgroup** (with the `[MPH]` tag) for questions about this document, or send a **ticket** to `<yaka@yaka.epita.fr>` otherwise.
- ▷ In examples, `42sh$` is our prompt: use it as a reference point.
- ▷ Do **not** wait for the last minute to start your project!

[1] If an executable file is required, please provide its sources **only**. We will compile it ourselves.

Project data

Instructors:

- Adrien Toubiana <toubia_a@yaka.epita.fr>
- Nils Layet <layet_n@yaka.epita.fr>
- Tania Situm <situm_t@yaka.epita.fr>

Important dates:

start:	Monday, April 18, 2016	09:00 PM
end:	Sunday, May 8, 2016	11:42 AM
duration:	19 days, 14 hours, 42 minutes	

Dedicated newsgroup: `epita.cours.java with [MPH]`

Members per team: 1

Languages & compilers:

Java *with* Java v.1.8 ➛ `javac`

Directory tree[1] of your submission:

`./src/` *	Contains your source files.
`./tests/` *	Contains everything related to your tests.
`./TODO` *	Lists what still needs to be implemented. It must be updated regularly.
`./README` *	Describes your project, its algorithms and how to use it, in a correct English.

`./AUTHORS` *	Lists the project authors' logins (leader first) as follows: an asterisk `*`, a space, a *login* (e.g. `login_x`) and a newline. Thus:
	`42sh$ cat -e AUTHORS`
	`* login_x$`
	`42sh$`

[1] Asterisk-marked files are mandatory. The dot "`.`" is the root of your repository. This list is not exhaustive!

1 Introduction

At the end of the next three weeks, you will have built a full-featured application by yourself. The myPhotoshop project aims to gather what you have learned last week. For those who do not already know what myPhotoshop is, your work is to implement an image processing platform and its graphical interface in Java. This project is really fun, however, you should consider doing it seriously. You could easily have a good grade at this project if you produce a **high-quality** work.

2 Objectives

First of all, here is a **non-exhaustive** list of the main objectives of this project:

- build a GUI based on Swing;
- save and load your myPhotoshop projects;
- package your filters in `*.jar` files and load them at runtime;
- understand and implement a software architectural pattern;
- make your application multi-threaded.

3 Mandatory

The mandatory part forces you to build a robust application. You will learn a lot of things here and be well-prepared to implement all the features you want. Do it seriously and the rest of the project will be painless.

Not having the mandatory will lower your grade drastically. Therefore, do not forget to have everything complete and working before jumping to the free features.

3.1 The Swing GUI

3.1.1 Minimal architecture

The Graphical User Interface of your application must be built using the Swing library. Swing is a set of Java classes from the Java Foundation Classes (JFC), a framework that lets you build, guess what, graphical applications. Even if you are completely free to choose what your application will look like, we have some expectations. Consequently, a minimal interface must consist, at least, of:

- a menu;
- a dedicated space to display an image;
- an insert where you display the history of the project.

Finally, the default interface must be written in well-formed **English**.

3.1.2 MVC design patterns and event handling

Swing is by nature a Model-View-Controller (or MVC) platform. As you will see, Swing is very easy to understand and manipulate, however, things can quickly go wrong if you don't *think* your interface. That's why we want you to base your application on the MVC design pattern.

We will check during the defense that the architecture of your application follows the MVC rules. Your application needs to implement at least three interconnected parts **with clarity**: - Model - View - Controller

Moreover, you should have already noticed that the Model needs a simple way to notify the View of any change in its state. Thus, we want the MVC in your application to rely on the Observer design pattern or any other event handling pattern. However, we highly recommend you to use the Observer in your interface.

3.2 Image handling

To be usable, your application must handle several image formats. We expect your application to be able to load and save images of the following formats:

- Bitmap;
- JPEG;
- GIF;
- PNG.

3.3 Filters

Since your application **is** an image processing tool, you should implement some filters. All the filters you will create **must not** be hard-coded into your Java application. You will have to develop each filter in a separate class which will implement the `Filter` interface we provide you with. Once compiled, all your filters will turn into `*.class` files and have to be gathered in one of the two following sets of **filters**:

- `BasicFilters.jar` that will contain all the mandatory filters;
- `BonusFilters.jar` that will contain your extra filters.

To sum up, you have to load some `.jar` files at runtime containing your filters which are implementing the `Filter` interface.

All your filters must be developed independently from your application. It is always a good practice to have your code as independent as possible. In your case, it must be possible to launch your filters on a different implementation of myPhotoshop, and your application must be able to launch any filters that follows the Filter interface.

3.3.1 Filters architecture

In addition to the `./src/{...}` directory, we ask you to make another directory for your filters:

```
./FilterSource/
    |-- src/
        |-- filter/
            |-- basic/
            |-- bonus/
            |-- Filter.java
```

The above schema should be sufficiently self-explanatory, but do not hesitate to call an assistant if you have any doubt.

Speaking of mandatory filters, here is a list of all the filters you will need to implement:

Invert Inverts the colors of the image
Binary Turns your image into a black and white image
Grayscale Turns your image into a grayscale version of it
Rotate left Rotate your image of +90 degrees
Rotate right Rotate your image of -90 degrees
Vertical flip or *horizontal symmetry* makes the upside down
Horizontal flip or *vertical symmetry* makes the left-side right

Your application must load every filter located in the `./filters/` folder. You can consider all the filters as well-formed, i.e. you don't have to handle any kind of errors. All the `*.class` files inside your filters JARs will implement the `Filter` interface.

3.4 History

Every time you apply a filter on an image, it must be possible to cancel what you have done. Conversely, it must be possible to *redo* what you've canceled.

3.5 Project handling

The load/modify/save process does not let you keep track of the history. To do so, you will have to serialize your image **and** the history of all its transformations. Obviously, you will have to implement the reverse operation, i.e. the loading of your project so that you can retrieve your image and its history. Your project files must be saved under the `*.myPSD` extension.

To sum it up, you must implement:

- the **creation** of a new project with or without a base image;
- the **loading** of an existing project;
- the **saving** of the current project;
- the **closing** of the current project.

3.6 Threading

It would be a shame if your application froze each time you apply a filter. To avoid this annoying effect, when a filter is used, the computation must happen in a **separate thread**.

Some filters you will write can take a while to end. Thus, we ask you to set up a *Cancel* functionality that lets the user...cancel the filter computation.

3.7 Project tabs

Your application must be capable of handling several projects opened at the same time in a single window using tabs for navigation to switch between them.

By default, your application will open images or projects as tabbed documents, and it switches between each of them when clicking on the desired tab. Each tab must display the name of the project and be able to be closed independently.

3.8 Mandatory features

In addition to the base mandatory part that your project must contain, you have to add special features to your project. The features are assigned to a level, according to their difficulty. You are required to implement enough levels to get a total of 6 to fulfill the mandatory (for example: level 3 + 3 * level 1 = 6). Furthermore, you can choose any features from any categories, regardless of the previous feature implemented.

Be careful: if a feature is not fully implemented, it will not be taken into account at all. Finally, if you choose higher levels, you will be rewarded accordingly.

3.8.1 Image

Open / Closing images (level 2)

- Open rights are handled
- Bad files are handled (wrong format, corrupt files, ...)
- Open an image from the clipboard
- Ability to open several files at the same time
- Previsualization of the image in a FileChooser before opening
- Ability to close all files in the same time
- Safe quit of the application (pop-up in case of change)

Saving (level 2)

- Written rights are managed
- Save / Save as opens a FileChooser
- Save is enabled only if a modification has been done
- A pop-up opens in case of existing file
- Compression is achieved when saving a project

Layers (level 3)

- Layers are orderable, duplicable and mergeable
- Opacity of a layer is editable
- Layers can be hidden
- Apply a filter on a layer only
- Apply a tool on a layer only

Gif (level 2)

- Gif images can be opened
- Frames of the gif can be displayed and edited separately
- Transparency in gif is handled

Histogram (level 1)

- View and edit the RGB histogram of the image
- View and edit the luminosity histogram of the image

3.8.2 Tools

Selection (level 3)

- Square selection
- Oval selection
- Free selection (with the mouse)
- Select all
- Deselect all
- Selection inversion
- Selection by contiguous color (with editable threshold)
- Apply a filter only in a selection

Pencil (level 1)

- Pencil size, shape, color and opacity are editable
- Layers / history are updated during drawing

Eraser (level 1)

- Opacity is editable
- If working with layers, only the current layer must be erased

Zoom / Pan (level 2)

- Zoom with the scroll wheel (with the origin at the mouse position)
- Percentage of the zoom is displayed and can be edited
- Pan the image view with the mouse

Rescaling (level 1)

- The image can be scaled to given values
- The image can be scaled to the canvas
- A rescaling algorithm must be used

Polygons (level 2)

- Rectangle tool
- Rectangle with rounded corner (editable radius) tool
- Oval tool
- Regular polygons (with editable number of vertices)

Text (level 1)

- Ability to write text anywhere on the image
- Font, size, color are editable

Convolution matrix (level 2)

- The size, scale and values of the matrix are editable
- Multi-threaded application of the matrix
- Choose the color to change in a convolution matrix
- Predefined matrices are proposed (emboss, blur, ...)

3.8.3 Interface

I18N (level 2)

- New languages are easily addable (e.g. from a configuration file)
- Language can be changed at runtime
- 3 languages are available

Look and feel (level 1)

- Look and feel is editable at runtime

Better history (level 2)

- Previous states of the history are displayed in thumbnails
- The size of the history is editable
- Ability to jump to a specific state
- The current state of the history is highlighted

Better filters (level 1)

- A loading animation is displayed when a filter is being applied
- A notification is displayed when the filter is applied

Status bar (level 1)

- A status bar shows real time information
- All actions are displayed (image saving, tool selection, undo, ...)

Splash screen (level 1)

- A splash screen is shown when the program starts up
- A loading bar shows the loading status (not a fake one!)

Misc interface settings (level 3)

- Ability to copy / cut / paste in the project
- Inconsistent buttons are disabled
- Context menu (right-click) is used to quickly access some actions
- Drag and drop from the asset directory in a JTree
- The position of the mouse (relatively to the image) is displayed

3.8.4 Misc features

Client / server mode (level 3)

- Ability to open a project on multiple clients at the same time
- Clients see all modifications in real time
- Modifications can be saved on the local machine or on the server.

4 Free features

The mandatory part of this project is not sufficient enough to get the best grades. Your grade and your pride will only be transcended by doing this free part. You are **totally** free to implement everything you want. The only two conditions are:

- Post a news explaining all the features you have implemented;
- **Do not** post a news containing a feature that was already announced in previous news.

We do not prohibit you from doing the same features as your friends, you simply have to list the features which were not already described by another student before you. **Penalties** will be applied if you don't respect this last rule.

Obviously, the more your features are useful and well-implemented, the better you will be rewarded. For those who don't know where to start, you can pick up any feature from the list of mandatory features **if it's not already in your mandatory**. Furthermore, here is a list of some features, not ordered by difficulty, that you should consider:

- Filter preview
- Project-file compression
- Apply filters through the pencil
- Mnemonics accelerators
- Logging system
- Handling *exotic* color spaces
- Saving / loading the user settings
- Web interface to display the image
- Editing with collaborators over the network

- Cloud storage integration (Google Drive, OneDrive…)
- Steganography
- Completely guided interface with help
- Personalized icons and buttons
- Implementation of other design patterns (*State* maybe)
- All Adobe Photoshop tools…

To avoid you a lot of pain, we highly discourage you to try to handle the original `.PSD` format. But if you do so, post a news.

5 Given tools

5.1 ImagePanel

`ImagePanel` is a wrapper class around the `BufferedImage` class that we provide. When you will try to serialize your projects, you will realize that a `BufferedImage` object cannot be serialized. `ImagePanel` helps you solving this problem. However, this class does not necessarily suit your needs. Feel free to modify it since it is just a *starter* file.

5.2 Filter

As we explained above, all your filters have to implement the `Filter` interface. It ensures you to have a method to retrieve the name of your filter and another one to actually apply the filter on a `BufferedImage` and to get a modified one.

Remember that it is **not possible** to modify this interface: all your filters must be working flawlessly on a totally different implementation of this project.

6 Conclusion

Again, we hope you will enjoy this project, make it fun and usable! A last advice: choose to build things that actually work instead of a lot of unusable features, and think your interface before you start.

You've got to code before you can sleep.