Assignment 6: Image Processing (Part 2)

Due Jun 19 by 8:59pm **Points** 0

LIME: Layered Image Manipulation and Enhancement (Part 2)

In part 2 of this series, you will some new image processing operations in your application, add a major feature and implement a text-based view.

1 Support for conventional file formats

Although PPM is a simple and standard image format, it is not very popular. In order to make your application more usable, you will add support to import and export images in popular file formats, such as JPEG and PNG. Look at the **ImageIO class (https://docs.oracle.com/javase/7/docs/api/javax/imageio/ImageIO.html)** in Java to see how one can read and write images of many popular file formats. Find a way to incorporate this into your application.

2 Layering

2.1 Introduction

Programs like Photoshop (or Gimp, a free, open-source competitor) are based on "layers" of images. Each layer is like an image of its own. All layers of an image have the same dimensions (equal to the dimensions of the image itself). This allows the user to create complicated workflows, such as (not all of them will be supported in our application):

- "Rainbow filter" on Facebook:
  1. Load a regular image as one layer.
  2. Load a rainbow image as another layer.
  3. Blend the two layers.
- Astrophotography: photographing celestial bodies too dim to capture in a single picture
  1. Take several pictures of a dark night sky.
  2. Load them into separate layers.
  3. Blend them by averaging the value for every pixel across layers. This effectively reduces noise in the image, which is a huge problem for photos with very low light.
- Better portraits for modeling: (A portrait is a photograph that shows only one person, against a background)

1. Load the portrait picture in a layer.
2. Select the pixels that are part of the person (there are many tools available for this, ranging from manual painting to AI-based tools). This is the "foreground", and all remaining pixels are "background".
3. Copy foreground and background pixels into separate layers, with the "unused" pixels in each layer being fully transparent.
4. Sharpen the foreground (make outlines crisper). Brighten image if necessary.
5. Blur the background layer.
6. Blend the foreground and background layers into a single final layer. This produces a prominent foreground "popping" against a blurry background.

Any image processing operation can be applied to any individual layer (e.g. by setting a layer as "current"). Layers can be turned on (visible) or off (invisible/transparent). Layers are stacked on each other, and often given meaningful, user-defined names. Normally when one is viewing a multi-layered image in Photoshop, one is actually viewing the top-most visible layer. A layer may be added (blank, or a copy of an existing layer) or removed.

2.2 Saving a multi-layered image

A multi-layered image can be saved simply as a collection of files: one for each layer (as regular images), and one (text) file that stores the locations of all the layer files.

3 Image manipulation by batch commands

Until now the "driver" of your program may have been a meaningless `main` method, or a test. A better way would be to allow the user to load an image, apply various effects to it and save the results. One way to accomplish this is if the user can provide input in a "script", as follows:

```
create layer first
create layer second
current first
load manhattan.png #load it in layer first
blur
save manhattan-blur.ppm
current second #make another layer, this is topmost and visible
load manhattan.ppm #load it in layer second
sepia
save manhattan-sepia.jpg #save second layer because it is topmost visible
invisible second #make second layer (topmost) invisible
save manhattan-blur-2.jpg #save first layer because it is the topmost visible

...
```

(The '#' symbol above illustrates the start of a single line comment)

A user could (for example) type the above in a file, and then provide it to the program. They could also type this interactively. The program would then read the script one line at a time, process it and create several files as requested by the user.

You must build a textual view that supports such scripting, both interactive (the user types the script manually) or file-based (the user specifies a file that contains the script).

Your script should support all features that you have implemented thus far (load/save images, add/remove layer, blur, sharpen, make the layer greyscale or sepia). You are not expected to support exactly the commands shown in the script above. The above was just an illustrative example of the kinds of things a user should be able to do. You may choose different words, or syntax. Whatever words and syntax you choose should be reasonable for a user to provide. **Do not try to support expressive or intuitive syntax: start with basic, easy-to-parse commands and enhance if you have time.** You are expected to do error-checking when the script is invalid.

4 What to Do

In this assignment, you must enhance your application to:

- Support the ability to create and manipulate individual layers, as above. More layering operations may be requested in future.
- Support the ability to save and load a multi-layered image. You may create any suitable format for the text file in the description above. Your program is required to only load in a multi-layered image that is saved in the same format (i.e. saved by your own program). You should organize all layer files and the main file in a single folder.
- Support the ability to import existing JPEG and PNG files, and the ability to export a topmost visible layer into JPEG and PNG files.
- Retain the ability to supporting filtering and color transformations, with the possibility of adding more in the future.
- Retain support for the PPM file format, and being able to switch between them (e.g. import a PPM file, work on it and save it as a JPEG file, etc.)
- Support the ability to type in image commands as a script, or load and execute a script stored in a file.

5 Design Considerations

As you add these features, pay attention to your design.

- What changes in your existing design do you make, and why?

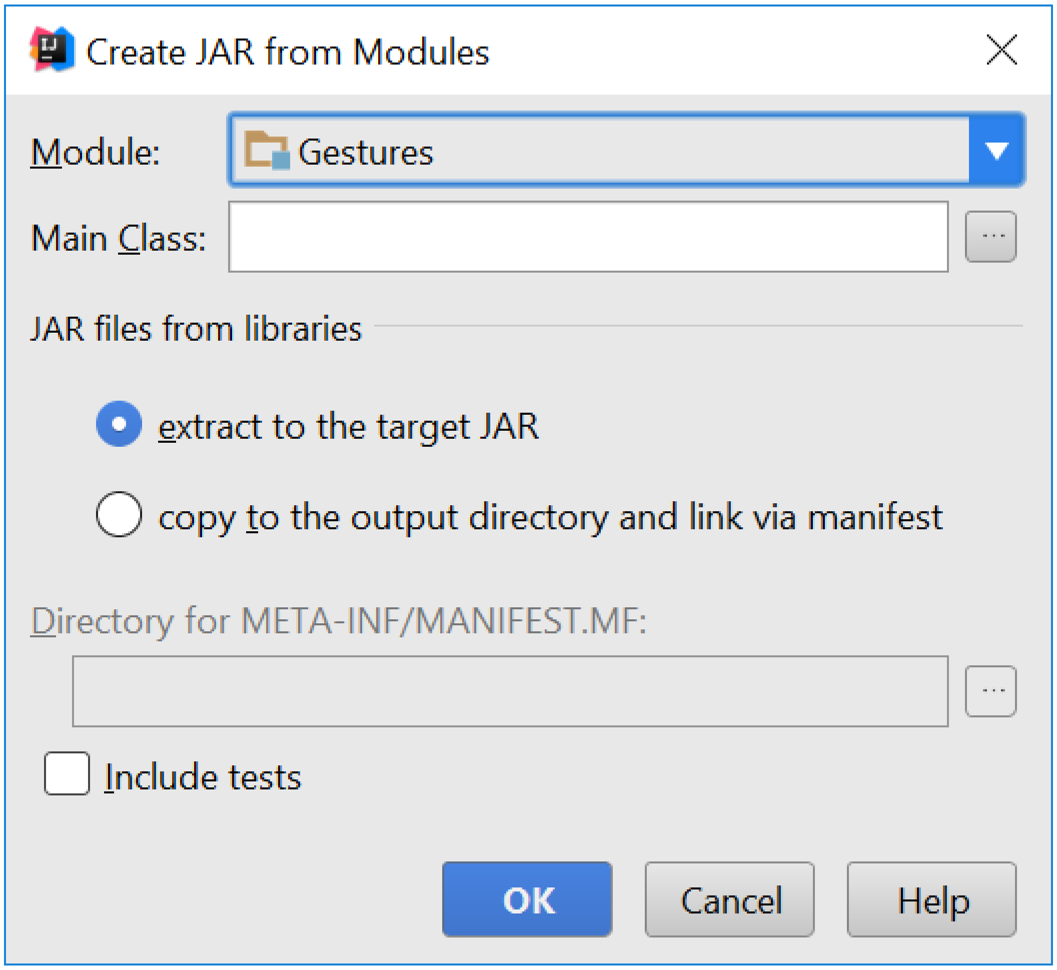
- What is the best way to incorporate new features into an existing application? Do you support all features, or do you release incremental versions with different features (e.g. Starter Edition, Pro Edition, etc.)?
- As you make changes, are you still adhering to the MVC architecture? Are you putting each class and operation where it belongs?
- Have your design enhancements gone beyond the specific operations to be implemented? How easy would it be to add similar extensions in the future?
- Whatever design choices you make, what are its advantages and limitations?

While you are allowed to change your design, you should be able to justify it. The bigger the change, the better we expect your justification to be. Please document and justify each change in a README file (maintain this file as you complete this assignment, rather than summarize after you are done and risk missing something).

6 Create a JAR file of your program

To create a JAR file, do the following:

- Go to File -> Project Structure -> Project Settings -> Artifacts
- Click on the plus sign
- Choose JAR -> From Modules with dependencies. You should now see the module in your project that you are working on (may be different than what is shown in the image below)



Module: Gestures

Main Class:

JAR files from libraries

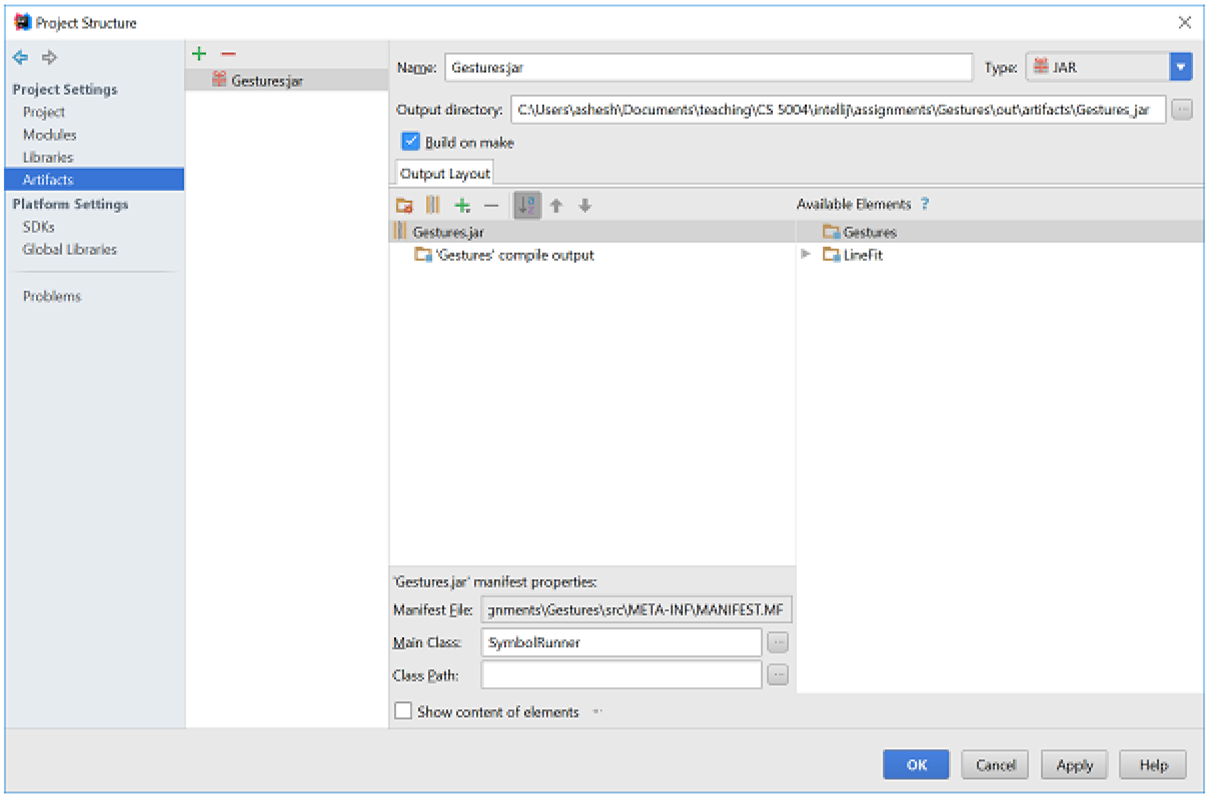
[✓ extract to the target JAR]

copy to the output directory and link via manifest

Directory for META-INF/MANIFEST.MF:

Include tests

- Select the main class of your program (where you defined the `main(String[] args)` method)
- If you see a checkbox labelled "Build on make", check it.
- Hit ok
- You should now see something like



Name: Gestures.jar
Type: JAR
Output directory: C:\Users\ashesh\Documents\teaching\CS 5004\intellij\assignments\Gestures\out\artifacts\Gestures_jar
Build on make: ✓
'Gestures.jar' manifest properties:
Manifest File: gnments\Gestures\src\META-INF\MANIFEST.MF
Main Class: SymbolRunner
Class Path:

If now you see a checkbox labelled "Build on make", check it now.

- Make your project (the button to the left of the run configurations dropdown, with the ones and zeros and a down-arrow on it). Your .jar file should now be in /out/artifacts/.
- **Verify that your jar file works** . To do this, copy the jar file to another folder. Now open a command-prompt/terminal and navigate to that folder. Now type java -jar NameOfJARFile.jar and press ENTER. The program should behave accordingly. If instead you get errors review the above procedure to create the JAR file correctly. **You can also run the jar file by double-clicking on it.**

7 What to submit

Please verify that you submit everything below, as each part is worth some points in this assignment.

A complete submission must include:

1. All your code in the src/ folder.
2. All your tests in the test/ folder.
3. At least two examples of a script file that shows all the working features of your application in the res/ folder. If a command is missing from your example files, we will assume that it does not work and grade accordingly. Your script file should run as-is when we run your JAR file.
4. A correct JAR file in the res/ folder. We should be able to run your program from this jar file.
5. An updated class diagram in the res/ folder, which shows names of classes and interfaces, method signatures and inheritance relationships.

6. A README file in the root submission folder (that contains src, test and res). The README file should document which parts of the program are complete, and design changes and justifications.
7. A USEME file in the root submission folder (that contains src, test and res). The USEME file should summarize which script commands are supported by your application, examples of using them and conditions if any (e.g. X command should be typed before Y, etc.)