# Hands-on Experiment # 9: Worksheet

Section 2 Date 31 October 2019

No more than 3 students per one submission of this worksheet.

Student ID 6238063321 Name Nattawat Pornthisan

Student ID 6238084521 Name Tananyapat Wareechuensuk

## Part A: Multi-dimension Array with Text File Processing (Golf score and Handicap)

The file *golf-score.csv* contains scores of golfers (total stroke in each hole that each golfer taken to finish each hole) for 18 holes. The file is in the “Comma-separated Value” format (<http://en.wikipedia.org/wiki/Comma-separated_values>) with the first line being the header labels describing par value of each hole.

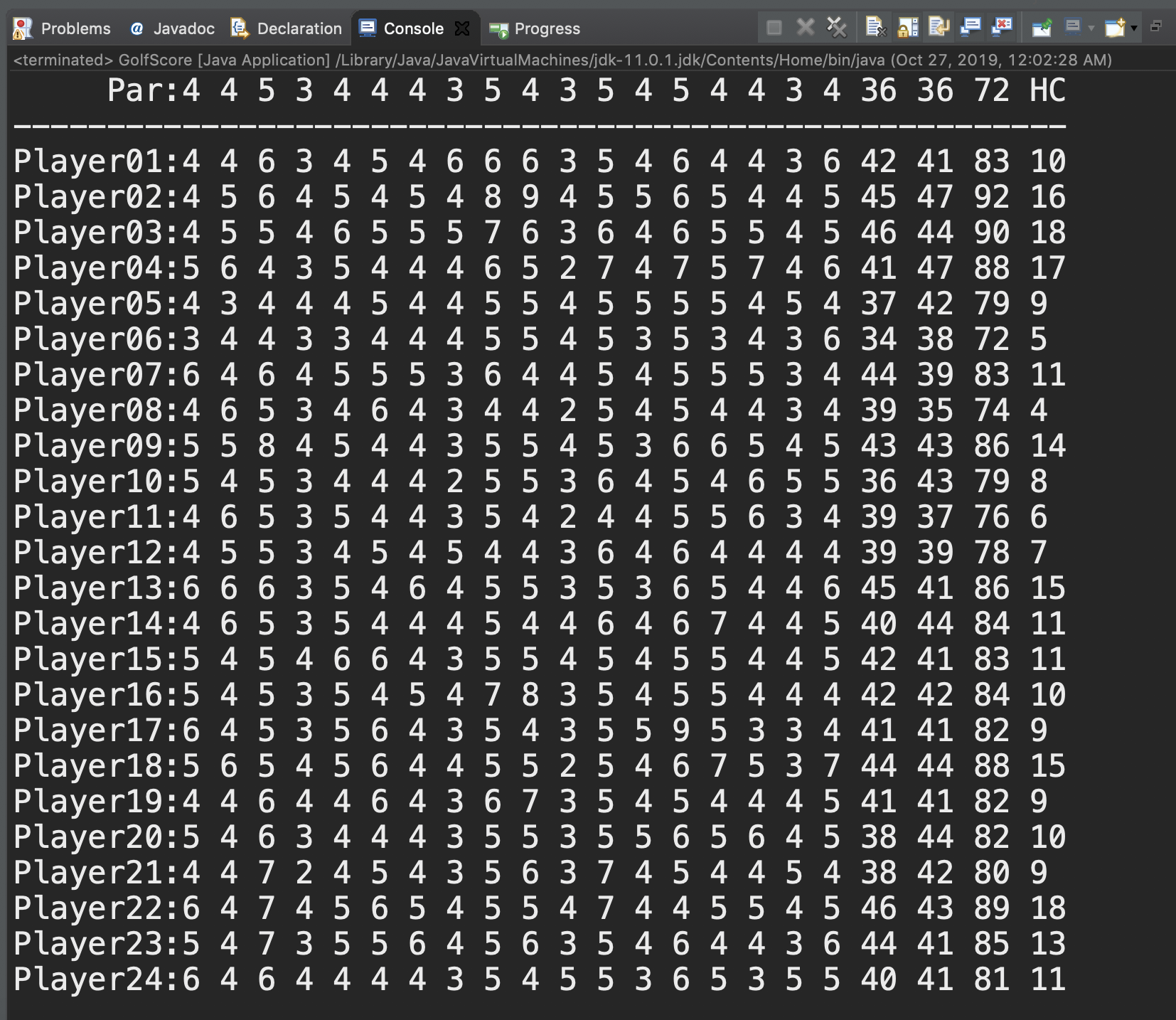
* Read <http://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html> to learn how to read a text file using an instance of the Scanner class.
* **Par**: A hole is classified by its par, meaning the number of strokes a skilled golfer should require to complete play of the hole. Normally, there are par 3’s, par 4’s and par 5’s.
* **Score**: The goal is to play as few strokes per round as possible. A golfer's number of strokes in a hole, course, or tournament is compared to its respective par score, and is then reported either as the number that the golfer was "under-" or "over-par", or if it was "equal to par". A hole in one (or an "ace") occurs when a golfer sinks their ball into the cup with their first stroke from the tee. Common scores for a hole also have specific terms.
* A **handicap (HC)** is a numerical measure of an amateur golfer's ability to play golf over the course of 18 holes. A player's handicap generally represents the number of strokes above par that the player will make over the course of an above-average round of golf. The better the player the lower their handicap is.
* To calculate handicap using System-36 formula, use the following table:

|  |  |
| --- | --- |
| Score - Par | HC get for that hole |
| <= 0 | 0 |
| 1 | 1 |
| >= 2 | 2 |

* Total HC is total HC of each hole.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Par | 4 | 4 | 5 | 3 | 4 | 4 | 4 | 3 | 5 | 4 | 3 | 5 | 4 | 5 | 4 | 4 | 3 | 4 | (72) |
| Score | 4 | 4 | 6 | 3 | 4 | 5 | 4 | 6 | 3 | 5 | 6 | 4 | 4 | 5 | 4 | 5 | 5 | 5 | 82 |
| HC | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 11 |

* Gross score is total strokes = 82
* Handicap = sum of hc for each hole = 11
* Net score = Gross score – Handicap = 82 -11 = 71
* Write a Java program (GolfScore.java) to generate golfer result as shown in the following screen
  + Hint: The code to read text file is close to Lab07!
  + Hint: Please follow the template code “GolfScoreTemplate.java”!



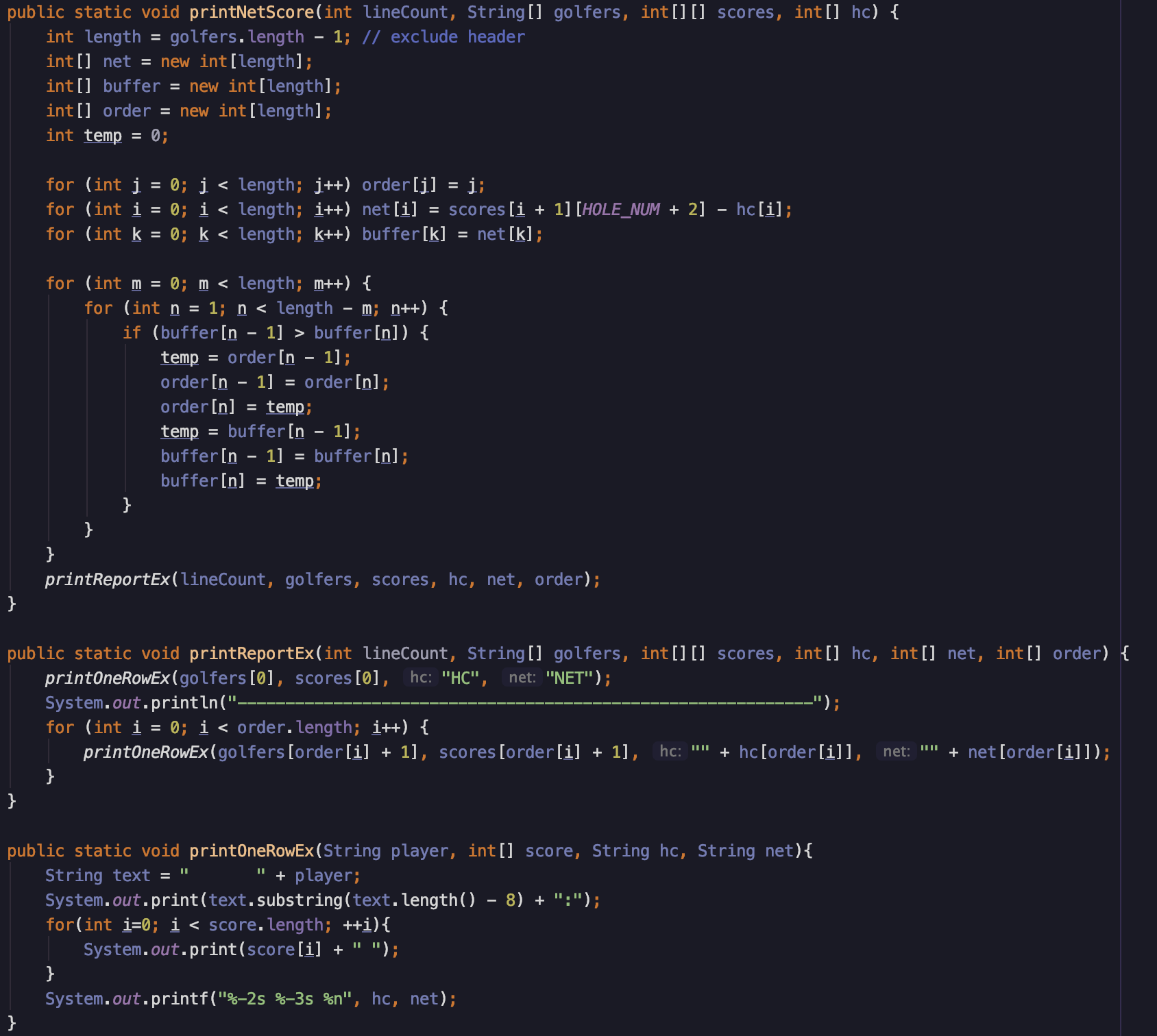
List your code here.



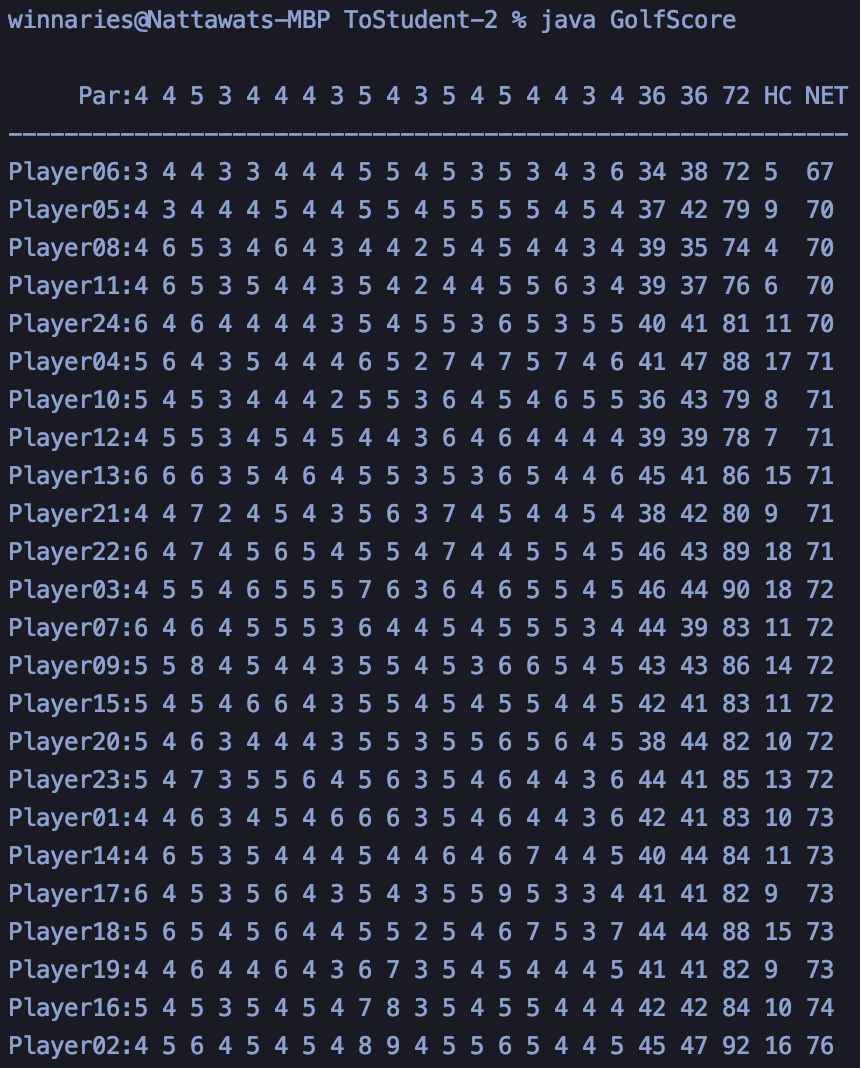
* Add method **printNetScore(…)** to show

Playername, out, in, gross, hc, net

In order for lowest net score to highest net score



Run the program an capture the screen here.



## Part B: Understanding Given Resources/Backgrounds

1. Get yourself familiar with the RGB color model. Play around with the color picker on <http://www.colorpicker.com/> and answer the following questions.
   1. What color is it that has the maximal value in B, G and 0 in R? Capture the picture of the color and post it here.



* 1. What color is it that has the maximal value in R, and middle value in G and B? Capture the picture of the color and post it here.



* 1. What are the requirements on the RGB values for all shades of gray?

Each of the R, G, and B values must be equals.

1. Read the API specification of the class *Java101ImageUtil* in Lecture9.
   1. How many static methods are there in the class?

According to the Lecture9, it specified a total of 4 methods. (However, in the source code itself, there are 6.)

* 1. How many overloaded methods are there in the class?

There are 3 overloaded methods named showViewer().

* 1. Write the “method signatures” of all the overloaded methods. (\*\* Write only the signature)

showViewer(int [][][] rgb, String title)

showViewer(int [][][] rgb1, int [][][] rgb2,String title)

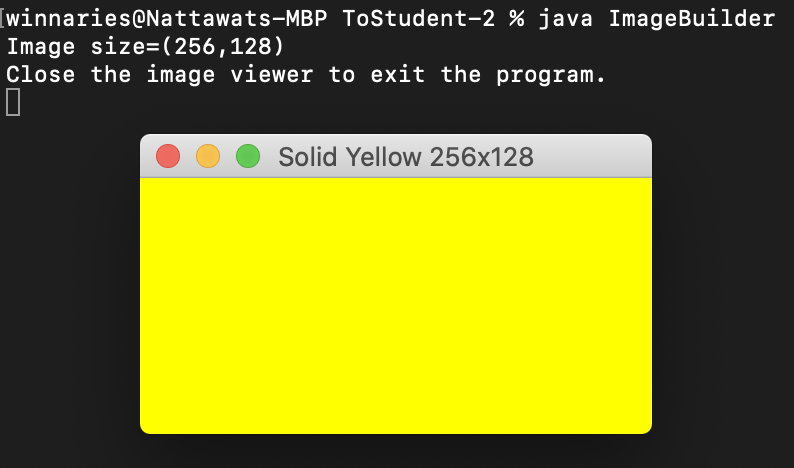
showViewer(int [][][][] rgbs, String title)

1. Read the source code of *Java101ImageUtilExample.java* and try executing the program. Briefly explain what the program does. (\*\* It is recommended NOT TO open big images. The program was not optimized in any ways. Try the program on some images with a few hundreds of pixels in their width/height)

What it does is basically loading an image and opening that particular image with a twist. The program first requires you to choose the image file as an input of the program. Then, it will provide you three choices of how you want it to show the input image, in which you can then input a number between 1 and 3 to choose each choice respectively. The first choice shows only an original version of the image. The second choice shows you an original image and its flipped version, the former on the left and the latter on the right. The third choice shows both image that the second choice shows, and it will show an additional 64x128 image of a solid red patch on the rightmost of the overall image. After the image have been shown to the user, the program will again provide you with three choice and so forth, repeating the process until you input a number that is not in range of 1 and 3 inclusively.

## Part C: Creating RGB arrays for Desired Images

1. Write a program performing the following steps.
   1. Create a 3-D array of int that when used with showViewer(int [ ][ ][ ],String), the program shows a 256-pixel (width) x 128-pixel (height) **all-yellow** image. Note the method to refer to each pixel is “image[columIndex][rowIndex]”
   2. Show the image with showViewer(int [ ][ ][ ],String)



## Part D: Image Manipulation

Modify *DesaturateIt\_Template.java* to obtain a Java program performing the following steps.

1. Ask the user to select a gif or a jpg file. (Some example files are provided.)
2. Show the original image, grayscale, and its sapia version using showViewer() as shown below

Note: Converting a color image into sepia image is very simple. All we have to do is repeat 3 simple steps for each pixels of the image.

1. Get the RGB value of the pixel.
2. Calculate tr, tg and tb using the formula

tr = 0.393R + 0.769G + 0.189B

tg = 0.349R + 0.686G + 0.168B

tb = 0.272R + 0.534G + 0.131B

Take the integer value.

1. Set the new RGB value of the pixel as per the following condition:

If tr > 255 then r = 255 else r = tr

If tg > 255 then g = 255 else g = tg

If tb > 255 then b = 255 else b = tb

Example of images



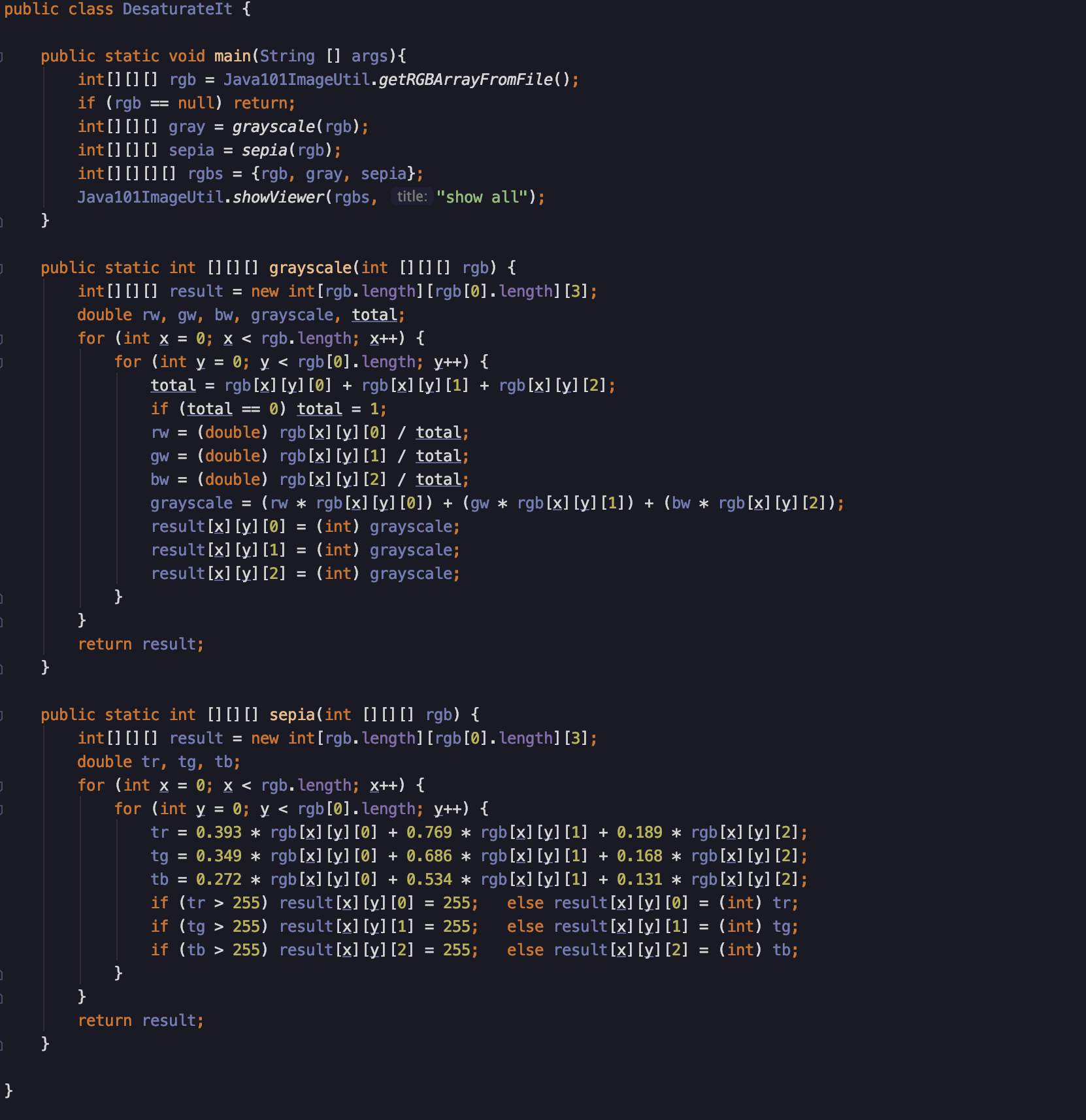
Explain how the grayscale values are computed.

We can calculate the grayscale value by using weighted methods, in which we combine each RGB values and weight each of it rather than just divide by 3. For example, if we have a RGB value of 100, 150, and 200 respectively, the weighted value of each color is itself divide by the total RGB value, which are (100 / 450), (150 / 450), and (200 / 450) respectively. Lastly, the grayscale value will be the sum of each color value multiply its weight, as such the grayscale value of our example would be (100 / 450) \* R + (150 / 450) \* G + (200 / 450) \* B. We then finally replace all RGB value with the grayscale value.

If a color pixel has RGB value = 100, 150, 200 for R, G, and B respectively. Calculate the value of tr, tg, tb.

|  |  |  |  |
| --- | --- | --- | --- |
| RGB 100, 150, 200 | tr | tg | tb |
| 192 | 171 | 134 |

List your source code here.



Submit this worksheet (by only one member of the group) via <http://www.myCourseVille.com> (Assignments > Hands-on Experiment # 9) **within the day after your lecture.**