CS 5004 Object-Oriented Design

Assignment 10: An Image-Processing Application View

Due Date: Monday, April 18th, 11:59 pm.

This homework should be done with your partner from part 1 & 2.

1 Additional Features

In this iteration of this project, you will build a view for your image processing application, featuring a graphical user interface. This will allow a user to interactively load, process and save images. The result of this iteration will be a program that a user can interact with, as well as use batch scripting from the previous iteration.

2 Image Mosaicing

If you did not complete image mosaicing as part of the previous iteration, you are now required to do it. Please refer to the earlier iteration for details.

3 View

A view is the part of the program that interfaces with the user. Views can be non-interactive as well as interactive: in this iteration you will build an interactive view with a graphical user interface.

3.1 Graphical user interface

Build a graphical user interface for your program. While the choices about layout and behavior are up to you, your graphical user interface should have the following characteristics, and obey the following constraints:

1. You must use Java Swing to build your graphical user interface. Besides the features from lecture, the provided code example illustrates some other features of Swing that you may find useful.

2. The user should see the image that is being processed on the screen. The image may be bigger than the area allocated to it in your graphical user interface. In this case, the user should be able to scroll the image.

3. The user interface must expose all the required features of the program: blur, sharpen, sepia, greyscale, mosaics, dithering, image generation (only the required parts) and the ability to load and save images.

4. The user should be able to specify suitably the image to be loaded and saved that the user is going to process. That is, the program cannot assume a hardcoded file or folder to load and save.

5. The user interface must provide a way for a user to interactively create and execute a batch-script as part of the program (in the same format you designed in the earlier iteration).

6. When the user specifies an operation, its result should be visible to the user.

7. The user interface must expose all its features through menus. A subset of the operations may also be exposed in other ways (e.g. buttons, etc.).

8. Each user interaction or user input must be reasonably user-friendly (e.g. making the user type the path to a file is poor UI design). We do not expect snazzy, sophisticated user-friendly programs. Our standard is: can a user unfamiliar with your code and technical documentation operate the program correctly **without reading your code and technical documentation?**

3.2 View and controller

Carefully design the interaction between a view and a controller, and formalize the interactions with view and controller interfaces. You may design a single controller that processes the batch-script input from the user, as well as the graphical user interface. Different controllers for different views are also possible if the views are very different from each other. However be mindful of the MVC principles and separation between the model, view and controller. When designing, always ask: "can I change one part with no/minimal changes to the others?"

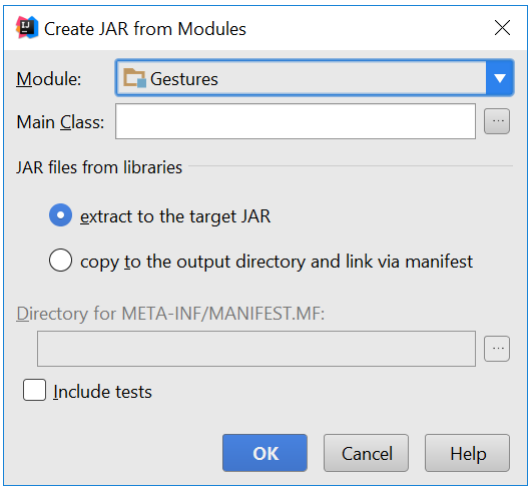
4 Extra credit

For extra credit, add the ability to undo and redo various operations. All image operations should be undoable (and redoable), other than generating images. Expose this functionality suitably in your graphical user interface (undoing and redoing makes less sense in a non-interactive script, so you do not have to support it there).

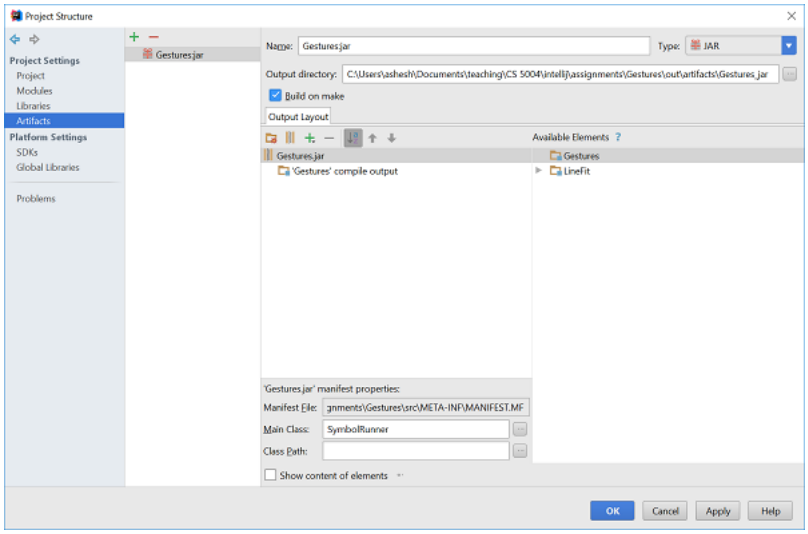
5 Create a JAR file of your program

To create a JAR file, do the following:

- To 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 from what is shown in the image below depending on your operating system):



- 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.
- Click **Ok**
- You should now see something like:



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

- Make your project (the button on the left of the run configurations dropdown, with the ones and zeroes and a down-arrow on it). Your `.jar` file should now be in `<projectRoot>/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 any-command-line-arguments-separated-by-spaces
  ```

  and press **Enter**. The program should behave accordingly. If instead you get errors, review the above procedure create the JAR file correctly.

6 Command-line arguments

Your program (from IntelliJ or the JAR file) should accept command-line inputs. Two command-line inputs are valid:

- `java -jar Program.jar -script path-of-script-file`: when invoked in this manner the program should open the script file, execute it and then shut down.
- `java -jar Program.jar -interactive`: when invoked in this manner the program should open the graphical user interface.

Any other command-line arguments are invalid: in these cases the program should display an error message suitably and quit.

7 Criteria for grading

You will be graded on:

1. The completeness, layout and behavior of your graphical user interface.
2. Your design (interfaces, classes, method signatures in them, etc.)
3. Whether your code looks well-structured and clean (i.e. not unnecessarily complicating things, or using unwieldy logic).
4. Correctness of your implementations, evidenced in part by the images you submit.
5. Whether you have written enough comments for your classes and methods, and whether they are in proper Javadoc style.
6. Whether you have used access modifiers correctly: `public` and `private`.
7. Whether your code is formatted correctly (according to the style grader).

8 What to submit

Your zip file should contain three folders: src, test and res (even if empty).

1. All your code should be in src/.
2. All your tests should be test/.
3. Submit a correct JAR file in the res/ folder. We should be able to run your program from this jar file.
4. Submit a screen shot of your program with an image loaded.
5. Submit a README.md file that documents how to use your program, which parts of the program are complete, and design changes and justifications. The file should also include citations for each image you have used. It is your responsibility to ensure that you are legally allowed to use any images, and your citation should be detailed enough for us to access the image and read its terms of usage. If an image is your own photograph or other original work, specify that you own it and are authorizing its use in the project.