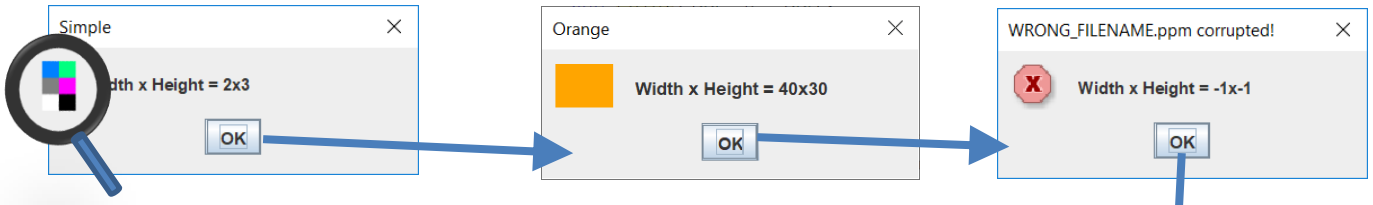
Due date: 18 November 2021 (Thu) **Assignment 5** Full mark: 100

Expected normal time spent: 8 hours

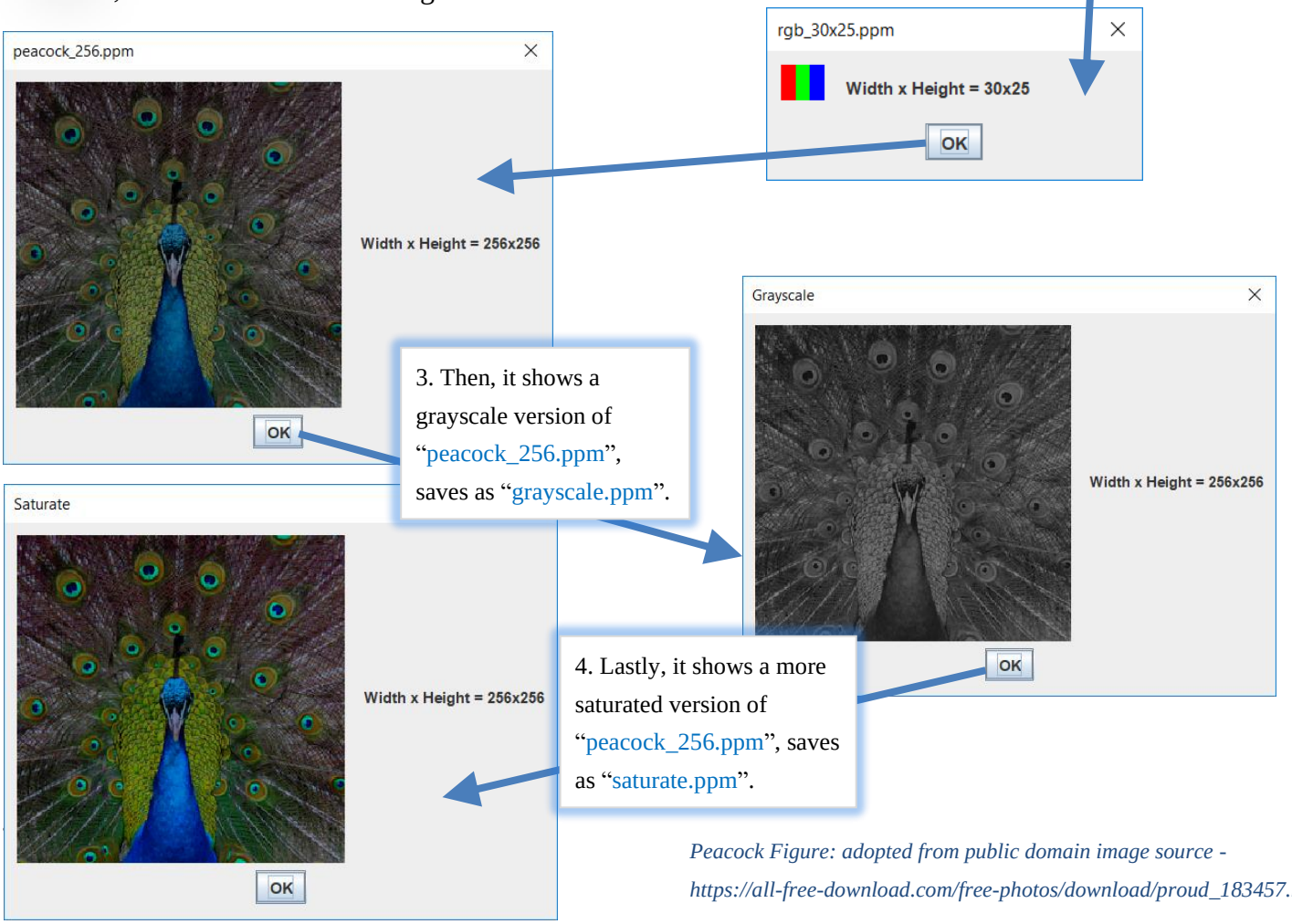
Photo Kiosk

Aim: - build an image processing app to process PPM [a] images.
- practise using existing classes/objects and creating our own methods.
- practise file I/O, String and 1D/2D array processing.

1. Firstly, the program prepares and displays some simple "default" images for your reference.



2. Then, it reads and shows some given PPM files:



Peacock Figure: adopted from public domain image source - https://all-free-download.com/free-photos/download/proud_183457.html

5. The program may display some text via System.out for your own debugging purposes. If the user clicks **`Close (×)`** button in any message dialog, act as if the user had clicked **`OK`**.

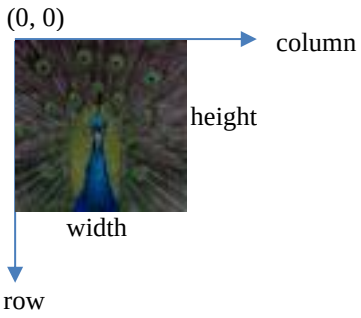
[a] Portable Pixel Map (PPM) is a simple image file which supports RGB full colour images in ASCII text format. Reference: https://en.wikipedia.org/wiki/Netpbm_format#PPM_example

6. A client class PhotoKiosk is given. You are expected to complete the implementation of another class PPM.

7. When reading a PPM image file, there may be various kinds of errors and issues such as file not found, wrong file format, etc. Follow the given sample code to throw an Exception to indicate header problem. You need NOT consider other kinds of file format issues. You may keep the given Exception handling (try-catch) blocks.

8. Here is the expected standard PPM file format, *free from additional features* such as comment:

```
P3
width height
max_value
R G B R G B R G B...
R G B R G B R G B...
...
R G B R G B R G B...
```



The first two characters on the first line must be "P3".

Then 3 numbers follow: *width* and *height* define the size of the image;

max_value is usually 255 which indicates maximum possible value of a pixel component value.

R, G, B's are red, green and blue pixel component values defining a full colour image in RGB. All values shall fall within 0 and *max_value*, i.e., usually within 0 – 255.
There are *height* pixel lines run from top to bottom row-by-row. Each pixel line contains *width* number of pixels, well, R, G, B triplets.

All numbers are delimited by white-spaces (including space, tab and newline.)

Therefore, Java Scanner class is well suited for reading the PPM file in plain text ASCII format.

The *origin* of the image and window coordinates system is always at the top-left corner.

9. In order to grayscale or saturate the image, you shall do processing in the YUV color space (a model used in the PAL video standard) which defines one luminance component (Y) and two chrominance components, called U (blue projection) and V (red projection); after processing, you convert back to a RGB color. RGB to/from YUV Conversion [b] (scaled and offset version) is given by:

```
Y =  (0.257 × R) + (0.504 × G) + (0.098 × B) + 16
U = -(0.148 × R) - (0.291 × G) + (0.439 × B) + 128
V =  (0.439 × R) - (0.368 × G) - (0.071 × B) + 128

R = 1.164 × (Y - 16) + 1.596 × (V - 128)
G = 1.164 × (Y - 16) - 0.813 × (V - 128) - 0.391 × (U - 128)
B = 1.164 × (Y - 16) + 2.018 × (U - 128)
```

In both these cases, you have to clamp the output values to keep them in the [0-255] range, i.e., check and rectify the resultant values in order to make negative values zero and make over-size values 255.

[b] "Video Demystified", Keith Jack (ISBN 1-878707-09-4).

Procedure (Step-by-Step):

1. Before you code, revise the concepts in OOP, specifically how to define methods and constructors. You will deal with 2D array of Color [c] in this exercise.

2. **Create a new NetBeans project `PhotoKiosk` with a default main class `photokiosk.PhotoKiosk`.** Copy the given code, Java source files and PPM files from Blackboard into the proper locations under your project folder.

3. Client class `PhotoKiosk` is given. You are expected to complete the implementation of another class `PPM`.

4. Your class `PPM` should provide methods to read PPM, write PPM, do color space conversion, as well as grayscale/saturate the image.
   a) To **grayscale** the image, you shall replace the R, G, B values with the luma component Y on the current image.

```
public PPM grayscale() {
    /*** student's work here to generate a grayscaled image based on Y ***/
}
```

   Newly generated image should be saved into a file named "`grayscale.ppm`", and then go to the next saturate image step.

   b) To **saturate** the image more, you shall manipulate the two chrominance components U and V with a simple linear scaling at middle-chroma by:

$$U' = (U - 128) \times S + 128$$
$$V' = (V - 128) \times S + 128$$

   where U' and V' denote the new chrominance components while S is the saturation scale factor, we assume $S = 2$ in this assignment. Updated YUV is mapped back to RGB; output values should be clamped to keep them in the [0-255] range.

```
public PPM saturate() {
  /*** student's work here to generate a more saturated image based on U & V ***/
}
```

   Newly generated image should be saved into a file named "`saturate.ppm`", then program exits.

5. The newly saved files should usually reside in your NetBeans application project folder by default, in the PPM format, again *free from additional features* such as comment.

6. Zip and Submit your *whole NetBeans project folder* in an archive file **`PhotoKiosk.zip`** via our Online Assignment Collection Box on Blackboard `https://blackboard.cuhk.edu.hk`.

[c]. java.awt.Color documentation: https://docs.oracle.com/javase/7/docs/api/java/awt/Color.html

Marking Scheme and Notes:

1. The submitted program should be free of any typing mistakes, compilation errors and warnings.
2. Comment/remark, indentation, style are under assessment in every programming assignments unless specified otherwise. Variable naming, proper indentation for code blocks and adequate comments are important. Include also your personal particulars and your academic honesty declaration statement in a header comment block of each source file.

3. Remember to upload your submission and **click Submit** before 6:00 p.m. of the due date. No late submission would be accepted.

4. If you submit multiple times, ONLY the content and timestamp of the latest one would be counted. You may delete (i.e. take back) your attached file and re-submit. We ONLY take into account the last submission.

University Guideline for Plagiarism:

Attention is drawn to University policy and regulations on honesty in academic work, and to the disciplinary guidelines and procedures applicable to breaches of such policy and regulations. Details may be found at http://www.cuhk.edu.hk/policy/academichonesty/.

With each assignment, students are required to submit a statement that they are aware of these policies, regulations, guidelines and procedures, in a header comment block.