

**Object Oriented Programming (OOP)**  
CCIT-4023, HKU SPACE Community College, 2018-19

**Course Project (25%)**

Class:		Name:		Date:	
--------	--	-------	--	-------	--


## 1. Introduction

In this course project, students are required to demonstrate their basic understanding of Object-Oriented Programming (OOP) concepts, and their ability to apply them with programming techniques learnt in the course for designing and developing *application software* in specified development platform. This project also helps students to consolidate their programming skills, and prepare for their examination.

### 1.1 Project Description

#### **Background of this Course Project:**

Your project group is required to develop a **Standalone, GUI-based, Multi-User Software System** for supporting certain *Specific Task* – **Image Editing**. This software includes 1) a recording system for multi-users and their activities, and 2) tools supporting the specific task (details in later section).

### 1.2 Objectives

- Initialise and identify the project nature and scope within the given specification
- Design and develop an application software with Object Oriented (OO) concepts
- Implement, test and evaluate the system with self-defined and standard classes

### 1.3 Grouping and Project Management

Students should form a group (of **4** to **6** members) to finish this project. Project group may further divide into sub-groups for handling different sub-systems. The following stages are expected during the development process, and should be introduced in the documentation and presentation:

- **Initiation and Planning**
  - All members of the whole group discuss and identify the project scope (e.g. the topic, objectives). A plan and schedule should be settled to keep track on the project progress.
- **Design and Implementation**
  - The group should then design (e.g. with UML class diagrams), and implement the program.
- **Testing and Evaluation**
  - The project work should be properly tested and evaluated, esp. based on the requirements set in sections 2.

Project group and members should have proper project management. The following lists some *highly recommended* project stages, *for reference*.

Lesson	Recommended Project Progress and Works
4	<b>Initiation:</b> Project release and briefing
6	<b>Grouping:</b> Project group settled; knowing each other; refining the project scope
7	<b>Design:</b> Settle general design (of main classes, e.g. in UML); may refine details later
8	<b>Implementation:</b> Start doing and finishing the distributed individual works
10	<b>Implementation (Integrate &amp; Test): <i>Finish an Alpha version (A basic integrated one)</i></b> ○ Integrate individual parts/classes, and start for the FINAL version
11	<b>Implementation, Testing &amp; Evaluation: <i>Finish the FINAL version</i></b> ○ Integrate all for the final version, <i>prepare project submission and presentation</i>
12	<b>Project submission:</b> with presentation + demonstration

Note that each group member should take part in code writing as well as presentation and project report. Strong group work is expected. All members are expected to take part in the design stage, and distribute the work evenly. You should consider that each member is implementing one or more sub-system(s) of the whole project, and these sub-systems can be integrated at a later stage.

**\*IMPORTANT\*** Each project group member must understand and be able to explain the basic/general project issues and their own implementation/contribution part, even though not all members participate in writing all codes of the whole project.

## 2. System Requirements

Project groups are going to build a **Software System** to meet the functional and programming requirements listed below. Groups should define their own classes and use the standard APIs to accomplish different tasks. (*Third party APIs could not be used, without prior approval.*)

### ***Required Development Platforms and Programming Language***

- *Targeted Running Platform:* Java Platform with JRE (version 8, or above)
- *Development Environment:* JDK (version 8, or above)

\* Report to teacher (in report form, report and presentation slides) the version your group uses.

The software system in this project includes two major aspects: 1) the general platform supporting multiple users using the software system and recording certain user activities, and 2) tools / sub-system supporting the specific task (details in later section).

## 2.1 Functional Requirements

The software system provides interfaces for different users. There are three types of the User accounts: Admin (*Administrator*), SUser (*Special User*) and CUser (*Common User*).

Common User (CUser) **has to pay** (e.g. HK\$8) for the tools each time they invoke and use for the specific tasks, while Special User (SUser) is free to use the tools as many times as they want. Each usage of the specific task tools (from a successfully authenticated user) is recorded and the balance of the associated user will be updated if necessary. Below shows a typical operational flow of working with the system:

- a) A new user self-registers and requests a new common user account for using the system, and waits for approval. (Data file stores this request.)
- b) Administrator logs in the system, the system prompts the administrator of new account requests if any existing, and the administrator approves these requests. (Data file stores and updates accordingly.) Administrator browses the records of all users and their task records.
- c) A valid user (with valid User ID and Password) logs in the system successfully, invoke to use the specific task tools, and the system records it. The user then logs out the system once finished. (Data file stores and updates accordingly if necessary.)
- d) User re-logs in the system, updates personal information, and lists task records for checking.

*\* Project group should consider other situations, such as invalid login, etc.*

The system should *at least* manage the following data/attributes/fields properly:

- **User :** User\_ID, Encrypted\_Password, Name, Time\_Created, Current\_Balance, etc.
- **Task\_Record :** TRecord\_ID, User\_ID, Time\_Recorded, Description, Fee, etc.

*\* Project group should properly design the types and structures of these data and attributes for the system, and decide whether specific attributes are compulsory or optional for valid data. Project group may also add other attributes to specific topic they have refined, e.g. photo of user.*

You are required to build a system to perform *at least* the following:

### 2.1.1 General Functions (these apply to ALL user types):

- a. Login / Logout (for testing purpose, default accounts must be included as given later.).
- b. After successfully login, Load required stored data and display user account information.
- c. Let user display and update their information properly

### 2.1.2 Admin (Administrator) Functions:

- a. Approve User Account Registration Request
- b. User (All user accounts): Display / Create / Modify / Delete
- c. Task\_Record (All records): Display / Create / Modify / Delete
- d. Analyse simple useful information (e.g. overall balance, negative balance accounts, etc.)

*\* Project group may decide their proper factors and criteria of analysis and evaluation.*

### 2.1.3 Users - SUser (Special User) and CUser (Common User) Functions:

- Self-register a new *User* account, and wait for approval by Administrator.
- Task\_Record (User's Own record): Display
- Use tools for specific task: if enough balance left for CUser, but it is free for SUser.

### 2.1.4 Tools of specific task: Image Editing

Tools / sub-system of this specific task (image editing) let user open, edit and save JPEG image files. Common users are expected to pay for using this tool / sub-system, with balance recorded in their accounts (e.g. charge HK\$8 for each time using this tools, HK\$80 for saving the result or using more advanced functions). Once a valid user enters the system successfully, the user calls this "sub-system" tools (and the system records this), and interacts with it to do tasks of image editing.

Using and extending the given "uncompleted" sources of the tools *if there are any given*, the project group has to finish this sub-system (tools) which *at least* provides extra functions of the following:

- ONE editing function from: rotation / flipping / scaling
- ONE editing function from: inverse / brightness / blurring
- ONE self-selected function decided by the project group

\* **These Extra Functions** should be implemented in your own class (e.g. for group G22, named **G22OOPImgFn**, similar to the given reference code sources)

### 2.1.5 Default Settings and Testing: For testing and evaluation, submission work should include:

- 4 or more default accounts (MUST use information of ALL project group members) –
  - one Admin, two Special Users, *one or more* Common Users below:

Type	Login Names	User Information	Password, <i>one letter only</i>
Administrator	<b>a</b>	Group representative	<b>"a"</b>
Special User	<b>s1, s2</b>	2 group members	<b>"s"</b>
Common User	<b>c1, c2...</b>	Other members	<b>"c"</b>

- Three sets of Testing Record Files (in three different folders) - for testing your system:

Nature	Number of existing <b>Task records</b>	Folder name
* Default File Set * (normal)	<b>9</b>	<i>Default (for demo as well)</i>
Normal Testing File Set	<b>90</b>	<b>Norm</b>
Abnormal Testing File Set (Errors are detected and handled when the system is launched)	E.g. File corrupted, inconsistent and/or uncompleted data	<b>Abnorm</b> (with a readme.txt to briefly explain abnormal data cases)

### 2.1.6 Project Scope and Assumptions

- Project group may decide other essential functions within the scope of this project. There may be reasonable scenarios not mentioned above. **However, this is NOT a complete system which handles ALL aspects.** You should ignore other issues, e.g. system concurrency, payment, etc.
- For Image Editing, assessments are limited to JPEG images, no larger than 2000x2000 pixels.
- Recommendation:** **Students should start the project and first focus on those basic required parts,** before further enhancements on more advanced features if time and resources are available.

**[Optional]** Extra Advanced Features / Functions:

- a) **Image Mosaic:** *Advanced Image Editing* (for matching, transformation, non-rectangular etc.)
- b) **Interactive Regional Image Editing:** *with Mouse Event*, select editing regions interactively
- c) **Advanced Login:** *Simple Biometric Authentication (Behavioural)*: with mouse and/or keyboard input (e.g. matching with temporal/spatial pattern, manner, rhythm)

**\* IMPORTANT \*** These advanced features are more open-ended, but challenging. Each project group should only attempt at most one. Students should properly finish the basic requirements first. If groups intend to try these extra features, contact and discuss with the teacher for more details.

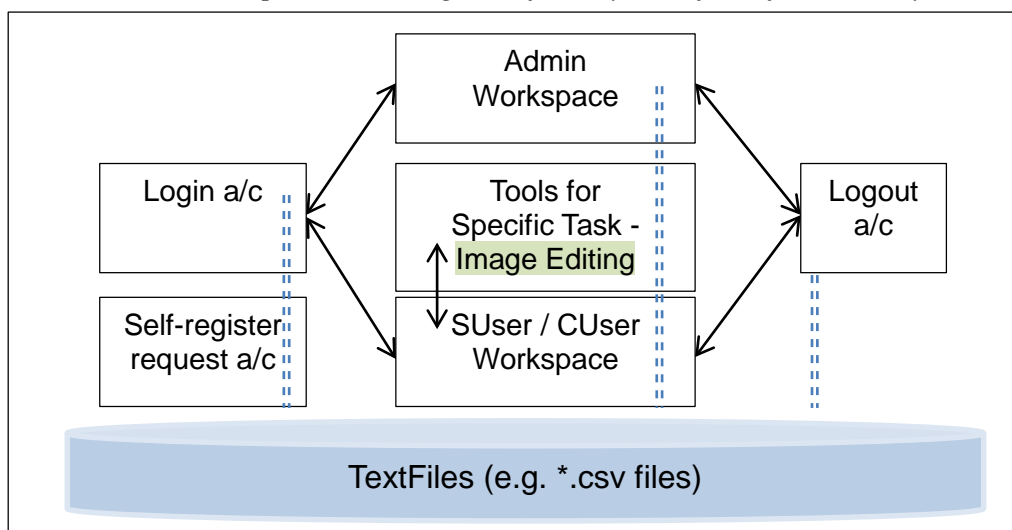
## 2.2 Programming Requirements

You should also apply the following programming techniques to the project:

- a) Basic Programming Techniques: Selection, Repetition, Array / ArrayList, etc.
- b) Object Oriented Features: Encapsulation, Inheritance, and Polymorphism
- c) Proper Exception Handling, and User Input Validation
- d) Data Handling, with Text File (in csv format)
- e) Basic Graphical User Interface (GUI), with Event Handling
- f) Naming: ALL your own defined Java classes and interfaces must start with **GroupNum**, e.g. name your Admin. User Class as **G22AdmUser**, for the project group **G22**.
- g) Main class naming: name your main class for launching the system as **G22MainSys**

**Remarks:** Students should also consider other programming issues of their project (i.e. future modification, extension, re-use or re-design). Proper code *indentation*, *naming* and *comments* should be applied for better managing and understanding the sources and materials developed.

*A Sample Block Diagram of the System, for reference only*



### 3. Project Submission and Presentation (*Lesson 12*)

- **Hardcopy** of *Project Submission Form*, complete and print out the given form (Colour-printed NOT required)
- **SOUL**: Submit the followings to SOUL
  - ☐ **Complete Program**: Compress/Zip all sources for developing and running the program in a zip file gpNum\_oop.zip (e.g. g22\_oop.zip). It should include:
    - All source codes (\*.java), the compiled files (\*.class), other associated folders & files
    - Data Text files (e.g. \*.csv), and other supporting files (e.g. images), if any
    - Two batch files (gpNum\_c.bat for compilation; gpNum\_r.bat for starting program)
  - ☐ **Project Documentation**: Submit the following documents:
    - ☐ **Project Submission Form**, complete the given form.
    - ☐ **Project Presentation Slide** (e.g. *pptx, pdf*) named gpNum\_present.pptx/pdf
    - ☐ **Project Report**: named gpNum\_report.docx/pdf (about 12 pages), which contains details of software design and development process:

Cover Page	Course name, year & semester; Project name & Title; Class number, group number, names and SIDs of group members
Table of Contents	
<b>Introduction</b>	Brief introduction & objectives of the project
Initiation and Plan	Project requirement, specification, plan, schedule & job allocation
Development Platform and Tools	Major Hardware, OS & software used, e.g. JDK ver., notepad++, etc.
<b>Design</b>	Problem analysis, ideas and concepts, detailed design, with proper UML class diagrams for major classes if necessary
<b>Implementation (Development)</b>	Function content and user interface, coding, debugging, testing (Major issues only, NO detailed explanation of program codes)
<b>Evaluation</b>	Evaluation of the final product and the development process
Conclusion	General Conclusion; Strength & weakness; Difficulty & limitation; Possible improvements, etc.
Others	If any. (E.g. References)

\* *Contents on documents (submission form, presentation slides, and report) could often be reused.*

#### ▪ **Presentation and Demonstration:**

Each project group has about 15~25 minutes, including presentation, demonstration, and optionally Q&A sessions. All group members should participate in this session. The demonstration should focus on the functional requirements and start with the default settings as specified above.

**\*IMPORTANT\*** *Project work without presentation and demonstration may not be assessed.*

- Because of the tight presentation schedule, it is important to have a good *preparation and rehearsal* before coming for the presentation and demonstration.

## 4. Assessment Criteria

**\*IMPORTANT\*** All programs must *at least* be compiled and run under normal operation without serious problems in our college computer room.

Weight (Total 100%)	Descriptions
40	Functional requirements (section 2.1)
20	Programming requirements (section 2.2)
20	Project development progress, documentation, and submission
20	Project presentation and demonstration

**\*Remark\*** Although all group members share the same project grade in general, variations may be applied to individuals in certain special cases such as strong unbalance workloads were reported.

## 5. Others

- Students should consult the teacher for project details, in case of any doubts.
- Double-check your submission. *Choose a project group representative to coordinate the project and submit your group project work.*
- Proper acknowledgements and/or citations should be given.
- Other project issues may be explored by students, including version control (e.g. with Git) and testing (e.g. with JUnit Test).
- *Collusion and plagiarism are serious offences and may result in disciplinary action. Zero mark may be given for the piece of coursework and, in addition, the final grade of the course may be affected. Carefully read the section about Collusion and Plagiarism in Student Handbook.*

## 6. References

### 6.1 Comma-Separated Values (CSV) File

[https://en.wikipedia.org/wiki/Comma-separated\\_values](https://en.wikipedia.org/wiki/Comma-separated_values)

A comma-separated values (CSV) file stores tabular data (numbers and text) in plain text. Each line of the file is a data record (while the first row is the attribute/field names). Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format. Example:

ID	Name	Role	Ability	Salary
12	CHAN Tai Man	Manager	Eat and Talk	30000
13	LAU Ann	Worker	Do, Do and Redo	40000
14	CHAN Kate	Supervisor	Talk and Eat	50000

The data table above could be saved in the CSV format as below:

```
ID,Name,Role,Ability,Salary
12,CHAN Tai Man,Manager,Eat and Talk,30000
13,LAU Ann,Worker,"Do, Do and Redo",40000
14,CHAN Kate,Supervisor,Talk and Eat,50000
```



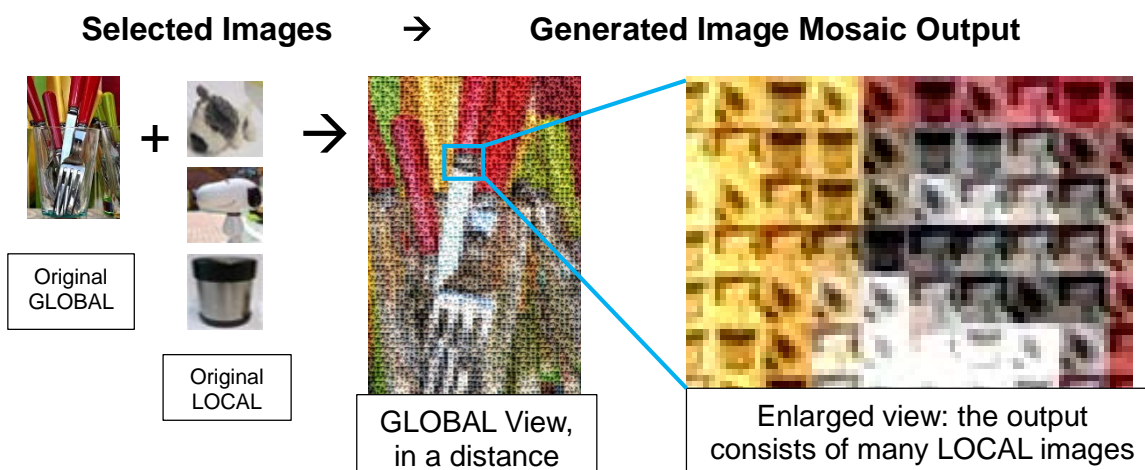
## 6.2 Image Mosaic

[https://en.wikipedia.org/wiki/Photographic\\_mosaic](https://en.wikipedia.org/wiki/Photographic_mosaic)

Image mosaic (or photographic mosaic), also known under the term Photomosaic, a portmanteau of photo and mosaic, is a picture (usually a photograph) that has been divided into (usually equal sized) tiled sections, each of which is replaced with another photograph that matches the target photo. When viewed at low magnifications, the individual pixels appear as the primary image, while close examination reveals that the image is in fact made up of many hundreds or thousands of smaller images. Most of the time they are a computer-created type of montage.

There are two kinds of mosaic, depending on how the matching is done. In the simpler kind, each part of the target image is averaged down to a single color. Each of the library images is also reduced to a single color. Each part of the target image is then replaced with one from the library where these colors are as similar as possible. In effect, the target image is reduced in resolution (by downsampling), and then each of the resulting pixels is replaced with an image whose average color matches that pixel.

In the more advanced kind of photographic mosaic, the target image is not downsampled, and the matching is done by comparing each pixel in the rectangle to the corresponding pixel from each library image. The rectangle in the target is then replaced with the library image that minimizes the total difference. This requires much more computation than the simple kind, but the results can be much better since the pixel-by-pixel matching can preserve the resolution of the target image.



## 6.3 Useful Links and Information for Working with Swing and Image in Java

**Using Swing Components** <https://docs.oracle.com/javase/tutorial/uiswing/components/index.html>

**Working with Images** <https://docs.oracle.com/javase/tutorial/2d/images/index.html>

**Class BufferedImage** <https://docs.oracle.com/javase/10/docs/api/java/awt/image/BufferedImage.html>

<code>int</code>	<code>getRGB(int x, int y)</code>	Returns an integer pixel in the default RGB color model (TYPE_INT_ARGB) and default sRGB colorspace.
<a href="#"><u>BufferedImage</u></a>	<a href="#"><u>getSubimage</u></a> (int x, int y, int w, int h)	Returns a subimage defined by a specified rectangular region.

**Class ImageIO** <https://docs.oracle.com/javase/10/docs/api/javax/imageio/ImageIO.html>

<code>static <a href="#"><u>BufferedImage</u></a></code>	<code><a href="#"><u>read</u></a> (<a href="#"><u>File</u></a> input)</code>	Returns a <code>BufferedImage</code> as the result of decoding a supplied <code>File</code> with an <code>ImageReader</code> chosen automatically from among those currently registered.
--	--	--



<code>static BufferedImage</code>	<code>read(InputStream input)</code>	Returns a BufferedImage as the result of decoding a supplied InputStream with an ImageReader chosen automatically from among those currently registered.
<code>static BufferedImage</code>	<code>read(URL input)</code>	Returns a BufferedImage as the result of decoding a supplied URL with an ImageReader chosen automatically from among those currently registered.

## 6.4 Extracted Given Code Sources for the Specific Task:- Image Editing

```

////////////////////////////////////
// STUDENTS MAY IGNORE OTHER IMPLEMENTATION DETAILS           //
// STUDENTS ADD Codes in 1) and 2) for their own functions //
////////////////////////////////////
/**
 * <h1> OOPGUI_ImgEdit: An Image Editing Application, with GUI </h1>
 * Part of course project, for An Image Editing Application with GUI.
 * ONLY for JPEG images. MAY CONTAIN MANY BUGS.
 * <p>
 * @version 1.0
 * @since 2018
 */
public class OOPGUI_ImgEdit extends JFrame{
    ////////// OPTIONS of Editing Image, for selection //////////
    public static final String [] imgEditOptions = {
        "Reset Original", // for option case 0
        "G99-getRedOnly", // for option case 1
        "G99-blendTwoImg" // for option case 2
    };
    ////////// 1) MORE OPTIONS ADDED from other Groups HERE //////////
    };
    ////////// Method for running selected option of Image Editing //////////
    public void runImgEditOpt(int imgEditOpt){
        infoLabel.setText("Selected Option: "+imgEditOptions[imgEditOpt]);
        switch (imgEditOpt){
            case 0: // "Reset Original"
                newImg = G99OOPImgFn.resetOriginal(orgImg); break;
            case 1: // "Reset Original"
                newImg = G99OOPImgFn.getRedOnly(orgImg); break;
            case 2: // "Reset Original"
                newImg = G99OOPImgFn.blendTwoImg(orgImg, newImg); break;
            ////////// 2) MORE OPTION CASES ADDED from other Groups //////////
            default: // UNKNOWN cases, do nothing
                infoLabel.setText("ERR: Unknown option!\n Do Nothing!");
                break;
        }
        showImg();
    }
    // ...

```

HERE add codes of calling your option functions of different cases, similar to others.

HERE add codes of your option String (and case below) to be listed, similar to others.

```

/**
 * <h1>G99OOPImgFn: Class providing Options of Image Editing </h1>
 */
public class G99OOPImgFn {
// ...
    public static BufferedImage resetOriginal(BufferedImage orgImg){
// ...

// ...
    public static BufferedImage blendTwoImg(BufferedImage orgImgA,
                                           BufferedImage orgImgB){
// ...
/**
 * Get Red Color ONLY, ONLY for RGB type
 * @param orgImg Original image for editing
 * @return The newly edited image (a simple image of RED color of original
image)
 */
    public static BufferedImage getRedOnly(BufferedImage orgImg){
        if (orgImg==null) return null; // if null original, exits
        int w = orgImg.getWidth(); // get width of image
        int h = orgImg.getHeight(); // get height of image
        BufferedImage newImg = new // create new image of same size
            BufferedImage(w, h, orgImg.getType());










        int orgRGB, newRGB, r, g, b;
        // LOOP for get pixels from original, and assign with new pixel values
        for (int x = 0; x < newImg.getWidth(); x++) {
            for (int y = 0; y < newImg.getHeight(); y++) {
                orgRGB = orgImg.getRGB(x, y); // get RGB from original image
                r = (orgRGB>>16 & 0xff); // get original RED
                g = 0x00; // for original GREEN; (orgRGB>>8 & 0xff);
                b = 0x00; // for original BLUE; (orgRGB & 0xff);
                newRGB = (r<<16) + (g<<8) + b; // assign combined-RGB value
                newImg.setRGB(x, y, newRGB);
            }
        }
        return newImg;
    }
}
// ...

```

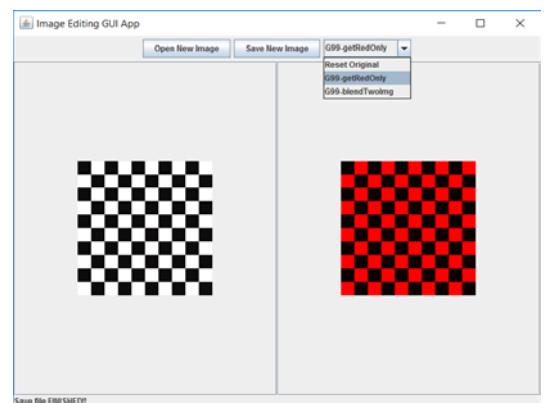
/\*

Integer in Hexadecimal to color RGB: 0xRRGGBB → RGB

e.g. In hexadecimal → (RGB) Color:

0xffffff	→		White
0x808080	→		Grey
0x000000	→		Black
0xff0000	→		Red
0x00ff00	→		Green
0x0000ff	→		Blue
0x00ffff	→		Cyan
0xff00ff	→		Magenta
0xffff00	→		Yellow

\*/



// ... Sample Display

~ END ~