






CSCI 136 Programming Exam #2
Fundamentals of Computer Science II
Spring 2013

This part of the exam is like a mini-programming assignment. You will create a program, compile it, and debug it as necessary. This part of the exam is open book and open web. You may use code from the course web site or from your past assignments. When you are done, submit all your Java source files to the Moodle exam #2 dropbox. Please ***double check you have submitted all the required files.***

You will have 100 minutes. No communication with any non-staff members is allowed. This includes all forms of real-world and electronic communication.

Grading. Your program will be graded on correctness and to a lesser degree on clarity (including comments) and efficiency. You will lose a substantial number of points if your code does not compile or crashes on typical inputs.

Overview. You are implementing a command-line utility that can perform batch processing of image files. A given input picture can undergo 3 different kinds of transforms:

Name	Code	Description	Input	Example output
				
Flip	F	Flips the image horizontally	-	
Color	C	Multiplies the red, green and blue components of each pixel by a red, green and blue floating-point factor	1.0 1.0 2.0	
Mask	M	Turns black certain pixels in the image based on another image. Wherever the mask image has white pixels, the original image remains the same, the rest is turned black.		

Input files. You will be completing a program BatchTransform that takes two command-line arguments. The first is a text file that specifies one transform per line. Here is the contents of `trans.txt`:

```
F flip
C red2 2.0 1.0 1.0
C blue2 1.0 1.0 3.0
C bright 1.5 1.5 1.5
M heart heart.png
```

The first letter on each line specifies the type of transform using one of the code letters from the above table. This is followed by a string used to uniquely specify a particular transform (since there may be multiple transforms of the same type but with different parameters). You can assume these identifier strings are unique within the transform file.

Following this are 0 or more options that configure the particular type of transform. The flip transform takes no additional options. The color transform takes three floating-point numbers for the red, green, and blue factors respectively. The mask transform takes the filename of its mask image file. You can assume the code letter will be uppercase and the number and types of parameters for a given code will be correct.

The second text file is a script that specifies one or more input files, what transforms to do on that input and the filename for the resulting output. Here is the contents of `scripts.txt`:

```
dont_panic.png out_flip.png flip
dont_panic.png out_blue.png blue2
dont_panic.png out_heart.png heart
```

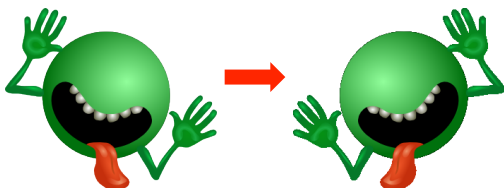
Each line contains a separate request for image processing. The first token is the input image filename. The second is the filename to write the processed image result to. This is followed by ***zero or more*** strings specifying transform identifiers from the first file. Your program should ***ignore any unknown identifiers***.

Getting stated. Download the file <http://katie.mtech.edu/classes/csci136/transform.zip>. You will be making use of the provided `Picture.java` class to manipulate image data. Here is all of the `Picture` API that you need to know:

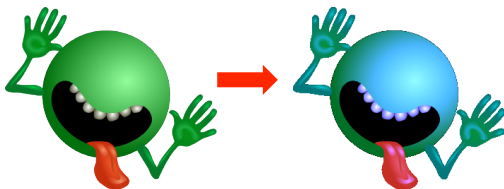
```
public class Picture
-----
    Picture(String filename) // Load a picture from the specified image filename
    Picture(int w, int h) // Create a blank w-by-h picture, where each pixel is black
    Color get(int i, int j) // Get color of pixel at (i, j), note (0,0) is upper-left
    void set(int i, int j, Color c) // Set color of pixel (i, j) to c, note (0,0) is upper-left
    void save(String name) // Save the picture to a file in a standard image format
    void show() // Display the picture in a window (for debugging)
```

You will need to complete the image transform logic in the `TransformFlip.java`, `TransformColor.java`, and `TransformMask.java` files. You have been provided with a partially completed `BatchTransform.java` program and the `Transform.java` abstract base class. Depending on how you implement `BatchTransform`, you may or may not need to add instance variables/methods to the `Transform` base class and its subclasses.

Flip transform. This one is pretty simple, the `transform()` method just flips the pixels in the horizontal plane. Note that the `transform()` method in all your concrete subclasses should not modify the input `Picture` object but instead return a new `Picture` object.



Color transform. An instance of the color transform has specific red, green, and blue factors. If someone attempts to construct a color transform with a ***negative factor***, you should ***throw a runtime exception***. To calculate the color of a transformed pixel, multiply the original pixels red, green and blue components by its corresponding factor. Note you may have to “clip” color values to 255 if they exceed 255 (when specifying a `Color` with integers) or 1.0 (when specifying a `Color` with a floating-point value).



Mask transform. Turns pixels black in the original image if the corresponding pixel in the mask image is not completely white. Completely white is when the red, green and blue color components are all 255. If the input image and the mask image have different sizes, you should throw a runtime exception.



Batch transform. Besides the help message that the main program already prints, you should also add console output to display the number of loaded transforms as well as messages showing the input and output files processed:

```
% java BatchTransform trans.txt script.txt
Loaded 5 transforms
Processing dont_panic.png -> out_flip.png
Processing dont_panic.png -> out_blue.png
Processing dont_panic.png -> out_heart.png
```

Sample output. Here is another sample run on some of the provided input files:

```
% java BatchTransform trans2.txt script2.txt
Loaded 9 transforms
Processing dont_panic.png -> out1.png
Processing dont_panic.png -> out2.png
Processing dont_panic.png -> out3.png
Processing dont_panic.png -> out4.png
Processing dont_panic.png -> out5.png
Processing dont_panic.png -> out6.png
Processing elmo.jpg -> out7.png
Processing elmo.jpg -> out8.png
```



out1.png



out2.png



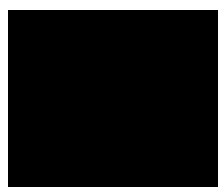
out3.png



out4.png



out5.png



out6.png



out7.png



out8.png