DM575: Exercises and Labs

Set 2

Writing client classes

One of the simplest ways to represent images is as a bidimensional array specifying for each dot (pixel) on the image its red, green and blue components. The class `Color` provides facilities for representing colours from their red, green, and blue components; it contains the following members.

- a constructor with three arguments that creates an object representing the colour with given red, green, and blue components (assuming their values are in the range $[0, 255]$);
- a constructor with one argument, an integer that encodes the red, green, and blue, in its three least significant bytes (`0xff0000` is red, `0x00ff00` is green, and `0x0000ff` is blue);
- methods `int red()`, `int green()`, `int blue()` that return the value of the red, green, and blue components of the colour;
- a method `int rgb()` that returns an integer encoding the red, green, and blue components in its three least significant bytes;
- a number of static fields with instances representing common colours (yellow, orange, etc.).

Observe that the same colour may be represented by distinct instances of `Color`, you should compare colours using their method `equals` instead of `==`. The class `Image` provides a simple interface for manipulating images in this format; it contains the following members:

- a constructor with two arguments that creates a black image with the specified dimensions;
- a constructor with three arguments that creates an image with the specified dimensions and background colour;
- a static method with one argument that takes the path to a file containing an image and returns an object representing that image (or `null` if it fails to load the file);
- methods `int width()` and `int height()` that return the width and height of this image;
- a method `Color pixel(int x, int y)` that returns the colour of pixel (x,y), assuming that these coordinates represent a valid pixel;
- a method `void setPixel(int x, int y, Color color)` that sets the colour of the pixel with coordinates (x,y) to the given values (assuming that the coordinates represent a valid pixel and that the color is not `null`);
- a method `void display()` that displays this image on screen.

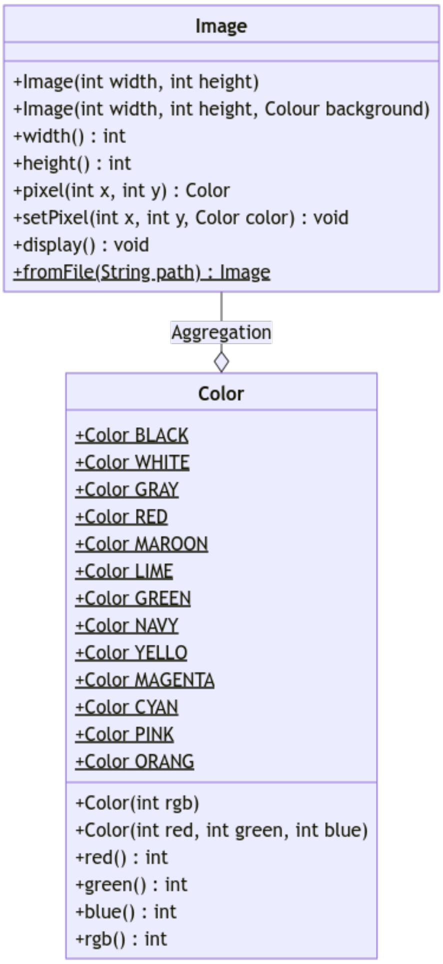

Figure 1: Class diagram for `Image` and `Color`.

By convention, pixels are counted from the top left corner of the picture.

Classes `Image` and `Color` contain assertions (`assert expr;`) for check that their contracts are respected. In java assertions are disabled by default (differently from Python) and can be enabled calling `java` with the flag `-ea` e.g., `java -ea ImageUtils`.

1. Your task is to develop a class `ImageUtils` implementing some of the usual image transformations available in most picture editors. In particular, you should implement the following (static) methods.

   (a) `Image flipHorizontal(Image image)` and `Image flipVertical(Image image)` that return a new image obtained by flipping the original image horizontally or vertically.

   (b) Three rotation methods `Image rotateLeft(Image image)`, `Image rotateHalf(Image image)` and `Image rotateRight(Image image)` that return a new image obtained by rotating the original image by performing a quarter turn, a half turn or a three-quarter turn counter-clockwise.

   (c) A method `Image stretchHorizontal(Image image)` that returns a new image with the same height as the original one and double the width, where every pixel is duplicated horizontally, and a similar method `Image stretchVertical(Image image)` operating on the vertical direction.

(d) A method `Image crop(Image image, int x, int y, int width, int height)` that returns a new image obtained by cropping the original one to the given area.

(e) Three methods `Image switchRedGreen(Image image)`, `Image switchRedBlue(Image image)` and `Image switchGreenBlue(Image image)` that return a new image where the components for two colours indicated have been switched.

(f) Three methods `Image grayscaleAverage(Image image)`, `Image grayscaleLightness(Image image)`, and `Image grayscaleLuminosity(Image image)` that return a new image where colours are converted to greys using the average, lightness, and luminosity methods, respectively.

- the average method assigns to each component value $\frac{(red+green+blue)}{3}$;
- the average method assigns to each component value $\frac{(\min(red,blue,green)+\max(red,blue,green))}{2}$;
- the average method assigns to each component value $0.3 \cdot red + 0.59 \cdot green + 0.11 \cdot blue$.

(In RGB greys have equal values for their red, green, and blue components.)

(g) A method `Color averageColor(Image image)` that computes the average colour of the given image, given by the average value of the red, green and blue components for every pixel in the image.

(h) A method `Image resample(Image image)` that returns a new image obtained from the original as follows: for each pixel not on the border, its red component is replaced by the average of the red components of the nine pixels in a 3×3 rectangle centred on the current pixel. The same process is applied to the green and blue components.

2. Develop an utility class `ImageUtilsSE` similar to the previous one, but where all methods are `void` and change the original image, rather than returning a new one. (Note that this class only includes the methods that preserve the dimensions of the image, since class `Image` does not allow these to be modified.) Additionally, you should implement the following (static) methods.

(a) A method `void addRectangle(Image image, int x, int y, int width, int height, Color color)` that draws a rectangle of the given color with size `width`×`height` with upper left corner in position `x`,`y`, truncated to fit the picture.

(b) A method `void addCircle(Image image, int x, int y, int radius, Color color)` that draws a circle of radius `radius` with centre in position `x`,`y`, truncated to fit the picture;

3. Develop an utility class `ImageCoder` that provides (static) methods for encrypting and decrypting an image using an external key. The methods that you should implement are the following:

(a) `void encrypt(Image image, String key)` that encrypts the image as follows: change each colour component of each pixel by adding it to the same colour component of the previously visited pixel and to the character code of the next character in `key` (modulo 256);

(b) `void decrypt(Image image, String key)` that decrypts an image, assuming it was encoded by the previous method using key.