COMP 132: Advanced Programming

Spring 2023

Programming Project

Due: May 24, 2023, 11:59 pm (Late submissions will not be graded.)

PhotoCloud Application Design and Development

This is an **individual programming project**. All work you submit must belong to you. For any questions regarding this programming project, use the corresponding Blackboard discussion forum and post your questions there. Besides, Project Info Session will be conducted by the TAs.

You must include and sign (by writing your name and student id number) the following Pledge of Honor statement at the beginning of your main method source code file. Otherwise, your project will not be graded.

```
/************** Pledge of Honor ****************************************
I hereby certify that I have completed this programming project on my own
without any help from anyone else. The effort in the project thus belongs
completely to me. I did not search for a solution, or I did not consult any
program written by others or did not copy any program from other sources. I
read and followed the guidelines provided in the project description.

READ AND SIGN BY WRITING YOUR NAME SURNAME AND STUDENT ID

SIGNATURE: <Name Surname, Student id>
*********************************************************************/
```

Blackboard submission: You should submit one ***.zip** file named your <name-surname> (for example, Ayse-Yilmaz.zip) that contains the whole Java project folder and a report file as a PDF. The report must be written in two parts. The first part should be a guideline describing the execution steps of your application. In the second part, you should describe your project design, implemented types and their hierarchies, graphical user interfaces, and references to the resources you have used.

Please note that the report grade is part of your project grade. You should prepare it as clearly as possible. There is no page limit for the report, and a report template is provided. **Bonus:** The top projects with the best GUI design, report, and demonstration will get bonus points.

Project Overview

In this project, you are expected to use object-oriented programming (OOP) and Java Swing skills to design a graphical user interface (GUI) based photo editor and sharing application. The application manages information about users, photos, and photo filters.

The programming project is different from the programming labs. In the labs, we provided you with all the required steps. In the project, the objectives are to practice with large-scale programming, apply OOP concepts you have learned, do research on Swing GUI, and practice GUI programming. We provide some general constraints and leave the implementation details up to you. You are expected to design and implement a large-scale Java application from scratch, and use Swing GUI components in the design and implementation of your application.

This document is a guideline to describe some of the design choices while you will decide on the implementation. You should use software practices such as type hierarchies, abstraction, code cleanliness, and documentation.

This is an **individual project**. You can use the provided course materials. You can discuss concepts with the TAs and your friends but **cannot** share any code. The project code and report you submit **must** belong to you. You are not allowed to copy-paste code from anywhere. **If you'd like to use a UI designer tool**, We expect you to be aware of what all of the code does, and you must describe how you used the designer in your report, explain in the source code, and during the project demonstration. You must also add references to your project report. We would not tolerate any violation of the rules stated in the project description.

In the following sections, we describe the project and specify its constraints. You should make your own decisions for anything that we have not established.

The recently founded company named **PhotoCloud** needs an application for its users. The rules and details of the operations not mentioned in this project description are left to your design. Make sure to state your assumptions in this direction clearly. Do not try to overextend the project by adding any other unrequired large-scale features. The features that **PhotoCloud** should have are listed below.

1. Users

Users interact with the PhotoCloud application. Each user has a **unique nickname**, a password, real name and surname, age, and a unique email address. A user can optionally have a profile photo. If a user does not have a profile photo, a default profile photo must be displayed. Users can modify their passwords and their personal information except their nicknames.

Users can upload some of their photos from the computer to the application. Uploaded photos are first private to the user; and if wanted by the user, the photos can be shared with other users.

Users can also modify the uploaded photos and save them separately from the original photo. The users can apply an unlimited number of filters they have access to their photos. The details about photo editing are explained later.

Available filters for the users are based on the tier of the user, that are explained below.

a. User Tiers

There are three tiers of Users: Free, Hobbyist, and Professional.

- Free tier users can only use simple editing that is Blurring and Sharpening an image.
- Hobbyist tier, additional to the Free tier, allows the ability to change the color of the photo via changing brightness and contrast.
- Professional tier is the most comprehensive tier, which, in addition to Hobbyist, allows the ability to apply Grayscale and Edge detection filters.

b. Photo Sharing

All users can post their original or modified photos to the Discover page (see Section 4). From there, users can see who posted what photo, their tier, an optional description of the post, and how much it is liked or disliked. Also, users can comment on the photo, which will be shown under the post's comments section.

c. Profile Page

There will be a Profile page for every User. Another User's profile page should be accessed by searching the User’s nickname or clicking the nickname of the posted user inside the Discover Page. Inside this posted user’s profile page, a user can see the posts that the user has made so far within the Discover Page. Besides, users can see publicly available information.

The profile page of the local user should be accessed easily. The user should see **private information** (e.g., email address, password) of herself and should be able to modify it in addition to **public information** (e.g., name, surname, profile photo). The user should also be able to delete her posted photos from the profile page. Deleting from the profile page should also delete from the Discover Page.

2. Administrator

a. The administrator is the manager of the whole system and is a specialized User of the PhotoCloud application.

b. The administrator has Professional tier user privileges and can remove any photo put on the Discover page by any user.

- When removing any photo from the Discover page, you should remove it from the owner's Profile page as well.

3. Photo Editing and Filters

We provide you two class files named **ImageMatrix.java** and **ImageSecretary.java**. The ImageMatrix class is to store an image in the memory and it performs some basic operations on images. The Image Secretary class is for IO operations i.e., reading an image and writing to/from a location. In your design and implementation, you're welcome to use the provided code, and you're also allowed to modify these classes as you wish.

You should implement the following filters, which are described next.

Blur, Sharpen, Grayscale, Edge detection, Contrast, and Brightness.

Blur Filter Implementation

Blurring an image is a very easy yet important operation. Images consist of pixels, comprising 4 different values: red, green, blue, and alpha. **To blur an image, we replace all pixels with the average value of the neighboring pixels.** The number of neighboring pixels defines our blurring degree. Figure 1 shows an example of blur filter.

Figure 1: Blur filter example

For more clarification and hints check Figures 2 and 3. After applying the mentioned tasks in these figures the resulting image is blurred.

In figure 2, the red rectangle is called “the kernel”. Its size is usually 3x3, 5x5, 7x7, 9x9,11x11, and the center value corresponds to the pixel we’re about to replace. You can check what happens when we increase the kernel size. **Note :** You can ignore the “missing border pixels” after any filtering operations.

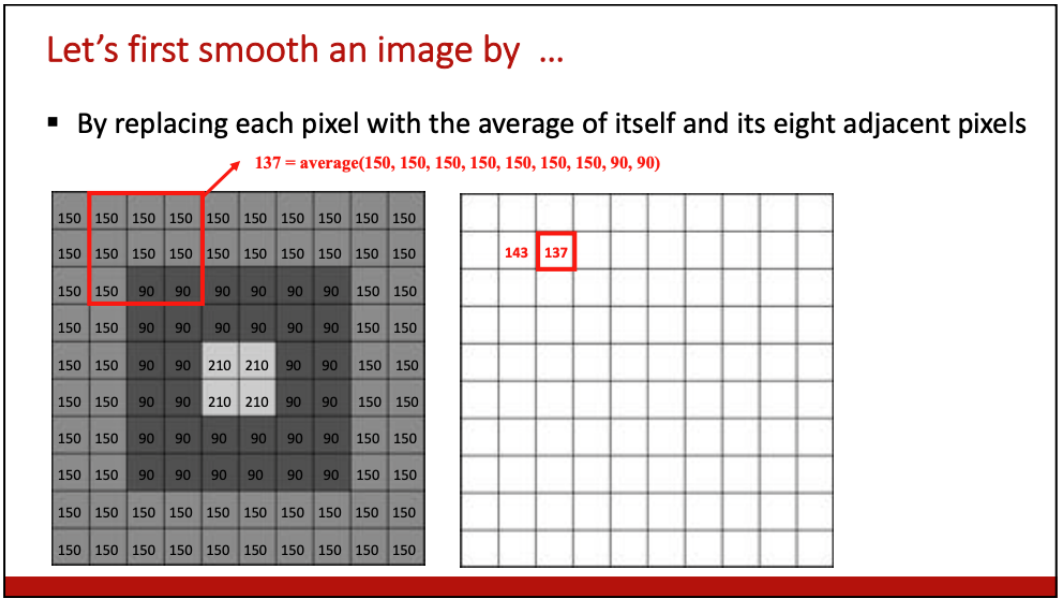


Figure 2: Guide for implementing blur filter

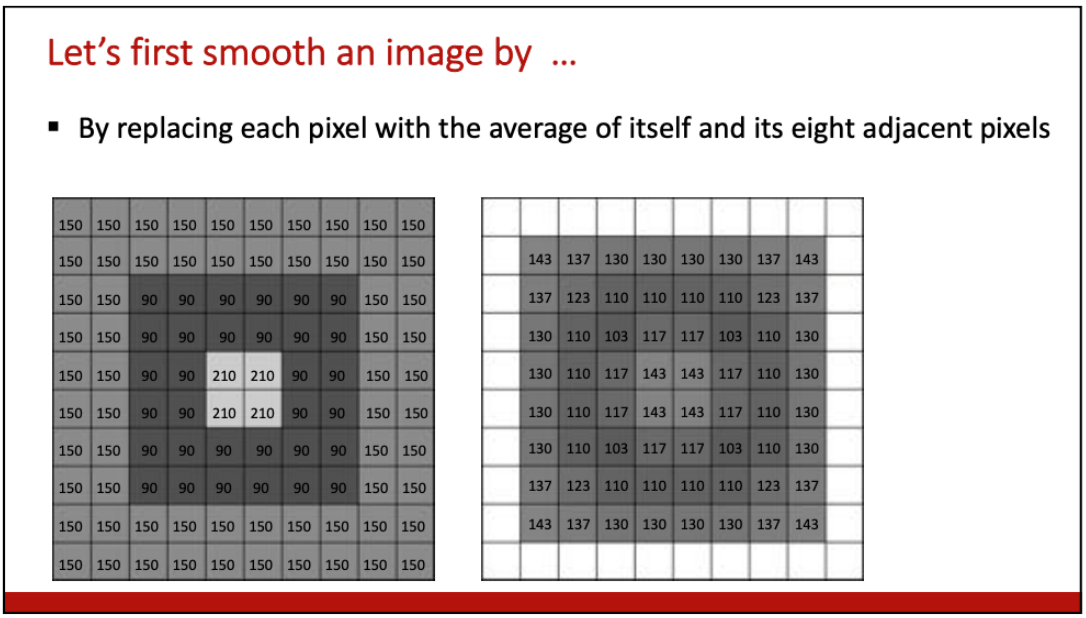


Figure 3: Guide for implementing blur filter

Sharpen Filter Implementation

Sharpening an image is very simple if you correctly implement blurring. To sharpen an image, first blur the image, then subtract the blurred image (pixel-wise) from its original form. That operation gives us the details. Then add details to the original image to make it sharp (See figure 4). The guidance of implementation is provided in figure 5.

Figure 4: Sharpen filter example

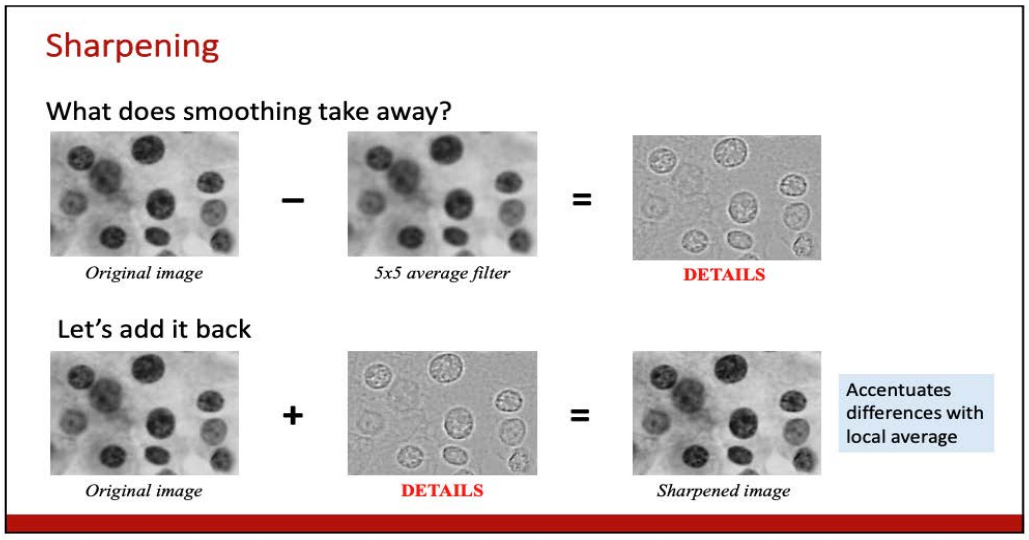


Figure 5: Guide for implementing sharpen filter

Hint: take into account the ignored border pixels from the blurring operation when you perform pixel-wise addition.

GrayScale Implementation

To convert a colored image to a grayscale image, we need to alter the RGB value of each pixel. You can search for this online. We expect a modifier to change how much the image is grayed out. Figure 6 demonstrates an example of this filter.

Figure 6: GrayScale filter example

Edge Detection Implementation

To find the borders, we need to move a “predefined” kernel on the image. There’re various kernels for this job. Check out Prewitt and/or Sobel Operators and apply one of them. (hint: before applying this filter, converting the image to grayscale and slightly blurring the image improves the quality of the resulting image. These kinds of operations are known as image pre-processing). Figure 7 illustrates an example of edge detection.

Figure 7: Edge detection example

Brightness

To change the brightness of an image, you need to increase each RGB value of each pixel by a specified amount. Search this online. We are expecting a modifier to change how much the image is getting brighter. Figure 8 displays an example of this filter.

Contrast

To change the contrast of an image, you need to alter each RGB value of each pixel by a specified amount and a formula. Search this online. We expect a modifier to change how much you change the contrast. Figure 9 shows an example of Contrast.

Figure 8: Brightness filter example

Figure 9: Contrast example

IMPORTANT NOTE: For all filters, the user should provide the filtering degree. Sliders or text inputs (e.g., percentage) can be used to take this from the user.

4. Discover Page

The Discover page is an essential feature of the PhotoCloud application, providing a space where users can view photos shared by others. **You can implement this as you wish,** but in terms of code quality, below are the hints to give you some insight.

Photo Grid Layout

You can design the Discover page with a grid layout to display shared photos. Each cell in the grid should contain a thumbnail of the shared photo and the nickname of the user who posted it.

Photo Interaction

When a user clicks on a thumbnail in the grid, you can open a larger view of the photo in a modal or a separate page. This view should display the full-sized photo, the user's profile photo, nickname, and optional description.

User Profile Access

When a user clicks on the nickname or profile photo associated with a shared photo, you should navigate to that user's profile page. The profile page should display the user's public information and posts.

Remember that these instructions are the starting points; you should make design and implementation decisions that best fit your application.

5. Login Page and Signup Page

The Login and Signup pages are crucial components of the PhotoCloud application, as they allow users to access their accounts and join the platform. This section will guide you through implementing Login and Signup pages.You can use .txt files to store nece

Login Page

Design the Login page with input fields for the user's nickname and password. Include a button or a form submission mechanism to submit the login credentials. Validate the entered credentials and display an error message if the nickname or password is incorrect. If the credentials are valid, navigate the user to their profile page or Discover page. Also, provide a link to the Signup page for users who still need an account.

Signup Page

Design the Signup page with input fields for the user's nickname, password, real name, surname, age, and email address. You may also include an optional field for uploading a

profile photo. Implement form validation to ensure the entered information is valid (e.g., checking for valid email format, minimum password length, and required fields). Display any error messages if the entered data does not meet the requirements. Once the user submits the form and the information is validated, create a new account and navigate the user to their profile page or the Discover page. Provide a link to the Login page for users with an account.

6. Logging

You should log events that would be beneficial to have a log entry. You can implement your logging system as you wish, but it would be beneficial if all logging functions could be called from one location as follows.

```
BaseLogger.info().log(“your message”) // logs to application_log.txt
BaseLogger.error().log(“your message”) // logs to application_error.txt
```

You must log errors that occur throughout the application (i.e., exceptions) to “application_error.txt” file. You must log some info messages to the “application_info.txt” file.

The logging format should be something similar to the following, however you can put your logging structure instead.

```
[Sun Mar 19 16:22:13 TRT 2023][INFO] <info message>
[Sun Mar 19 16:22:16 TRT 2023][ERROR] <exception message>
```

Info logs must contain application start-end time, user login operations, who logged in, the image files that are read and written, filters that you applied. Measuring the time spent for a filter would be helpful information for further analysis such that we can have something like this in the application_info.txt file

```
[Sun Mar 19 16:22:13 TRT 2023][INFO] Blurring filter applied to <...> file,
took: 3ms
```

Error logs must contain any kind of exception messages that are thrown.

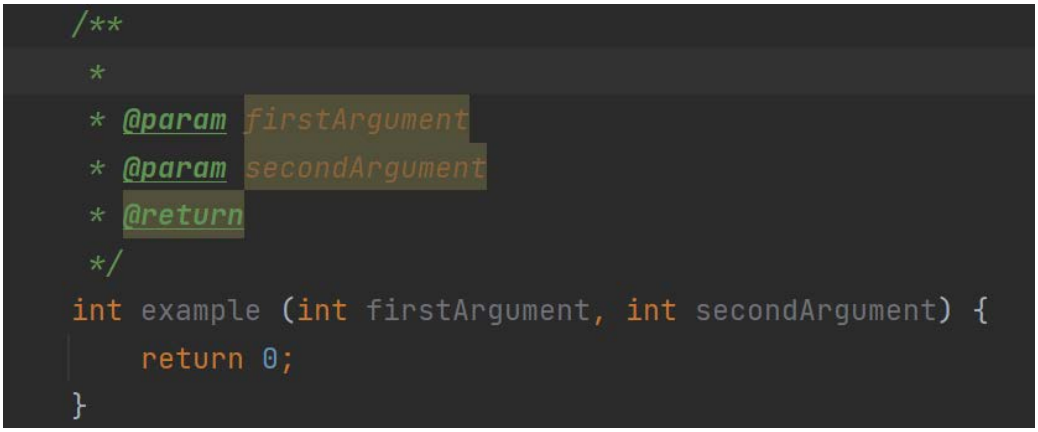
For example: since images are generally stored in arrays, accessing an index that exceeds the size is a kind of error that must be logged.

Trying to read an image file that does not exist is also a kind of error that must be logged.

In your project design and implementation, you should:

- Use inheritance and type hierarchies.
- Apply polymorphism: abstract classes, interfaces.
- Use Java Collections Framework types.
- Use Java Swing GUI components.
- Apply code documentation
  - Explain what each method does, the method's parameters, and the return type.
  - Your documentation for a method should be explanatory to a first-time reader yet concise enough that a reader should understand it within a few lines. You can get inspiration from Java Language's official documentation.
  - You may use Javadoc for documentation. quick introduction or https://www.oracle.com/java/technologies/javase/javadoc-tool.html. Then, you can generate the Javadoc website for your project by using the tools that your ide provides for you and show the "index.html" file.

Hint: you can have your IDE generate a documentation string (docstring for short) template by typing /** above a method declaration and hitting enter.



```
/**
 *
 * @param firstArgument
 * @param secondArgument
 * @return
 */
int example (int firstArgument, int secondArgument) {
    return 0;
}
```

Note: Not providing sufficient documentation would lead to grade reduction, even if your code is perfect.

Design and implement a Graphical User Interface using the **Java Swing Framework**. You can use components such as **JTextField, JButton, JComboBox, and JOptionPane** to design your GUI. You can use any layout, but your **GUI** should be easy to understand and use.

To meet the project demo requirements, your application must *have at least*:

- 9 users with 3 for each type of user (Free tier, Hobbyist tier, Professional tier).
- 12 different photos for the whole app, with at least 1 photo per user.
- All described image filters (blur, sharpen, grayscale, edge detection, brightness and contrast)
- An administrator user.

During the **demonstration** of your project, you should show the execution of the following operations via your application's **Graphical User Interface:**

- Sign-up and log in as a User.
- Display/Modify User information from its Profile Page.
- Upload/Delete photos to/from the User's profile.
- Add/Remove photos to/from the Discover page from the profile page.
- Go to another user's profile page, display another user's public information, and display posted photos to the Discover page in the profile page.
- Visit the Discover page. Show the required information described in Section 1.c.
- Apply available filters to images multiple times based on the tier of a user and show the resulting image in the application.
- Save the resulting image in the application and as a file.
- Log the actions and errors of the application to the individual files.

Good Luck!