P05 Memeage 5000

Programming II (CS300) Fall 2019
Department of Computer Sciences
University of Wisconsin-Madison

Due: 9:59PM on October 16, 2019
Pair Programming NOT Allowed for P05

Overview and Learning Objectives:

This assignment involves implementing a steganography system that hides a text message inside an image. In the examples below, the image on the left does not have a hidden message (that I am aware of), but the image on the right does. If you cannot tell the difference between these two images, that is the point of steganography.

Through the process of completing this assignment, you will practice making use of inheritance to extend the capabilities of some pre-existing class. You will also get more experience writing tests for your own code to help you gain confidence in its correctness.

Grading Rubric:

5 points	**Pre-Assignment Quiz:** You should not have access to this write-up without first completing this pre-assignment quiz through Canvas. If you have not already, please complete this quiz as soon as you have access to this course on Canvas.
15 points	**Immediate Automated Tests:** Upon submission of your assignment to Gradescope, you will receive feedback from automated grading tests about whether specific parts of your submission conform to this write-up specification. If these tests detect problems in your code, they will attempt to give you some feedback about the kind of defect that they noticed. Note that passing all of these tests does NOT mean your program is correct. To become more confident in this, you must write and run additional tests of your own.
10 points	**Manual Grading Feedback:** After the deadline for an assignment has passed, the course staff will begin manually grading your submission. We will focus on looking at your algorithms, use of programming constructs, and the style and readability of your code. This grading usually takes about a week from the hard deadline, after which you will find feedback on Gradescope.
20 points	**Supplemental Automated Tests:** When your manual grading feedback appears on Gradescope, you will also see the feedback from these additional automated grading tests. These tests are similar to the Immediate Automated Tests, but check your code against different requirements within this specification.
50 points	**TOTAL**

1. MemeageTests

Create a new Java Project in Eclipse (using Java 8) that is called: P05 Memeage 5000. Add this provided FourBytes.java source file to your project. Create a new class called MemeageTests to define tests for this and other classes within this P05 project. Familiarize yourself with the provided FourBytes class code. Focus your attention on the JavaDoc descriptions, since the bit operations are NOT a topic that is covered in CS300. Implement the following test methods inside your MemeageTests class. These tests should check the specific examples described in the provided JavaDocs for each respective method. They should then return true when the method's behavior matches the JavaDoc example, and false otherwise.

```
public static boolean testFourBytesGetBits()
public static boolean testFourBytesSetBits()
```

2. Color

Next create a new class called Color to represent the color of a single pixel (picture element) within an image. For this assignment, we are only considering 32bit colors that fit within 4bytes of memory (aka one of our FourBytes objects). You should use inheritance to share the implementation details of the provided FourBytes class with your new Color class.

Each color is broken into four separate bytes: alpha, red, green, and blue. Red, green, and blue are the primary colors of light that can be mixed together in different intensities (0-255) to make over 16 million different colors. The 8-bits of alpha are commonly used to represent the transparency vs opacity, but can also be used for other purposes. The organization of these bytes within a color should be as follows:

Most Significant Bits :	8-bits Alpha	8 bits Red	8-bits Green	8-bits Blue	: Least Significant Bits

Your color class must implement the following constructors and methods: to set and retrieve the specified parts of this data:

```
public Color(int argb)
public Color(int alpha, int red, int green, int blue)
public Color(Color other)

public int getARGB()
public int getAlpha()
public int getRed()
public int getGreen()
public int getBlue()

public void setARGB(int argb)
public void setAlpha(int alpha)
public void setRed(int red)
public void setGreen(int green)
public void setBlue(int blue)
```

You are **NOT ALLOWED** to declare any fields inside the Color class.

Create a test method inside your MemeageTests class with the following signature. This method should test and make use of at least one Color constructor, an accessor (getter) method, and a mutator (setter) method.

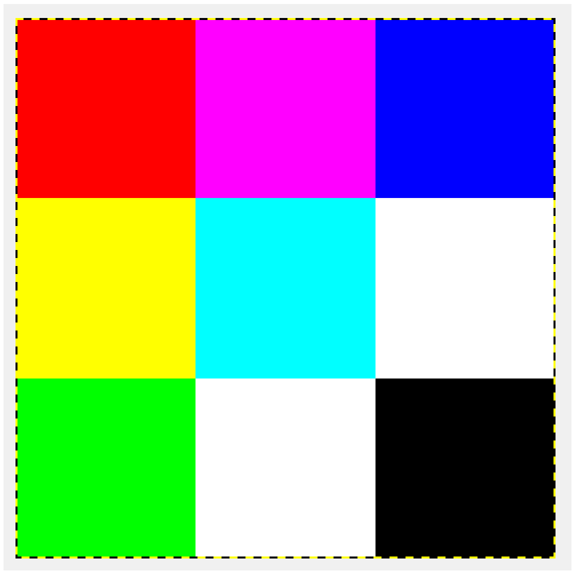
```
public static boolean testColor()
```

3. Image

Add this provided Image.java source file to your project. Download and add to your project folder this provided testImage.png. Familiarize yourself with the provided code in this class. Then implement the following test method inside your MemeageTests class. This method should load the provided image file and then ensure that 1) the width and height of this image are both three, and 2) the color of the center pixel within this image has 255 (the maximum byte value) for its Green and Blue color components and 0 (the minimum byte value) for its Red color component.

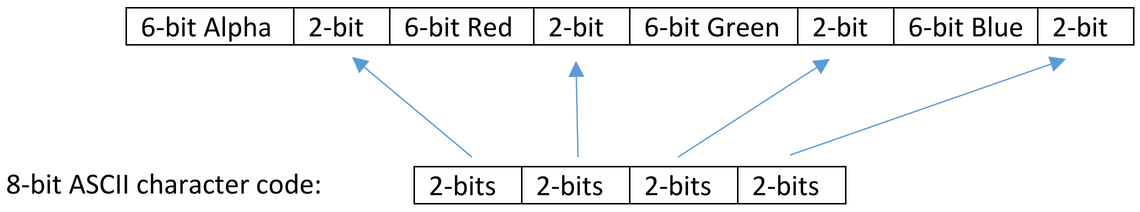
```
public static boolean testImage()
```

Here is a zoomed in visualization this 3x3 image's contents:



4. ColorPlusChar

Create a new class called ColorPlusChar which inherits from Color. The purpose of this class is to hide a single 8-bit ASCII character code within the two least significant bits of each color component (alpha, red, green, and blue). Since these least significant bits/digits have a small impact on the color that we perceive, it will be difficult to detect these changes for most human eyes. Here's how these bits are organized:



The constructors and methods within this ColorPlusChar class that you must implement include the following:

```
public ColorPlusChar(Color color, char character)
public ColorPlusChar(Color color)
```

```java
// stores 8-bit character within the least significant bits of color components
public void hideChar(char character)

// retrieves 8-bit character from the least significant bits of color components
public char revealChar()
```

As with the Color class, you are **NOT ALLOWED** to declare any fields inside the ColorPlusChar class.

Create a test method inside your MemeageTests class with the following signature. This method should test and make use of at least one ColorPlusChar constructor, the accessor (getter) method, and the mutator (setter) method.

```java
public static boolean testColorPlusChar()
```

5. Memeage

The final class that you will create for this assignment is called Memeage (for meme-image), and represents an image that may possibly include a String meme hidden within its colors. This class should inherit from the provided Image class, and should additionally implement the following constructors and methods:

```java
public Memeage(File file) throws IOException
public Memeage(File file, String meme) throws IOException, IllegalArgumentException

public void setMeme(String meme) throws IllegalArgumentException
public String getMeme() throws IllegalStateException
```

As with the Color and ColorPlusChar classes, you are **NOT ALLOWED** to declare any fields inside the Memeage class.

When storing a String meme within a Memeage, you will store one character per color/pixel location. The first character in the String should be stored in the color at the upper-left most corner of the image. Subsequent characters should be stored in colors that are one pixel to the right of the previous character, except when you reach the end of a row. When there are no more pixel locations left in a row, the next character should be stored in the left-most position of the next row down. Note that this order that we are storing characters within this image is the same order that we read English: left to right and top to bottom.

After all of the characters within a String meme message are stored, one additional character must be stored in the next Color/pixel location of your image: the null character: `'\0'`. The purpose this extra character is to mark the end of the meme, for when it is read out later.

There are two circumstances under which an IllegalArgumentExceptions may be thrown while attempting to store a new meme within a Memeage: 1) when the number of characters in the String meme is greater than or equal to the number of Colors/pixel locations within the image, and 2) when any character within the meme message has a character code that is greater than 127 (which typically indicates a non-ASCII characters). Ensure that any exceptions thrown in either circumstance include a descriptive message.

The job of getMeme() is to recover the string that was previously stored in this image: this could be a Memeage that setMeme() was just called on, or it could be one that was just loaded from a file. There are two circumstances under which an IllegalStateException may be thrown while attempting to read a meme out of a Memeage: 1) a character with a character code that is greater than 127 is extracted from a Color within the memeage, or 2) none of the characters extracted from this image contain the null character that should exist to mark the end of the meme message: `'\0'`. Ensure that any exceptions thrown in either circumstance include a descriptive message.

You are encouraged (but not required) to add additional test methods to your MemeageTest class to gain confidence that it is working correctly. Here are some more images that you can use your new Memeage implementation to decode: image01.png, image02.png, image03.png.

6. Assignment Submission

Congratulations on completing this assignment! The only four files that you should submit for this assignment include: Color.java, ColorPlusChar.java, Memeage.java, and MemeageTests.java. Your final score for this assignment is based on your submission that is marked "active" within gradescope.