برنامه نویسی پیشرفته: تمرین سری چهارم

تاریخ تحویل: تا دوشنبه ۱۲/۳۰ ساعت ۱۱:۵۵ (حداکثر تا سه روز بعد از این تاریخ قادر به ارسال تمرینات هستید و به ازای هر روز ۲۵% نمره کسر می‌گردد)

توصیه می‌شود درباره تمرینات با همکلاسی‌های خود به صورت گروهی به بحث و تبادل نظر بپردازید اما این به معنای تقلب، کپی کردن و... نمی‌باشد و تمام تمرینات باید توسط شما حل و پیاده سازی شود.

تمام تمرینات در یک پوشه با شماره دانشجویی و شماره تمرین و با ساختار به صورت زیر قرار گیرد، در غیر این صورت به عنوان عدم دریافت تلقی می‌شود.

HW_4_9413901

***توجه:** تمامی تمرین‌ها می‌بایست در قالب یک پروژه ارائه شوند که در آن کلاس‌های مربوط به هر سوال درون یک package مجزا (به عنوان مثال com.aut.hw4.question1 ، com.aut.hw4.question2 و ...) قرار می‌گیرد. همچنین یک کلاس برای اجرای سوالات در نظر بگیرید که حاوی متد main بوده (مثلا Main.java که در بسته‌ی com.aut.hw4 قرار می‌گیرد) و برای اجرای کد هر سوال یک متد در این کلاس بنویسید (به عنوان مثال private static void runQuestion1()) که تمامی کارهای مربوط به این سوال در آن انجام شده و این متد در main فقط فراخوانی می‌شود.

****توجه**: قسمتهای توضیحی در یک فایل pdf و درکنار پوشه تمرینات و همگی در یک فایل Zip ارسال شوند.

۱) مجموع مربعات (۱۰)

یک کلاس با نام SumSqures طراحی کنید:

أ. یک ArrayList از اعداد صحیح وجود داشته باشد.

ب. نوشتن javaDoc مناسب برای تمام متغیرها، سازنده‌ها، و توابع (به جز getter/setter)

ج. یک تابع setter داشته باشد که یک ArrayList از اعداد به عنوان ورودی دریافت کند.

د. یک تابع writeOutSquare داشته باشد که اعدادی که در ArrayList وجود دارند را به توان ۲ برساند و در یک فایل با نام sumSqure.txt ذخیره کند و همچنین در خط دوم فایل مجموع این اعداد را ذخیره کند.

توجه کنید که **به هیچ عنوان** از کلاس Scanner در داخل SumSqures استفاده نکنید و باید در خارج از کلاس از آن استفاده کرده و فقط لیستی از اعداد به این کلاس فرستاده شود (یک عدد m از کاربر گرفته و به همان تعداد از کاربر عدد دریافت کند).

Sample input:

1 2 3 4 5 10

Sample output:

1 4 9 16 25 100 //first line of your output file

145 //second line of your output file

۲) تعداد کاراکترها (۲۰)

أ. یک کلاس با نام CharDictionary بنویسید که یک تابع با نام charFrequency داشته باشد که به عنوان ورودی آدرس یک فایل را گرفته و سپس تعداد کاراکترهای استفاده شده در این فایل را در یک HashMap ذخیره کند (در صورتی که حروف بزرگ استفاده شده بود آن را با حروف کوچک یکسان درنظر بگیرید).

ب. یک تابع با نام displayFrequency داشته باشد که به ترتیب حروف الفبا تعداد هر کاراکتر را نمایش دهد (کاراکترهایی که استفاده نشده‌اند در خروجی نمایش داده **نشود**).

ج. یک تابع با نام saveFrequency داشته باشد که مانند displayFrequency حروف و تعداد تکرار آنها را در یک فایل با نام frequency.txt ذخیره کند.

د. برای تمام عناصر کلاس (متغییرها، توابع و سازنده‌ها) javaDoc مناسب نوشته شود.

Sample input:

Assume this is a text on text file zzz.

Sample Output:

a = 2

e = 4

h = 1

i = 3

l = 1

n = 1

….

z = 3

. = 1

۳) طولانی ترین زیر رشته مشترک (۲۵ نمره)

أ. یک کلاس با نام LCS ایجاد نمایید که در آن یک تابع با نام longestCommonSubsequence وجود داشته باشد که دو آدرس به عنوان ورودی دریافت کند (هرکدام آدرس یک فایل متنی است) و سپس بزرگترین زیر رشته مشترک بین این دو فایل را در یک فایل دیگر با نام longest.txt ذخیره کند. همچنین برای تمام متغیرها، سازنده‌ها، و توابع (به جز getter/setter) javaDoc مناسب نوشته شود.

Sample Input:

File1.txt =

abcaaadoooeewe

File2.txt =

a01299ooooeewezzzzzzzzzz

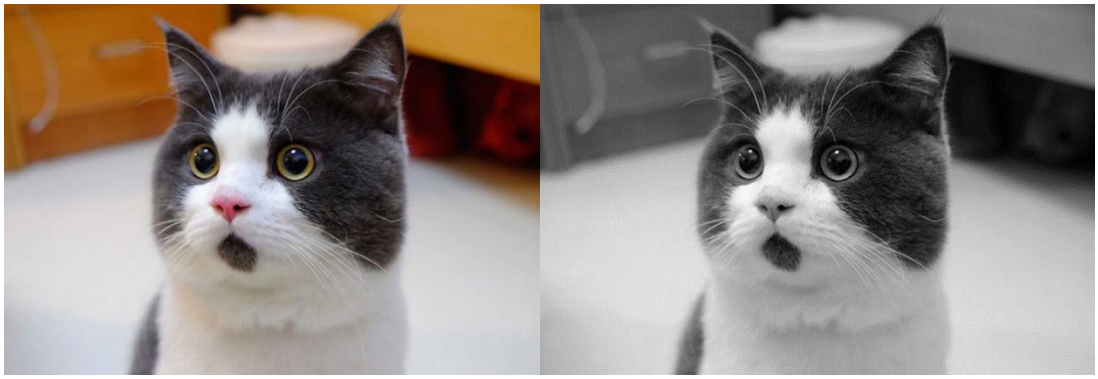
Sample Output:

oooeewe

punlic String getLongestCommonSubsequence(String pathFile1, String pathFile2){}

۴) !!!Instagram (۴۵ نمره)

در این تمرین میخواهیم ببینیم که چگونه میتوان یک عکس با فیلترهای مختلف (مشابه آن چیزی که در اینستاگرام می‌بینید!) ایجاد کرد.



همانطور که میدانید هر عکس از چندین هزار پیکسل تشکیل شده است. به عنوان مثال اگر یک عکس با اندازه ۱۰۰*۱۰۰ را در نظر بگیرید این عکس از ۱۰۰۰۰ پیکسل تشکیل شده است که هر کدام از این پیکسل‌ها نشان دهنده یک یک عدد ۳۲ بیتی هستند که از ترکیب چهار مولفه اصلی آلفا، قرمز، سبز، آبی (Alpha, Red, Green, Blue) ایجاد شده اند (بیشتر بخوانید). که آلفا میزان شفافیت (Transparency) و قرمز، سبز، آبی رنگ آن پیکسل را مشخص می‌کنند.

ALPHA								RED								GREEN								BLUE							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

هر پیکسل از ۳۲ بیت تشکیل شده است و هر ۸ بیت نشاندهنده یک مولفه رنگی هستند.

با استفاده از کلاس ImageIO می‌توانید یک تصویر از ورودی بخوانید یا در خروجی بنویسید و با استفاده از تابع getRGB(i,j) می‌توانید مقدار پیکسل در خانه i ,j در تصویر بدست آورید. با استفاده از کد داده شده، موارد زیر را پیاده سازی کنید.

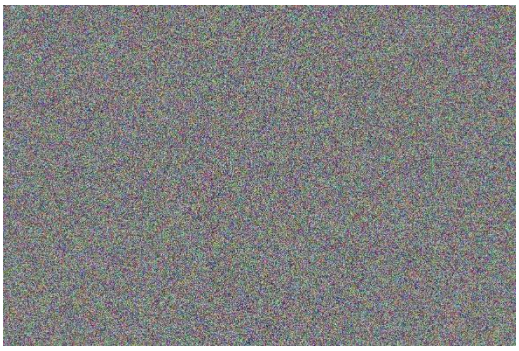
أ. یک کلاس با نام FilterImage بنوسید که یک سازنده با دو ورودی داشته باشد که یکی از آنها آدرس فایل ورودی و دیگری آدرس فایل خروجی باشد.

ب. سه تابع filterRedColor، filterBlueColor و FilterGreenColor وجود داشته باشد که در کدام از آنها یکی از مولفه‌های رنگی (قرمز، آبی، سبز) صفر باشد (به این ترتیب تصویر شما فاقد آن رنگ خواهد شد).

ج. یک تابع blackWhite بنویسید که تصویر ورودی را به سیاه و سفید تبدیل کند (کافی است مقدار مولفه‌های رنگی در هر پیکسل را برابر میانگین (r+g+b) / 3 درنظر بگیرید).

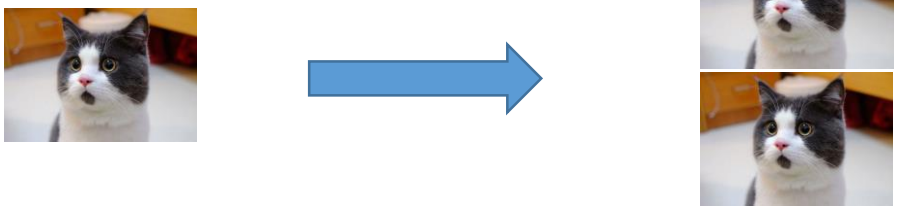
عکس شما بعد از فراخوانی تابع blackWhite مشابه این تصویر خواهد شد.

د. یک تابع randomImage بنویسد که بتواند یک عکس تولید کند. برای اینکار ابتدا با استفاده از کلاس BufferedImage یک عکس خالی با اندازه ۲۰۰*۲۰۰ بسازید سپس مقدار هر مولفه رنگی را به صورت تصادفی با استفاده از کلاس Random بین 0-255 مقداردهی کنید.



یک تصویر تصادفی که توسط تابع randomImage تولید شده است.

ه. یک تابع duplicateImage بنویسید که آدرس یک عکس را به عنوان ورودی گرفته و مشابه همان عکس را دقیقا در **زیر** آن کپی کند. یعنی آدرس یک عکس ۱۰۰*۱۰۰ را به عنوان ورودی تابع دهید و تابع شما یک عکس ۲۰۰*۱۰۰ ذخیره کند.



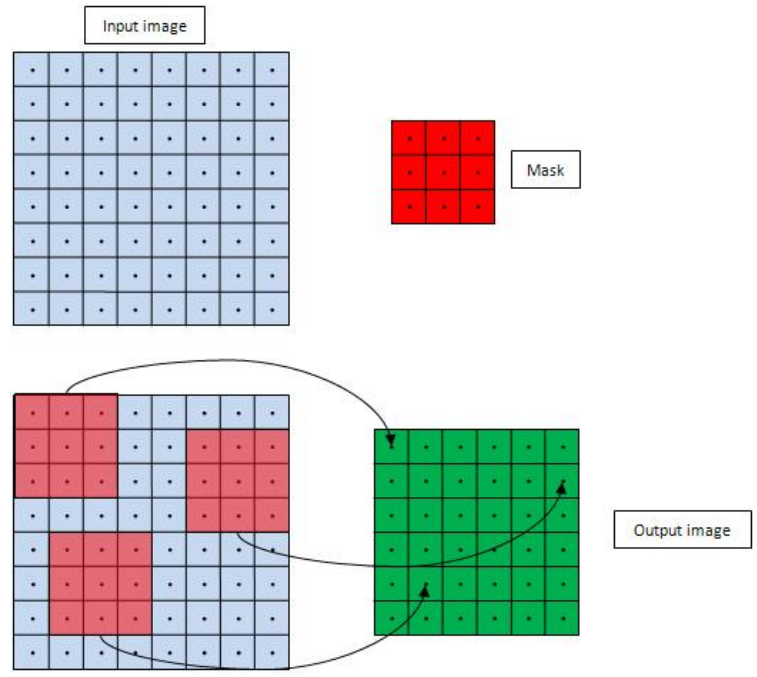
و. یک تابع negativeImage بنویسد که مطابق زیر عمل کند.

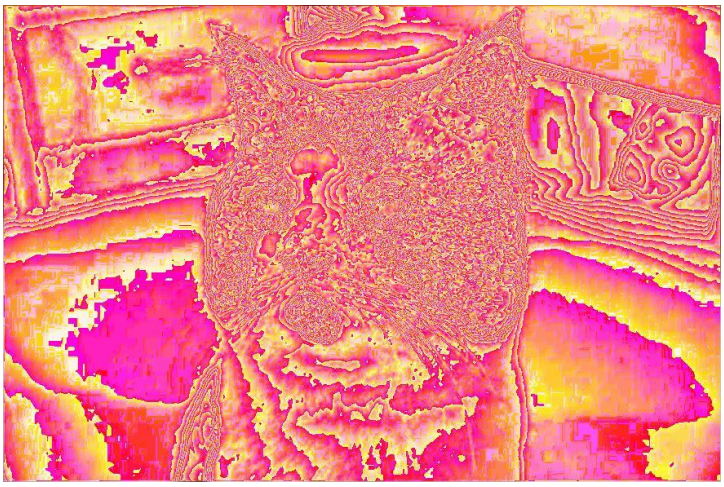
a. ابتدا یک ماتریس ۳*۳ به نام mask در نظر بگیرید که تمام درایه های آن با مقدار ۰٫۲۵ پر شده باشد.

b. یکی یکی پیکسلهای تصویر ورودی را پیمایش کرده (مشابه تمرین قبل) و mask را برروی آن اعمال کنید. به این معنا که برای محاسبه پیکسل [1,1] باید فرمول زیر را محاسبه کنید:

pix[1,1] = (pix[0,0]*0.25 + pix[1,0]*0.25 + pix[2,0]*0.25+ pix[0,1]*0.25 + pix[1,1]*0.25 + pix[2,1]*0.25 + pix[0,2]*0.25 + pix[1,2]*0.25 + pix[2,2]) / 9

همانطور که واضح است در سطرهای کناری نمیتوان ماسک را اعمال کرد به همین دلیل میبایستی به اندازه طول و عرض تصویر به طرفین عکس صفر اضافه کنید.





عکس شما بعد از اعمال تابع negativeImage مشابه این عکس خواهد شد.

ه. یک تابع با نام motionBlurImage بنویسید که دقیقا مشابه سوال قبل عمل کند با این تفاوت که ماتریس mask به ترتیب به صورت زیر باشد:

```
double mask2[][] = {{ 1, 0 , 0, 0 , 0} , { 0, 1, 0, 0, 0} , { 0, 0, 1, 0 , 0}, {0,
0, 0, 1, 0}, {0, 0, 0, 0, 1} }
```

توجه: یک عکس را به عنوان نمونه انتخاب کنید و تمام نتایج حاصل را برروی آن اعمال کنید و تمامی عکسها را به همراه کد ارسال نمایید.

بیشتر بدانید: برای مشاهده فیلترهای دیگر میتوانید به اینجا مراجعه کنید و آنها را برروی عکسهای خود آزمایش کنید.

Sample code:

```
//in your Main class
ImageFilter imageFilter = new ImageFilter("D:\\image\input.jpg","D:\\image\output.jpg");


// in your ImageFilter class
File f = new File(inputFilePath);
BufferedImage img = ImageIO.read(f); // read image (Handle exception yourself)
int width = img.getWidth();
int height = img.getHeight();
 for ( int i = 0; i < width; i++){
      for (int j = 0; j < height; j++){
         int pixel = img.getRGB(i,j);  // get each pixel with getRGB method
         int alpha = … //shift pixel 24 bit to get alpha
         int red =  …// shift pixel 16 bit to get red color
        //… do the same thing for blue & green
        int newPixel = … //contact alpha, red, green, blue to create new pixel
        img.setRGB(i, j, newPixel);
    }
}

File newFile = new File(outputFilePath);
ImageIO.write(img, "jpg", newFile); // save result as jpg image (Handle exception yourself)
```